Modulare und Wandlungsfähige Robotersysteme

Modellbasierte Softwareentwicklung basierend auf AutomationML und ontologischer Semantik

Yingbing Hua, Michael Mende und Björn Hein, Karlsruher Institut für Technologie

Die Softwareentwicklung bei Industrierobotern benötigt erhebliches interdisziplinäres Wissen und viel technische Erfahrung. Vor allem die Heterogenität der herstellerspezifischen Programmiersprachen und -werkzeuge verursacht hohen Aufwand beim Einsatz von Industrierobotern, obwohl Roboter per se frei programmierbar sind und für eine Vielzahl von Aufgaben eingesetzt werden können. Um verschiedene Rollen, wie z. B. den Komponentenzulieferer, Anwendungsentwickler sowie Systemintegratoren und Endanwender, bei der Programmierung und Integration von Robotern zu unterstützen, wurde im Rahmen des Forschungsprojekts ReApp ein modellbasierter Ansatz entwickelt. Das Datenaustauschformat AutomationML wurde hierbei für die Modellierung der Roboterkomponenten und -systeme eingesetzt. Auf Basis von Domäne-Ontologien wurden die AutomationML-Modelle semantisch verarbeitet und zu einem maschineninterpretierbaren Informationsmodell umgewandelt, aus dem Quellcodes generiert werden konnten.

Industrieroboter sind flexible Maschinen mit hoher Stabilität und Genauigkeit und sind mittlerweile essenzielle Voraussetzung für die Massenproduktion in z. B. der Automobil- und Elektroindustrie. Für Industrie 4.0-Lösungen wird jedoch die Anforderung gestellt, dass kleine Stückzahlen bei hoher Variantenvielfalt gefertigt werden sollen. Aufgrund der aufwändigen Programmierung und Konfiguration der Robotersoftware sind Industrieroboter für die individuelle Produktion aus wirtschaftlichen Gründen noch nicht geeignet.

Um die Einsetzbarkeit von Robotern zu erhöhen, insbesondere für kleine- und mittlere Unternehmen (KMUs), die deutlich mehr Produktvarianten mit kleinen Losgrößen fertigen wollen, wurden im Rahmen des Forschungsprojekts ReApp modellbasierte Methoden der Softwareentwicklung untersucht und herstellerneutrale Werkzeuge für die Roboterprogrammierung entwickelt. Zur Modellierung der Roboterkomponenten und Robotersysteme wurde das Datenaustauschformat AutomationML eingesetzt. Darüber hinaus wurde eine ontologische Semantik für die Interpretation der AutomationML-Modelle angewendet. Mit dieser formalen Semantik werden logische Konsequenzen automatisch hergeleitet, welche Anforderungen an die Funktionalitäten bzw. Schnittstellen der Robotersoftware ermitteln und für die angeschlossene Codegenerierung zur Verfügung stellen. Dadurch entsteht ein konsistenter und kontinuierlicher Entwicklungsprozess für alle beteiligten Rollen.

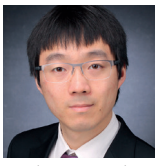
Modellbasierte Ansätze für die Roboterprogrammierung

Die Best Practice-Strategie des Softwareengineerings hat gezeigt, dass sowohl die Qualität als auch die Wirtschaftlichkeit der Software mit fortschrittlichen modellbasierten Technologien verbessert werden können. Für die industrielle Automatisierung wurden hierfür zwei fundamentale Ansätze betrachtet [1]. Der eine ist die komponentenbasierte Softwareentwicklung (Component based Software Engineering, CBSE), bei der wiederverwendbare Komponenten angestrebt werden. Jede Komponente ist ein eigenständiger Funktionsblock, der ohne große Änderung in unterschiedlichen Systemen integriert werden kann. Im Gegensatz dazu basiert die modellgetriebene Softwareentwicklung (Model-driven Software Development, MDSD) auf der automatisierten Codegenerierung aus abstrakten Modellen. Diese Methoden wurden in einem breiten Spektrum des industriellen Engineerings umgesetzt, wie z. B. der modellbasierten Validierung von Steuerungssystemen [2], der objektorientierten PLC-Programmierung [3]

Modular and Adaptable Robot Systems – Model Based Software Development Based on AutomationML and Ontological Semantics

Software development of industrial robots requires interdisciplinary knowledge and technical experience. Due to the heterogeneity of the manufacturer-dependent programming languages and tools, robot programming remains highly complex, although robots themselves are flexible and can be used for a wide range of applications. To support different roles during the development, including component provider, application developer, system integrator and end user, a model based approach was developed in the research project ReApp. The data exchange format AutomationML was used for the modelling of robot components and systems. Based on domain ontologies, the AutomationML models were processed semantically and converted to a machine-interpretable information model, from which source code was generated.

