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

doi.org/10.37544/1436–4980–2024–06–20

Ein neuartiger Ansatz zum Engineering von digitalen Zwillingen in Industrie-4.0-Fabriken

Model-based DevOps für digitale Zwillinge

J. Zhang, A. Wortmann, O. Riedel

ZUSAMMENFASSUNG Der digitale Zwilling ist ein wesentlicher Bestandteil der Digitalisierung in der Industrie 4.0. Das Zwillingssystem besteht aus physischer und virtueller Repräsentation in Zusammensetzung mit einem komplexen Softwaresystem, wofür ein aufwendiger Engineering-Prozess nötig ist. Dieser Prozess soll durch einen modellbasierten DevOps-Ansatz mittels Modellabstraktion unterstützt werden. Dieser Beitrag gibt einen Einblick in die Forschung und die Herausforderungen der modellbasierten DevOps.

STICHWÖRTER

Advanced Systems Engineering, Industrie 4.0, Nachhaltigkeit

Model-based DevOps: A novel approach for engineering digital twins in industry 4.0 factories

ABSTRACT The digital twin is an essential component of digitization in Industry 4.0. The twin system consists of physical and virtual representation, in combination with a complex software system, for which a complex engineering process is necessary. This process is to be supported by a model-based DevOps approach using model abstraction. This article provides an insight into the research and challenges of model-based DevOps.

1 Einleitung

Fabriken der Industrie 4.0 nutzen Fortschritte in komplexen Systemen sowie in der Vernetzung und Digitalisierung, um flexible und optimierte Produktionsprozesse bereitzustellen [1]. Sie erlauben vor allem die Produktion kundenorientierter Lösungen aus Daten, die während des Lebenszyklus eines Produkts gesammelt werden. Das „Model-Based DevOps“ für digitale Zwillinge soll die Transformation traditioneller Fabriken hin zu nachhaltigen Industrie-4.0-Fabriken [2] unterstützen. Es soll durch agile Methodiken die Optimierung von komplexen Systemen in Industrie-4.0-Fabriken ermöglichen. Die ausschlaggebende Neuheit bei diesem Ansatz besteht in der Kombination aus agilen Model-Based-DevOps-Methodiken mit Methodiken zum Engineering von digitalen Zwillingen. Durch diese Kombination soll Model-Based DevOps nicht nur zu optimierten Produktionsprozessen führen, sondern auch wertvolle Erkenntnisse über die effektive Anwendung von Optimierungstechniken für komplexe Softwaresysteme und eine nachhaltige Industrie 4.0 liefern [3].

1.1 Digitale Zwillinge

Eine populäre, qualitative Definition digitaler Zwillinge nach *Kritzinger* [4] (**Bild 1**) beruht auf der Unterscheidung zwischen digitalem Modell, digitalem Schatten und digitalem Zwilling.

Das Augenmerk liegt auf den automatisierten Datenflüssen zwischen physischem und digitalem Objekt. Sind die Datenflüsse in beide Richtungen manuell, so handelt es sich bei dem digitalen Objekt um ein klassisches Modell (wie etwa ein CAD-Modell). Ist der Datenfluss vom physischen Objekt zum digitalen Objekt automatisch, die andere Richtung jedoch manuell, handelt es sich bei dem Modell um einen digitalen Schatten [5] (beispielsweise im Rahmen eines Dashboards). Sind beide Richtungen automatisch, handelt es sich um einen digitalen Zwilling. Um dies umzusetzen, ist der digitale Zwilling als komplexes Softwaresystem ausgelegt (vergleiche ISO 23247 [6]), das Daten vom physischen Objekt aufnimmt, diese auswertet und damit das Verhalten des physischen Objekts steuert. Typische Aufgaben dieses komplexen Softwaresystems sind die Synchronisation und Auswertung von Daten zwischen dem physischen Objekt und dem digitalen Objekt [7].

In einer Industrie-4.0-Fabrik entsteht ein cyber-physisches System, das Hardware und Software umfasst sowie ein digitales System, welches das digitale Objekt in Form von Modellen und Software zur bidirektionalen Übertragung von Daten umfasst.

Beim Engineering eines solchen Systems sind eine Vielzahl an Personen aus unterschiedlichen Hintergründen beteiligt, von Maschinenbetreibern über Produktionsplanern bis hin zu Softwareentwicklern.

1.2 Model Driven Engineering

Model Driven Engineering (MDE) ist eine Methodik der Softwareentwicklung [8, 9], in dem Modelle die primäre Entwicklungsartefakte sind. Über die in Modellen inhärente Abstraktion ermöglicht MDE sowohl die effizientere Automatisierung von Entwicklungstätigkeiten als auch die konzeptuelle Kluft [10] zwischen den Domänenexperten des Problemraums und den Software-Experten des Lösungsraums zu reduzieren. Modelle können dabei abstrakter (und gleichzeitig präziser) als der klassi-

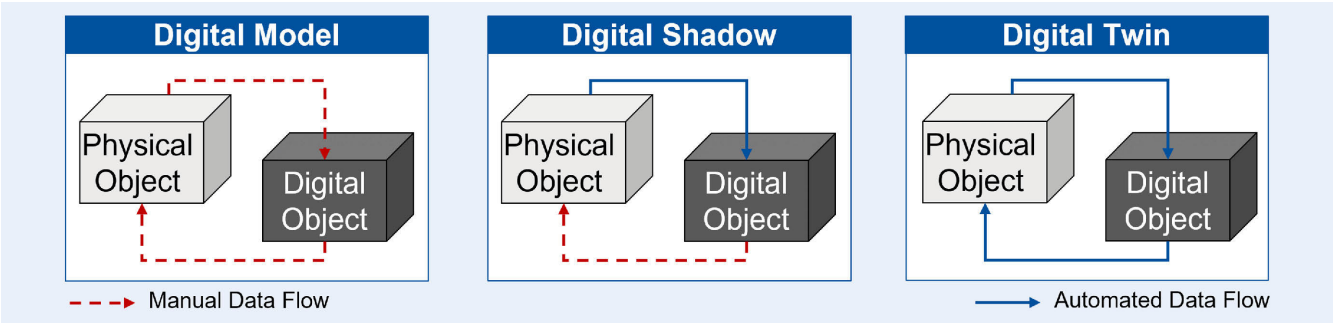

Bild 1. Klassifikation von digitalem Modell, digitalem Schatten und digitalem Zwilling. *Grafik: in Anlehnung an [4]*

sche Quellcode sein, da diese von technologischen Komplexitäten abstrahieren und gleichzeitig präzise Terminologien der Anwendungsdomänen verwenden können. Aus entsprechenden Modellen (wie AADL, EAST-ADL, UML, oder verschiedene Low-Code-Techniken) lassen sich mithilfe geeigneter Generatoren verschiedene Artefakte wie Dokumentation, Code und Tests automatisch generieren.

