Industrial DevOps

Schnelle Entwicklungszyklen und hohe Softwarequalität sind in der Automatisierung kein Widerspruch mehr

vorausrobotik

Carl-Buderus-Str. 7 | 30455 Hannover | **voraus**robotik.com

1 Zusammenfassung

Die industrielle Automatisierung ist in immer kürzeren Zyklen wechselnden Anforderungen unterworfen. Dazu gehört die Automatisierungsaufgabe selbst (z. B. Produktvielfalt und Variantenanzahl, Individualisierung, Fehlerbehebung). Aber auch normative Regelungen, z. B. aufgrund des Cyber Resilience Acts (CRA), oder die durch die zunehmenden geopolitischen Unsicherheiten geforderte digitale Souveränität tragen hierzu bei. Die Umsetzung dieser, zum Teil bei der Konzipierung der Anlage noch unbekannten Anforderungen, stellt die hierfür Verantwortlichen vor große Herausforderungen: Die Komplexität und Größe der Automatisierungsprojekte steigt, vor allem im Software-Bereich. Gleichzeitig sinkt bedingt durch den demographischen Wandel die Verfügbarkeit der erforderlichen Experten. Verschärft wird die Situation durch die heterogene und in großen Teilen inkompatible Softwarelandschaft: In der industriellen Automatisierung sind verschiedenste Software-Module zu erstellen, die häufig in proprietären, veralteten und unterschiedlichen Sprachen programmiert werden müssen. Im Ergebnis gestalten sich (nachträgliche) Anpassungen als sehr aufwändig, und Softwaretests können erst gegen Ende des Projektes an der realen Anlage erfolgen – dies führt zu hohen Risiken bspw. in Bezug auf Anlagenstillstand oder -beschädigung. Schlussendlich erhöhen sich zwangsläufig die Kosten und es verlängert sich die Laufzeit der Automatisierungsprojekte, was deren Rentabilität in Frage stellt. Diesen Herausforderungen ist mit den aktuell verfügbaren Software-Werkzeugen für die industrielle Automatisierung nicht effizient zu begegnen.

Anderseits beweist der aus der IT-Welt bekannte DevOps-Ansatz, dass schnelle Entwicklungszyklen, flexible und trotzdem kompatible Toollandschaften sowie eine hohe Software-Qualität kein Widerspruch sind. Dieses White-Paper zeigt auf, welche Voraussetzungen in der OT-Welt erfüllt sein müssen, um den erfolgreichen DevOps-Ansatz in die industrielle Automatisierung zu übertragen. Wie unten detaillierter dargelegt, liegt die Lösung in einer Software-Plattform die zugleich hochsprachen- als auch echtzeitfähig ist, um alle Komponenten einer Automatisierungslösung einheitlich zu orchestrieren. Darüber hinaus ist eine Virtualisierung ebendieser Komponenten erforderlich. So wird

eine effiziente Entwicklung auch ohne real verfügbare Hardware möglich. Die damit einhergehende hohe Testabdeckung, vor dem Deployment auf die Anlage, erlaubt die frühzeitige Erkennung von Fehlern und deren Behebung bei geringen Kosten. Im Ergebnis entkoppelt sich die Software-Entwicklung von der Hardware-Verfügbarkeit und es resultieren schnelle Entwicklungszyklen bei hoher Qualität. Unser Fallbeispiel zeigt, dass die Kosten für das Einspielen von Updates um mehr als 90 % gesenkt werden können.

Wir nennen es **Industrial DevOps.**

2 Einleitung

Industrielle Automatisierungstechnik ist im internationalen Wettbewerb von entscheidender Bedeutung – sie ermöglicht nicht nur die wirtschaftlich konkurrenzfähige Produktion einer Vielzahl von Wirtschaftsgütern, sondern sie trägt auch dazu bei, die Auswirkungen des demographischen Wandels abzufedern. Trotzdem sehen sich sowohl Hersteller von Automatisierungslösungen bzw. deren Komponenten als auch produzierende Betriebe mit zahlreichen Anforderungen konfrontiert, die Aufwände und Kosten treiben und daher zunehmend die Wirtschaftlichkeit in Frage stellen.

Als erster Grund ist hier die Inflexibilität insbesondere von älteren, über die Jahre gewachsenen Anlagen zu nennen. Diese lassen sich nur mit hohem Aufwand auf neue Produkte umrüsten – Mitarbeiter und Integratoren, die über entsprechendes Know-How verfügen, sind nicht mehr verfügbar, Softwarestände unbekannt, Softwareschnittstellen sind inkompatibel zur IT-Infrastruktur etc.