Keywords:
robotics, model based software development, AutomationML, semantics



Yingbing Hua arbeitet als wissenschaftlicher Mitarbeiter am Karlsruher Institut für Technologie, Fachgebiet Robotik und Industrieengineering.

Michael Mende arbeitet als wissenschaftlicher Mitarbeiter am Karlsruher Institut für Technologie, Fachgebiet Industrierobotik.

Prof. Dr.-Ing. habil. Björn Hein leitet die Forschungsgruppe Intelligente Industrieroboter des Instituts für Anthropomatik und Robotik am Karlsruher Institut für Technologie.

yingbing.hua@kit.edu
www.ipr.kit.edu

und verteilten Steuerungssysteme [4]. Darüber hinaus existieren mehrere populäre Modellierungssprachen wie z. B. die Unified Modelling Language (UML) der Object Management Group (OMG) [5] für die Anforderungsanalyse und Spezifikation der Software und das Eclipse Modelling Framework (EMF) für die effiziente Codegenerierung innerhalb der Entwicklungsumgebung Eclipse.

In der Domäne der Robotik wurde CBSE weitgehend erforscht [6, 7]. Es existieren bereits Frameworks, wie z. B. das Robot Operating System (ROS) [8], das sowohl von der Forschungsgemeinschaft als auch von zahlreichen Komponentenherstellern intensiv genutzt und erweitert wird. ROS bietet ein multimodales Kommunikationsnetzwerk für den Aufbau eines Robotersystems. Dazu zählen das Publisher-Subscriber- und das Server-Client-Konzept. Durch standardisierte Softwareschnittstellen von einfachen Datenstrukturen bis hin zu komplexen Geometrie- und Sensordaten kann eine Roboterkomponente in einer ROS-Laufzeitumgebung leicht angebunden werden. Umfangreiche kostenlose Robotersoftware, wie z. B. Komponententreiber, Bildverarbeitungs- und Bahnplanungsalgorithmen, sowie zusätzliche Entwicklungswerkzeuge, wie z. B. Visualisierung, Simulation und Netzwerkdiagnostik, stellen eine zuverlässige Basis für die Entwicklung neuer Applikationen. Allerdings kann ROS eine integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) nicht ersetzen. Insbesondere für MDSD sind grundsätzliche Funktionalitäten, wie z. B. die Modellvalidierung und die automatische Codegenerierung unerlässlich. Das Forschungsprojekt BRICS hat auf Basis von ROS das BRICS Component Model entwickelt, um MDSD für die Roboterprogrammierung voranzutreiben [9]. Die vier Abstraktionsebenen der OMG wurden umgesetzt und das EMF wurde für die Codegenerierung in einer Werkzeugkette eingesetzt.

Trotz dieser existierenden Lösungen lässt sich die effiziente Roboterprogrammierung nicht erreichen, da dort stets umfangreiche Expertise bei jedem Entwickler vorausgesetzt wurde. Für einen wirtschaftlichen Einsatz von Industrieroboter sollen die Aufgaben der an der Wertschöpfungskette beteiligten Akteure analysiert und individuell behandelt werden. Ziel ist eine konsistente und durchgängige Softwareentwicklung, wobei sich jeder Akteur nur auf seine eigene Aufgabe konzentriert. Die Heterogenität der Roboterdomäne stellt die primäre Herausforderung zur Erreichung des Ziels dar, die im Rahmen des Projekts ReApp durch einen semantikbasierten Ansatz gelöst wurde. Die Automation Markup Language (AutomationML) [10] wurde für die Beschreibung der Roboterkomponenten bzw. -systeme verwendet. AutomationML ist ein XML-basiertes Datenformat für kontinuierliches Engineering in einer heterogenen Werkzeug-Landschaft. Mit standardisierten Semantiken und wohldefinierten Modellierungsverfahren kann der Standard für die Beschreibung und den Austausch von Fabrik-Topologie, Geometrie, Kinematik, Logik sowie Kommunikation eingesetzt werden. Um

