Bei diesem Beitrag handelt es sich um einen wissenschaftlich begutachteten und freigegebenen Fachaufsatz („reviewed paper“).

doi.org/10.37544/1436–4980–2024–04–20

Blockbasiertes Programmieren und Steuern von kamerageführten Robotern via OPC-UA-Skills

Flexible BPMN-Steuerung für Robotersysteme

S. Kreuter, J. Abicht, P. Hold, M. Heinrich, T. Wiese, S. Schlund

ZUSAMMENFASSUNG Bei der Programmierung und Steuerung von Robotersystemen fehlen Standardisierungen, wodurch Experten bei Änderungen in den Randbedingungen mit Zeit- und Personalaufwand nachprogrammieren. Um den Programmieraufwand zu senken, wird ein Software-Framework zur BPMN-basierten Programmierung und Orchestrierung von skillbasierten Robotersteuerungen vorgestellt. Ein Bildverarbeitungs-Modul unterstützt bei der Parametrierung der Skills.

STICHWÖRTER

BPMN, Skillbasierte Steuerung, OPC-UA

Flexible BPMN-based control for robotic systems

ABSTRACT The programming and control of robot systems lacks standardization, resulting in time-consuming and costly reprogramming by experts when boundary conditions change. To reduce the programming effort, a software framework for BPMN-based programming and orchestration of skill-based robot control systems is presented. An image processing module supports the parameterization of the skills.

1 Einleitung

1.1 Ausgangssituation und Problemstellungen

Zwischen 2010 und 2025 wird sich der globale Konsum verdoppeln [1] und das Kaufverhalten vom Konsum der Massenproduktion hin zu individuell gefertigten Produkten wandeln [2]. Dieses geänderte Kaufverhalten spiegelt sich in der Industrie in der Fertigung kleiner, variantenreicher Lose wider und macht damit flexible, anpassbare Produktionssysteme unerlässlich [3]. Automatisierte Produktionssysteme kämpfen aber mit mangelnder Flexibilität [4], während demografischer Wandel und Fachkräftemangel die Stabilität und Wirtschaftlichkeit manueller Produktion beeinträchtigen [5]. Mit einem erwarteten massiven Rückgang der Arbeitskräfte durch Pensionierungen in den nächsten zehn Jahren [6] verschärft sich dieses Problem sogar noch.

Um diesem Problem entgegenzuwirken, werden vermehrt mobile oder statische Robotersysteme in die Produktion integriert, die repetitive oder präzise Aufgaben übernehmen, zum Beispiel Maschinenbeschickungen oder Montageaufgaben [7, 8]. Der Mensch bleibt dabei mit seiner Flexibilität und Fachwissen für komplexere Aufgaben essenzieller Bestandteil bei der Inbetriebnahme und Überwachung der Robotersysteme [9]. Aufgrund der Individualisierung der Produkte müssen sich Robotersysteme stärker an veränderte Umgebungsbedingungen (zum Beispiel Bauteile, Maschinen, Speicherart) anpassen und wandelbar sein, was einen höheren Inbetriebnahmeaufwand und Nebenzeiten nach sich zieht. Modulare Steuerungskonzepte, wie skillbasierte Steuerungen (skill-based control, SBC), verringern die Programmierzeit bei der Anpassung an neue Prozesse und Produkte durch einheitliche Steuerungsarchitekturen und Kommunikationsinterfaces [10]. Trotzdem sind derartige Ansätze aktuell nur wenig vereinheitlichte Insellösungen für dedizierte Steuerungen. Trotz vereinfachter Anpassbarkeit und Erweiterbarkeit der Programme, verbleibt jedoch die Notwendigkeit, die abstrakten Programmbausteine zeitintensiv zu reparametrieren (zum Beispiel durch Skillparameter in Skills). Der stärkere Einsatz von Sensorik, wie 2D-Kameras, die Umgebungsänderungen automatisiert detektieren und daraufhin Programme anpassen, sind für die Reduktion des Inbetriebnahmeaufwands zukünftig ebenso essenziell. Der geringe und aufgabenspezifische Einsatz von Sensorlösungen unterstützt aktuell nur begrenzt bei der Programmierung von Robotern und wird meist für Einzelaufgaben, wie Positionsregelungen verwendet.

Um Nebenzeiten gering und die Produktivität dennoch hochzuhalten, ist es notwendig, Steuerungen von Robotersystemen sowie deren Schnittstellen und Programmierung zu vereinheitlichen und durch Nutzung von Sensorsystemen zu beschleunigen. Zeitgleich sind Benutzerschnittstellen nötig, die es den vorhandenen Fachkräften ermöglichen, den relevanten Programmcode der Steuerungen ohne zusätzliches Expertenwissen effizient anzupassen.

Zusammengefasst sind bei der Programmierung und Steuerung von Robotersystemen folgende drei Defizite identifizierbar:

1. Die geringe Standardisierung im Bereich von Steuerungen von Robotersystemen führt zu einer hohen Herstellerabhängigkeit, sowie geringer Austauschbarkeit und Erweiterbarkeit der Programme.
2. Die Programmierung ist weiterhin Personen mit Expertenwissen vorbehalten, da Maschinenbedienenden aufgrund fehlender nutzerzentrierter Interfaces (angepasst an deren Qualifikati-

Tabelle. Vergleich aktueller modularer Steuerungsframeworks und Programmiersysteme [10, 18–23].

Technologie	Grafische Programmierung	Funktionsbausteine	SBC	Vorgestelltes Framework
Programmiermethode	Teach-in, grafisch, Funktionsbausteine	Playback, Funktionsbausteine	Programmieren von Skills zu Jobs	Deklarativ, Funktionsbausteine
Anwendungsgebiet	Fügen, Fertigung	Handling, Prüfen, Fertigung	Fertigung	Fertigung, Handling
Bildverarbeitung	Teilweise	Teilweise	Teilweise	Ja
Roboterkompatibilität	Erweiterbar	Herstellerbeschränkung	Erweiterbar	Erweiterbar
Einlerndauer	1–5 Tage	1–5 Tage	~ 2 Tage	< 1 Tag
Aufwand für Modifikation	< 5 Minuten	< 5 Minuten bis 1 Tag	> 15 Minuten	< 5 Minuten

onstiefe) für die Programmänderungen das entsprechende Knowhow fehlt. Es sind dedizierte Schulungen für jede Steuerung in der Produktion notwendig, um das nötige Knowhow aufzubauen und auf verschiedene Applikationen zu übertragen

3. Trotz vereinzelter Ansätze für modulare Steuerungskonzepte und der Anwendung intelligenter Hardwarekomponenten fehlen einheitliche Steuerungslösungen, die Fachkräfte durch eine ganzheitliche, automatisierte Detektion und Verarbeitung aller relevanten Umgebungsänderungen bei der Programmierung der Robotersteuerungen unterstützen.

