TECHNISCHE UNIVERSITÄT DRESDEN

CIMTT

Diplomarbeit

Abstraktion verteilter Produktionsmaschinen in cyber-physischen Produktionssystemen

Betreuer

Dipl.-Medieninf. Gordon Lemme,
Dr.-Ing. Sebastian Götz

21. September 2016

Verantwortl. Hochschullehrer

Peter Heisig
Matr.-Nr.: 3521226

Prof. Dr. Uwe Aßmann

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Abstraktion verteilter Produktionsmaschinen in cyber-physischen Produktionssystemen

unter Angabe aller Zitate und nur unter Verwendung der angegebenen Literatur und Hilfsmittel selbstständig angefertigt habe.

Dresden, den 21. September 2016

Peter Heisig

Inhaltsverzeichnis

1 Einleitung

Seit der Mitte des 18. Jahrhunderts ist die industrielle Fertigung in stetigem Wandel. Mit der Entwicklung dampfgetriebener Arbeits- und Kraftmaschinen um 1750 wurde die erste industrielle Revolution eingeleitet. Zum Ende des 19. Jahrhunderts ermöglichte die Einführung arbeitsteiliger Massenproduktion und wissenschaftlicher Betriebsführung das erste Transportband in der fleischverarbeitenden Industrie. Knapp einhundert Jahre nach dieser zweiten Revolution wurden 1969 erste speicherprogrammierbare Steuerungen (SPS) zur variantenreichen Serienproduktion eingesetzt. Informations- und Kommunikationstechnologie sind seither der Grundstein automatisierungsgetriebener Rationalisierungen. Im Jahr 2011 wurde ausgehend von Lean Production der Begriff Industrie 4.0 geprägt, der die vierte Industrielle Revolution beschreibt (vgl. zu diesem Abschnitt [Gau+14]). Sie zeichnet sich durch neue Ansätze wie das Internet of Things (IoT) und cyber-physische Systeme (CPS) im Kontext industrieller Produktion aus [SG16].

1.1 Motivation

Vor der vierten Revolution war klassische Produktionssteuerung zentralisiert und Steuerungstechnik monolithisch strukturiert. Zukünftig wird die Fertigung in cyberphysische Systeme von Systemen zerlegt und mit offenen Standards dezentral betrieben (vgl. zu diesem Absatz [Mil14]).
Moderne Produktionseinrichtungen beherbergen jedoch Maschinen jeden Alters, die zu einem gemeinsamen System verwachsen müssen. Die Technologie zur numerische Kontrolle von Werkzeugmaschinen existiert bereits seit den frühen 1950er Jahren [LHL04]. Gerade diese älteren Anlagen besitzen häufig keine Möglichkeit der Integration in die IT-Systeme einer künftigen Fertigungsstrecke [Wan+04]. Das schlichte Ersetzen dieser Altmaschinen ist aufgrund hoher Kosten meist keine Lösung [Fra16]. Jedoch behindern diese vorrangig die nahtlose Machine-To-Machine (M2M) Kommunikation durch fehlende Infrastrukturanbindung, womit die Kette von Bearbeitungsschritten für ein Produkt zahlreiche manuelle Eingriffe erfordert.
Vor einigen Jahren wurden bis zu 60% der Arbeitszeit eines Werkers auf die Übertragung des Entwurfs eines Fertigungsschrittes in die Umsetzung an der Maschine

verwendet [GML00]. So besitzt eine Altmaschine als Teil des Fertigungsprozesses keine Möglichkeit externer Kommunikation und kein Application Programming Interface (API) [DP11]. Bei jüngeren Konstruktionen treten Integrationsschwierigkeiten an anderer Stelle auf. So sind selbst bei bestehender Netzwerkfähigkeit geschlossene Soft- und Hardwarearchitekturen und fehlende Schnittstellen verantwortlich für eingeschränkte Überwachung und Steuerung, respektive für die Verhinderung von ökonomisch sinnvoller Automatisierung (vgl. zu diesem Absatz [DP11; FC07]). Weiterhin erschweren die unzureichende Umsetzung von Industriestandards und -normen die Integration der Maschinen [Wan+04; Hop14].

Technische Komponenten, wie eine Netzwerkanbindung, sind nicht die einzigen Barrieren moderner Produktionsautomatisierung. Fehlerbehaftete Kommunikationsmechanismen, sowie die Gefahr der Veräußerung betriebsinterner Daten, sind Probleme die heute gelöst werden können. Auch erfordern sinkende Losgrößen und steigende Produktvariabilität eine flexible Automatisierung von Echtzeitüberwachung und -kontrolle verteilter, rekonfigurierbarer Fertigungssysteme (vgl. zu diesem Absatz [Wan+04; LS01]).
Produktionseinrichtungen basierten bisher auf dem manuellen Sammeln und Verteilen von Daten für Überwachung, Steuerung und Wartung der Maschinen. Doch gegenüber hohen Kosten, menschlichen Fehlern, dem teilweise schlechten Zugang zur Anlage und Aspekten der Datensicherheit, sind Automatisierungslösungen heute günstig, sicher und attraktiv für die Fertigungsindustrie (vgl. zu diesem Absatz [DP11]).

1.2 Zielsetzung

Nach der Motivation und der damit einhergehenden Identifikation des Kernproblems, werden nun die Ziele dieser Arbeit beschrieben. Den Schwierigkeiten in der industriellen Praxis wird wie folgt begegnet:

- Durch die entfernte Kontrolle einer Altmaschine werden manuelle Tätigkeiten wie das Übertragen eines Maschinenprogramms gemindert. Der operative Einsatz einer vormals nicht integrierten Anlage kann damit stärker automatisiert werden und beschleunigt den übergeordneten Produktionsablauf.

- Die zentrale Auswertung von Prozessdaten ermöglicht einen gesamtheitlichen Einblick in die Produktion, wobei jene Daten nicht notwendigerweise zentral zu persistieren sind. Diagnosen geschehen damit nicht mehr vor Ort, wodurch Wartungszyklen besser überprüft und eingehalten werden können. In der Konsequent wird außerdem die Planung der Fertigung vereinfacht und die Zeit bis zur Produktion gesenkt. Weiterhin sollen Störfälle wie Werkzeugbruch und -wechsel ad hoc an Verantwortliche kommuniziert werden.

Im Kontext dieser Arbeit gilt eine Anlage als integriert, wenn die infrastrukturelle Einbettung in ein cyber-physisches Gesamtsystem den Anforderungen (vgl. Kapitel 3) genügt. Neben den praktisch orientierten Vorgaben wird die Forschung zur Anlagenmodernisierung für die Industrie 4.0 durch weitere Ziele unterstützt:

- Eine dezentrale Informations- und Kommunikationsarchitektur verbessert die Resilienz, Produktionsstabilität und Skalierbarkeit von verteilten Fertigungssystemen.
- Kommunikationskanäle zwischen einzelnen Maschinen werden aufgrund durchgängig offener Schnittstellen nicht mehr unterbrochen. Durch damit einheitlich mögliche Machine-To-Machine (M2M) Kommunikation wird die Kontrolle und Überwachung hierarchisiert und dezentralisiert.
- Die Modellierung von Komponenten und Funktionalität einer Maschine wird durch Standardentwicklungswerkzeuge und -austauschformate vereinfacht.
- Das Optimierungspotential der Gesamtanlage kann durch statistische Auswertung der anfallenden Daten zu Maschinenoperation und -auslastung ausgeschöpft werden.

Die Hierarchisierung von Kontrolle und Überwachung bezieht sich auf das Beispiel der flexiblen Fertigungszelle in denen ein Verbund von Maschinen eine gemeinsame Aufgabe bearbeitet (vgl. [Gro08]). Innerhalb eines solchen Verbunds wird zunehmend über Ethernet kommuniziert, wodurch eine Basis für die TCP/IP Protokollfamilie zur Verfügung steht.

> „Aktuell sind nach einer Studie der Fachhochschule Südwestfalen 86% der SPS-Systeme über Ethernet angebunden, wobei von den verbleibenden 14% der Befragten 6% angaben, Ethernet wahrscheinlich in Zukunft einzusetzen.“ [WJN15][1]

[1] ursprüngliche Quelle: *M. Rothhöft, „Marktstudie: Industrielle Kommunikation,“, VDMA, 2013.*

Um nun die Ziele im Rahmen dieser Arbeit effektiv erreichen zu können, unterliegen Konzept und Implementierung verschiedenen Einschränkungen und Voraussetzungen.

Eine Ethernet-basierte Netzwerkinfrastruktur erlaubt das Einbinden eines virtuellen Maschinenabbilds in die Fertigungsstrecke. Zugang zur Anlage, regelungstechnische Modifikationen und das Anbringen von Sensorik und Aktuatoren sind gegeben. Die zu modernisierende Werkzeugmaschine wird durch rechnergestützte numerische Steuerung (CNC) kontrolliert. Automatisierte Werkzeugkomponenten, wie Einspannvorrichtungen oder Schutztüren, sind an eine Speicherprogrammierbare Steuerung (SPS) gekoppelt. Einplatinencomputer sind ausreichend leistungsfähig für die Steuerung und Überwachung von CNC-Maschinen [GM16].

Auch ist das vorgestellte Konzept der Anlagenmodernisierung auf diskrete Fertigung mit bestehender Netzwerkinfrastruktur beschränkt. Unter Berücksichtigung der besprochenen Ziele und Einschränkungen, wird eine konzeptuelle und prototypische Lösung durch die folgenden Schritte erreicht.