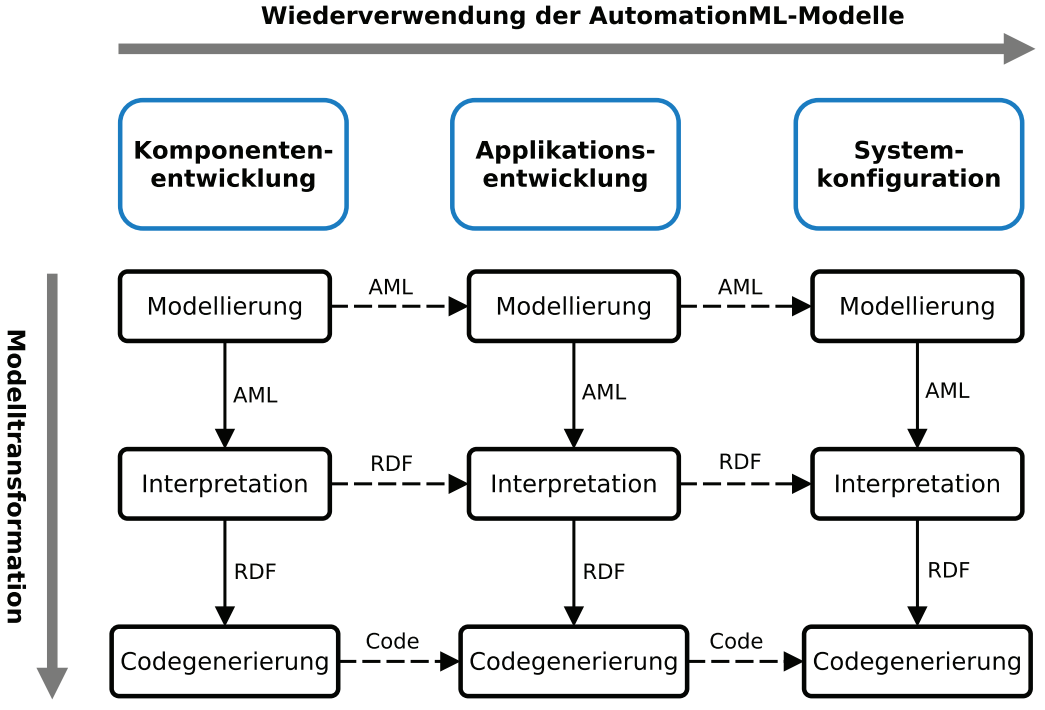


Bild 1: Ablauf des modellbasierten Ansatzes für die Roboterprogrammierung.