1.2 Stand der Technik bei der Programmierung und Steuerung von Robotersystemen

Robotersysteme dienen der Automatisierung industrieller Produktionsprozesse durch eine prozessspezifische Anordnung von Fertigungsmodulen mit einem zentralen Roboter als Manipulator [11]. Neben zahlreichen ortsfesten Lösungen, wie bspw. für Fräsbearbeitungen [12, 13], wurden in den letzten Jahren auch mobile Roboterzellen [8, 14, 15], teilweise als kollaborative Lösungen [13, 16], entwickelt. Aktuelle Bestrebungen bestehen darin, Robotersysteme für ihre Aufgabe weiter zu flexibilisieren (Plug-and-Produce-Paradigma) [14], unter anderem im Bereich der flexiblen Steuerung und intuitiven Programmierung von Robotersystemen [11], und damit die industrielle Verbreitung zu fördern.

Ein Ansatz für eine verbesserte Steuerung und Programmierung sind SBC, die durch Abstraktion der Hardwarefunktionen in Softwaremodulen über funktionsspezifische Skills eine schnellere Rekonfigurierbarkeit und aufgabenorientierte Programmierweise ermöglichen [10, 17]. Basierend auf einer objekt-orientierten Steuerungsarchitektur werden Template-Klassen (Skill-Basisklassen) bereitgestellt, die für das Erstellen neuer Skills als Guideline verwendet werden. Die Kommunikation zwischen den Modulen und deren Skills sind durch die Templates vorbestimmt, sodass sich der Programmierer in einer SBC auf die reine Funktionalität und Parametrierbarkeit der Hardwarefunktion fokussiert. In Sequenzen (oder auch Jobs) werden danach Skills zu aufgabenspezifischen Programmabläufen kombiniert und von einer Leitsteuerung orchestriert. Bei komplexen Arbeitsprozessen steigt trotz hoher Modularität jedoch der Programmieraufwand, da die Anzahl der Skills schnell ansteigt. Außerdem besteht weiterhin eine Abhängigkeit von Expertinnen und Experten, die sich in die Einsetzbarkeit und Parametrierung der Skills einarbeiten. Verbessert werden könnte dies durch eine einheitlichere, nutzerzentrierte Programmierung der Abläufe und der Erhöhung der Autonomiefähigkeit.

Die Programmierung von Robotersystemen erfolgt sehr steuerungsnah und benötigt daher umfangreiche Vorkenntnisse. Eine Vereinfachung durch die Verwendung von grafischen Blöcken und funktionsorientierter Programmierung wird daher von diversen Projekten in Forschung und Industrie vorangetrieben, um die notwendigen Qualifikationshürden zu reduzieren. Hierfür wird unter anderem der Einsatz von Funktionsbausteinen [18, 19], wie beispielsweise die Implementierung von Google Blockly [20, 21], oder grafische Programmierung erforscht [22, 23]. Um die Unabhängigkeit von speziellen Hardwarelösungen zu gewährleisten, werden diese Ansätze um Plugins zur Kommunikation erweitert, welche die programmierten Bausteine in die hardwarespezifischen Grundbefehle übersetzen [24, 25]. Plugins existieren für Industrieroboter und benötigte Peripherien. Eine Vereinheitlichung (zum Beispiel über OPC-UA) der Kommunikation stärkt hier die Herstellerunabhängigkeit der Programmierparadigmen und industrielle Verbreitung.

Die automatisierte Detektion und Verarbeitung von Umgebungsänderungen im Programmcode wird maßgeblich durch Forschungsbemühungen im Bereich der Bildverarbeitung getrieben. Visual Perception bedeutet in diesem Zuge die Aufnahme, Organisierung und Interpretation von Bildinformationen im Kontext einer bestimmten Prozessaufgabe [26]. Grundproblem an Robotersystemen ist unter anderem die Detektion eines zu handhabenden Bauteils und die Bestimmung der Position im Raum. In der Robotik kommen hierfür außerdem RGB-, Monochrom- oder Stereovision-Kameras für 2D- und 3D-Rohdaten zum Einsatz. Die benötigten Informationen werden danach algorithmisch ermittelt und als Parameter an die Steuerung weitergegeben [27, 28].

Vergleichend zu den Kerntechnologien im Stand der Technik, gibt die **Tabelle** einen Überblick über aktuelle Ansätze im Bereich der Programmierung und Steuerung von Robotersystemen.

2 Vorgehen

Das entwickelte Software-Framework verfügt über drei aufeinander aufbauende Softwarekomponenten, die die drei Defizite heutiger Steuerungen von Robotersystemen synergetisch lösen. Als Basis wird dabei das zu steuernden Robotersystems als SBC umgesetzt. Hierauf aufbauend besteht das Software-Framework aus:

- Abstrahierten Robotersystemfunktionen als Skills einer SBC und Entwicklung von OPC-UA-Konnektoren,