Ein weiterer Vorteil besteht in der Wiederverwendbarkeit und Portierbarkeit von Modellen [11]. Durch die Verwendung standardisierter Modellierungssprachen und -werkzeuge fördert MDE die Wiederverwendbarkeit von Softwarekomponenten und die gemeinsame Nutzung von Best Practices innerhalb und zwischen Organisationen. Modelle werden heute in einer Vielzahl von Domänen, inklusive Automotive (EAST-ADL) [12], Avionik (AADL) [13] und Produktionstechnik (OPC-UA-Informationsmodelle) [14], erfolgreich eingesetzt.

Model-Based DevOps soll durch die Verwendung von MDE-Techniken solche Vorteile erhalten und bei der Entwicklung und Rekonfiguration von digitalen Zwillingssystemen helfen.

1.3 DevOps

DevOps ist ein Softwareentwicklungsansatz [15], welcher die traditionelle Kluft zwischen Entwicklungs- (Dev) und Betriebsteams (Ops) in einer Organisation überbrückt. Das Hauptziel von DevOps ist die Verkürzung der Entwicklungs- und Bereitstellungszyklen zwischen Dev und Ops auf der Grundlage einer kollaborativen Beziehung und einer agilen Zusammenarbeit zwischen beiden Fraktionen. DevOps führt die Automatisierung in allen Phasen des Lebenszyklus der Softwareentwicklung ein, von der Integration über das Testen und die Freigabe bis zur Bereitstellung und dem Infrastrukturmanagement. Somit werden Softwareentwicklung und -evolution effizienter und zuverlässiger.

Im Beitrag werden zuerst die wichtigsten Herausforderungen im Model-Based DevOps für digitale Zwillinge analysiert und danach Ansätze zur Bewältigung der Herausforderungen dargelegt. Abschließend wird anhand eines Demonstrators die Machbarkeit aufgezeigt und es werden offene Herausforderungen für die Untersuchung von Model-Based DevOps dargestellt.

2 Stand der Technik

Das folgende Kapitel gibt einen Überblick zu Ansätzen der modellbasierten Softwareentwicklung und DevOps für komplexe Systeme in der Industrie 4.0.

2.1 Model-Driven Engineering für Industrie 4.0

In einer konzeptionellen Architektur eines digitalen Zwillings für Industrie-4.0-Fabriken wird der digitale Zwilling als containerisierter Dienst mit mehreren Schnittstellen dargestellt [16]. Jede Schnittstelle bietet spezifische Funktionalitäten für physische Maschinen oder andere digitale Dienste. Der digitale Zwilling stützt sich auf ein Modell, welches seinen Zustand, sein Design und sein Verhalten umfasst. Die präsentierte Architektur nutzt verschiedene Softwareentwurfsmuster, um die Beziehungen zwischen den Komponenten des Zwillingsdienstes, die Beziehungen zwischen dem Dienst und seiner externen Umgebung sowie die Anpassungsfähigkeit des Zwillings an verschiedene Kontexte zu beschreiben. Dieser Ansatz schlägt ein Konzept für die Implementierung des digitalen Zwillings vor. Jedoch berücksichtigt er im Gegensatz zu Model-Based DevOps weder die Auswirkungen von Laufzeitdaten auf das Entwurfsmodell des Zwillings noch die Ableitung von Betriebsmodellen aus Laufzeitdaten.

2.2 Model-Driven Engineering für digitale Zwillinge

Modelle können in allen Phasen des Entwicklungsprozesses für komplexe Systeme eingesetzt werden [17]. Entwicklungsmodelle in der Entwicklungszeit beschreiben etwa Anforderungen, Architekturkomponenten und funktionale Belange. Die Codegenerierung aus solchen Modellen kann die Entwicklungseffizienz erhöhen und aufgrund des erleichterten Refactorings auch die Agilität steigern [18]. Betriebsmodelle unterstützen die Betriebszeit, indem sie beispielsweise Komponentenkonfigurationen und Deployments spezifizieren. Betriebsmodelle der Betriebszeit können in Anlehnung an „Models@run.time“ die Rekonfiguration laufender Systeme durch Modellanpassungen unterstützen [19]. MDE zur Erstellung digitaler Zwillingsarchitekturen anhand von Architekturbeschreibungssprachen verwendet [20]. Ebenso existieren Vorarbeiten, welche einen Rahmen für die Entwicklung und Wartung von digitalen Zwillingsinfrastrukturen auf der Grundlage von AutomationML-Modellen vorschlagen [21].

Keiner der Ansätze zielt bisher darauf ab, den gesamten Lebenszyklus des Systems und die Beziehungen zwischen Dev- und Ops-Phasen mit Modellen abzudecken.

2.3 DevOps von Softwaresystemen und Übertragung auf komplexe Industrie-4.0-Umgebung

Obwohl DevOps-Praktiken bei herkömmlicher Softwareentwicklung von Vorteil sind, stellen sie bei der Entwicklung von cyber-physischen Systemen [22] eine besondere Herausforderung

dar. Cyber-physische Systeme integrieren Berechnungen, Netzwerke und physikalische Prozesse. Die komplexen und verflochtenen Systeme sowie ihre Echtzeit- und Sicherheitsanforderungen sind für DevOps-Anwendungen eine große Herausforderung.

3 Forschungsschwerpunkte im Model-Based DevOps

Wie angedeutet, wird in dieser Forschung untersucht, wie eine Kombination aus MDE-Methoden zur Entwicklung von digitalen Zwillingen (Kapitel 2.1 und 2.2) mit dem Model-Based DevOps (Kapitel 2.3) zusammengeführt werden kann, um Engineering-Prozesse der Produktion zu verbessern. Der folgende Abschnitt gibt einen Überblick über Model-Based DevOps und erklärt danach die Herausforderungen zur Model-Based-DevOps-Methodik für Industrie 4.0.

Model-Based DevOps in **Bild 2** fokussiert sich vor allem auf die bidirektionale Beziehung zwischen Entwicklungs- (links) und Betriebsmodellen (rechts) sowie auf die systematische Ableitung von Modellen zwischen der Entwicklungs- und der Betriebszeit des Systems in der Produktion [23].