1. Ermitteln der Anforderungen für eine Integration von Altmaschinen in moderne, verteilte Produktionsumgebungen – im Folgenden als Retrofitting bezeichnet.
2. Recherchen zum heutigen Stand der Technik und die Einbeziehung vorhandener Systeme.
3. Konzeption einer virtuellen Repräsentation als Schnittstelle der zu integrierenden Anlage.
    - Einsatz von Einplatinencomputern als Integrationsequipment
4. Ermöglichen von dezentraler Kontrolle und Überwachung im Hinblick auf cyber-physische Produktionssysteme.
    - Transfer und Ausführung von Maschinenprogrammen.
    - Erfassen von Produktionsdaten durch angeschlossene Sensoren.
    - adaptive Reaktion auf Zustandsänderung durch Rückkopplung.
5. Vorstellung eines skalierenden, erweiterbaren Frameworks für die softwaretechnologische Seite der virtuellen Repräsentation.
6. Eine prototypische Implementierung belegt die prinzipielle Durchführbarkeit.

Retrofitting ist nicht nur die Integration von Altmaschinen. Im Rahmen dieser Arbeit gilt die Definition von Bergweiler, nach der Retrofitting die Erweiterung des Equipments einer Anlage durch zusätzliche Hardware meint. Der funktionale Umfang einer Maschine wird durch neue Module für die Übertragung und verteilte Verarbeitung der Daten ausgebaut. Dadurch wird die Kommunikation zwischen individuellen Geräten und Produkten der Fertigung ermöglicht, bis die Fabrik den künftigen Standards, Direktiven und Prinzipien der Industrie 4.0 genügt [Ber15]. In Verbindung mit den Zielen muss die zusätzliche Hardware, für die Erweiterung des Equipments, in Einplatinencomputern bestehen.

Nach Klärung der Ziele, Beschränkung des Konzepts und dem Aufzeigen eines groben Lösungswegs werden in dieser Arbeit folgende Fragen zu beantworten sein.

> Welchen softwaretechnologischen Konzepten muss die Modernisierung und der infrastrukturelle Kontext einer Altmaschine unterliegen, um eine ganzheitliche Integration in cyber-physische Produktionssysteme (CPPS) gewährleisten zu können?

1. Welche System- und Softwarearchitektur ist für ein flexibles Retrofitting zur Steuerung und Überwachung veralteter Fertigungsanlagen im Kontext von CPPS geeignet?
2. Wie und wo werden Informationen zu Maschinenzustand und -operation erfasst, verarbeitet, persistiert und Fremdsystemen zur Verfügung gestellt?
3. Welche standardisierten Protokolle und Datenstrukturen eignen sich für M2M-Kommunikation in einem CPPS?

1.3 Methode und Aufbau

Angelehnt an die Design Science Research Methodology wurden bisher grundlegende Probleme identifiziert und die Arbeit motiviert [Gee11]. Durch die folgenden Grundlagen (Kapitel 2) werden essentielle Technologien und Konzepte beschrieben. Die sich anschließenden Anforderungen (Kapitel 3) spezifizieren die Zielvorgaben der darauf entwickelten Lösungskonzepte (Kapitel 4) für die Abstraktion von Maschinen in cyber-physischen Produktionssystemen. Durch die prototypische Implementation (Kapitel 5) des Frameworks und das virtuelle Maschinenabbild wird die

prinzipielle Durchführbarkeit des Vorhabens belegt. Die Evaluation (Kapitel 6) hat eine qualitative und quantitative Bewertung von Konzept und Implementation des Prototyps zum Ziel. Schlussendlich werden in der Zusammenfassung ein Fazit und Ausblick (Kapitel 7) auf weitere Forschung gegeben.

2 Grundlagen

Für diese Arbeit relevante Technologien und Konzepte werden in folgendem Kapitel erläutert. Grundlegende Eigenschaften von Fertigung und Automatisierung sind die Basis für das Verständnis von semantischen Informationsmodellen, respektive dem virtuellen Abbild der Realität. Durch dieses Modell ist ein System in der Lage die Produktion und deren Schritte zu überwachen, Differenzen zu erwartetem Verhalten festzustellen und autonom darauf zu reagieren.

2.1 Fertigung und Automatisierung

Fertigung, als Unterbegriff der Produktion, beschreibt Verfahren zur Umwandlung oder Erzeugung von Stoffen mit Hilfe von Energie und Informationen innerhalb eines Prozesses [Lin15]. Automatisierung, nach der DIN 19233, ist „das Ausrüsten einer Einrichtung, so dass sie ganz oder teilweise ohne Mitwirkung des Menschen bestimmungsgemäß arbeitet“. Eine Verknüpfung dieser beiden Konzepte ist in Abbildung 2.1 dargestellt. Die Rückkopplung von Prozessdaten in eine Automatisierungseinrichtung befähigt das System, unter Berücksichtigung von Zielen, steuernd auf die Fertigung zu wirken. Direkte Eingriffe in den Prozess, sowie dessen Beobachtung durch den Menschen, werden verringert - im ökonomischen Zusammenhang rationalisiert (vgl. zu diesem Absatz [Lin15]).

Um die verschiedenen Bereiche der klassischen[1] Automatisierung darzustellen, wird eine Schichtenorganisation herangezogen. Die Struktur eines Unternehmens entspricht damit einer Automatisierungspyramide. Unterschieden werden diese Ebenen aufgrund der unterschiedlichen Anforderungen an Datendurchsatz und Übertragungsgeschwindigkeit (vgl. zu diesem Absatz [Lin15]). Wird eine neue Komponente in diese Pyramide integriert, geschieht dies entweder horizontal oder vertikal. Ersteres bedeutet die Verbindung der Komponente mit Geräten gleicher Ebene. Ist sie vertikal integriert, verbindet sie Komponenten unterschiedlicher Ebenen. Die Ebenen des Beispiels der Abbildung 2.2 erläuterte Linke wie folgt:

[1] Automatisierung vor der vierten industriellen Revolution.

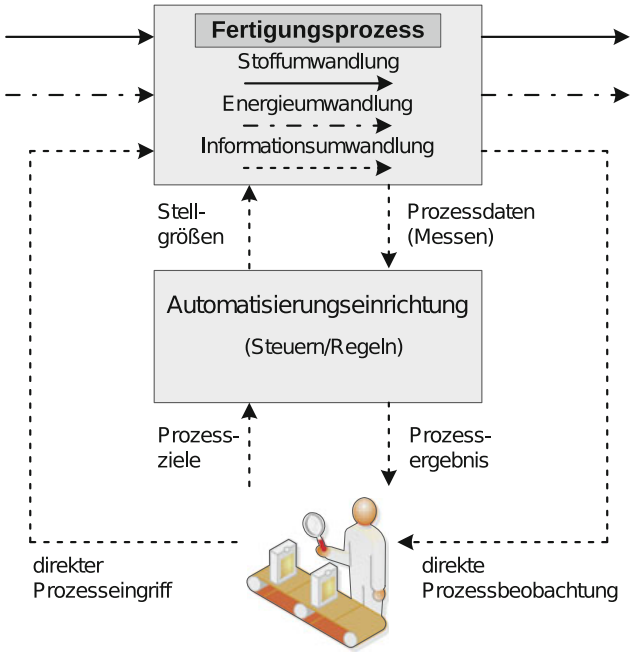


Abbildung 2.1: Automatisierte Fertigung aus [Lin15]

- **Unternehmensleitebene**
  Marktrelevante Unternehmensprozesse, wie Unternehmensführung, Personal-, Investitions- und Produktionsplanung, finden unter Zuhilfenahme eines Systems für Enterprise Resource Planning (ERP) befinden sich auf dieser Ebene.
- **Betriebsleitebene**
  Prozesse wie Auftragsbearbeitung und Produktionsplanung, Terminüberwachung und Kostenanalyse sichern auf dieser Ebene den täglichen Fertigungsbetrieb eines Unternehmens. Die softwareseitige Unterstützung übernimmt beispielsweise ein Manufacturing Execution System (MES).
- **Prozessleitebene**
  Je nach Größe der Anlage wird in detaillierteren Ebenen die Steuerung und Regelung von Produktionsprozessen und deren Überwachung vollzogen. Verbindungen, wie die zwischen einzelnen Fertigungszellen, werden auf dieser Ebene abgebildet.

- **Steuerungs- & Feldebene**
  Aktuatoren und Sensoren und deren Schnittstelle zu Ein- und Ausgangssignalen befinden sich auf der Feldebene. Die von diesen Komponenten konsumierten, beziehungsweise erfassten Daten, werden auf der Steuerungsebene mit dem Prozess verbunden. Daraus entstehende Information werden in die Reaktion des Systems einbezogen, wodurch minimale Latenzen in der Übertragung notwendig sind.