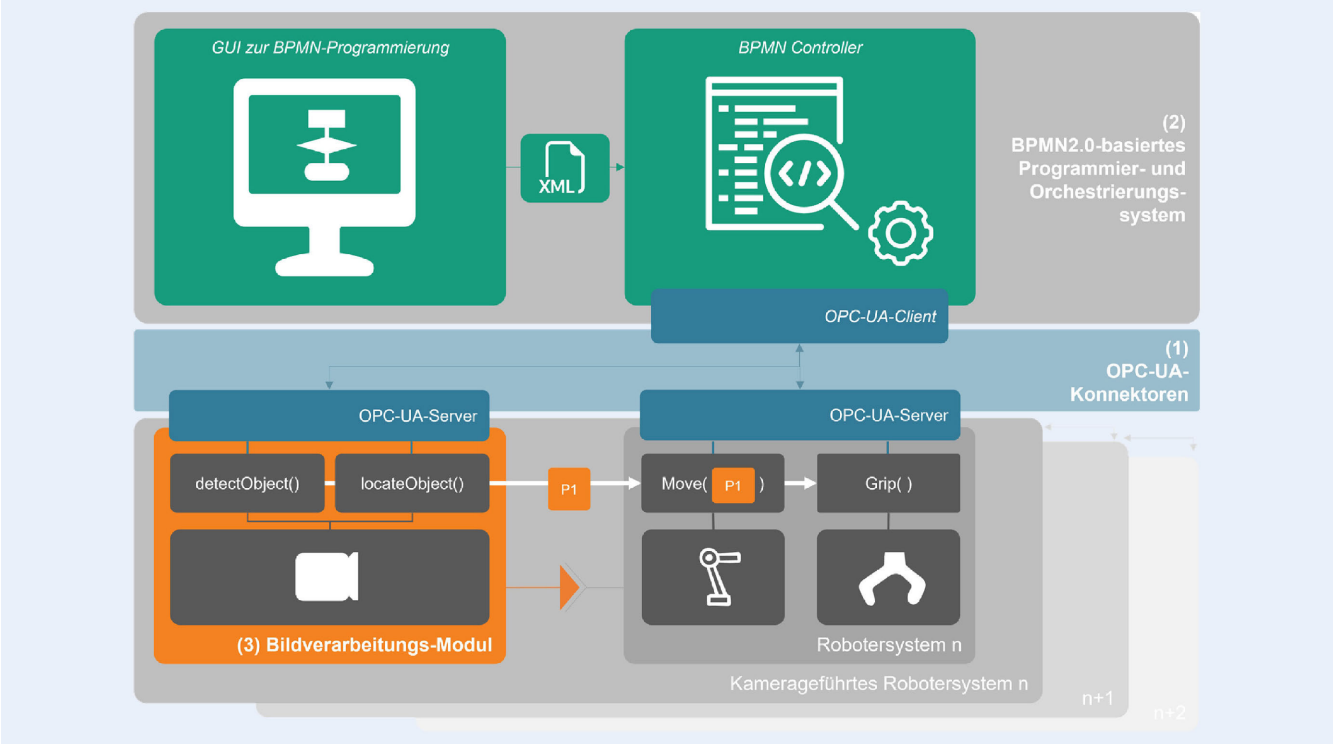

Bild 1. Darstellung des entwickelten Software-Frameworks. *Grafik: Fraunhofer IWU, Fraunhofer Austria*

- Business Process Model and Notation (BPMN) 2.0 Programmier- und Orchestrierungssystem zur Skillansteuerung (BPMN Controller),
- Bildverarbeitungs-Modul zur Parametrierung der aufgerufenen Skills.

Der schematische Aufbau des Frameworks ist in **Bild 1** dargestellt. Über den BPMN2.0-Standard wird ein Prozessablauf in einem Graphischen User Interface (GUI) als BPMN 2.0-Modell programmiert und als XML-File textbasiert gespeichert. Das XML enthält die Informationen der auszuführenden Skills in abstrakter Form. Ein BPMN-Controller lädt die Informationen und leitet einen Ablaufplan zur Orchestrierung ab. Über OPC-UA-Konnektoren werden durch den BPMN-Controller die jeweiligen Skills getriggert und überwacht. Das Bildverarbeitungs-Modul unterstützt den Programmierer als kamerageführtes Robotersystem, indem vorgeschaltete Bildverarbeitungs-Skills zur automatisierten Ermittlung von Skillparametern des Robotersystem dienen.

Das Framework adressiert damit die Herausforderungen heutiger Robotersysteme. Über ein zentrales, einheitliches und intuitiveres Programmier- und Steuerungssystem sind heterogene Roboter orchestrierbar. Mit der BPMN-Programmierung wird zudem eine an die Qualifikationstiefe von Maschinenbedienenden angepasste, deklarative und damit nutzerzentriete Programmierumgebung bereitgestellt.

3 Methodik

3.1 Abstrahieren der Robotersystemfunktionen in Skills und Entwicklung von OPC-UA-Skillkonnektoren

Zur flexibleren Nutzung abstrahiert das Framework die Hardwarefunktionen des Robotersystems in einer SBC zuerst als Skills in einer objektorientierten Programmierweise. Die SBC läuft dabei in einem Leitrechner, womit die Hardwaremodule, wie Roboter, Greifer oder Kameras, über einen Feldbus kommunizieren. Die Skills dienen als Datenschnittstellen zu übergeordneten Orchestrierungssystemen, um die Hardwarefunktionen eines Robotersystems einheitlich bereitzustellen und anzusprechen, siehe [10]. Die Skills werden über eine Zustandsmaschine nach dem Module Type Package (MTP) [29] verwaltet. Wie in **Bild 2** zu sehen, wurde zur Implementierung der Skills eine Skill-Basisklasse (BaseSkill) entwickelt, womit die notwendigen Datenstrukturen definiert werden.

- ST_SkillCommand: Liste an Schaltbefehlen basierend auf dem MTP [29], um Skills in einen spezifischen Zustand zu versetzen, z.B. in einen Ausführungszustand (execute),
- ST_SkillData: Liste an ST_Parameter, die eine endliche Anzahl an Skillparametern mit Namen, Wert, Einheit und Beschreibung definieren, um Skills zu parametrieren,
- ST_SkillDataDefault: Default-Parameter von ST_SkillData, die bei der Initialisierung des Skills ST_SkillData beschreiben,
- ST_SkillState: Aktueller Zustand des Skills nach dem MTP [29].

Das Software-Framework definiert Skill-Basisklassen für variable Steuerungsarten und -hersteller. Umgesetzt sind die Skill-Basisklassen als Funktionsbausteine in TwinCAT 3.1 (Beckhoff), Programmbausteine in TIA V16 (Siemens) und Automation Studio (B&R) sowie als Klassen für Python-basierte Skills. Die Skill-Basisklassen werden jeweils als Bibliotheken in die Steuerungsumgebungen eingebunden, neue Skills über eine Vererbung erzeugt und die vordefinierten Datenstrukturen bei der Implementierung skillspezifisch überschrieben. Das Ausführen eines Skills erfolgt durch Aktivieren des ST_SkillState über den passenden ST_SkillCommand. Über ST_SkillData wird der Skill aufgabenspezifisch vor Ausführung parametriert. Die Vorgehensweise