Dies betrifft aber auch neue Anlagen – häufig werden proprietäre, zum Teil nur eingeschränkt miteinander kompatible Komponenten verbaut. Insbesondere herstellerspezifische Programmiersprachen und -umgebungen erzwingen die gleichzeitige Tätigkeit verschiedener Experten und machen eine holistische Anlagensimulation, -programmierung und -pflege nahezu unmöglich. In letzter Konsequenz können wesentliche Funktionen erst an der realen Anlage umgesetzt und getestet werden – hier auftretende Fehler oder erforderliche Änderungen führen zu einer Explosion von Kosten und Aufwänden. Dies veranschaulicht untenstehendes Bild 1 eindrucksvoll – es zeigt, wann Fehler in der Software-Entwicklung entstehen (zu 85 % während der Programmierung) und wann sie entdeckt werden. Je später dies ist, umso höher werden die Kosten – dies ist insbesondere bei der heutigen Entwicklung von Automationslösungen gegen Ende des Projektes (aktuell ca. 75 % während der Systemtests). Ziel muss also sein, die Fehlererkennung und -vermeidung nach vorne zu verschieben.

Regelmäßige Software-Aktualisierungen, wie sie zukünftig durch den CRA zur Pflicht werden, Integration in die bestehende und einem stetigen Wandel unterworfene IT-Infrastruktur oder sich im Laufe des Betriebes ergebende Änderungen (z. B. aufgrund von

neuen Produkten oder Produktanpassungen) erfordern einen hohen personellen Einsatz, der durch das Fachkräfteproblem noch verschärft wird, und bergen große Risiken für Fehler und Stillstand, wie oben beschrieben. Ein kleines Beispiel: Ein mittelständisches Unternehmen liefert jährlich 100 Anlagen aus, die jeweils 10 Jahre laufen. So ergeben sich 1.000 Anlagen im Feld, die gewartet, aktualisiert etc. werden müssen. Wenn wir von einem erforderlichen Update (bspw. aufgrund des CRA oder von Produktänderungen) pro Anlage und Quartal ausgehen, so ergeben sich konservativ angenommen 4.000 Einsätze pro Jahr – in Anbetracht der Expertenverfügbarkeit und dem Risiko eines Produktionsausfalls ist dies wirtschaftlich kaum darstellbar.

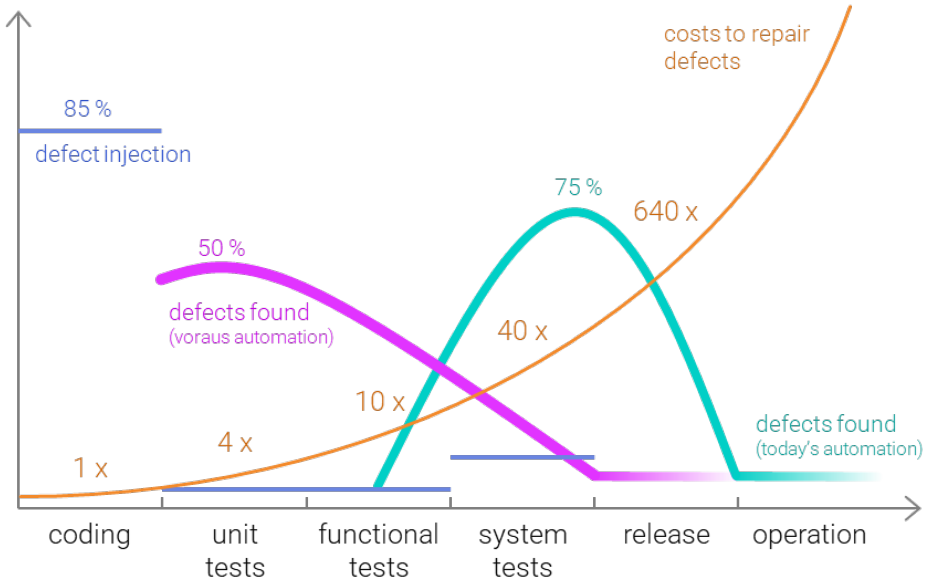

Bild 1: Fehlerentstehung und -entdeckung in der Software-Entwicklung
(Grafik nach Jones, Caspers. Applied Software Measurement: Global Analysis of Productivity and Quality)

Was also ist zu tun? Wir brauchen dringend eine Lösung, die vergleichbar zum DevOps-Ansatz in der IT, eine holistische Betrachtung des gesamten Software-Lebenszyklus einer Anlage über alle Komponenten beinhaltet. Nur so wird es möglich, Entwicklungszeiten drastisch zu reduzieren, die Anzahl der benötigten Experten zu begrenzen, Fehler frühzeitig zu entdecken und Anlagen im Feld mit geringem Aufwand zu aktualisieren. Gleichzeitig darf dabei nicht die digitale Souveränität der Anlagenbetreiber gefährdet und neue Abhängigkeiten geschaffen werden.