Dabei werden die Phasen Engineering des cyber-physischen Systems (gelb), Entwicklung des Softwaresystems des digitalen Zwillings (grau), Betrieb des digitalen Zwillings (blau) und Betrieb des cyber-physischen Systems (grün) unterschieden. Dazu werden heterogene Modelle in unterschiedlichen Detailgraden sowie für unterschiedliche Entwicklungsphasen verwendet. Beispielsweise werden CAD-Modelle als Engineering-Modelle für die Engineering-Phase des cyber-physischen Systems genutzt, während Komponentendiagramme für die Entwicklung des Softwaresystems verwendet werden. Daten aus dem Betrieb können mithilfe von Modellen systematisch aus dem cyber-physischen System gewonnen werden.

Das gesammelte Wissen der digitalen Zwillinge aus der Laufzeit soll so automatisiert in die Engineering-Modelle zurückgeführt werden, welche zur Konzeptualisierung und Optimierung des realen Produktionsprozesses dienen sollen. Im Folgenden werden die zentralen Herausforderungen genauer erörtert.

3.1 Definition eines Referenzmodells für digitale Zwillinge und Datenmodelle

Um digitale Zwillinge mit Datenmodellen in Verbindung zu stellen, ist ein Referenzmodell erforderlich.

Dieses Referenzmodell soll das gemeinsame Verständnis der Kernelemente von digitalen Zwillingen und Datenmodellen konzeptualisieren und insbesondere die Beziehung zwischen ihnen verdeutlichen, um MDE-Methoden anzuwenden (Kapitel 2.1 und 2.2). Das Referenzmodell soll die Möglichkeit eröffnen, bestehende Erkenntnisse und Methoden für digitale Zwillinge und Datenmodelle in kombinierter Weise zu nutzen, um digitale Zwillinge als Vermittler zwischen verschiedenen Modellen von Laufzeitdaten des cyber-physischen Systems einsetzen zu können.

3.2 Nachhaltigkeit von Produktionsprozessen mit digitalen Zwillingen

Des Weiteren sollen die digitalen Zwillinge die Vorarbeiten (Kapitel 2.3) in der Produktion zur Optimierung von Produktionsprozessen entlang der drei Säulen der Nachhaltigkeit erweitern [24]:

- Wirtschaft: Modellbasiertes DevOps soll zur Simulation von Rekonfigurationen des Produktionsprozesses zu Optimierungszwecken eingesetzt werden. Konkret ist geplant, mehrere mögliche Produktionslayouts zu untersuchen, um ein optimales Layout zu finden, das die Effizienz maximiert.
- Umwelt: Umweltbezogene modellbasierte DevOps-Simulationen liefern auf dieser Ebene tiefere Einblicke in den Ressourcenverbrauch und die Emissionen von Fabriken. Diese können mit der Produktqualität und der Prozesseffizienz in Bezug auf die wirtschaftliche Ebene abgeglichen werden.
- Sozial: Model-Based DevOps erhöht die Transparenz für die Stakeholder in den Fabriken durch Abstraktion der Modelle. Durch die erhöhte Transparenz kann Vertrauen in die Produktionstechnik geschaffen werden.

Dabei werden Produktionsabläufe anhand von Sensordaten ausgewertet und mithilfe von Modellen analysiert, die zur Vorhersage genutzt werden [25].

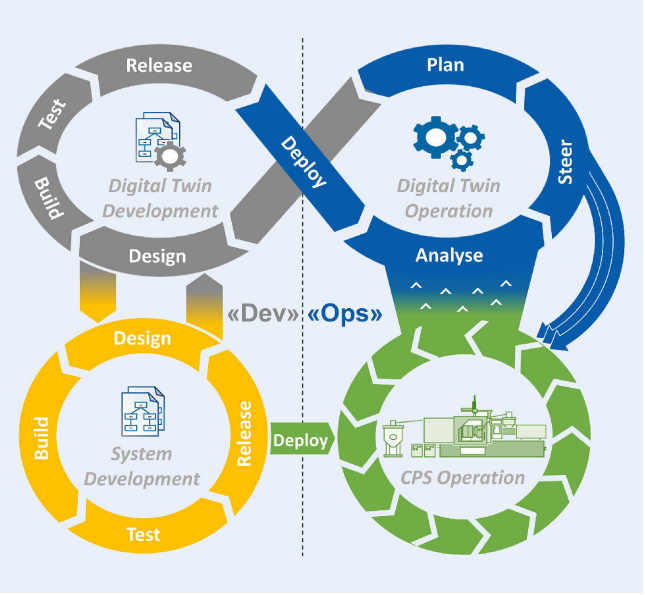

Bild 2. Erweitere DevOps-Schleife mit Unterteilung in Phasen zur Entwicklung von komplexen Systemen. *Grafik: [19]*

3.3 Komposition von digitalen Zwillingen

Das Konzept verbindet modellbasiertes DevOps (Kapitel 2.3) mit modellbasierter Entwicklung von digitalen Zwillingen (Kapitel 2.2), welche die Verbindungen zwischen Entwicklungs- und Betriebsmodellen widerspiegeln. Da komplexe cyber-physische Systeme in der Regel viele heterogene Modelle umfassen, gibt es eine Vielzahl durch Model-Based DevOps induzierte digitale Zwillinge. Dies wirft die Frage auf, wie sie zusammengesetzt werden können und wie sich die Möglichkeiten von Model-Based DevOps zur kontinuierlichen Verbesserung auf die Komposition von modularen digitalen Zwillingen auswirken.

Ebenfalls eine Frage ist, wie Model-Based DevOps auf logisch oder physisch zusammengesetzte digitale Zwillinge angewendet werden kann.

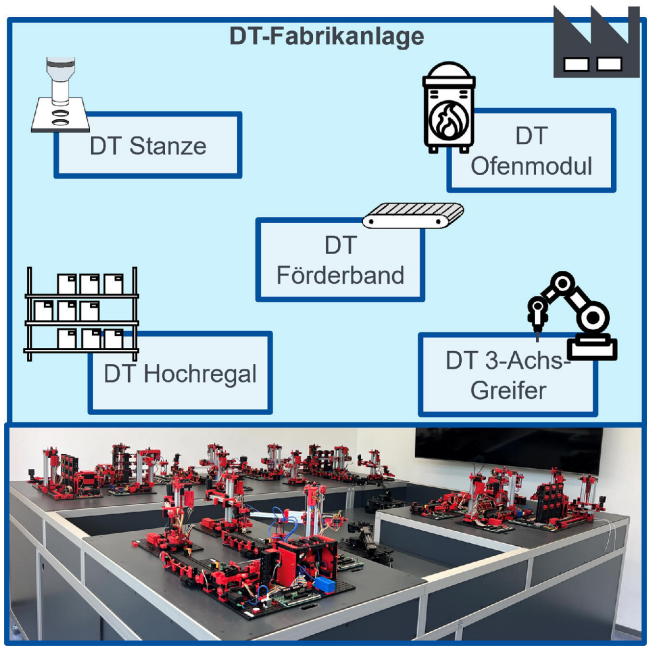

Bild 3. Aufbau des Demonstrators einer Fabrikanlage mit vier Produktionsstationen und Turtle Bots. *Grafik und Foto: ISW, Demonstrator: [2] und Turtle Bots: [23]*