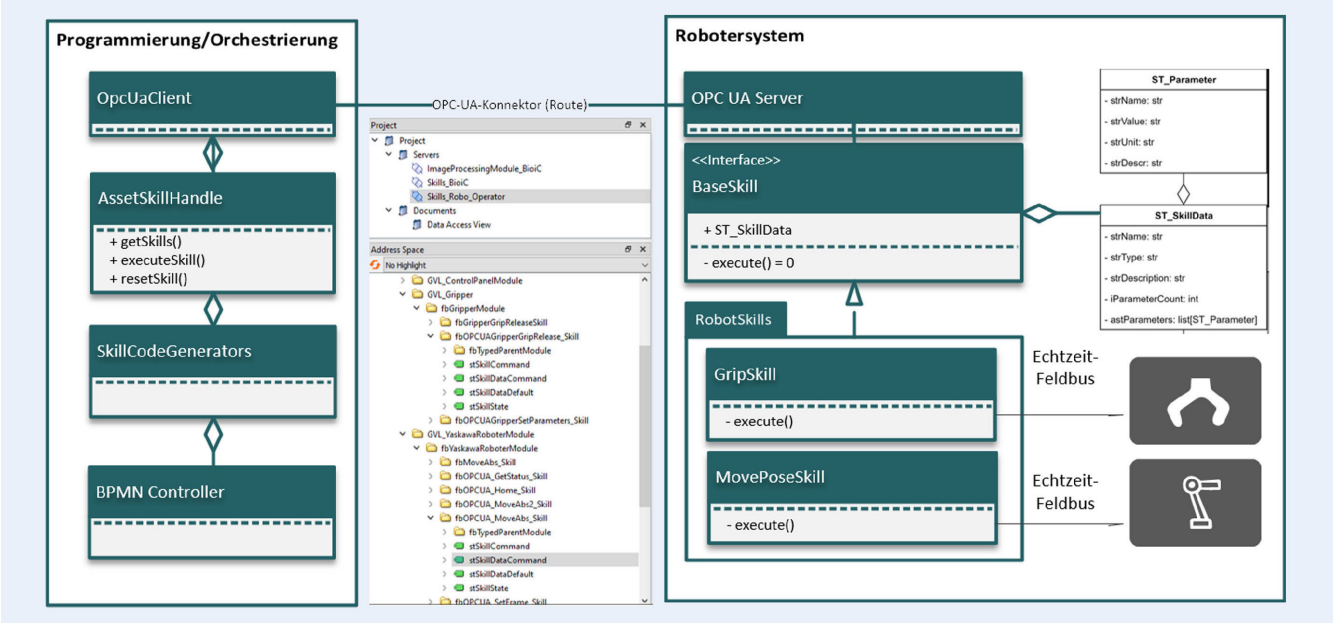

Bild 2. Implementierung von Skill-Basisklassen und Aufbau von OPC-UA-Skillkonnektoren. *Grafik: Fraunhofer IWU*

ermöglicht damit ein konsistentes Steuern und Überwachen aller Hardwarefunktionen des Robotersystems.

Um die Skills anzusprechen, werden Datenstrukturen der einheitlichen Skill-Basisklasse als Nodes über einen OPC-UA-Server gehostet, wie Bild 2 mittig zeigt. Alle Hardware-Module (oder auch Assets) stehen in einer konsistenten Nodestruktur bereit. Ein Verbindungsmanagement AssetSkillHandle baut nach Angabe der Verbindungsparameter (Route, Zertifikat, Nutzer und Passwort) eine OPC-UA-Route zum OPC-UA-Client des Robotersystems auf. Der AssetSkillHandle durchsucht über die Route den OPC-UA-Server nach der Datenstruktur der Skill-Basisklasse (getSkills()). Für jeden Nodebereich, der einer Skill-Basisklasse zugeordnet wird, wird ein OPC-UA-Skillkonnektor als Merker instanziiert. Bei der parallelen Kommunikation mit mehreren Assets werden mehrere AssetSkillHandle mit unterschiedlichen Verbindungsparametern verwendet. Der AssetSkillHandle ermöglicht damit das zentrale Schreiben und Lesen der Datenstrukturen aller gefundenen Skills der Skill-Basisklasse für ein Asset. Ein SkillCodeGenerator erzeugt zuletzt mithilfe des AssetSkillHandle jeweils eine Python-Methode pro Skill, welche bei Ausführung den Skill über den Konnektor startet. Die Methoden-Inputs sind hierbei die Skillparameter des Skills. Die Methoden dienen so als kompakte Schnittstelle für einen beliebigen, angebundenen Python-Controller zur Orchestrierung.

Die Abstraktion der Robotersystemfunktionen über Skill-Basisklassen und das Bereitstellen über einen OPC-UA-Server ermöglichen es, die vom BPMN-Controller orchestrierten Programme für mehrere Steuerungssysteme zu nutzen und somit eine Austauschbarkeit zu gewährleisten. Hierfür wird die Struktur der Skill-Basisklasse für alle Hardwarefunktionen der Robotersysteme in deren Steuerungen umgesetzt. Stehen auf einem Robotersystem die gleichen Skills bereit, so sind die Programme ohne Anpassung des Quellcode übertragbar und herstellerunabhängig.

3.2 Entwicklung eines BPMN 2.0 Programmier- und Orchestrierungssystems

Um eine Abstrahierung auf verschiedene Benutzerlevel zu ermöglichen, wird zunächst eine schnellere und verständlichere Programmieroberfläche benötigt, welche ergänzend eine Standardisierung sicherstellt. Aus diesem Grund werden die Abläufe eines Robotersystems im BPMN 2.0 Standard (BPMN 2.0) modelliert und in einem entsprechenden Controller orchestriert. Vorteile von BPMN 2.0 sind dabei neben der intuitiven und einfachen Darstellungsart, einheitliche Standards, erhöhte Flexibilität sowie die Möglichkeit der Abstrahierung eines Prozesses für verschiedene Benutzerlevel. Die unterschiedlichen Abstraktionslevel werden durch die Möglichkeit der Modellierung von beliebig tief verschachtelten Sub-Prozessen in BPMN 2.0 realisiert, die Skills zu kleineren Teilabläufen zusammenfassen, speichern und anschließend in übergeordneten Gesamtabläufen verwenden. Diese Möglichkeit der Verschachtelung von Prozessen durch die Verwendung von BPMN Sub-Modellen erhöht einerseits die Flexibilität und Geschwindigkeit in der Programmierung sowie ermöglicht andererseits die Programmierung des Robotersystems durch Personen mit geringen Fachkenntnissen in Bezug auf Skills und hardwareseitige Implementierung.

Die Struktur des Programmier- und Orchestrierungssystems ist in **Bild 3** ersichtlich. Zunächst werden die einzelnen Skills mittels Drag and Drop zu einem BPMN 2.0 Prozess zusammengefügt. Nach Abschluss des erzeugten BPMN 2.0 Diagramms wird dieses in einer XML-Datei abgespeichert. Wird ein Prozess gestartet, wird ein Graph aus der XML-Datei erzeugt, wobei jede Node des Graphs einen Skill darstellt. Diese Skills werden im Durchlauf mit den hardwarespezifischen Skills abgeglichen, wobei fehlende Skills ergänzt werden. Ergänzend werden entsprechende Parameter hardwarespezifisch abgeglichen und im Falle von fehlenden Parametern entsprechende Default Werte verwendet. Ist der Abgleich erfolgreich abgeschlossen und sind die erforderlichen Skills und Parameter vollständig geladen, iteriert der Controller durch den Graphen und aktiviert für jeden Skill die