3 Das Problem an der Wurzel packen

Die oben genannte Problematik ist nicht neu und wird von vielen Anbietern adressiert. Diese fokussieren jedoch fast immer Teillösungen und setzen dabei auf der existieren SW-Landschaft auf. Dies löst zwar punktuell ein Problem, bleibt jedoch prinzipbedingt Stückwerk. Unserer Meinung nach muss man zum Kern vordringen.

3.1 Was ist DevOps?

DevOps ist ein Kunstwort aus der IT, das für **Dev**elopment und **Op**erations steht, also Entwicklung und Betrieb ganzheitlich betrachtet. Dies betrifft nicht nur die Prozesse, sondern auch die dahinterstehende Toolchain. Ziel ist eine schnelle Software-Entwicklung und -Auslieferung bei gleichzeitig hoher Qualität. In diesem Kontext wird auch immer gerne auf die „liegende Acht" verwiesen (Bild 2).

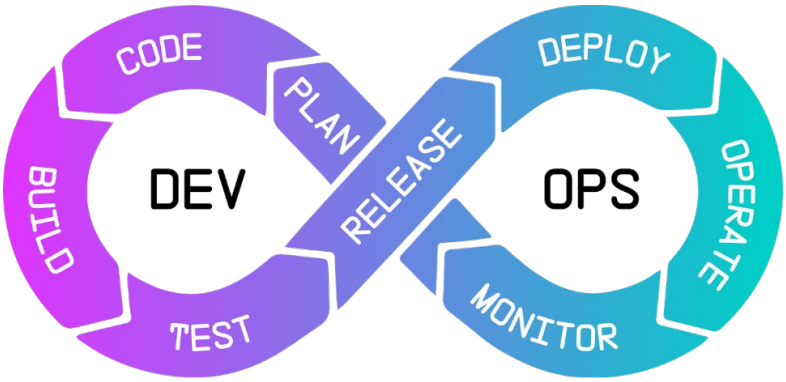

Bild 2: DevOps – Prozessschritte der modernen Software-Entwicklung

Neben der Planung, der Programmierung, dem Erstellen der Software und dem Test (typische Entwicklungsphasen) – linker Teil der Abbildung – beinhaltet der Begriff auch die Erzeugung von Releases, deren Auslieferung und den Betrieb der Software sowie das Monitoring. Die jeweiligen Prozessschritte werden von leistungsstarken, miteinander kompatiblen Tools unterstützt. Entscheidend für die hohe Softwarequalität ist das frühzeitige (und automatisierte) Testen der Software. In der klassischen IT ist DevOps inzwischen fest etabliert und zum Standard geworden – so werden bspw. die Internetseiten bei großen Versandhändlern mehrere Hundert Mal am Tag aktualisiert, ohne, dass der Endnutzer dies bewusst wahrnimmt.

3.2 Was sind die Herausforderungen in der Automatisierungstechnik?

Warum ist es so schwer, die in der IT bewährte Vorgehensweise, die viele der in Abschnitt 2 genannten Probleme löst, in die Automatisierung (OT) zu übertragen – also die so oft geforderte IT-OT-Konvergenz durchzuführen?

Hierfür gibt es im Wesentlichen zwei Gründe:

- Die in der Automatisierung eingesetzten Programmiersprachen sind veraltet (haben ihren Ursprung zum Teil in den 70er Jahren des letzten Jahrhunderts) und häufig proprietär. Als Beispiel seien hier der (zwar in der EN 61131-3 standardisierte) strukturierte Text für die SPS-Programmierung oder die roboterspezifischen Programmiersprachen genannt. Dies führt einerseits dazu, dass moderne, in der Software-Entwicklung etablierte Tools nicht zugänglich sind und anderseits, dass die vom Hersteller mitgelieferten Entwicklungsumgebungen nicht

miteinander kompatibel sind und nur den jeweiligen Teilbereich abdecken. Daraus resultiert schlussendlich eine starke Fragmentierung der Software- und Toollandschaft mit zahlreichen Brüchen.
- Der zweite wesentliche Grund ist die fehlende Möglichkeit zum ganzheitlichen Testen von Anlagen. So ist es zwar mit den herstellerspezifischen Tools denkbar, virtuelle (d. h. hardwareunabhängige) Tests durchzuführen, diese sind jedoch nur punktuell (also an die jeweilige Komponente gebunden). Die gesamte Steuerung einer Anlage (inkl. sog. Edge-Cases) kann erst an der realen Zelle final geprüft werden. Dies führt, wie oben gezeigt, zu hohen Kosten bei der Fehlerbehebung und setzt die Verfügbarkeit aller Komponenten voraus, die je nach Lieferzeiten erst spät im Projekt gegeben ist. Dadurch sind Verzögerungen und Kostensteigerungen zumeist unvermeidbar.