3.4 Modellierung von Betriebsaktivitäten und Entwicklungsaktivitäten

Im Kontext des Model-Based DevOps konzentriert sich diese Herausforderung auf den Entwurf und die Implementierung modellbasierter Methoden (Kapitel 2.3) zur Darstellung von Laufzeitdaten in den Phasen des Betriebs und der Entwicklung. Eine initiale Aufgabe besteht darin, einen konzeptionellen Rahmen für die Definition von Laufzeitdaten zu entwickeln, woraufhin eine domänenspezifische Sprache für diesen Zweck erschaffen wird. Ein zentrales Merkmal dieser domänenspezifischen Sprache ist die Fähigkeit, modellbasierte Verbindungen zwischen den erfassten Laufzeitdaten und digitalen Zwillingen sowie zwischen Entwicklungsmodellen und digitalen Zwillingen zu repräsentieren.

4 Versuchsaufbau: Eine Demo-Fabrik in der Forschung

Um die Praktikabilität und den Nutzen von Model-Based DevOps für Industrie-4.0-Fabriken zu erforschen, besteht der Demonstrator aus einer modularen Produktionslinie, die mit dem „fischertechnik Baukasten“ realisiert wird. Jedes Modul wird von digitalen Zwillingen begleitet, um seinen Zustand zu überwachen und das Potenzial von Model-Based DevOps für die Optimierung von Entwicklungs- und Verhaltensmodellen der Module zu untersuchen.

Bild 3 zeigt in der unteren Hälfte den Fabrikdemonstrator und seinen Aufbau für einen Herstellungsprozess mit vier Produktionsstationen [26, 27]. Der digitale Zwilling der Fabrikanlage enthält digitale Zwillinge der fischertechnik-Module.

Die erste Station nimmt Rohstoffe entgegen und lagert diese in Hochregalen. Die zweite Station nimmt Rohstoffe entgegen, um diese mit Stanzen zu verarbeiten. Die dritte Station nutzt Werkzeugmaschinen, um die Montage durchzuführen. In der finalen Station wird das Produkt veredelt und bis zum Weitertransport gelagert. Jede Station simuliert einen Schritt aus einem Produktionsprozess, um einen Rohstoff bis zum Endprodukt zu verarbeiten. Außerdem gibt es Stationen für logistische Aufgaben wie Lagerung und Sortierung. Jede Station führt einen mehrstufigen Prozess aus, der den Einsatz von digitalen Zwillingen der entsprechenden Maschinen erfordert.

Die verschiedenen Produktionsstationen erfordern die folgenden Maschinen, die in Bild 3 in der unteren Hälfte dargestellt sind:

- 3-Achs-Greifer und Förderbänder: Die 3-Achs-Greifer sind mit Greifern ausgestattet, um Produkte anzuheben und abzusetzen. Die Förderbänder bieten eine effiziente Möglichkeit zum Transport von Produkten.
- Stanzmaschinen: Stanzmaschinen simulieren Stanzen, welche Materialien wie Metall in gewünschte Formen bringen.
- Hochregale: Hochregale dienen dem Sortieren und Lagern von Rohstoffen und auch Produkten.
- Ofenmodul: Eine Ofenmaschine, die neben einem Förderband auch mehrere Sensoren sowie LEDs bietet, um Erwärmungsprozesse zu simulieren.
- Turtle Bots: Turtle Bots dienen dem Transport der Produkte zwischen den Fabrikinseln sowie zum Abtransport des finalen Produktes.

Die digitalen Zwillinge der Demonstrationsmodule entsprechen ISO 23247 [10], da sie eine digitale Repräsentation der physischen Produktionsstationen enthalten und von einer digitalen Zwillingseinheit verwaltet werden.

Anhand dieses Versuchsaufbaus werden die Fortschritte in der Forschung aus Kapitel 3 genauer betrachtet. In einer ersten Studie wird anhand des Turtle Bots die Umsetzung des digitalen Zwillings untersucht. Es ist anzumerken, dass dies nur eine Aufnahme des aktuellen Zustands ist und noch aktiv geforscht wird.

5 Bisherige Fortschritte

Die bisherigen Ergebnisse bestehen im Wesentlichen aus zwei Teilen, zum einen der Herleitung der erarbeitete Referenzarchitektur für digitale Zwillinge und zum anderen eine prototypische Realisierung zur Komposition von digitalen Zwillingen.

5.1 Konzept eines selbstadaptiven digitalen Zwillings

Obwohl Entwicklungszeit und Betriebszeit von MDE profitieren können, sind die entsprechenden Modelle im traditionellen DevOps typischerweise getrennt. Daher hat der überwachte Systemzustand keinen direkten Einfluss auf die Entwurfsmodelle. Softwareingenieure müssen Designanpassungen implementieren, testen und erneut bereitstellen, nachdem sie Fehler im Systemzustand erkannt haben. In diesem Fall existiert nur eine manuelle Rückführung von der Betriebs- zur Entwicklungszeit.

Bisher sind selbstadaptive Systeme [28] prominent, mit dem Ziel, mittels einer Analyse und einer Reflexion, die modelliert werden können [29], die Anpassung von Modellen und Aktionen zu automatisieren (**Bild 4**).

Ein solches System für digitale Zwillinge umfasst bereits eine ausgereifte Implementierung der Adaptionsfähigkeit, die sich in fünf Schlüsselkomponenten (MAPE-K) gliedert:

- Monitor (Monitor): Überwacht kontinuierlich den Zustand des Systems und sammelt relevante Daten.