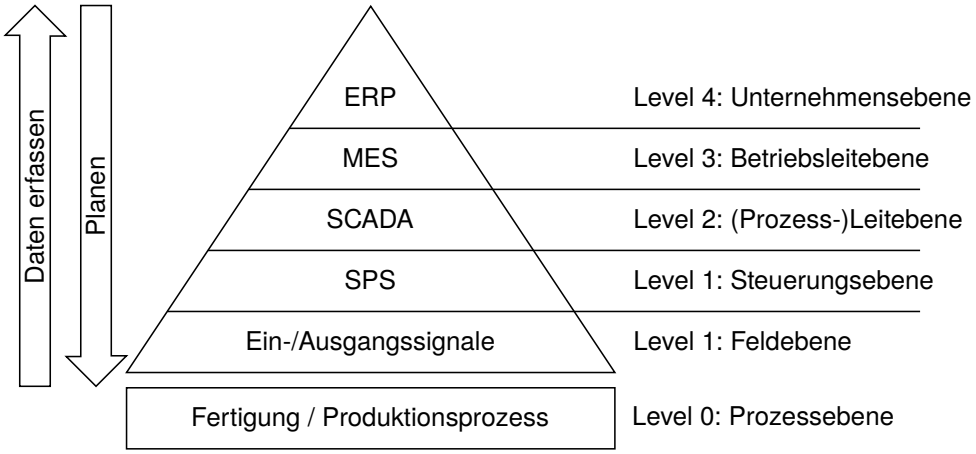


Abbildung 2.2: Beispiel einer klassischen Automatisierungspyramide[1]

Auf der Prozessebene (Level 0) des Beispiels geschieht die physikalische Fertigung des Produkts. Dabei fallen große Mengen von Rohdaten an, die in jeder folgenden, höheren Schicht zu abstrakteren Informationen verarbeitet werden. Die Toleranz bezüglich der Übertragungsgeschwindigkeit ist auf diesen unteren Ebenen am geringsten. Speicherprogrammierbare Steuerungen (SPS, vgl. Abschnitt 2.1.3) und numerische Kontrolle (NC, vgl. Abschnitt 2.1.2) erhalten Befehle in Echtzeit und automatisieren den Produktionsablauf. Über einen Feldbus (vgl. Abschnitt 2.1.1) werden diese Instruktionen und Messdaten an ein Supervisory Control and Data Acquisition (SCADA) System gekoppelt. Derlei Systeme sind verantwortlich für die Überwachung und Steuerung technischer Prozesse und kontrollieren die übergeordnete Fertigungszelle, respektive Verbünde von Werkzeugmaschinen, Robotern und automatisierten Komponenten [Lin15]. Ein MES, beziehungsweise Fertigungsmanagementsystem bildet dann die Schnittstelle zum Ressourcenmanagementsystem (ERP) der Unternehmensleitebene.

[1] Darstellung durch Wikipedia-Nutzer UlrichAAB.

2.1.1 Kommunikationssysteme

In der industriellen Fertigung werden Feldbusse als Kommunikationskanal in Feld- und Steuerungsebene genutzt. Neben dem Feldbus existieren weitere, teils veraltete Kommunikationskanäle, die zu der in Abbildung 2.3 dargestellten, heterogenen Automationsstruktur führen.

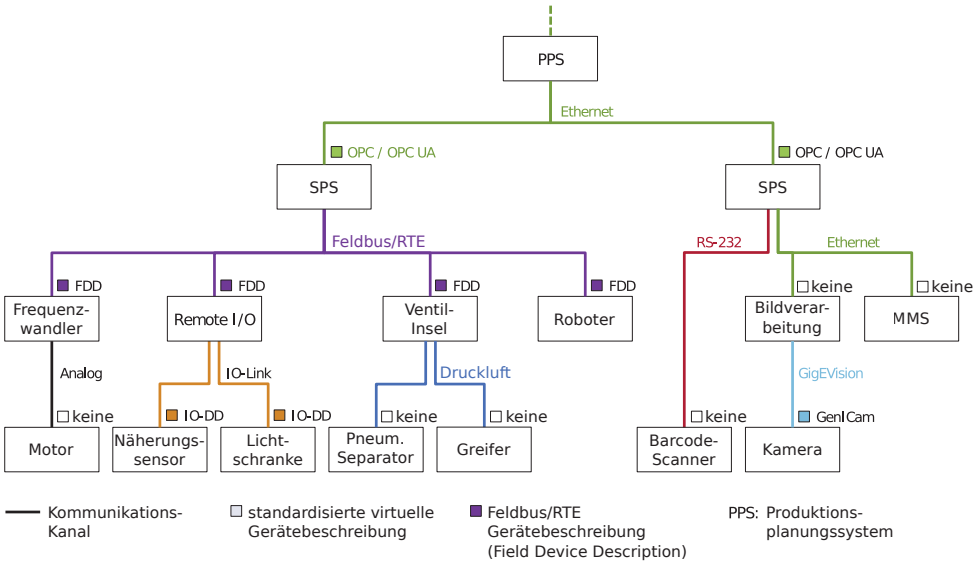


Abbildung 2.3: Repräsentative Automationsstruktur nach [HR15]

Hammertringl und Reinhart fanden vier Klassen von Kommunikationssystemen in dieser durch eine Befragung ermittelten Struktur. Sensoren und Aktuatoren der ersten Klasse sind direkt verbunden und übermitteln binäre oder analoge Informationen rein physikalischer Natur, wie Strom mit 24V oder Druckluft. Sie stellen keine standardisierte, digitale Beschreibung ihrer Funktion bereit, wodurch dem angeschlossenen Gerät diese händisch mitgeteilt werden muss.
Feldgeräte mit IO-Link[1]-Fähigkeit können durch die IO Device Description (IO-DD) beschrieben werden und sind Teil einer zweiten Klasse mit Direktverbindung. Innerhalb dieser können minimale Protokolle beschrieben und die Identifikation der Komponenten durchgeführt werden. Dadurch kann innerhalb dieser Klasse extern parametriert und eine maschinenlesbare Beschreibung übertragen werden. Letzteres wird in der Praxis jedoch kaum genutzt. Ein weiterer, stark verbreiteter Repräsentant ist die serielle Schnittstelle RS-232.
Die dritte Klasse von Kommunikationssystemen verbindet Bus-basierte Geräte. Traditionelle Bussysteme und Real-Time Ethernet (RTE) sind Stand der Technik und erlauben die Definition der Strukturen durch einen Bus-Master. RTE ist hierbei

[1] Implementierung der IEC TR 61131-9

ein Überbegriff verschiedener Netzwerkstandards wie Profinet IO, EtherNet/IP und EtherCAT [Dür+14]. Bildverarbeitungssysteme, ihre Protokolle (z.B. GigE Vision[1]) und Beschreibungssprachen (z.B. GenICam[2]) sind hier verbreitet. Für Konfiguration und Überwachung der Systeme wird eine Mensch-Maschine-Schnittstelle (MMS) eingesetzt (vgl. zu diesem Absatz [HR15]).
Feldbusse sind digitale bidirektionale, serielle Kommunikationsnetzwerke für echtzeitfähige, verteilte Kontrolle von Instrumenten, Steuerungseinheiten und Aktuatoren [Mah03]. Trotz der Standardisierungsbemühungen durch IEC 61158, existieren unterschiedliche Feldbusse wie CAN, ProfiBUS oder EtherCAT. Jeder Hersteller von Maschinen, Robotern und automatisierten Komponenten stellt einen anderen Busstandard, weshalb die Kommunikation der Geräte nicht garantiert werden kann. Für deren Verbindung mit unterschiedlichen Systemen wird ein Adapter benötigt, wodurch der Aufwand bezüglich Bereitstellung und Konfiguration steigt (vgl. zu diesem Absatz [Pau+13]).

Auf den höheren Ebenen der Automatisierungspyramide (vgl. Abbildung 2.2) etablierte sich das nicht Echtzeit-fähige Ethernet. Die Variante des RTE verbreitet sich jedoch zunehmend auch auf den unteren Ebenen (vgl. Abschnitt 1.2) und erlaubt Kommunikation mit Remote Procedure Calls (RPC), TCP/IP-Sockets und OPC (ursprünglich OLE[3] for Process Control, vgl. Abschnitt 2.2) [Pau+13]. Die Homogenisierung der Infrastruktur, vom Ressourcenmanagement im ERP über die Speicherprogrammierbare Steuerung bis zum einzelnen Sensor auf der Feldebene, vereinfacht den Informationsaustausch und trägt zur Flexibilisierung des Gesamtsystems bei. Weiterhin stehen damit die Daten aller Schichten für jeden anderen Netzwerkteilnehmer zur Verfügung.
Diese Form der Kommunikations- und Informationsstruktur ist nach Hammerstingl und Reinhart in einer vierten Klasse zu finden. Abbildung 2.3 zeigt OPC und OPC UA (vgl. Abschnitt 2.2) als Standard für den Datenaustauch zwischen dem Produktionsplanungssystem (PPS) und den speicherprogrammierbaren Steuerungen (SPS, vgl. Abschnitt 2.1.3). Mit dieser Technologie stellen Geräte aktiv ihre virtuelle Beschreibung bereit, was durch die Hersteller unterstützt und vorangetrieben wird (vgl. zu diesem Absatz [HR15]). Auch mit numerischer Steuerung (CNC, vgl. Abschnitt 2.1.2) kontrollierte Werkzeugmaschinen können so Informationen zu Zustand und Produktionsfortschritt bereitstellen.