Das Ergebnis ist bekannt: Anlagen werden nach erfolgreicher Inbetriebnahme kaum mehr aktualisiert und optimiert, da das Risiko von Fehlern und damit von (ungeplanten) Anlagenstillständen zu groß ist. Die immer kürzer werden Produktlebenszyklen, die hohe Variantenvielfalt und nicht zuletzt die gesetzlichen Anforderungen aus dem CRA stehen dem jedoch grundsätzlich gegenüber.

4 Lösung: Industrial DevOps

Die gute Nachricht ist, dass eine Lösung für die Automatisierung existiert. Die aus der IT bekannte Toolchain ist prinzipiell auch für die OT anwendbar – nur ist sie mit den proprietären Programmiersprachen und inkompatiblen herstellerspezifischen Entwicklungsumgebungen derzeit nicht zugänglich.

Ein wichtiger Lösungsbaustein ist die Programmierung aller Komponenten einer Automatisierungszelle in Hochsprache, ohne Einschränkung der Echtzeitfähigkeit. Dadurch werden die Inkompatibilitäten aufgebrochen und die existierenden Software-Tools zugänglich. Bei voraus robotik stellt sich die Industrial DevOps Toolchain dann wie in Bild 3 gezeigt dar. Hierbei sind gängige (zum Teil kostenlose) IT-Werkzeuge den einzelnen Prozessen zugeordnet, die den gesamten Automatisierungs-Software-Stack abdecken und zueinander kompatibel sind. Des Weiteren übertreffen sie in ihrer Leistungsfähigkeit häufig die proprietären Teillösungen. Ein Vendor-Lock findet nicht statt – prinzipiell sind die Anwender in der Tool-Auswahl frei; ebenso existiert kein Cloud-Zwang – auch hier hat der Kunde die volle Kontrolle. Somit liefert Industrial DevOps einen wichtigen Beitrag zur digitalen Souveränität.

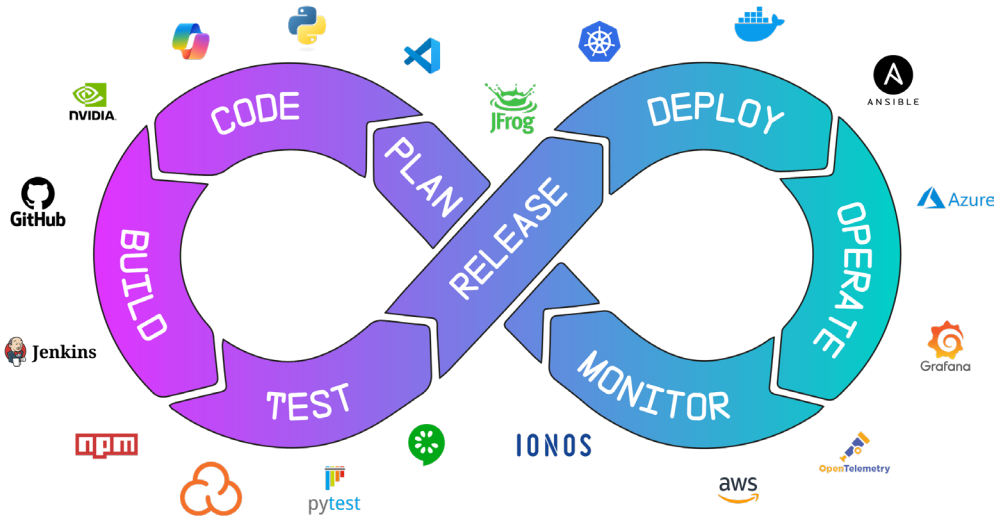

Bild 3: **Industrial DevOps** von voraus robotik inkl. Toolchain (Quelle Logos: Hersteller)

Der zweite Schlüssel ist die Virtualisierung wesentlicher Eigenschaften der Automatisierungskomponenten. So können Entwicklung und Test der Software gegen die virtuellen Komponenten erfolgen und Optimierungen sowie Fehlerbehebungen in einem frühen Stadium und damit kosteneffizient erfolgen. Da die Schnittstellen / APIs der virtuellen Komponenten identisch zu den realen sind, erfordert das Deployment keine Anpassungen. Es existiert eine einheitliche und durchgängige Code-Basis.