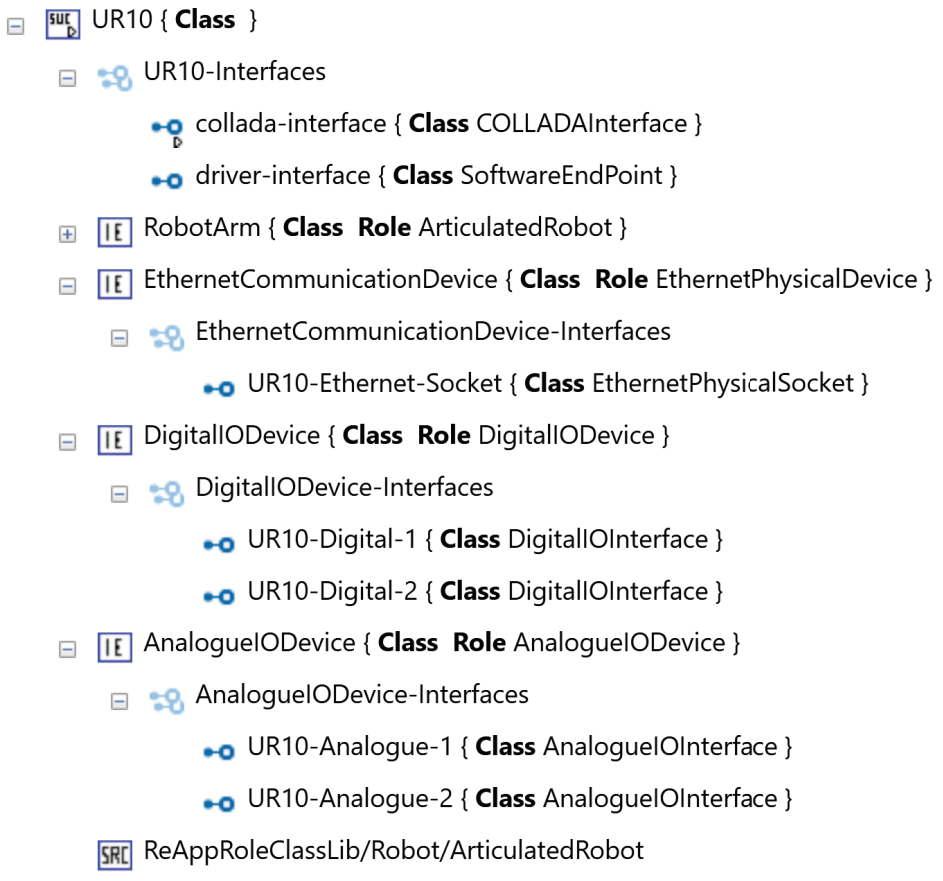


Bild 2: AutomationML SystemUnitClass des UR10 Roboters.

die implizite Semantik der XML-Daten zu extrahieren, werden die AutomationML-Modelle anschließend zu dem Datenformat Resource Description Framework (RDF) umgewandelt. Durch automatisierte Inferenz über das Domäne-Wissen der Robotik werden Anforderungen über die Funktionalitäten bzw. Schnittstellen der Robotersoftware abgeleitet. In Bild 1 wird der gesamte Ablauf des Ansatzes gezeigt. Die drei vertikalen Phasen in Form der Komponenten- und der Applikationsentwicklung sowie der Systemkonfiguration entsprechen der Rollentrennung der Akteure. In jeder Phase werden zielführende AutomationML-Modelle durch die zuständige Rolle erstellt und beim Bedarf in der nächsten Phase wiederverwendet. Die Codegenerierung bezieht sich auf die Ausgabe der semantischen Verarbeitung der AutomationML-Modelle. Die Interoperabilität der Modelle über die Phasen erfolgt durch die maschinenlesbare formale Semantik und der generierte Quellcode aller Phasen ist aufgrund der einheitlichen ROS-Schnittstellen integrierbar. In den folgenden Abschnitten wird die technische Umsetzung des Ansatzes detailliert aufgezeigt.

AutomationML als ein Modellierungswerkzeug

Im Kontext von Industrie 4.0 kann AutomationML ein wichtiges Format für die Speicherung von Struktur- und Ingenieurwissen werden [11]. Aufbauend auf den Industriestandard Computer Aided Engineering Exchange (CAEX) (IEC 62424) besteht die Architektur von AutomationML aus den folgenden Bausteinen:

- *RoleClass:* Eine Rollenklasse repräsentiert ein domänenspezifisches Konzept. Der AutomationML e.V. hat im Rahmen des Standards IEC 62714 grundsätzliche Konzepte der industriellen Automatisierung definiert, wie z. B. Sensor oder Aktor.
- *InterfaceClass:* Schnittstellen werden für die Verbindung zwischen Objekten genutzt und durch Schnittstellenklassen definiert.
- *SystemUnitClass:* Klassen der Systemobjekte sind Modelle für wiederverwendbare Komponenten. Zur Identifikation deren Semantik werden geeignete Rollenklassen referenziert. Eine SystemUnitClass kann durch Komposition interner Elemente weiter spezifiziert werden, um die gewünschte Granularität der Klassenbeschreibung zu erreichen. Verfügbare Schnittstellen der Komponente werden aufgezählt.
- *InstanceHierarchy:* Für den Aufbau eines gesamten Systems werden bereits modellierte Komponenten bzw. Subsysteme instanziiert und in einer Hierarchie angeordnet. Weitere Konfigurationen des Systems, wie z. B. Parametrisierung und Verbindungen zwischen einzelnen Komponenten, können vorgenommen werden.