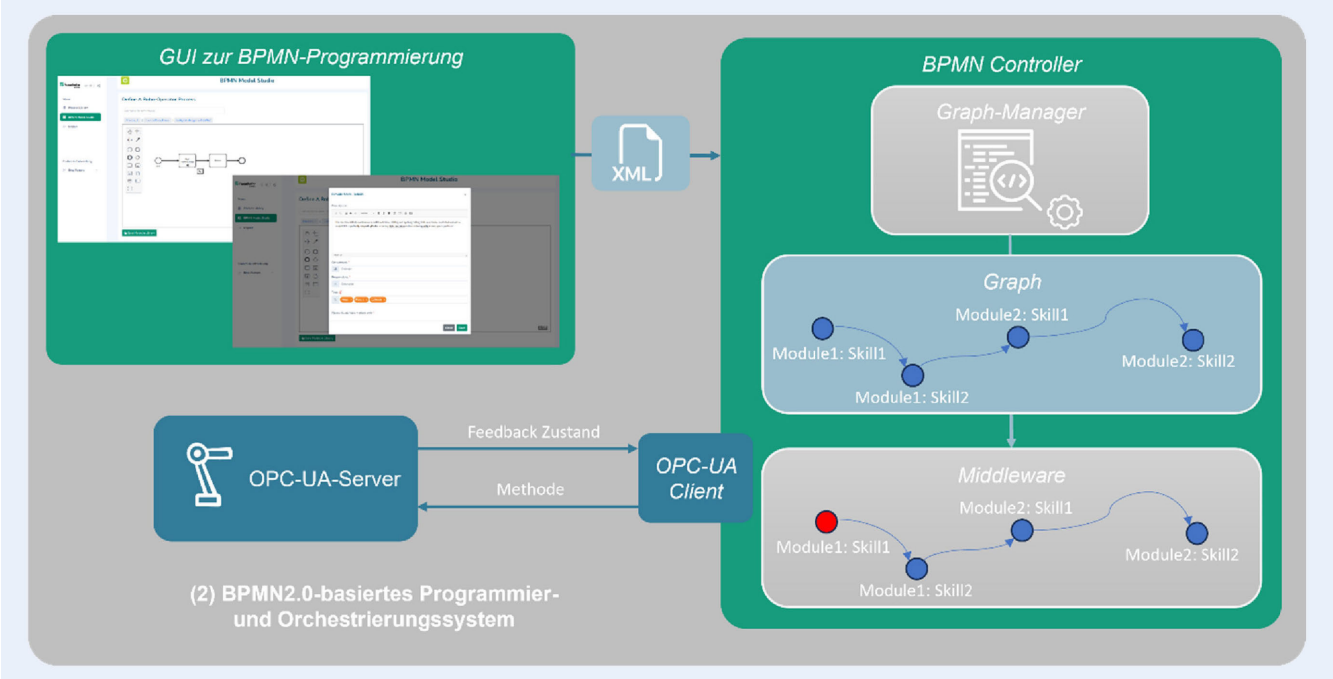

Bild 3. Schematische Darstellung des Ablaufs im BPMN Controller. *Grafik: Fraunhofer Austria*

entsprechende Methode zum Starten. Basierend auf dem Feedback über den Skillzustand wird nach erfolgreicher Ausführung eines Skills zur nächsten Node des Graphen übergegangen und der nächste Skill ausgeführt.

3.3 Parametrieren der Skills über ein Bildverarbeitungs-Modul

Wechselnde Umgebungsbedingungen im Ablauf führen zu einer notwendigen Umparametrierung der Skills an die neue Situation. Eine sich ändernde Speicherposition für ein zu greifendes Bauteil bedingt ein Nachteachen der Entnahmepose und damit ein manuelles Anpassen des Skillparameters im Skill. Die SBC ermöglicht durch klare Trennung von Aufgabe und Parametrierung, die Skillparameter vorher sensorisch zu ermitteln und dynamisch an die folgenden Skills zu übergeben. Das Software-Framework benutzt hierfür ein Bildverarbeitungs-Modul, welches Bildverarbeitungs-Skills zur automatisierten Berechnung der Parameter bereitstellt, siehe Bild 1 orange. Das Modul ist als Python-Runtime auf dem Steuerungs-PC des Robotersystem integriert. Eine Intel Realsense D453i (Stereo-Vision-Kamera mit RGB- und Tiefendaten) dient als Sensorinput. Alle Bildverarbeitungs-Skills implementieren die Skill-Basisklasse in Python, sodass die Skills analog zum Robotersystem über eine zweite OPC-UA-Route in einem zweiten AssetSkillHandle angesprochen werden. Hierzu startet das Bildverarbeitungs-Modul einen weiteren OPC-UA-Server.

Bild 4 zeigt die im Bildverarbeitungs-Modul vorhandenen Skills zur Objektlokalisierung sowie zur Objektinterpretation. Algorithmisch wird zur Lokalisierung von prozessveränderlichen Objekten (wie Bauteilen) ein Shape Matching eingesetzt, bei Equipment (wie Maschinen, Werkzeuge, Spannmittel) kommt ein Marker-basiertes Verfahren über ArUco-Marker zum Einsatz. Ein Feature-Detektor ermöglicht das semantische Interpretieren von Farb- und Textinformationen, um diese als Triggersignal für den Programmablauf zu nutzen. Workflow und Konfiguration der Algorithmik ist jeweils in Bild 4 gegenübergestellt.

Zur Programmierung einer flexiblen Roboterbewegung ermöglicht das Software-Framework nun die Erstellung eines generischen BPMN-Sub-Modells GripPart. In GripPart wird eine Bewegung zum Lokalisierungspunkt des Bauteil, die Bauteillokalisierung und das Bauteilgreifen über eine Kombination der Skills moveAbs(), locateObject(), moveAbs() und grip() kombiniert. Der Parametrierungsaufwand reduziert sich auf die Bestimmung der Greifweite im Skill grip(). Die anzufahrende Position wird hingegen automatisch ermittelt.

Mit der Nutzung des Bildverarbeitungs-Moduls in der SBC des Software-Frameworks ist es möglich, Robotersysteme mit geringem oder minimalem Programmieraufwand bei sich ändernden Objekten und Objektpositionen wieder in Betrieb zu nehmen. Die benötigten Informationen werden über Bildverarbeitungs-Skills an die folgenden Skills als Skillparameter übergeben. Die Abstraktion erfolgt in einem dedizierten BPMN-Sub-Modell.