Wo findet die eigentliche OT-IT-Konvergenz statt?

Bild 4 veranschaulicht dies anhand der bekannten Automatisierungspyramide (nach ISA-95). Die entscheidenden Ebenen sind Level 1 und Level 2: Werden hier die Inkompatibilitäten aufgebrochen und APIs vereinheitlicht, so wird ein nahtloser Durchgriff der IT-Ebene bis auf Feldebene möglich. Gerade für Big Data und Anwendungen der künstlichen Intelligenz ist dies eine zwingende Voraussetzung.

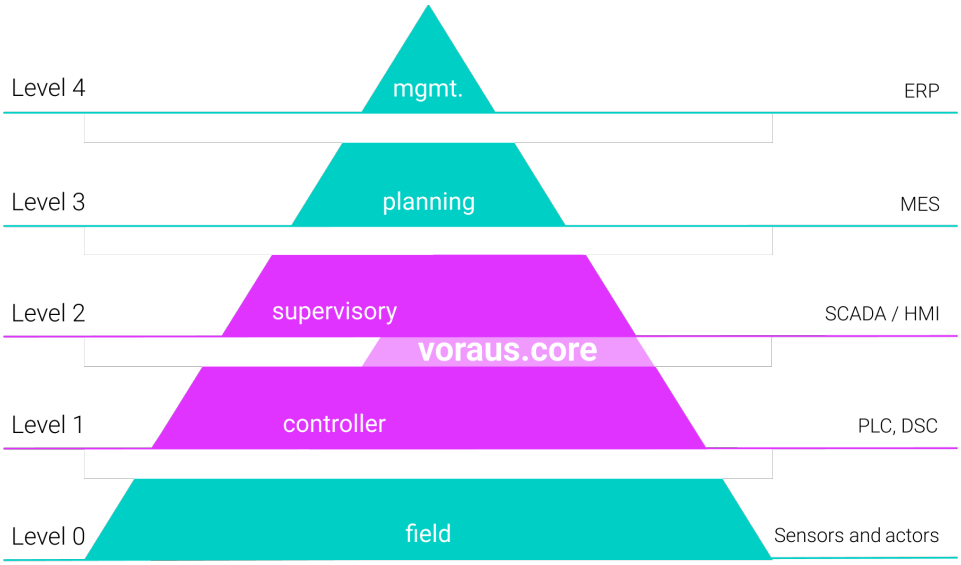

Bild 4: Der voraus.core macht Level 1 und Level 2 IT-fähig

4.1 Echtzeitfähige SW-Plattform – voraus.core

Auch wenn sämtliche Komponenten in Hochsprache programmiert werden, so muss die Orchestrierung der Anlage trotzdem robust und in Echtzeit erfolgen. Hierfür sind entsprechende Treiber erforderlich, die weitestmöglich auf industriellen Standards aufsetzen (z. B. EtherCAT, PROFINET, OPC UA) oder proprietäre Systeme einbinden (z. B. Roboter verschiedenster Hersteller oder existierende SPS). Dies sind in Bild 5 die beiden unteren Software-Layer. Um die Erweiterbarkeit sicherzustellen, ist der Software-Stack der Steuerungsplattform modular und containerisiert, die Schnittstellen orientieren sich an IT-Standards, sind offen und dokumentiert.

Basierend auf der API wird die eigentliche Applikation in Hochsprache (z. B. Python) programmiert, wobei einerseits auf existierende Software-Module (z. B. für Robotik) aufgebaut, anderseits eigene Funktionalität eingebracht werden kann. Dank standardisierter und moderner Interfaces ist die Anbindung an existierende IT (wie bspw. HMI, SCADA, Warenwirtschaftssysteme, Flottensteuerung für AMR) problemlos und in kürzester Zeit möglich. Des Weiteren ist Dank Hochsprache die Einbindung neuester KI-Methoden ebenfalls kein Problem mehr.