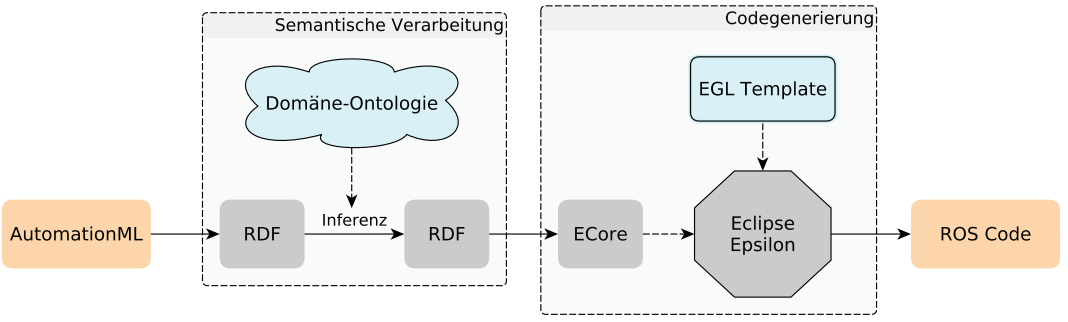


Bild 3: Modelltransformation von AutomationML zu Ecore und die anschließende Codegenerierung.

Durch die Unterstützung des objektorientierten Paradigmas eignet sich AutomationML für die Modellierung der Roboterkomponenten und -systeme, wobei die Rollenklassen und Schnittstellenklassen die Grundlage für die Modellierung darstellen. Zur ausführlichen Beschreibung der Roboterdomäne wurden die Standardbibliotheken von AutomationML um roboterspezifische Klassen erweitert. Diese Erweiterung stellt eine Taxonomie über die Roboterdomäne bereit, anhand der jede beliebige Roboterkomponente durch eine AutomationML-SystemUnitClass beschrieben werden kann. Mit dem Ziel, dass ein Quellcode für die Komponente generiert werden sollte, fokussiert das Komponentenmodell auf Softwarefunktionalitäten. Als Beispiel lässt sich ein UR10 Roboter durch einen Roboterarm, ein Ethernet Modul, eine digitale sowie eine analoge Ein- und Ausgabeeinheit beschreiben. Geometrische und kinematische Informationen des Roboters werden durch eine Referenz auf einer entsprechenden COLLADA-Datei in einer SystemUnitClass verwiesen. In Bild 2 ist das AutomationML-Modell des UR10 Roboters dargestellt, wobei ein großer Teil der Schnittstellen aufgrund der Granularität weggelassen wurde.

Ein Komponentenmodell kann durch den Komponentenzulieferer mit dem AutomationML Editor [12] mit geringem Aufwand erstellt werden und danach als Template für die Modellierung weiterer ähnlicher Komponenten dienen. Zum Beispiel kann ein UR5 Roboter durch einen einfachen Kopier- und Konfigurationsvorgang aus dem Modell des UR10 Roboters generiert werden. Diese Wiederverwendbarkeit des Modells bietet ein schnelles und konsistentes Engineering.

In der Applikationsentwicklung setzt sich ein Robotersystem aus Instanzen der zur Verfügung gestellten ROS-Schnittstellen zusammen. In der Phase der Systemkonfiguration werden die Instanzen entsprechend des realen Systems parametrisiert.

AutomationML bietet keine Mechanismen für die domänenspezifische Modellvalidierung. Die Kompatibilität der Verbindungen bzw. die Gültigkeit der Parametrisierung müssen für die Codegenerierung explizit sichergestellt werden. Zielorientierte Verifikationsroutinen können dafür implementiert werden, sind aber nicht generisch für jede Anwendung. Der ReApp-Ansatz stellt ein semantikbasiertes Verfahren vor, bei dem die AutomationML-Modelle auf Basis des allgemeinen Wissens über Robotik und ROS semantisch interpretiert und verifiziert werden.