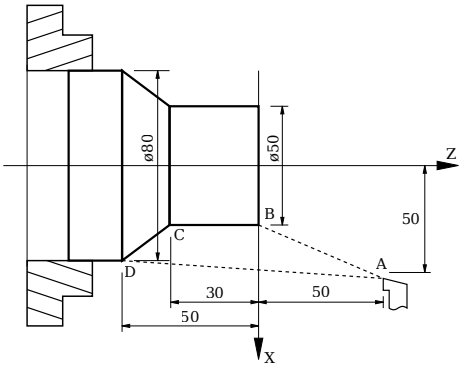
[1] Schnittstellenstandard industrieller Bildverarbeitung
[2] Schnittstellenstandard für industriell eingesetzte Kameras
[3] Object Linking and Embedding

2.1.2 Numerische Kontrolle

Für die Fertigung eines Produkts werden Bauteile benötigt, die durch Werkzeugmaschinen entstehen. In der DIN 69651 ist eine Werkzeugmaschine definiert als eine „mechanisierte und mehr oder weniger automatisierte Fertigungseinrichtung, die durch relative Bewegung zwischen Werkstück und Werkzeug eine vorgegebene Form am Werkstück oder eine Veränderung einer vorgegebenen Form an einem Werkstück erzeugt“ (Zitat aus [Hir00]). Ein Werkstück ist dabei ein Rohling eines bestimmten Materials, welcher auf eine bestimmte Art durch Bearbeitung verändert wird. Die aus diesem entstehende Form wird mit Software für Computer-Aided Design (CAD) entworfen, wobei eine zwei- oder dreidimensionale Visualisierung des Modells den Konstrukteur unterstützt. In einem zweiten Schritt werden die so entstandenen Konstruktionspläne in Bewegungsmuster umgewandelt. Der etablierte Kodierungsstandard für die Steuerungsinformationen zu diesen Mustern ist durch die DIN 66025, beziehungsweise ISO 6983 normiert und als G-Code bekannt. Eine vollständige Kompatibilität der Befehle zwischen den Anlagen wird aufgrund spezifischer Werkzeug- und Maschinenparameter, wie Drehzahlen oder Begrenzungskoordinaten der Arbeitsfläche, verhindert. Durch Präprozessoren und manuelle Anpassungen werden Steuerungsinformationen des zweiten Schritts auf die Maschine angepasst.
Wurde der vorverarbeitete G-Code auf eine Werkzeugmaschine mit Computer Numerical Control (CNC) übertragen, kann diese mit dem eigentlichen Fertigungsschritt beginnen. Derlei Maschinen besitzen mehrere durch Schrittmotoren und Servos angetriebene Bearbeitungsachsen, welche die Position des Werkzeugs relativ zum Werkstück durch Translation und Rotation (Hilfsachsen) verändern. Mit dieser relativen Bewegung wird sukzessiv Material entfernt, wodurch die im CAD entworfenen Bauteile physisch entstehen. Für das Entfernen von Material werden verschiedene Typen von Werkzeugmaschinen eingesetzt. Drehmaschinen und Fräsen sind hier die prominentesten Repräsentanten, wobei zum Beispiel auch spezielle Roboter mit Befehlen der CNC gesteuert werden können (vgl. zu diesem Absatz [Hir00]).

G-Code teilt sich in zwei Gruppen von Instruktionen, die wiederum herstellerspezifisch beziehungsweise generisch sind. Abbildung 2.4 zeigt das Beispiel einer Konstruktion und dessen Programm mit Bewegungsinstruktionen (G-Befehle) und sonstige Anweisungen (M-Befehle) für die Herstellung auf einer Drehbank[1]. Das Einspannen des Rohlings durch `M12` schließt die Klammern - in der Konstruktion als schraffierte Blöcke dargestellt. Ein Eilgang, wie durch `G0 X100 Z50`, richtet das

[1] nach www.helmancnc.com/cnc-lathe-simple-g-code-example-g-code-programming-for-beginners (abgerufen am 7.10.2016)




```
M12               ; Werkstück einspannen
G0 X100 Z50       ; Eilgang zu Punkt A
M3 S600           ; Starten der Spindel mit 600 U/min
M8                ; Kühlung/Schmierung einschalten
G1 X50 Z0   F600 ; Vorschub zu Punkt B mit 600 mm/min
   W-30     F200 ;          zu Punkt C mit 600 mm/min
   X80 W-20 F150 ;          zu Punkt D mit 150 mm/min
G0 X100 Z50       ; Eilgang zu Punkt A
M5                ; Stoppen der Spindel
M9                ; Kühlung/Schmierung ausschalten
M13               ; Einspannvorrichtung lösen
M30               ; Programmende
```

Abbildung 2.4: Beispielkonstruktion und G-Code für eine Drehbank

Werkzeug ohne das Entfernen von Material an einem bestimmten Punkt aus, wobei sich `A` im Beipiel an Position `(X = 100, Z = 50)` befindet. Der Befehl `G1` wird für den eigentlichen Fräsvorgang verwendet und lässt das Werkzeug mit einer Geschwindigkeit (Vorschub) von 600 mm/min durch das Material laufen.

Vielen CNC-Anlagen fehlt der Speicher für Programme mit mehreren tausend Bewegungsinstruktionen. Derlei Befehlslisten müssen während der Bearbeitung durch ein Fremdsystem an die Maschine sukzessiv übertragen werden. Das Konzept der Direct Numerical Control (DNC) steht für diese Verbindung via RS-232 oder Parallel Port. Fremdsysteme sind PCs oder dedizierte DNC-Transfergeräte, die den Code von Speichermedien wie USB-Sticks und SD-Karten beziehen. DNC, verstanden als Distributed Numerical Control, ermöglicht weiterhin die Verteilung von Programmen auf einen Maschinenverbund [Hir00].

Die Vorteile der Fertigung mit CNC liegen in der Wiederholbarkeit und Genauigkeit der Operation. Weiterhin wird die Rüstzeit, jene zum Einstellen der Maschine, verringert und damit die Produktivität erhöht (vgl. zu diesem Absatz [Smi08]).
Dennoch sind auch moderne CNC-Anlagen in ihrer Funktion limitiert, da der verwendete G-Code lediglich Instruktionen und prozedurale Daten abbilden kann, wodurch ein Großteil der Konstruktionsinformationen verloren geht. Zwei neue Standards, namentlich STEP-NC (ISO 10303-238) und Function Blocks (IEC 61499), etablieren sich aus diesem Grund. Durch diese sollen CNC-Maschinen mit mehr Informationen, für intelligentere Fertigung und bessere Interoperabilität, ausgestattet werden. Beispielsweise wird durch STEP-NC die Abhängigkeit von Parametern reduziert. Neuberechnungen zur Laufzeit beziehen unter anderem Verformungen durch Erhitzen des Werkstücks in die Fahrtenplanung ein. Function Blocks dagegen, sind Teil eines Standards für verteilte industrielle Prozesse und Kontrollsysteme. Sie kapseln Maschinendaten, wie Werkzeugeigenschaften oder Algorithmen, für CNC. (vgl. zu diesem Absatz [XLY06]).

Darüber hinaus besitzen Anlagen spezifische, automatisierte Maschinen- und Werkzeugkomponenten, die nur indirekt durch CNC steuerbar sind. Schließmechanismen, Abluftsysteme oder Materialzufuhr werden von maschinenspezifischen, internen speicherprogrammierbaren Steuerungen in die Fertigung integriert [Hir00].

2.1.3 Speicherprogrammierbare Steuerung

Daten und Zustände automatisierter Maschinen- Werkzeug- und Fertigungsprozesskomponenten werden durch speicherprogrammierbare Steuerungen (SPS) aufgenommen, verarbeitet und verändert. Durch der DINIEC 60050-351 7-2013 sind sie definiert als eine „rechnergestützte programmierte Steuerung, deren logischer Ablauf über eine Programmiereinrichtung, zum Beispiel ein Bedienfeld, einen Hilfsrechner oder ein tragbares Terminal, veränderbar ist“ (Zitat aus [HLG15]). Nach Heinrich et al. sind folgende Komponenten dafür notwendig. Ein Hardwaresystem stellte die Verbindung zum Fertigungsprozess und weiteren automatisierten Anlagen her. Programm- und Datenspeicher sind für die Persistenz des Anwenderprogramms, beziehungsweise der Prozessdaten verantwortlich und werden durch die Verarbeitung verändert. Dafür sind ein Betriebssystem sowie ein Anwenderprogramm zuständig.

Der prinzipielle Aufbau einer SPS umfasst Stromversorgungs-, Signalverarbeitungs- und vier Schnittstellenfunktionen für den Datenaustausch. Eine Mensch-Maschine-Schnittstelle ermöglicht dem Bediener den operativen Betrieb, z.B. durch Statusanzeigen, zu überwachen. Die Programmier- und Test-Schnittstelle wird durch einen Programmierer, neben der Implementierung von Instruktionen, auch zur Fehleranalyse genutzt. Kommunikationsfunktionen erlauben die Anbindung an externe Systeme, automatisierte Komponenten und Datenquellen. Aktuatoren und Sensoren werden über die Prozessschnittstelle an eine SPS gekoppelt. Der konzeptuelle Aufbau wird durch ein, in Abbildung 2.5 dargestellten, Hardwaresystem implementiert. Die Stromversorgungseinheit liefert die für den Betrieb notwendige Energie und kann an die jeweilige Quelle angepasst werden. Eine Zentraleinheit mit CPU, Speicher, Verarbeitungseinheit und Anschlüssen für Programmiergeräte und Netzwerk bildet die Steuerungslogik ab. Die Kopplung an digitale und analoge Datenquellen und -empfänger erfolgt über Ein-/Ausgabe-, beziehungsweise Signaleinheiten.