4 Verifikation

Das Software-Framework wurde an einer mobilen Roboterzelle („Robo Operator“) zur Werkzeugmaschinenbeschickung am Fraunhofer IWU verifiziert [10, 31]. Die Leitsteuerung ist als SBC in „TwinCAT3.1“ von Beckhoff programmiert, um alle Hardwarefunktionen als Skills bereitzustellen. Der Leitsteuerung ist ein 6-Achs-Industrieroboter „Yaskawa GP12“ untergeordnet. Als repräsentativer Programmablauf dient die Maschinenbeschickung einer „HaasVF“ Werkzeugmaschine. Der Programmablauf besteht aus dem Bauteilhandling zwischen Bauteilspeicher und Spannmittel, der Maschinenzustandsinterpretation und der Maschineninteraktion (Öffnen, Schließen von Türen, Starten).

Bild 5 stellt hierfür exemplarisch die Ausführung des BPMN2.0-Submodells Bewege_Roboter.XML dar. Nacheinander ist die Orchestrierung einer Roboterfunktion, hier Roboter-

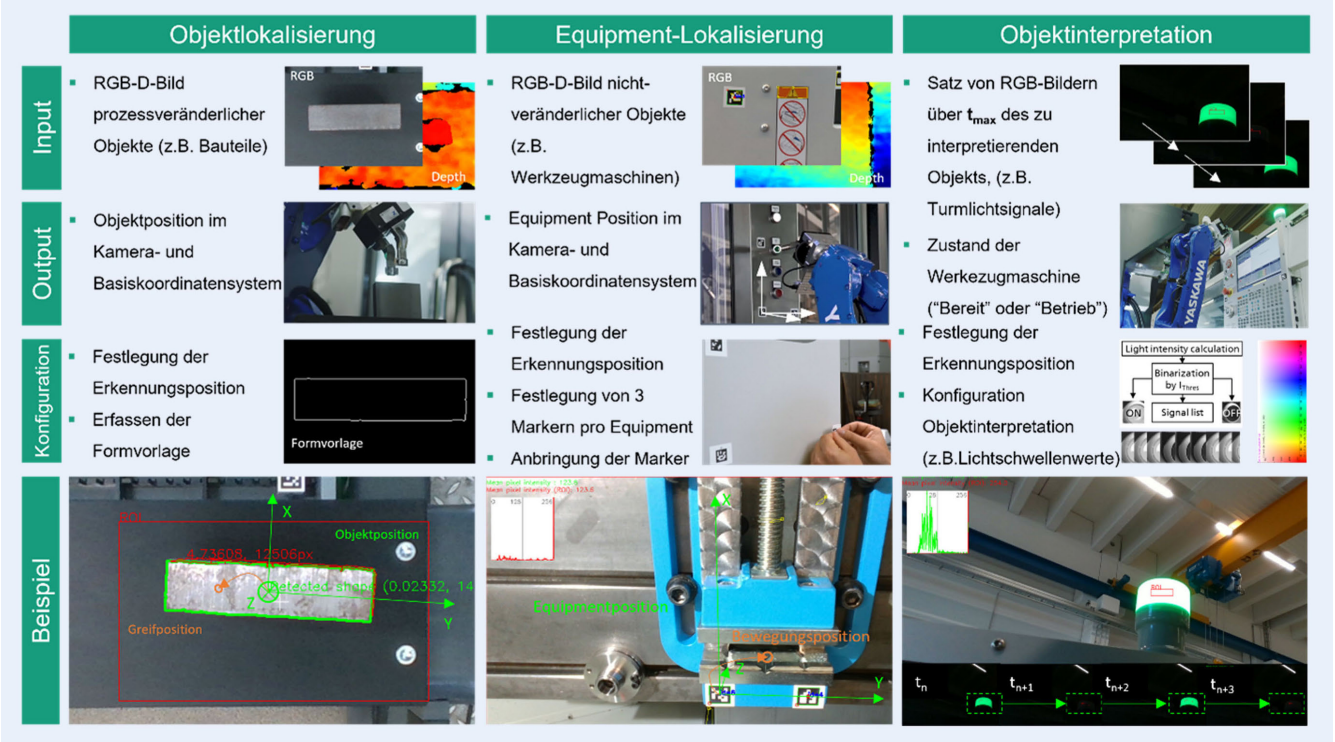

Bild 4. Parametrieren des Ablaufs über Skills eines Bildverarbeitungs-Moduls. *Grafik: Fraunhofer IWU [30]*

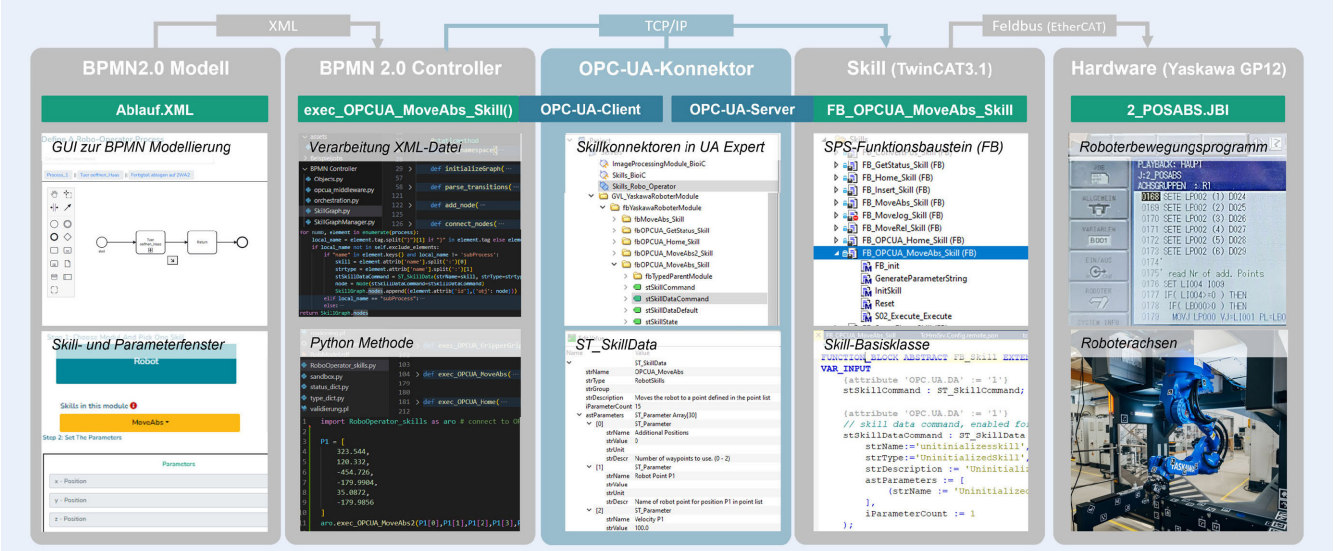

Bild 5. Ausführung einer Roboterfunktion über das Software-Framework. *Grafik: Fraunhofer IWU, Fraunhofer Austria*

bewegung (Absolutbewegung), vom BPMN-Modell zur ausgeführten Hardwarefunktion dargestellt.