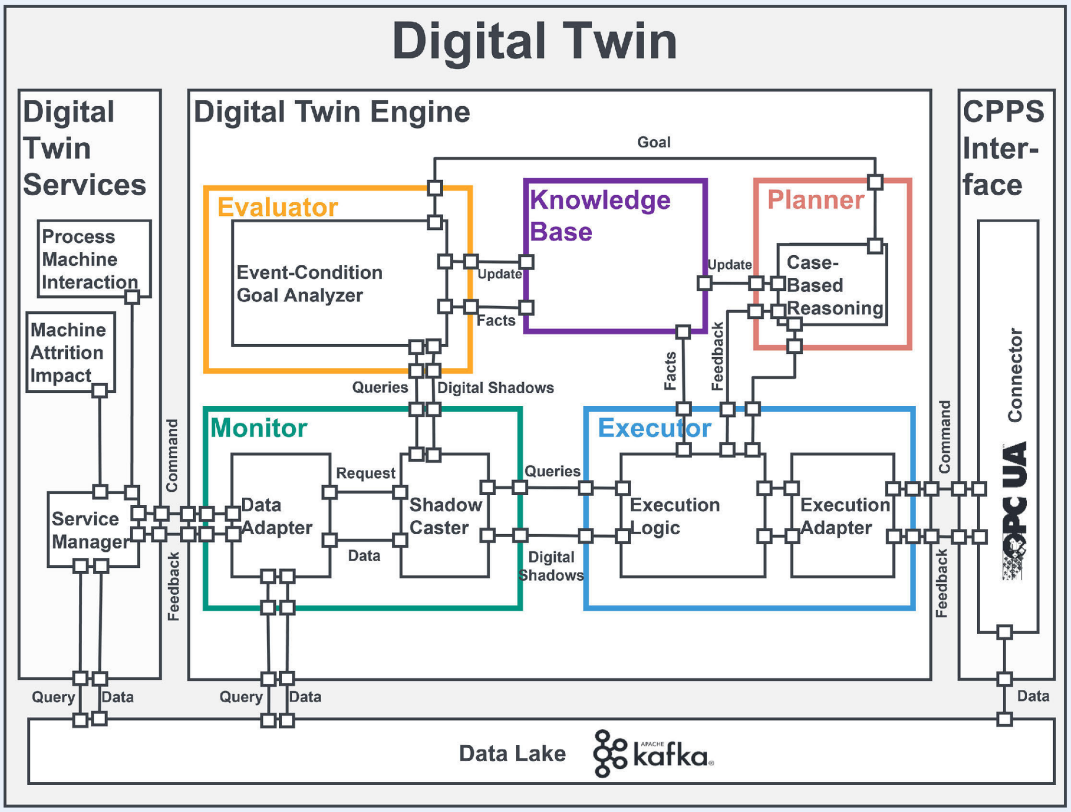

Bild 4. MAPE-K-Adaptionsmechanismus für adaptive digitale Zwillinge. *Grafik: [24]*

- Evaluator (Analyze): Analysiert die gesammelten Daten, um den aktuellen Zustand des Systems zu identifizieren und Veränderungen festzustellen.
- Planer (Plan): Entwickelt auf Basis der Analyseergebnisse einen Handlungsplan, der spezifische Ziele erreichen soll.
- Executor (Execute): Führt den entwickelten Plan aus, um die gewünschten Anpassungen am System vorzunehmen.
- Knowledge Base (Knowledge): Speichert Informationen über bekannte Szenarien und Erfahrungen, die zur Planerstellung und Entscheidungsfindung herangezogen werden.

Die Anpassungsfähigkeit ist meist auf spezifische Aspekte wie die Bereitstellung von Webservern beschränkt und muss im Vorfeld konzipiert werden. Außerdem besteht häufig keine direkte Verbindung zwischen der Entwicklungs- und der Betriebszeit, dies bedeutet, dass es keine Beziehung zwischen Entwicklungs- und Betriebsmodellen gibt.

5.1.1 Referenzarchitektur und Fallstudie am Turtle Bot

Im Folgenden wird eine modulare Referenzarchitektur (Kapitel 3.1) eingeführt. Diese wurde aus der MAPE-K-Architektur abgeleitet, indem die zentralen Komponenten betrachtet und in die funktionalen Komponenten eingegliedert werden, dargestellt in **Bild 5**. Danach wurde eine Machbarkeitsstudie mit der Implementierung des digitalen Zwillings durchgeführt. Die Architektur ist für eine automatisierte Synthese und flexible Anpassungen zur Laufzeit geeignet. Somit können Ingenieure passende Plugins für Service-Schnittstellen oder Gateways mit Low-Code-Lösungen definieren, erzeugen und in das digitalen Zwillingssystem integrieren.

Bild 5 zeigt die erarbeitete Referenzarchitektur eines digitalen Zwillings für den Turtle Bot. Dabei werden die funktionalen Anforderungen und Schnittstellen des digitalen Zwillings näher betrachtet. Die Komponenten sind:

– Ein Gateway zur Kommunikation zwischen dem digitalen Zwilling und anderen Systemen sowie dem cyber-physischen System, in diesem Fall dem Robot Operating System des Turtle Bots. Hier wurden Realisierungen der Schnittstellen erkundet, um Daten zu überwachen (Monitor) und Anweisungen zu senden (Execute). Durch den modularen Aufbau können die Monitor- und Executor-Komponenten ausgetauscht werden. Zum Beispiel kann die Schnittstelle zu bestehenden Systemen wie MES-Systemen gesichert werden, indem das digitale Zwillingssystem über die OPC-UA-Server mit den Daten der physikalischen Maschine arbeiten kann. Am Beispiel der fischertechnik-Fabrik [26] werden die Zustände der Maschinen auf OPC-UA erfasst und durch das digitale Zwillingssystem in der Monitor-Komponente interpretiert und abstrahiert.
– Ein Modell- und Datenverwaltungssystem (Knowledge Base), welches Wissen aggregiert. Es besteht aus einer Datenbank, die in regelmäßigen Abständen neue Daten ablegt und über die Engine Daten für Services bereitstellt. Zukünftig sollen Daten für KI-Algorithmen ebenfalls bereitgestellt werden.
– Eine Service-Schnittstelle, die es erlaubt, Dienste bereitzustellen und zu nutzen, etwa KI-Algorithmen in Form der Analyse- und Plan-Komponenten der MAPE-K. Hier können auch Visualisierungen und Steuerungen angebunden werden. Zudem bietet die Schnittstelle die Möglichkeit, mit maschinellem Lernen Optimierungsverfahren einzubinden. Die zugrunde liegende Technologie kann für nicht zeitkritische Anwendungen als