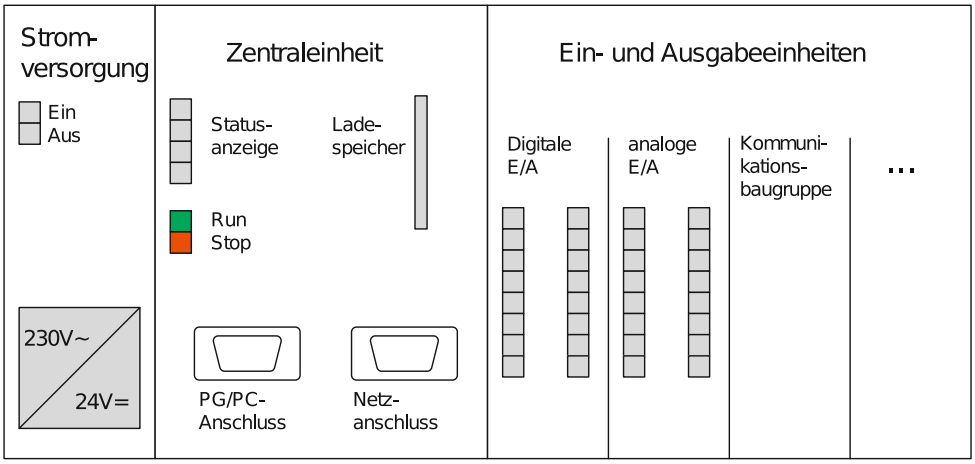


Abbildung 2.5: Grundlegender Hardwareaufbau einer SPS nach [HLG15]

Signalverarbeitungsfunktionen im Anwenderprogramm werden innerhalb der Verarbeitungseinheit zyklisch ausgeführt. In einem ersten Schritt werden dafür die aktuellen Zustände der Eingänge erfasst, z.B. Sensordaten, und der Datenspeicher gelesen. Das Anwenderprogramm wird nach dem EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe) abgearbeitet, woraufhin Ausgangssignale, z.B. Befehle für Aktuatoren, erzeugt werden. Diese Zyklen verbrauchen Zeit die kritische Aktionen verhindern können.
So existieren neben den zyklusorientierten auch unterbrechungsfähige SPS. Das aktuell laufende Programm kann dabei durch Interrupts pausiert und später wieder aufgenommen werden. Programmiert werden Anwenderprogramme mit normierten Methoden der EN 61131-3. In textueller Form existieren Standards für Anweisungslisten und strukturierten Text. Grafisch wird die Logik einer SPS durch Kontaktpläne, eine Funktionsbaustein- oder Ablaufsprache implementiert.
Eine weitere Kategorie bilden ereignisorientierte Steuerungen bei denen das Anwenderprogramm erst bei Statusveränderungen der Eingangssignale abgearbeitet wird. Weiterhin unterschieden werden modulare SPS, wie die *Simatic S7*-Serie von Siemens, und kompakt-SPS, wie zum Beispiel *LOGO!*. Letztere zeichnen sich durch fehlende Erweiterbarkeit aus. Neben hardwarebasierten SPS ermöglichen Software-Steuerungen (Soft-SPS) eine weitere Stufe der Flexibilisierung. Steuerungen mit Echtzeit-Betriebssystemen, auch in eingebetteten Recheneinheiten, übernehmen hier die Überwachung und Kontrolle des Prozesses, sind jedoch weniger verbreitet.
Eine Alternative zu SPS bietet die verbindungsprogrammierte Steuerung (VPS), bei der die Komponenten der Ein- und Ausgabe festverdrahtet und die Logik vordefi-

niert ist. Die speicherprogrammierbare Variante hat nicht nur den Vorteil der Flexibilität. Der Funktionsumfang, die Verarbeitung analoger und digitaler Daten sowie die geringen Betriebskosten etablierten die SPS als Standard in der industriellen Fertigungsautomatisierung (vgl. zu diesem Absatz [HLG15]).

Die EN 61131-3 ist nicht die einzige Möglichkeit der Programmierung einer SPS. Zugunsten vertikaler und horizontaler Integration hat die PLCopen[1], eine Vereinigung von Steuerungsherstellern, und die OPC-Foundation Funktionsbausteine in einer Spezifikation für OPC UA abgebildet. Damit kann die SPS eine aktive Rolle innerhalb der Automatisierungspyramide einnehmen und sich beispielsweise Produktionsaufträge eigenständig abholen (vgl. zu diesem Absatz [OPC14; PO10]).

2.2 OPC Unified Architecture

Der Austausch und die Modellierung von Daten kann in einem heterogenen Technologieraum nur durch standardisierte Beschreibungssprachen, Kommunikationsprotokolle und Modelle erreicht werden. Diese Aussage wird im Zusammenhang mit Industrie 4.0 durch eine Tendenzbefragung von BITKOM, VDMA und ZVEI aus dem Jahr 2013 gestützt. So sehen Mitarbeiter aus 278 Unternehmen, des Maschinen- und Anlagenbaus, Standardisierung als größte Herausforderung für die Umsetzung von Industrie 4.0 [KWH13].
Die OPC[2] Foundation ist ein Industriekonsortium, das für Entwicklung und Wartung solcher Standards verantwortlich ist. Sie schuf Interoperabilitätsstandards für den sicheren und zuverlässigen Austausch von Daten im Automatisierungsraum industriell produzierender Unternehmen auf Basis des Distributed Component Object Model (DCOM). Dieses ist ein von Microsoft definiertes System für entfernte Methodenaufrufe (Remote Procedure Calls, RPC) innerhalb eines Windows-Ökosystems, das für die heutigen heterogenen Informationssysteme ungeeignet ist. Neben der Plattformunabhängigkeit ist die Zusicherung des nahtlosen Übergangs von Informationen, zwischen Geräten unterschiedlicher Hersteller, die Hauptaufgabe querschnittlicher Spezifikationen im Kontext der OPC Unified Architecture (OPC UA)[3]. Das Konsortium berücksichtigte bei der Spezifikation folgende Ziele [OPC14].

- sicherer, zuverlässiger Informationsaustausch

[1] www.plcopen.org
[2] Open Platform Communications
[3] nach opcfoundation.org/about/what-is-opc (abgerufen am 23.09.2016)

- Plattform- und Herstellerunabhängigkeit
- standardisierte Kommunikation über Internet und Firewalls
- serviceorientierte Architektur (SOA)
- Schutz vor unerlaubtem Zugriff
- Erreichbarkeit und Zuverlässigkeit
- Vereinfachung durch Vereinheitlichung

Konkret bietet die OPC UA (auch IEC 62541) einen semantischen Kommunikations- und Datenmodellierungsstandard für den Informationsaustausch [Aya+13]. Ein erweiterbares Meta-Modell spezifiziert die Grundbausteine und Regeln für ein Informationsmodell und beinhaltet verschiedene Einstiegsknoten und Basis-Typen [OPC14]. Informationsmodelle sind Repräsentationen von Konzepten, Relationen, Beschränkungen, Regeln und Operationen zur Spezifikation der Bedeutung (Semantik) von Daten innerhalb einer bestimmten Domäne [Lee99]. Diese werden von Maschinen, Baugruppen und anderen Ressourcen im Adressraum angeboten, wodurch jede Entität innerhalb eines IT-Ökosystems mit der jeweilig anderen kommunizieren kann und deren strukturelle Eigenschaften kennt.

2.2.1 Informationsarchitektur

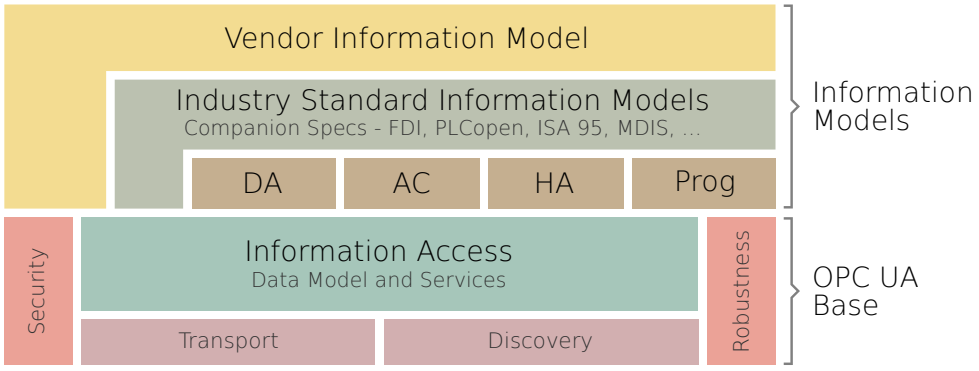


Abbildung 2.6: Spezifikationen von OPC UA

In Abbildung 2.6 ist die dafür notwendige Informationsstruktur dargestellt [OPC14]. Auf der untersten Ebene werden Transportprotokolle, das Meta-Modell und grundlegende Services beschrieben. Das bevorzugte Protokoll setzt auf TCP/IP auf und erlaubt einen performanten Austausch von beispielsweise Geräte-, Sensor-, Maschinen- und Prozessdaten innerhalb einer Client-/Server-Architektur. Eine für die Kommunikation über Firewalls taugliche Methode bietet die im Standard verankerte

HTTP/XML-Schnittstelle. Durch einen Discovery-Mechanismus können Funktionen und Eigenschaften von OPC-UA-fähigen Teilnehmern in einem Subnetz bekannt gegeben werden. Doch auch Dienste für Ereignisregistrierung oder Sitzungsmanagement sind Teil der elementaren Definitionen. Eckpfeiler dieser Basis von OPC UA sind Fehlertoleranz und Sicherheit als zentrale Aspekte der Spezifikation. Auf Details zur Sicherheitsinfrastruktur wird in dieser Arbeit nicht eingegangen. Darauf aufbauend werden generische Informationsmodelle definiert, die unter anderem den Adressraum eines Servers repräsentieren [OPC14]:

- **Data Access (DA)**
  Daten zur Automatisierung werden durch das Data Access Modell strukturiert. Sowohl analoge und diskrete Variablen, als auch Einheiten und Qualitätsattribute werden hier beschrieben. Sensoren, Steuerungen und Regler sind beispielsweise Quellen jener Variablen.
- **Alarms and Conditions (AC)**
  Die Behandlung von Dialogen und Alarmen wird mit diesem Modell definiert. Events, beziehungsweise Ereignisse auf die sich Clients registrieren, werden durch Veränderungen im Zustand eines Geräts ausgelöst.
- **Historical Access (HA)**
  Historische Variablenwerte und Events werden mit diesem Modell dargestellt. Die Persistenz dieser Daten ist konfigurierbar.
- **Programs**
  Komplexe Aufgaben werden durch Programme repräsentiert und mit Zustandsautomaten beschrieben.