Der vorgestellte Ablauf verhält sich für den weiteren Verlauf des Programms analog. Mit dem Software-Framework wird eine Methodik bereitgestellt, die damit folgende Vorteile im Vergleich zu den vorgestellten Defiziten und Ansätzen im Stand der Technik (vgl. Kapitel 1) aufweist:

1. Das Programmieren und Steuern von Robotern über Skill-Basisklassen und die angebundenen OPC-UA-Server macht den Programmcode einheitlich, herstellerunabhängig und somit austauschbar. Eine Erweiterung der Roboterfunktionen ist durch Hinzufügen neuer Skills aufwandsarm möglich. Die Skills werden im Controller dynamisch aktualisiert.
2. Die GUI zur Ablaufprogrammierung über BPMN-Modelle stellt eine intuitive Benutzerschnittstelle dar. Durch Abstraktion in Sub-BPMN-Modellen sind auf die Qualifikationstiefen von Benutzern und Benutzerinnen zugeschnittene Programmierweisen mit dem gleichen Framework möglich. Nach Vorbereitung der Robotersysteme (Umsetzung der SBC), werden herstellerspezifische Programmierschulungen auf ein Minimum reduziert. Die benötigten Kenntnisse zur Programmierung eines Ablaufs reduzieren sich auf Kenntnisse über einzelne

https://doi.org/10.37544/1436-4980-2024-04-20, am 05.11.2024, 07:23:07

Prozessschritte und dazugehörigen Skills und deren Parametrierung über das Bildverarbeitungs-Modul.
3. Die Integration eines Bildverarbeitungs-Moduls verringert die Zeit für die Wiederinbetriebnahme, da Steuerungsparameter für einheitliche Sub-Modelle automatisiert ermittelt werden.

5 Zusammenfassung und Ausblick

Zur flexibleren Programmierung von Robotersystemen wurde ein Software-Framework mit drei aufeinander aufbauenden Komponenten entwickelt, welches die Problematik der hohen Herstellerabhängigkeit aufgrund fehlender Standardisierung, die umständliche Programmierweise sowie das Fehlen von Methoden zur flexiblen Steuerung bei Umgebungsänderungen adressiert. Zu diesem Zweck wurden Robotersystemfunktionen als Skills in einer SBC abstrahiert sowie Skillkonnektoren zur Orchestrierung mittels OPC-UA entwickelt. Darauf aufbauend wurde eine BPMN-basierte Programmierumgebung mit BPMN-Controller zur Ansteuerung, sowie ein Bildverarbeitungsmodul zur Parametrierung der Skills entwickelt. Das Software-Framework wurde abschließend an einer mobilen Roboterzelle anhand einer Werkzeugmaschinenbeschickung verifiziert.

Nach der erfolgreichen Verifizierung steht für das erarbeitete Software-Framework zunächst eine Validierung der Zielkriterien aus. Für die Bewertung werden die vier Kriterien Programmierzeit, Wiederanlaufzeit sowie die Anlernzeit und Usability (über PSSUQ-Fragebogen) des Frameworks gemessen. Hiernach werden die Übertragbarkeit auf weitere Robotersystemen (Autonox-Roboter mit TwinCAT3.1-Leitsteuerung, UR5) und prozessspezifische Hardwarekonfigurationen (Beschickung, Palettierung) untersucht. Weiterführende Forschungspunkte konzentrieren sich u.a. auf ein einheitliches Ablaufmonitoring des automatisierten Prozesses im BPMN2.0-Modell (Anzeige von Skill-States und Zustandsvariablen), System-Feedback zur Vermeidung unzulässiger Prozessabläufe sowie die Entwicklung einer parallelen Prozessdatenbank für die statische und dynamische Parametrierung von Skills, um zum Beispiel ermittelte Parameter des Bildverarbeitungs-Moduls einheitlich an folgende Skills zu übergeben (beispielsweise über MongoDB). Weiterhin werden Methoden des Fehlerhandling zur automatischen Ablaufanpassung bei fehlerhafter Skillausführung eingebunden. Um den sicheren Remotezugriff auf die Robotersysteme über OPC-UA-Konnektoren zu ermöglichen, ist die Erweiterung des Frameworks um ein Zertifikatsmanagement geplant.

Mit erfolgreicher Validierung des Frameworks bieten sich aussichtsreiche Entwicklungsmöglichkeiten und Potentiale durch eine Generalisierung des vorgestellten Ansatzes. Durch Sub-BPMN-Modelle sowie Nutzung der Bildverarbeitung sind Standardaufgaben wie ein Bauteilgriff abstrahierbar und für einen Ablauf mit minimalem Anpassungsaufwand wiederverwendbar. Darüber hinaus ermöglicht die Bereitstellung von einheitlichen Skills eine Selbstbeschreibung von Hardwarekomponenten, was die Realisierung von Plug-and-Produce-Systemen unterstützt. Weiteres Potential liegt in der verbesserten Simulationsfähigkeit des Frameworks. Die Konnektoren in Python vereinfachen es, Testumgebungen zur Simulation von BPMN-Modellen und Skills, zum Beispiel für Plausibilitätsprüfungen, virtuelle Inbetriebnahmen oder Sicherheits- und Risikobewertungen, im Sinne einer durchgängigen Softwareintegration (Continuous integration and deployment) anzubinden.

FÖRDERHINWEIS

Die in diesem Beitrag vorgestellten Ergebnisse wurden im Rahmen des von der Fraunhofer Gesellschaft (FhG) geförderten Forschungsprojekts „ProViPer – Semi-Automatische Programmierung von Roboterzellen durch visuelle Umgebungswahrnehmung“ erzielt.