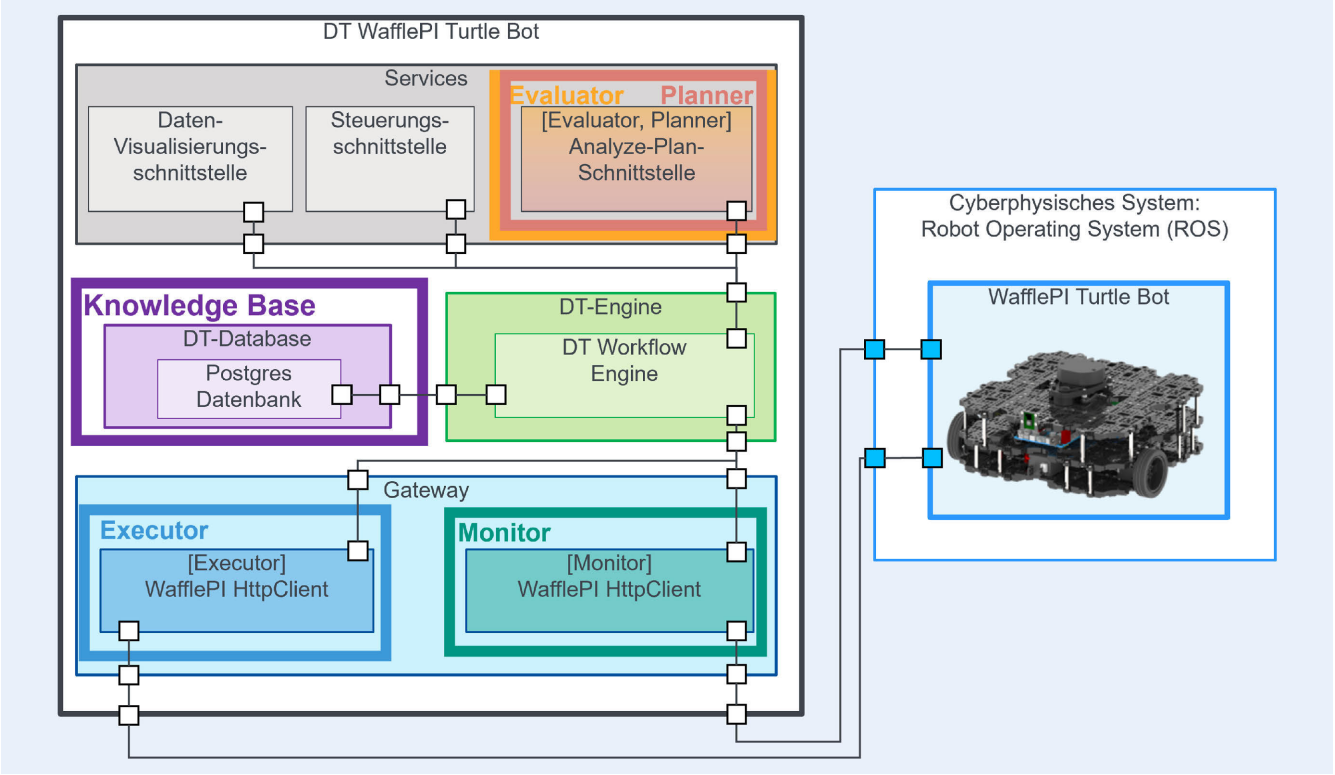

Bild 5. Beispielarchitektur eines Digitalen-Zwilling-Systems für einen Turtle Bot. *Grafik: ISW, Turtle Bot: [23]*

REST (Representational State Transfer)-Schnittstelle realisiert werden. Für zeitkritische Anwendungen müssen der digitale Zwilling sowie alle zeitkritischen Services auf der Maschine oder der Edge laufen.

– Eine Engine, die das Zusammenspiel der Komponenten orchestriert. Hier befindet sich ein Modul, das die Synchronisation der Daten zwischen Gateway und Knowledge Base sichert.

Um den Energieverbrauch (Kapitel 3.2) zu regulieren, wird das Verhalten des Turtle Bots zu Laufzeit angepasst. Hier wird eine Regel definiert, diese bewirkt eine Drosselung der Geschwindigkeit anhand der Vorhersage der Möglichkeit, die Arbeit zu vollenden. Wenn eine Tätigkeit, wie der Transport eines Produkts zwischen zwei Stationen, nicht möglich ist, wird der Turtle Bot zur Ladestation gefahren. Dies kann auch über eine Bedienoberfläche (**Bild 6**) gesteuert werden.

Eine Funktion zur Erkennung des Batteriestands könnte wie folgt lauten: Wenn die vorhergesagte Kurve außerhalb des möglichen Batteriestands liegt, wird der Turtle Bot aufgeladen.

Weitere Methoden des maschinellen Lernens bieten sich zudem zur Optimierung des Layouts oder einzelner Produktionsschritte hinsichtlich Wirtschaftlichkeit oder Umweltverträglichkeit an [30].

Bild 6 zeigt eine Bedienoberfläche für den Turtle Bot. Hier sind die Positionen der einzelnen Gelenke des Turtle Bots sowie Sensorwerte wie Geschwindigkeit und LIDAR (Light imaging, detection and ranging)-Sensorwerte unten ersichtlich. Zusätzlich ist der Batteriezustand in Relation zur Geschwindigkeit des Turtle Bots als Graph visualisiert.

Durch die Abstraktion der Bewegungsrichtungen des Turtle Bots und seines Greifarms kann der Benutzer den Turtle Bot bedienen, ohne Vorkenntnisse in Kinematik zu haben. Die gesendeten Befehle werden vom digitalen Zwilling ausgewertet und in Vektoren übersetzt, die zur Steuerung des physischen Systems verwendet werden.

5.2 Komposition von digitalen Zwillingen am Beispiel des Turtle Bots

Beim Turtle Bot als Ganzes handelt es sich um zwei digitale Zwillinge: ein digitaler Zwilling des WafflePI, der die Navigation in der Modellfabrik umsetzt, und ein digitaler Zwilling des Greifarms, der die Annahme und Ablage des Produktes und der Rohstoffe übernimmt.

Hier wird ein hierarchischer Ansatz zur Komposition (Kapitel 3.3) von digitalen Zwillingen verfolgt. Das Ergebnis ist ein komplexer digitaler Zwilling, bestehend aus dem digitalen Zwilling des WafflePI sowie einem digitalen Zwilling des Greifarms. Die Komposition wird auf Gateway-Ebene durchgeführt. Dabei wird ein neues Gateway-Modul erzeugt, das die beiden Gateways des WafflePI und des Greifarms kommuniziert und abstrahiert. Die Gateways gewährleisten die Interoperabilität der digitalen Zwillinge als ein gesamtheitlicher digitaler Zwilling des Turtle Bots.