2.2.2 Transportprotokolle

„Protokoll-Bindings“ erlauben den Transport der Daten zwischen einem OPC UA Client und Server durch unterschiedliche Mechanismen. Diese können parallel verwendet werden und arbeiten transparent, auch ohne Zutun des Entwicklers. Die Bereitstellung der drei Varianten kann ohne Anpassungen der Software verändert werden. Abbildung 2.7 stellt die Kommunikationsvarianten gegenüber und zeigt deren orthogonale Eingliederung.

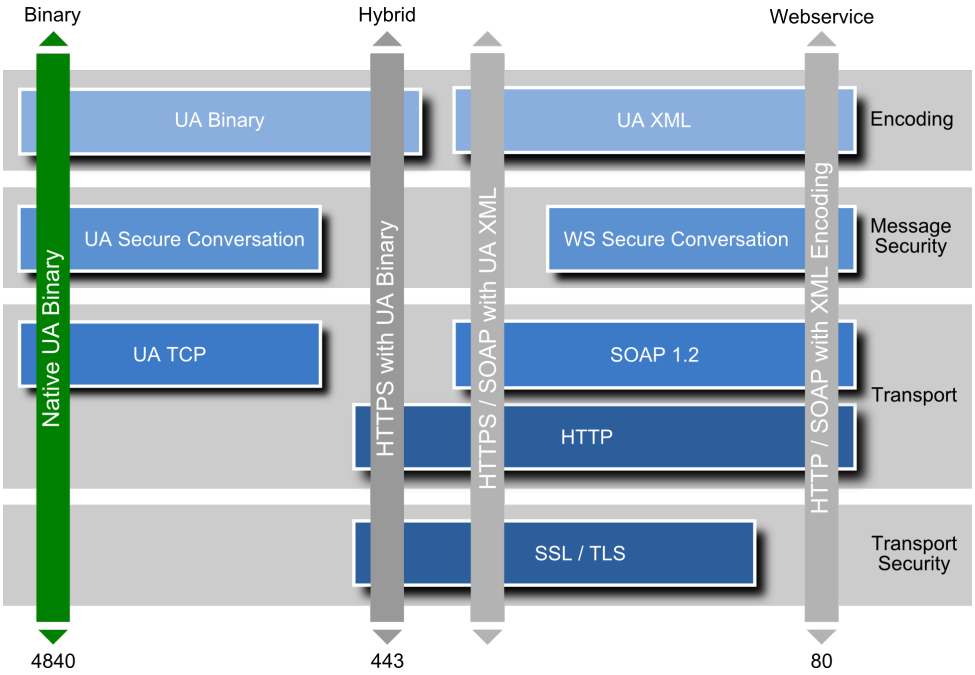


Abbildung 2.7: OPC UA Transportvarianten

Binärprotokoll (UA-Binary). Dieses vorgeschriebene Protokoll bietet die beste Performance und den geringsten Overhead. Da weder XML-Inhalte umgewandelt, noch SOAP[1] oder HTTP verwendet werden, ist es aufgrund des geringen Ressourcenverbrauchs für eingebettete Geräte geeignet. Außerdem bietet es die beste Interoperabilität, weil Binärdaten im Gegensatz zu XML-Dokumenten weniger Freiheitsgrade besitzen.

Webservice (XML-SOAP). Eine optionale Web-Service-Schnittstelle kann durch den OPC UA Server bereitgestellt werden. Der Overhead ist größer, da der Nachrichtenaustausch mit XML das Protokoll verlangsamt. Somit herrscht wenig bis keine Akzeptanz beim Einsatz mit kleinen und Kleinstgeräten. Dem gegenüber erfährt SOAP umfassende Werkzeugunterstützung und kann leicht durch .NET oder Java-Anwendungen verwendet werden. Weiterhin ist es durch die Verwendung von HTTP/HTTPS Firewall-freundlich.

[1] Simple Object Access Protocol: standardisierter Datenaustausch via XML, w3.org/TR/soap (abgerufen am 15.11.2016)

Hybrid (UA-Binary über HTTPS). Das Hybridprotokoll verbindet binären und SOAP-basierten Nachrichtenaustausch. Der Overhead ordnet sich zwischen den Verwendeten Kommunikationsmechanismen ein, wobei die die Vorteile beider vereinigt werden. Die binär kodierte Nachricht wird hierbei in den HTTPS-Frames versendet und ermöglicht eine Firewall-freundliche Kommunikation.

Dieser Abschnitt wurde der Website von ascolab[1] entnommen.

2.2.3 Modularität und Erweiterbarkeit

Viele bereits existierende Modelle, wie MTConnect, PLCopen, FDI und ISA95, unterscheiden sich von OPC UA durch fehlende Erweiterbarkeit. Semantische Zusammenhänge lassen sich oft nicht ohne weiteres darstellen, wie Hoppe schrieb [Hop14]:

- Wie geben sich z. B. ein „Temperatursensor“ oder eine „Ventilsteuerung“ zu erkennen?
- Welche Objekte, Methoden, Variablen und Ereignisse definieren die Schnittstelle für Konfiguration, Initialisierung, Diagnose und Laufzeit?

Die Erweiterbarkeit des Informationsmodells von OPC UA (vgl. Abbildung 2.6) ermöglicht Companion Specifications, die diesen Mangel ausgleichen und zusätzlich domänenspezifische Definitionen erlauben. Plattformunabhängigkeit wird durch frei verfügbare, aber auch proprietäre Implementierungen des Softwareinfrastruktur-Stacks ermöglicht. Ein API, Codegeneratoren für den Adressraum und Entwicklungswerkzeuge werden für die Programmiersprachen Ansi C/C++, .NET, Java und weitere durch die OPC Foundation bereitgestellt.

2.3 Cyber-physische Produktionssysteme

Die Verbindung von Überwachung und Kontrolle technischer Systeme mündet in Paradigmen, die Realität und virtuelle Umgebungen miteinander verschmelzen lassen. So wurde das Konzept cyber-physischer Systeme (CPS) 2006 durch Edward A. Lee erstmalig erläutert. Er versteht diese als Integration von Informationsverarbeitung und physischen Prozessen. Virtuelle und physische Abläufe werden durch

[1] ascolab.com/de/unified-architecture/protokolle.html (abgerufen am 11.11.2016)

Sensoren und Aktuatoren überwacht, beziehungsweise beeinflusst, stehen in unmittelbarer Wechselwirkung und sind durch Kontrollschleifen rückgekoppelt [Lee08]. Der historische Weg, hin zu darauf aufsetzenden Systemen, ist in Abbildung 2.8 dargestellt[1].