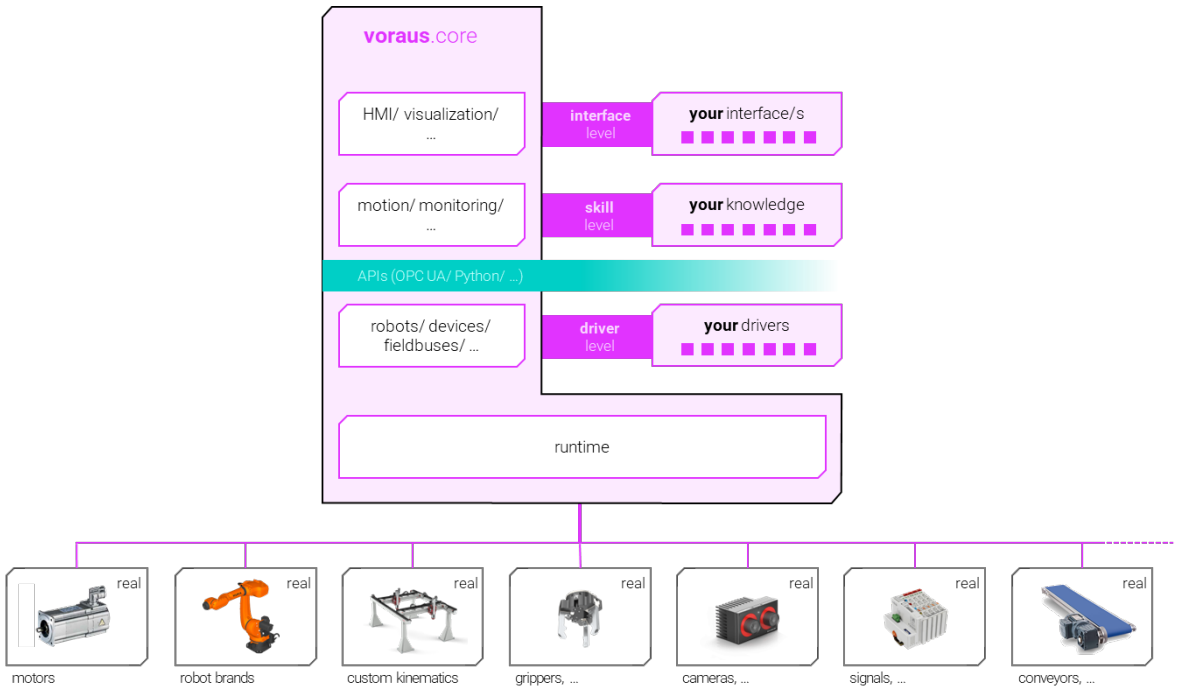

Bild 5: voraus.core – von der Feldbusebene bis zur Applikation

Die beschriebene Architektur des voraus.core stellt den rechten Teil der liegenden DevOps-Acht sicher.

4.2 Entwicklungsumgebung – voraus.pioneer

Der linke Teil der DevOps-Acht wird durch den linken Part von Bild 6 umgesetzt. Hier greifen zahlreiche moderne IT-Tools ineinander, um eine effiziente Programmierung zu

gewährleisten. Dabei ist die Auswahl der Software-Werkzeuge nicht fest vorgegeben und kann individuell den Bedürfnissen angepasst werden. Die oben beschriebene Virtualisierung der Komponenten trägt dazu bei, dass die Entwicklung der gesamten Applikation unabhängig von der Verfügbarkeit der Hardware erfolgt. Insbesondere können Anlagen schnell visualisiert und mit physikalischen Eigenschaften versehen simuliert und optimiert werden (dank der zahlreichen am Markt verfügbaren Physics-Engines). Die Erstellung von Testfällen erfolgt parallel zur Spezifikation bzw. Programmierung. Die automatisierten Tests stellen (nach jeder Änderung) sicher, dass z. B. die geforderten KPIs (Erreichbarkeit, Taktzeit etc.) eingehalten werden. Edge-Cases und Error-Injection garantieren ein robustes Verhalten der Anlage im Feld – so kann im Vorfeld eine Testabdeckung von über 95 % erreicht werden, während sie bei der klassischen SPS-Programmierung bei unter 30 % liegen dürfte. Dies ist jedoch nur möglich, wenn Entwicklung und Betrieb die gleiche Code-Basis nutzen und die virtuellen und realen Komponenten über dieselben APIs verfügen.