Literatur

[1] McKinsey & Company: Manufacturing the future: The next era of global growth and innovation. Internet: https://www.mckinsey.com/capabilities/operations/our-insights/the-future-of-manufacturing. Zugriff am 27.09.2022
[2] Wang, Y.; Ma, H.-S.; Yang, J.-H. et al.: Industry 4.0: a way from mass customization to mass personalization production. Advances in Manufacturing 5 (2017) 4, S. 311–320
[3] Vogel-Heuser, B.; Bauernhansl, T.; Hompel, M. ten: Handbuch Industrie 4.0 Bd.4. Berlin, Heidelberg: Springer Berlin Heidelberg 2017
[4] Lüdtke, A.: Wege aus der Ironie in Richtung ernsthafter Automatisierung. In: Botthof, A.; Hartmann, E. A. (Hrsg.): Zukunft der Arbeit in Industrie 4.0. Berlin, Heidelberg: Springer Berlin Heidelberg 2015, S. 125–146
[5] Fuchs, J.: Demografie und Fachkräftemangel. Die künftigen arbeitsmarktpolitischen Herausforderungen. Bundesgesundheitsblatt, Gesundheitsforschung, Gesundheitsschutz 56 (2013) 3, S. 399–405
[6] McKinsey & Company: Die Besten, bitte: Wie der öffentliche Sektor als Arbeitsgeber punkten kann. 2019
[7] Abicht, J.: Projekt “Merhabe” vorgestellt: Die applizierbare Roboterzelle. Stand: 01.06.2021. Internet: https://www.kognitive-produktion.de/projekt-merhabe-vorgestellt-bedienerfaehigkeiten-durch-mobile-flexible-roboterzellen-bereitstellen/. Zugriff am 27.09.2022
[8] Industrie-Partner GmbH: Robo Operator®. Internet: https://www.ip-equipmentrental.de/de/mietequipment/robo-operator.php. Zugriff am 27.09.2022
[9] Directorate-General for Research and Innovation: Industry 5.0 – Towards a sustainable, human-centric and resilient European industry. European Commission 2021
[10] Wiese, T.; Abicht, J.; Friedrich, C. et al.: Flexible skill-based control for robot cells in manufacturing. Frontiers in Robotics and AI 9 (2022)
[11] Sanneman, L.; Fourie, C.; Shah, J. A.: The State of Industrial Robotics: Emerging Technologies, Challenges, and Key Research Directions, 2020
[12] Verl, A.; Valente, A.; Melkote, S. et al.: Robots in machining. CIRP Annals 68 (2019) 2, S. 799–822
[13] Villani, V.; Pini, F.; Leali, F. et al.: Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. Mechatronics 55 (2018), S. 248–266
[14] Wojtynek, M.; Steil, J. J.; Wrede, S.: Plug, Plan and Produce as Enabler for Easy Workcell Setup and Collaborative Robot Programming in Smart Factories. KI – Künstliche Intelligenz 33 (2019) 2, S. 151–161
[15] Lienenluke, L.; Grundel, L.; Storms, S. et al.: Temporal and Flexible Automation of Machine Tools. 2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES), Las Palmas de Gran Canaria, 2018 – 2018, S. 335–340
[16] Bogue, R.: Europe continues to lead the way in the collaborative robot business. Industrial Robot: An International Journal 43 (2016) 1, S. 6–11
[17] Zimmermann, P.; Axmann, E.; Brandenbourger, B. et al.: Skill-based Engineering and Control on Field-Device-Level with OPC UA. 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019 – 2019, S. 1101–1108
[18] Universal Robots: UNIVERSAL ROBOT UR3e – A Flexible Collaborative Robot Arm. Zugriff am 27.09.2022
[19] Omron: Omron TMFlow. Stand: 26.09.2022. Internet: https://automation.omron.com/en/ca/products/family/Omron%20TM%20Software. Zugriff am 26.09.2022
[20] Weintrop, D.; Shepherd, D. C.; Francis, P. et al.: Blockly goes to work: Block-based programming for industrial robots. 2017 IEEE Blocks and Beyond Workshop (B&B), Raleigh, NC, 2017 – 2017, S. 29–36
[21] Winterer, M.; Salomon, C.; Koberle, J. et al.: An Expert Review on the Applicability of Blockly for Industrial Robot Programming. 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020 – 2020, S. 1231–1234

[22] Thomas, U.; Hirzinger, G.; Rumpe, B. et al.: A new skill based robot programming language using UML/P Statecharts. 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 2013 – 2013, S. 461–466
[23] drag&bot: drag&bot – Graphical robot operating system. Internet: https://www.dragandbot.com/. Zugriff am 26.09.2022
[24] Haspl, T.; M. Rathmair; M. Papa et al. (Hrsg.): Software toolchain for modeling and transforming robotic workflows into formally verifiable model representations. in press. Austrian Robotics Workshop. 2022 2022
[25] Komenda, T.; Garcia, J. B.; Schelle, M. et al. (Hrsg.): Sustainable utilization of industrial robotic systems by facilitat- ing programming through a human and process Sustainable utilization of industrial robotic systems by facilitating programming through a human and process centred declarative approach. International Conference on Competitive Manufacturing. 2022 2022
[26] Yang, J.; Wang, C.; Jiang, B. et al.: Visual Perception Enabled Industry Intelligence: State of the Art, Challenges and Prospects. IEEE Transactions on Industrial Informatics 17 (2021) 3, S. 2204–2219
[27] Du, G.; Wang, K.; Lian, S. et al.: Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. Artificial Intelligence Review 54 (2021) 3, S. 1677–1734
[28] Inceoglu, A.; Koc, C.; Kanat, B. O. et al.: Continuous Visual World Modeling for Autonomous Robot Manipulation. IEEE Transactions on Systems, Man, and Cybernetics: Systems 49 (2019) 1, S. 192–205
[29] Deutsches Institut fur Normung e.V.: 2658. VDI/VDE/NAMUR 2658 BLATT4. Beuth 08.2020
[30] Abicht, J.; Hellmich, A.; Wiese, T. et al.: New automation solution for brownfield production – Cognitive robots for the emulation of operator capabilities. CIRP Journal of Manufacturing Science and Technology 50 (2024), S. 104–112
[31] Abicht, J.; Wiese, T.; Hellmich, A. et al.: Interface-Free Connection of Mobile Robot Cells to Machine Tools Using a Camera System. In: Weißgraeber, P.; Heieck, F.; Ackermann, C. (Hrsg.): Advances in Automotive Production Technology – Theory and Application. Berlin, Heidelberg: Springer Berlin Heidelberg 2021, S. 468–477

Dipl.-Ing. Sebastian Kreuter
Foto: Autor
Tel. +43 1 5046906
sebastian.kreuter@fraunhofer.at

Dipl.-Wirtsch.-Ing. Dr. techn. Philipp Hold
Fraunhofer Austria Research GmbH
Theresianumgasse 7, 1040 Wien
www.fraunhofer.at

Univ. Prof Dr.-Ing. Sebastian Schlund
Tel. +43 1 58801 330 01
sebastian.schlund@tuwien.ac.at
Technische Universität Wien
Institut für Managementwissenschaften
Theresianumgasse 27, 1040 Wien
www.tuwien.ac.at

Dipl.-Ing. Johannes Abicht
Foto: Autor
Tel. +49 351 / 4772-2613
johannes.abicht@iwu.fraunhofer.de

Dipl.-Ing. Malte Heinrich

Dipl.-Ing. Torben Wiese
Fraunhofer-Institut für Werkzeugmaschinen und Umformtechnik IWU
Bürogebäude Dresden-Gittersee
Pforzheimer Str. 7a, 01189 Dresden
www.iwu.fraunhofer.de