Semantische Erweiterung der AutomationML-Modelle

AutomationML-Modelle sind aus Sicht des MDSD in der plattformunabhängigen Ebene der Abstraktionshierarchie der OMG. Um daraus einen Quellcode für eine spezifische Plattform erzeugen zu können (z. B. ROS), müssen diese Modelle erst auf die plattformabhängige (ROS) Abstraktionsebene übertragen werden. Das heißt, die Bedeutung der AutomationML-Modelle muss erst deklariert werden, um Fragestellungen wie z. B.: „Was bezeichnet eine Verbindung zwischen zwei Komponenten?“ oder „Welche Softwarefunktionalitäten soll ein Roboterarm zur Verfügung stellen?“ zu beantworten. Ein innovativer Ansatz dieser Modell-zu-Modell-Transformation ist der Einsatz einer formalen Semantik, die das Wissen mithilfe maschinell interpretierbarer Ontologien abbildet. Die W3C Standards Web Ontology Language (OWL) und Resource Description Framework (RDF) wurden für die Erstellung der Ontologien eingesetzt, in denen die inhärenten Eigenschaften einzelner Domäne-Klassen, wie z. B. Roboter und Kamera, spezifiziert wurden [13]. Auf Basis dieser Ontologien lässt sich eine AutomationML-Beschreibung zu einem kompatiblen RDF-Modell umwandeln. Zu jeder AutomationML-Rollen- bzw. Schnittstellenklasse wird die universelle Identifikation (URI) der entsprechenden Domänen-Klasse in der Onto-

Literatur

[1] Vyatkin, V.: Software engineering in industrial automation: State-of-the-art review. In: IEEE Transactions on Industrial Informatics 9 (2013) 3, S. 1234-1249.
[2] Estevez, E.; Marcos, M.: Model-based validation of industrial control systems. In: IEEE Transactions on Industrial Informatics 8 (2012) 2, S. 302-310.
[3] Obermeier, M.; Braun, S.; Vogel-Heuser, B.: A model-driven aproach on object-oriented plc programming for manufacturing systems with regard to usability. In: IEEE Transactions on Industrial Informatics 11 (2015) 3, S. 790-800.
[4] Yan, J.; Vyatktin, V.: Distributed software architecture enabling peer-to-peer communicating controllers. In: IEEE Transactions on Industrial Informatics 9 (2013) 4, S. 2200-2209.
[5] Object Management Group. URL: www.omg.org, Abrufdatum 26.09.2017.

logie vermerkt. Diese semantische Erweiterung der AutomationML-Modelle ermöglicht die automatische Inferenz über Domänen-Wissen, die in AutomationML nicht definiert wurde und zu einem endgültigen Informationsmodell für die Codegenerierung führt.

Im Falle des UR10 Roboters, der einen ArticulatedRobot enthält, muss sein ROS-Treiber aufgrund der Schnittstellendefinition in der Ontologie die entsprechenden Softwarefunktionalitäten zur Verfügung stellen. Hierbei sind ein Publisher des ROS-Topics vom Typ JointState und ein Client der ROS-Action vom Typ FollowJointTrajectory zu implementieren.

Im Falle eines Robotersystems, in dem mehrere Komponenten miteinander verbunden sind, muss zusätzlich die Kompatibilität gesichert werden. Für die Hardwarekommunikation können fehlerhafte Verbindungen durch semantische Abfrage herausgefunden werden. Beispielsweise wird eine Verbindung zwischen zwei digitalen Eingängen als inkompatibel angezeigt. Auf der Softwareebene darf eine ROS-Komponente nur über ROS-Schnittstellen des gleichen Typs mit einer anderen Komponente kommunizieren. Dazu wird der regelbasierte Inferenzmechanismus Semantic Web Rule Language (SWRL) angewendet. Eine Regel drückt ein Faktum der Domäne mittels formaler Semantik aus und ist allgemeingültig. Die Verifizierung durch die Inferenzmaschine läuft automatisch im Hintergrund der Modell-zu-Modell-Transformation und benötigt keine explizite Implementierung.

Automatische Codegenerierung

Die erfolgreiche Verifizierung stellt die Konsistenz des Informationsmodells sicher und ermöglicht die automatische Codegenerierung mit dem Eclipse Modelling Framework. Zuerst wird das semantische Informationsmodell durch eine weitere Modell-zu-Modell-Transformation in das Eclipse-konforme Datenformat Ecore konvertiert. Danach wird ein ROS-Paket mit Projektstruktur, Manifest und Quellcode (C++ oder Python) aus Eclipse Epsilon Templates erzeugt, die für ROS erstellt wurden. Im Falle einer Roboterkomponente lassen sich die grundsätzlichen Funktionalitäten der Komponente durch ROS-Schnittstellen darstellen. Attribute der Komponente werden als ROS Parameter für die Laufzeitumgebung zur Verfügung gestellt. Die COLLADA Beschreibung wird in ein ROS-konformes Datenformat konvertiert, um Open-Source-Komponenten, wie z. B. Koordinatentransformationen und kollisionsfreie Bahnplanung, nutzen zu können. Im Falle eines Robotersystems werden die Konfigurationsdaten in XML-Dateien gespeichert und für die Ausführung der Applikation bereitgestellt.