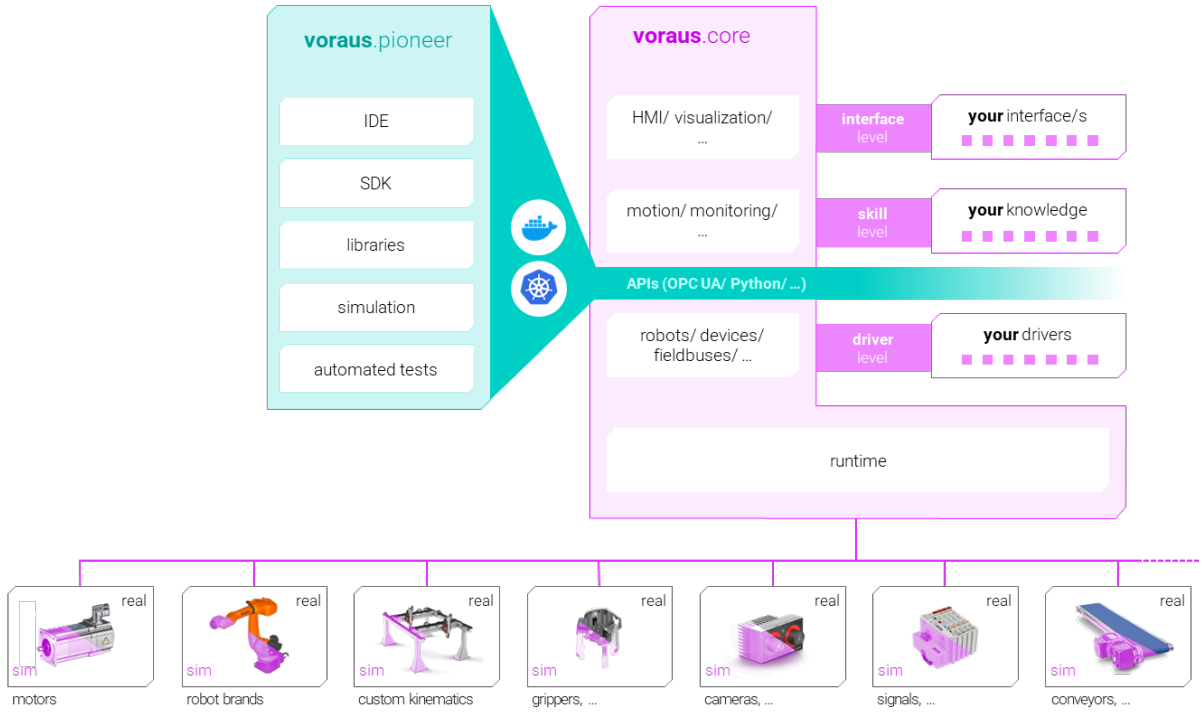

Bild 6: Holistische Umsetzung von Industrial DevOps durch voraus.pioneer und voraus.core

5 Fallstudien

In diesem Abschnitt wollen wir an zwei Beispielen exemplarisch betrachten, welche Vorteile der Industrial DevOps Ansatz im Vergleich zum klassischen Vorgehen liefert.

5.1 Update und Rollback

Um mit geringem Risiko Updates auf Anlagen zu verteilen (oder im Falle von Problemen ein Rollback zu veranlassen), ist es essenziell, Software mit hoher Qualität (also mit hoher Testabdeckung) zu erstellen. Dabei haben wir bei voraus robotik für interne wie externe Zwecke folgende Testpipeline etabliert: Die Jenkins-basierte Testpipeline folgt einem strukturierten Industrial DevOps-Workflow mit mehreren Stufen zur Sicherstellung der Softwarequalität und Release-Bereitstellung. Der Testzyklus beginnt mit der Pipeline-Vorbereitung, einschließlich Initialisierung von Workspace-Variablen und Formatierungsprüfungen. Die Artefaktverarbeitung umfasst das Abrufen, Verarbeiten und Ergänzen von Build-Informationen, gefolgt vom eigentlichen Build-Prozess, der Generierung von Dokumentation sowie dem Ausführen von Unit Tests und statischen Codeanalysen. Weitere Schritte decken Compliance-Prüfungen (z. B. OSS License Compliance), das Erstellen von Container-Images und das Packaging für Releases ab. Anschließend erfolgen Deployment- sowie System-, Integrations- und End-to-End-Tests auf der definierten Zielhardware. Abschließende Schritte der Testpipeline wie SonarQube-Analysen und das Speichern von Metadaten sichern die Nachverfolgbarkeit.

Die beschriebene Pipeline ermöglicht vollautomatisiert und in kürzester Zeit das Testen sowie das Erstellen von Releases. Das früher zumindest in Teilen übliche manuelle Abprüfen wird vollständig vermieden und der Personaleinsatz ist von zuvor mehreren Tagen auf jetzt wenige Stunden gesunken. Da das Erstellen von Testfällen ohnehin Teil der Software-Entwicklung ist, beschränkt sich der einmalige Mehraufwand auf das Aufsetzen und Befüllen der oben beschriebenen Testpipeline.