Zudem sollen Modelle der Entwicklungszeit durch adaptive Verfahren anhand der Betriebsdaten optimiert werden, um den DevOps-Zyklus zu schließen. Die vorliegende Fallstudie gibt Einblick in die Komposition des digitalen Zwillings des Turtle Bots. Es bleibt offen, welche Auswirkungen eine Beziehung zwischen Modellen für die Adaption von Entwicklungsmodellen hat.

In nachfolgender Untersuchung soll die Referenzarchitektur von Turtle Bots auf komplexe Fabrikanlagen erweitert werden. Dabei sollen Kompositionsmechanismen und Auswirkungen auf Entwicklungsmodelle für komplexe Systeme einer Fabrikanlage untersucht werden.

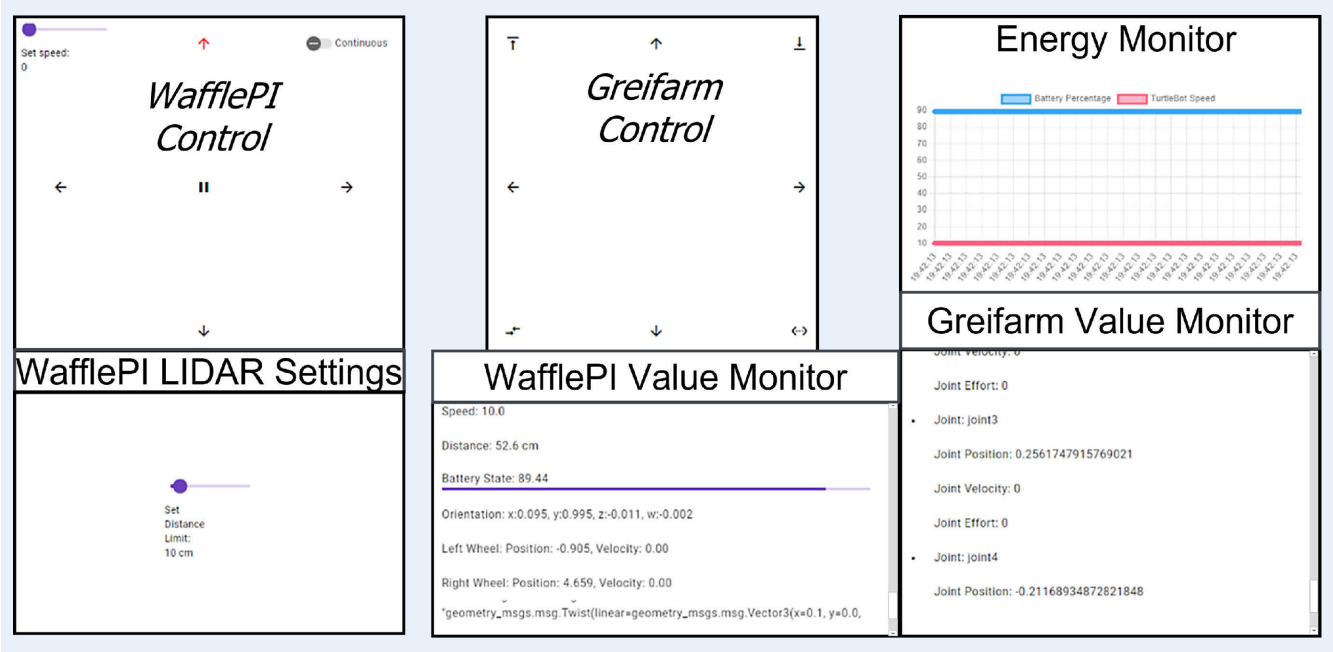

Bild 6. Bedienoberfläche des Turtle Bots. Oben links und zentral: Steuerung für WafflePI sowie dessen Greifarm. Unten zentral und rechts: Einstellungen des Turtle Bots sowie dessen Werte vom physischen System. Oben rechts: Batteriestand und Geschwindigkeit des WafflePI. *Grafik: ISW*

6 Diskussion zu offenen Herausforderungen

Im Folgenden werden die unbeantworteten Herausforderungen aus Kapitel 3 betrachtet und Ansätze zur Lösung der Herausforderungen erörtert.

Aufbauend auf dem definierten konzeptionellen Rahmen in Kapitel 3.4 wird die Entwicklung eines Orchestrationssystems zur Abfrage von Laufzeitdaten in Angriff genommen. Um Modelle effektiv zu synthetisieren, sollen Frameworks eingesetzt werden, die es ermöglichen, Entwicklungsmodelle – inklusive Details zu Maschinenstruktur, Geometrie, Kinematik und Verhalten – in Betriebsmodelle zu transformieren. Diese Transformation unterstützt die Anwendung der Modelle auf einen Demonstrator, um die theoretischen und praktischen Aspekte eines Model-Based-DevOps-Ansatzes zu verfeinern.

Dieses System soll zudem den Zugriff auf Informationen über vergangene, gegenwärtige und zukünftige Zustände von cyber-physischen Systemen ermöglichen. Es schafft eine konzeptionelle Verknüpfung zwischen den Entwurfsphasenmodellen, wie Zustandsmaschinen, und deren Laufzeitdarstellung, einschließlich der Zustandshistorie.

Bei der Nachrüstung bestehender Systeme mit Model-Based-DevOps-Funktionen ist es entscheidend, Laufzeitmodelle aus Entwicklungsmodellen und Zustandsdaten zu generieren. Diese Daten können beispielsweise durch Protokolldateien oder Datenbanken repräsentiert werden und erlauben eine modellbasierte Überwachung.

7 Fazit

Dieser Beitrag stellt Model-Based DevOps vor. Dabei handelt es sich um einen neuen Forschungsweg, der die Entwurfsmodelle von komplexen cyber-physischen Systemen mit ihrer Laufzeitüberwachung in Form von digitalen Zwillingen verbindet.

Die Anwendung von Model-Based DevOps macht die Generierung eines digitalen Zwillings aus seinem Entwicklungsmodell möglich. Zudem erlaubt Model-Based DevOps die Rückführung von Änderungen am Laufzeitmodell in die Entwurfsmodelle der digitalen Zwillinge und trägt somit zur Verbesserung der Nachhaltigkeit bei.

Ausgehend von bisherigen Forschungen wurden Forschungsschwerpunkte zu einer nachhaltigen Industrie 4.0 identifiziert, die für die Untersuchung von Model-Based DevOps von zentraler Bedeutung sind. In zukünftigen Studien soll eine Steigerung der Effizienz von Produktionsprozessen durch simulierte Rekonfigurationen und eine Optimierung des Ressourcenverbrauchs sowie die Reduktion von Emissionen der Fabrik in Bezug auf Produktqualität und Prozesseffizienz aufgezeigt werden. Zuletzt soll eine Erhöhung der Transparenz erfolgen.