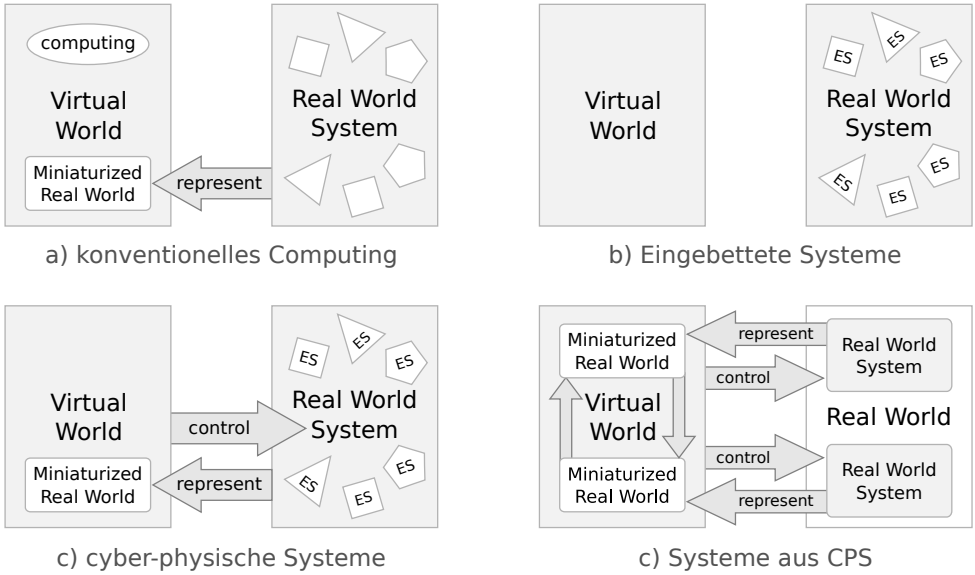


Abbildung 2.8: Der historische Weg zu CPSoS

Im konventionellen Computing (Abbildung 2.8a) sind Systeme der physischen Welt durch abstrakte Modelle repräsentiert. Berechnungen bezüglich der Realität werden in Simulationen auf diesen Modellen durchgeführt. Durch eingebettete Systeme (ES, Abbildung 2.8b) wird der Computer in das Realweltobjekt integriert, wodurch Berechnungen in die physikalischen Systeme getragen werden. Mit CPS (Abbildung 2.8c) existiert nicht nur ein passives Modell des Realweltsystems. Das Wissen um den Zustand des Realitätsausschnitts verhilft zur Steuerung der ES, wodurch neben dem realen Objekt ein synchrones, virtuelles Modellobjekt entsteht. Diese Konzept *dualer Realität* von Objekten steht für die Kontrolle von Dingen der physischen Welt. Um die Synchronität des Modells gewährleisten zu können, müssen Rückkopplungsschleifen die Effekte physischer Prozesse auf Berechnungen und Simulationen beziehungsweise vice versa verifizieren [Lee08]. Weiterhin sollen derlei Systeme autonom auf Diskrepanzen reagieren und korrigierende Maßnahmen einleiten.
MAPE-K ist ein geeignetes Konzept cyber-physischer Rückkopplung und besteht

[1] Abbildung 2.8 und folgender Absatz nach Vortrag *Life with Cyber-Physical Systems* von Prof. Dr. Uwe Aßmann, 29. Juni 2016, Technische Universität Dresden

aus den Phasen Monitor, Analyze, Plan und Execute. Eine übergeordnete Wissensbasis (Knowledge-Base) beinhaltet das physische Modell sowie Regeln und Ziele des verwalteten Elements, respektive der Fertigungsanlage. In Abbildung 2.9 sind die Phasen und ihre Beziehungen zueinander dargestellt.

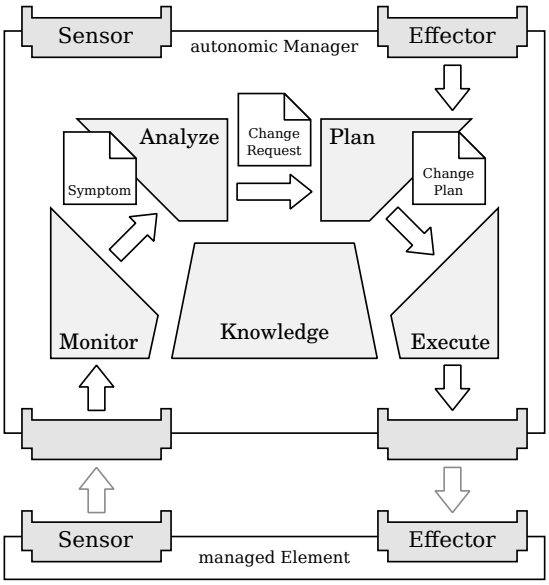


Abbildung 2.9: MAPE-K Kontrollschleife nach [IBM06]

Die Sensordaten werden vom Monitoring erfasst und im Kontextmodell der Wissensbasis abgelegt. Symptome, beziehungsweise Ereignisse die eine Analyse des Modells anstoßen, werden ermittelt. In der folgenden Analyse werden die Ziele der Rückkopplung im physischen Kontext überprüft. Ist ein Eingreifen in die Realität erforderlich um die Ziele zu erfüllen, wird ein Veränderungsanfrage (Change-Request) an die Planungskomponente von MAPE-K übergeben. Diese Phase mündet in einem Veränderungsplan (Change-Plan) der sich an Regeln und Zielen orientiert und mittels Execute über Aktuatoren ausgeführt wird (vgl. zu diesem Absatz [IBM06]).
In CPS werden außerdem semantische Kontrollregeln benötigt um die Rückkopplung mit MAPE-K zu ermöglichen. Die Regeln werden in der Kombination von Ereignis, Bedingung und Aktion festgehalten, besser bekannt als Event-Condition-Action (ECA, vgl. zu diesem Absatz [TGP08]). In der Monitor-Phase von MAPE-K werden die Ereignisse gesammelt. Die Analyse wertet die Bedingungen aus, woraufhin die Planung eine Aktion ermittelt, die von der Execute-Phase durchgeführt wird. So kann ein Ereignis beispielsweise die Veränderung eines Sensorwertes sein. In der Analyse ergibt sich das Überschreiten eines Schwellwertes. Die Planung sucht eine Aktion den Wert zu senken und überlässt die Ausführung der Execute-Phase. Das Konzept ist etabliert und bewerkstelligt die Rückkopplung in adaptiven, cyber-

physischen Systemen [WG14; Sei+16; SHS15].
In Systemen von CPS (CPSoS, Abbildung 2.8d) wird die physische Welt in Realweltsysteme gegliedert, die über ihre Modelle interagieren. CPSoS bieten Potential für die vierte industrielle Revolution und sind Grundlage cyber-physischer Produktionssysteme (CPPS). Produkte, Maschinen und andere Ressourcen werden in diesen durch CPS repräsentiert, welche Informationen und Dienste über das Netzwerk der gesamten Produktionsstrecke teilen. CPS sind fundamentale Elemente eines CPPS, die unmittelbaren Zugriff auf relevante Informationen, Maschinenparameter, Produktionsprozesse und deren Produkte besitzen. Durch die Dezentralisierung der Produktionslogik haben CPPS, im Gegensatz zu traditionellen Produktionssystemen, wesentliche Vorteile bezüglich Transparenz, Adaptivität, Ressourceneffizienz und Flexibilität. Auf Ebene der Automatisierung werden Informationen eines CPS-Netzwerk benötigt, um den Fertigungsprozess auf Basis von strategischen Entscheidungen erfolgreich durchzuführen. Für Entscheidungsfindung und Kontrolle der Fertigung werden konsistente, kohärente Informationen über die reale Welt benötigt [Ber15].

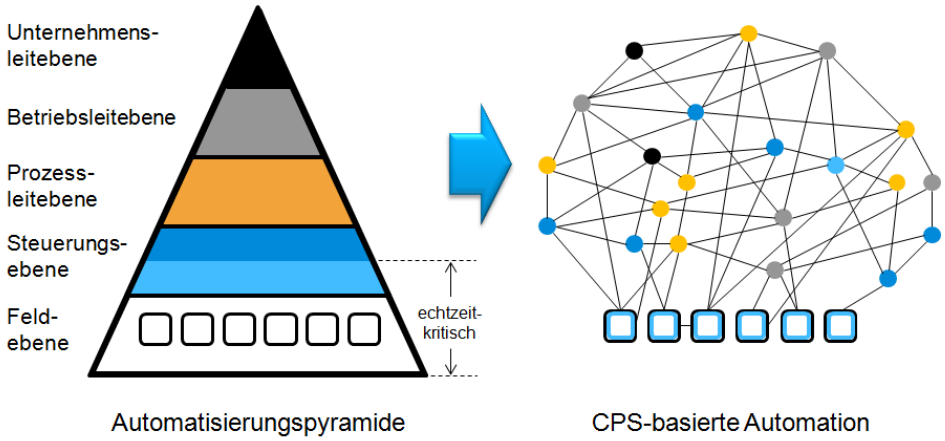


Abbildung 2.10: Auflösung der Automatisierungspyramide aus [Ver13]

Diese Informationen, Dienste und Funktionen werden an jener Stelle lokalisiert, die im Sinne einer flexiblen Entwicklung und Produktion den größten Vorteil bringt [Ver13]. Starre Strukturen, wie die der klassischen Automatisierungspyramide, sind ungeeignet für die dezentrale Verortung der genannten Ressourcen. Die demnach notwendige Auflösung dieser Architektur zu einem vernetzten System von Systemen, beziehungsweise CPSoS, wir in Abbildung 2.10 verdeutlicht. Sowohl Hard- und Software als auch die Verarbeitung der anfallenden Daten wird nicht länger in Schichten organisiert werden [SSG13]. Damit sollen die Produktionsressourcen

auf Knoten eines Netzwerks aufgeteilt und schrittweise auf ihre funktionale Struktur abstrahiert werden [Ver13]. Bis eine geeignete Architektur für CPPS andere Möglichkeiten bietet (vgl. [LBK15; Bor+14]), werden Echtzeit-Steuerungen in der Feldebene verbleiben [Ver13].

3 Anforderungen

Für die in Abschnitt 1.2 aufgestellten Ziele, werden in diesem Kapitel die spezifischen Kriterien zu deren Erfüllung erläutert.

Informationssysteme in der Produktion dienen der Verbesserung der Wettbewerbsfähigkeit und müssen Innovations- und Zeitdruck standhalten. Moderne Produktionsumgebungen helfen Arbeitsabläufe zu optimieren und vereinfachen Beteiligten die Ausführung ihrer Arbeit. Jedoch verhindern Altmaschinen aufgrund fehlender Infrastrukturanbindung und geschlossener Architekturen (vgl. [DP11]) die Vollautomatisierung dieser Arbeitsabläufe und erfordern die physische Anwesenheit einer Fachkraft [Wan+04].