Auch das bisher übliche Aufspielen von Updates verändert sich drastisch: Die containerisierte Software kann entweder (kundenindividuell) zum Download angeboten werden (und ist mit wenigen Klicks installierbar) oder wird per Kubernetes-Cluster verteilt. Betrug vorher die Zeitdauer für ein Update 60 Minuten, so sind es jetzt weniger als 5 Minuten. Greifen wir das Beispiel des Maschinenbauers mit 1.000 Anlagen im Feld und 4 Updates pro Jahr nochmal auf, so ergeben sich folgende Zahlen:

	Klassischer Ansatz	**Industrial DevOps**
Dauer je Update	60 Minuten	5 Minuten
Kosten je Update (200 €/h)	200 €	17 €
Kosten je Anlage (4 Updates je Quartal)	800 €	68 €
Kosten für 1.000 Anlagen	800.000 €	68.000 €

Die mit Industrial DevOps erzielbare Einsparung beträgt in diesem Beispiel über 90 %.

5.2 Optimale Auswahl und Programmierung von Robotern

Die Firma autonox Robotics GmbH bietet eine Vielzahl von Roboterkinematiken für verschiedenste Anwendungen an. Doch welcher Roboter ist für meine Applikation der

richtige? Die Beantwortung dieser Frage nahm bisher viele Tage in Anspruch. Um Kunden die Auswahl zu erleichtern, haben wir beispielhaft den autonox Finder, der mehrere Hundert Kinematiken beinhaltet, mit unserem voraus.pioneer gekoppelt – dank offener Schnittstellen war dies problemlos möglich. So können automatisch in die Simulation der Applikation verschiedenste Roboter geladen und getestet (Erreichbarkeit, Störkonturen, Taktzeit etc.) werden (siehe Bild 7). Der dabei entstandene Applikationscode kann nach Auswahl direkt in der realen Anlage weiterverwendet werden. Eine Neuprogrammierung ist nicht erforderlich. Der Aufwand für die Auswahl wurde auf wenige Stunden statt mehrerer Tage reduziert.

Bild 7: autonox Finder und voraus.pioneer: optimale Auswahl von Kinematiken einfach gemacht

6 Fazit

Schnelle Entwicklungszyklen, wie sie vermehrt auch in industriellen Automatisierungsanwendungen gefordert werden, stellen derzeit eine große Herausforderung dar. Neben der Verfügbarkeit von Experten liegt die wesentliche Ursache in proprietären Komponenten verschiedenster Lieferanten, deren Software und Entwicklungswerkzeuge nicht kompatibel sind. Die moderne Softwareentwicklung zeigt hingegen mit dem DevOps-Ansatz, dass schnelle Updatezyklen und hohe Qualität kein Widerspruch sein müssen. Um Industrial DevOps zu ermöglichen, müssen zwei Voraussetzungen geben sein: Programmierung aller Komponenten in Hochsprache und Virtualisierung ebendieser Komponenten für Entwicklung und Test bei Beibehaltung der API.

Sind diese Bedingungen erfüllt, so steht eine durchgängige, moderne und leistungsstarke Toolchain (ohne Vendor-Lock) zur Verfügung und schnelle Entwicklungszyklen bei garantierter Qualität werden im OT-Bereich endlich Realität. Anwender von Automatisierungslösungen erhalten ihre digitale und technologische Souveränität – gerade in Zeiten geopolitischer Unsicherheiten und fragiler Lieferketten ein wichtiges Merkmal. Die mit Industrial DevOps geschaffene Durchgängigkeit der Daten über alle Schichten der Automatisierungspyramide ist des Weiteren Voraussetzung, Automatisierungslösungen effizient in bestehende IT-Infrastruktur einzubinden und um Verfahren der künstlichen Intelligenz gewinnbringend umzusetzen.

7. Unternehmensinformationen und Kontaktdaten

Weitere Informationen zu unserer Industrial DevOps Lösung und wie wir Sie bei der Umsetzung unterstützen können finden Sie auf unserer Homepage www.vorausrobotik.com. Unsere Dokumentation mit vielen Beispielen gibt es hier: https://docs.vorausrobotik.com/ – jetzt auch mit KI-Suche.

Unsere Kontaktdaten

voraus robotik GmbH | Carl-Buderus-Str. 7 | 30455 Hannover | **voraus**robotik.com

Ansprechperson

Tobias Ortmaier | tobias.ortmaier@vorausrobotik.com