Das generierte ROS-Paket stellt einen guten Ausgangspunkt für die Entwicklung der Komponente bzw. des Systems dar. Der Komponentenentwickler kann sich auf die Implementierung der tatsächlichen Funktionalitäten der Komponente und der Applikationsentwickler kann sich auf die Realisierung der Ablauflogik des Programms konzentrieren. Zur Konfiguration eines existierenden Systems kann der Systemintegrator bzw. der Endanwender das Systemmodell in AutomationML modifizieren. Bild 3 zeigt die Transformationskette der Modelle. Die gesamte Modell-zu-Modell- sowie die angeschlossene Modell-zu-Code-Transformation sind für den Nutzer transparent.

Fazit

Modellbasierte Softwareentwicklung nutzt die Abstraktionsebenen, um die Rollentrennung zu fördern. Im Kontext von Industrie 4.0 wurde in diesem Beitrag ein semantikbasierter Ansatz vorgestellt, in dem Roboterkomponenten bzw. -systeme in AutomationML modelliert und semantisch verarbeitet wurden. Auf Basis der Domänen-Ontologien können logische Schlussfolgerungen über die Anforderung der Software abgeleitet und daraus ein Informationsmodell für die Codegenerierung zur Verfügung gestellt werden. Die Codegenerierung erfolgt unmittelbar mit dem Template-basierten Ansatz des Eclipse Epsilon Projekts. Jede Phase des Ansatzes kann durch einen bei der Softwareentwicklung Beteiligten selbstständig durchgeführt werden. Sowohl die Modelle als auch der Quellcode sind aufgrund der standardisierten Semantik sowie der ROS-Schnittstellen leicht zu konfigurieren und für verschiedene Applikationen wiederverwendbar.

Schlüsselwörter:
Robotik, modellbasierte Softwareentwicklung, AutomationML, Semantik

Dieser Beitrag entstand im Rahmen des Projekts „ReApp – Wiederverwendbare Roboterapplikationen für flexible Roboteranlagen basierend auf Industrial ROS", das von dem Bundesministerium für Wirtschaft und Technologie (BMWi) unter dem Kennzeichen 01MA13001J gefördert wird.

[6] Brugali, D.; Scandurra, P.: Component-based robotic engineering. In: IEEE Robotics & Automation Magazine 16 (2009) 4, S. 84-96.
[7] Brugali, D.; Shakhimardanov, A.: Component-based robotic engineering (part ii). In: IEEE Robotics & Automation Magazine 17 (2010) 1, S. 100-112.
[8] Robot Operating System (ROS). URL: www.ros.org, Abrufdatum 26.09.2017.
[9] Bruyninckx, H.; Klotzbücher, M.; Hochgeschwender, N.; Kraetzschmar, G.; Gherardi, L.; Brugali, D.: The brics component model: a model-based development paradigm for complex robotics software systems. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC) 2013.
[10] AutomationML Verein. URL: www.automationml.org, Abrufdatum 26.09.2017.
[11] Bedenbender, H.; Bentkus, A.; Epple, U.; Hadlich, T.; Hankel, M.; Heidel, R.; Hillermeier, O.; Hoffmeister, M.; Huhle, H.; Kiele-Dunsche, M.; Koziolek, H.; Lohmann, S.; Mendes, M.; Neidig, J.; Palm, F.; Pollmeier, S.; Rauscher, B.; Schewe, F.; Waser, B.; Weber, I.; Wollschlaeger, M.: Beziehung zwischen I4.0-Komponenten – Verbundkomponenten und intelligente Produktion. Berlin 2017.
[12] AutomationML Tools. URL: https://www.automationml.org/o.red.c/tools.html, Abrufdatum 26.09.2017.
[13] Zander, S.; Heppner, G.; Neugschwandtner, G.; Awad, R.; Essinger, M.; Ahmed, N.: A model-driven engineering approach for ros using ontological semantics. In: 6th International Work-shop on Domain-Specific Languages and models for ROBotic systems (DSLRob-15) co-located with the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2015.