FÖRDERHINWEIS

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) 505496753.

[1] Lasi, H.; Fettke, P.; Kemper, H. G. et al.: Industrie 4.0. Wirtschaftsinformatik 56 (2014) 4, S. 261–264

[2] Braun, S.; Dalibor, M.; Jansen, N. et al.: Engineering digital twins and digital shadows as key enablers for industry 4.0. In: Vogel-Heuser, B., LiteraturWimmer, M. (eds.): Digital transformation. Heidelberg: Springer 2023, pp. 3–31

[3] Fur, S.; Heithoff, M.; Michael, J. et al.: Sustainable digital twin engineering for the internet of production. In: Karaarslan, E.; Aydin, Ö.; Cali, Ü. et al. (eds.): Digital Twin Driven Intelligent Systems and Emerging Metaverse. Singapore: Springer Nature 2023, pp. 101–121

[4] Kritzinger, W.; Karner, M.; Traar, G. et al.: Digital Twin in manufacturing: A categorical literature review and classification. Ifac-PapersOnline 51 (2018) 11, pp. 1016–1022

[5] Becker, F.; Bibow, P.; Dalibor, M. et al.: A conceptual model for digital shadows in industry and its application. In: Ghose, A.; Horkoff, J.; Silva Souza, V.E.et al. (eds.): Conceptual Modeling. ER 2021. Cham: Springer 2021, pp. 271–281

[6] ISO/DIS 23247–1: Automation systems and integration—digital twin framework for manufacturing—Part 1: overview and general principles. International Organization for Standardization Geneva, Switzerland 2020

[7] Dalibor, M.; Jansen, N.; Rumpe, B.et al.: A cross-domain systematic mapping study on software engineering for digital twins. Journal of Systems and Software 193 (2022), #111361

[8] Bass, L.; Weber, I.; Zhu, L.: DevOps: A software architect's perspective. Boston: Addison-Wesley Professional 2015

[9] Stahl, T.; Völter, M.; Efftinge, S. et al.: Modellgetriebene Softwareentwicklung: Techniken. Engineering, Management. Heidelberg: dpunkt.verlag 2007, S. 64–71

[10] Kirchhof, J. C.; Michael, J.; Rumpe, B. et al.: Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems, 2020, pp. 90–101

[11] Brambilla, M.; Cabot, J.; Wimmer, M.: Model-driven software engineering in practice. San Rafael, California (USA): Morgan Claypool Publishers 2017

[12] Debruyne, V.; Simonot-Lion, F.; Trinquet, Y.: EAST-ADL—An Architecture Description Language: Validation and Verification Aspects. World Computer Congress, IFIP TC-2 Workshop on Architecture Description Languages (WADL), Toulouse/France, 2004, pp. 181–195

[13] Zhang, Q.; Wang, S.; Liu, B.: Approach for integrated modular avionics reconfiguration modelling and reliability analysis based on AADL. IET Software (2016), pp. 18–25, doi.org/10.1049/iet-sen.2014.0179

[14] Leitner, S. H.; Mahnke, W.: OPC UA – Service-oriented architecture for industrial applications. Softwaretechnik-Trends 26 (2006): no page

[15] Ebert, C.; Gallardo, G.; Hernantes, J. et al.: DevOps. IEEE software 33 (2016) 3, pp. 94–100

[16] Bellavista, P.; Bicocchi, N.; Fogli, M.et al.: Requirements and design patterns for adaptive, autonomous, and context-aware digital twins in industry 4.0 digital factories. Computers in Industry 149 (2023), #103918

[17] Combemale, B.; France, R.; Jézéquel, J. M. et al.: Engineering modeling languages: Turning domain knowledge into tools. Boca Raton, Florida: CRC Press 2016

[18] Blair, G.; Bencomo, N.; France, R. B.: Models@run time. Computer 42 (2009) 10, pp. 22–27

[19] Morin, B.; Barais, O.; Jézéquel, J. M.et al.: Models@run time to support dynamic adaptation. Computer 42 (2009) 10, pp. 44–51

[20] Bibow, P.; Dalibor, M.; Hopmann, C.; Mainz, B. et al.: Model-driven development of a digital twin for injection molding. In: Dustdar, S.; Yu, E.; Salinesi, C.et al. (eds.): Advanced Information Systems Engineering. CAiSE 2020. Cham: Springer International Publishing 2020, pp. 85–100

[21] Lehner, D.; Sint, S.; Vierhauser, M. et al.: AML4DT: A model-driven framework for developing and maintaining digital twins with automationml. 26th IEEE international conference on emerging technologies and factory automation (ETFA). IEEE 2021, pp. 1–8

[22] Lee, J.; Bagheri, B.; Kao, H. A.: A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufacturing letters 3 (2015), pp. 18–23

[23] Combemale, B.; Jézéquel, J. M.; Perez, Q. et al.: Model-Based DevOps: Foundations and Challenges. ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). IEEE 2023, pp. 429–433

[24] Giddings, B.; Hopwood, B.; O'brien, G.: Environment, economy and society: fitting them together into sustainable development. Sustainable development 10 (2002) 4, pp. 187–196

[25] Jiang, Y.; Yin, S.; Dong, J. et al.: A review on soft sensors for monitoring, control, and optimization of industrial processes. IEEE Sensors Journal 21 (2020) 11, pp. 12868–12881

[26] fischertechnik: Homepage. Internet: www.fischertechnik.de. Zugriff am 05.06.2024

[27] Robotis: Homepage. Internet: www.robotis.us. Zugriff am 05.06.2024

[28] Splettstößer, A. K.; Ellwein, C.; Wortmann, A.: Self-adaptive digital twin reference architecture to improve process quality. Procedia CIRP 119 (2023), pp. 867–872

[29] Amin, M.: Toward self-healing infrastructure systems. Computer 33 (2000) 8, pp. 44–53

[30] Weichert, D.; Link, P.; Stoll, A. et al.: A review of machine learning for the optimization of production processes. The International Journal of Advanced Manufacturing Technology 104 (2019) 5, pp. 1889–1902

Jingxi Zhang M.Sc.
Foto: ISW
jingxi.zhang@isw.uni-stuttgart.de
Tel. +49 711 685-82325

Prof. Dr. rer. nat. habil. Andreas Wortmann

Prof. Dr.-Ing. Oliver Riedel

Universität Stuttgart, Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW)
Seidenstr. 36, 70174 Stuttgart
www.isw.uni-stuttgart.de