3.1 Überwachung

Im Wartungs- und Störfall muss der Zustand der Anlage bekannt sein. Dieser kann bei nicht integrierten Altmaschinen nur an deren Terminal eingesehen werden. Ein Techniker muss die Betriebs- und Prozessdaten vor Ort erfassen um eine Diagnose stellen zu können und unter anderem das ERP-System darüber zu informieren. Im Sinne der Industrie 4.0 wird diese Form ortsgebundener Arbeitsplätze an Bedeutung verlieren und einer dezentralen Nutzungsschnittstelle weichen [GSL14]. Damit müssen die Daten über geeignete Schnittstellen zur Verfügung stehen. Subsysteme können dann auch automatisiert über Zustandsänderungen der Maschine in Kenntnis gesetzt werden. Weiterhin braucht ein CPPS diese Informationen um adäquat reagieren zu können (vgl. Abschnitt 2.3, [Lee08]). Abgesehen von der notwendigen Dezentralisierung und dem Informationsgewinn für Rückkopplungsschleifen gilt es einen Werkzeugbruch zugunsten von Maschinenverfügbarkeit und Produktqualität, respektive der Ökonomie des gesamten Produktionssystems, zu verhindern [Amb+15].

R1 Die Überwachung von Betriebs- und Prozessdaten der Altmaschine und ihrer automatisierten Maschinen- und Werkzeugkomponenten ist ortsunabhängig, so dass Zustandserfassung und Störfalldiagnose durch Subsysteme des CPPS erfolgen kann.

3.2 Steuerung

Um einen bestimmten Fertigungsschritt an einer numerisch kontrollierten (NC) Anlage durchzuführen, muss das auszuführende Programm nach DIN 66025 übertragen werden [Hir00]. Auch Speicherprogrammierbare Steuerungen (SPS) benötigen ein Anwenderprogramm nach EN 61131-3 oder der PLCopen-Spezifikation [HLG15; OPC14]. Diese Befehlsketten werden entweder mit einem Speichermedium auf den Steuerungs-PC kopiert oder direkt an dessen Terminal kodiert. Der zeitliche Aufwand und das notwendige Personal verlangsamen die Fertigung des Endprodukts und führen zu einer suboptimalen Fertigungsstrecke [Aya+13]. Für das Retrofitting der Anlage muss die entfernte numerische Kontrolle ermöglicht werden. Weiterhin sind Produktionsmaschinen mit zusätzlichen automatisierten Komponenten wie Schließmechanismen für Schutztüren, Kühl-, Entlüftungs- oder Einspannsystemen ausgestattet. Auch die Steuerung dieser muss ortsunabhängig sein, damit ein CPPS ganzheitlich in den Produktionsprozess eingreifen kann [GSL14].

R2 Die Steuerung der Altmaschine und ihrer automatisierten Maschinen- und Werkzeugkomponenten ist ortsunabhängig, so dass Übertragung, Ausführung und Abbruch von NC-Programmen, beziehungsweise produktionsbedingter Steuerbefehle, durch Subsysteme des CPPS erfolgen kann.

Die steigende Automatisierung zur Optimierung der Produktionsabläufe wird in einem CPPS durch Rückkopplung erreicht. Mit den Einhalten der Anforderungen zu Überwachung und Steuerung hat das System die Möglichkeit automatisch auf veränderte Bedingungen zu reagieren.

3.3 Standardisierung

Nach Ferrolho et al. entstehen auch mit Netzwerkanbindung und Programmierschnittstellen noch zu überwindende Probleme [FC07]. CNC-Maschinen basieren auf einer geschlossenen Architektur numerischer Kontrolle und sind nicht für die Integration mit anderen ausgelegt. Die Kontrolleinheit der Anlage lässt die Steuerung von einem entfernten PC nicht zu. Programmierumgebungen sind nicht ausreichend leistungsfähig um komplexe Aufgaben, wie die kollaborative Operation innerhalb einer flexiblen Fertigungszelle, zu entwickeln. Unterschiedliche Hersteller verwenden eigene Programmiersprachen und Entwicklungstools, wodurch Integration und gemeinschaftliche Produktion erschwert werden. Die sich damit ergebende Heterogenität der Anlagen einer Produktionsstrecke ist ein bereits betrachtetes Problem cyber-physikalischer Systeme (vgl. [SG16]). Im Falle proprietärer Schnittstellen und geschlossener Architekturen muss ein Adapter die Standardisierung von Protokollen und Informationen durchsetzen [Aya+13; GSL14]. Für SPS gelten in diesem Zusammenhang die gleichen Anforderungen.

R3 Standardisierte Informationsprotokolle und -modelle werden für die Integration heterogener Altmaschinen eingesetzt, so dass Datenaggregation und M2M-Kommunikation gesamtheitlich gewährleistet werden kann.

3.4 Lokalität

CPPS müssen in geringstmöglicher Zeit Betriebs- und Prozessdaten der Maschine analysieren, bewerten und in den Produktionsprozess eingreifen können. Die Synchronisation des virtuellen Modells der Realität wird jedoch durch stetig wachsende Datenvolumina aufgrund steigender Geräteanzahl erschwert. Damit verlangsamt sich die Verarbeitung der Daten mit der geografischen Entfernung zwischen Gerät und System. Bei der Integration von Altmaschinen muss demnach die Datenanalyse, -persistenz und Historie sowie die Reaktion auf dadurch erkannte Veränderungen möglichst nahe an der Anlage geschehen. Läuft eine Rückkopplungsschleife direkt an der Maschine, muss außerdem nur ein Teil der anfallenden Daten veräußert und die Kontrolle nur teilweise an hierarchisch übergeordnete Systeme abgegeben werden (vgl. zu diesem Absatz [Bon+12]). Durch den verminderten Austausch zwischen den Systemen werden die Sicherheit der Daten verbessert und Kommunikationsfehler minimiert (vgl. [Wan+04]).

R4 Die Erfassung und Persistierung anfallender Betriebs- und Prozessdaten sowie die Interpretation von Maschinenbefehlen geschieht geografisch nahe der Anlage, wodurch zeitliche Latenzen, Kommunikationsaufwände und -fehler minimiert werden.

Auch wenn die Zeit für die Kommunikation von Steuerbefehlen und Sensordaten durch die Nähe zur Maschine minimiert wird, ist weder harte noch weiche Echtzeit ein Kriterium. Es wird davon ausgegangen, dass die Interpretation und Ausführung der Maschinenbefehle sowie die Aggregation der Daten direkt an der Maschine geschieht. Um in adäquater Zeit reagieren zu können, unterliegen die für CPPS erforderlichen Kontrollschleifen damit ebenfalls dem Lokalitätskriterium [Bon+12].

3.5 Integrationshardware

Die Leistungsfähigkeit von Einplatinencomputern, wie dem Raspberry Pi[1] oder Arduino[2], sowie deren Tauglichkeit im Bereich der Maschinensteuerung (vgl. [GM16]), ist in der industriellen Fertigung nicht zu ignorieren. Die Ökonomie eines Fertigungssystems hängt unmittelbar an den Kosten für zusätzliche Hardware, wodurch preisgünstige, eingebettete Systeme, nicht nur durch cyber-physische Produktion, an Attraktivität gewinnen [Lee08; KWH13]. Weiterhin können Einplatinencomputer durch Echtzeitbetriebssystemen auch zeitkritische Steuerungsaufgaben (vgl. Abschnitt 2.1.3) übernehmen. Demnach müssen Einplatinencomputer für das hardwareseitige Retrofitting eingesetzt werden.

R5 Einplatinencomputer werden als zusätzliche Hardware zum Retrofitting eingesetzt, wodurch die Kosten der Modernisierungsmaßnahmen gering ausfallen.

[1] raspberrypi.org (abgerufen am 10.11.2016)
[2] arduino.cc (abgerufen am 10.11.2016)

4 Forschungsstand

Nach der Spezifikation der Zielvorgaben werden in diesem Kapitel der aktuelle Stand der Technik sowie bereits bestehende Forschungsarbeiten zum Thema erläutert und mit den aufgestellten Kriterien für eine Lösung abgeglichen.

4.1 Überwachung des Maschinenbetriebs

In cyber-physischen Produktionssystemen ist Adaptivität durch Rückkopplung zentrales Element. Doch um korrigierend auf einen Produktionsprozess wirken zu können, muss der Zustand des physischen Systems bekannt sein. Dieser wird durch Sensorik an der Maschine erfasst und in einem Modell persistiert (vgl. zu diesem Absatz Abschnitt 2.3).

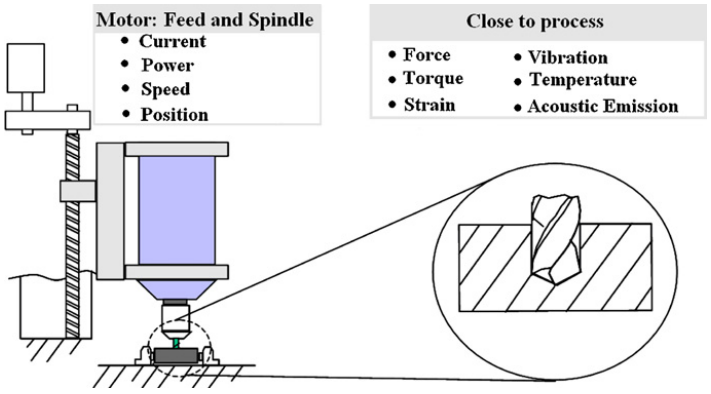


Abbildung 4.1: Messgrößen bei der Maschinenüberwachung aus [Tet+10]

Vielschichtige Möglichkeiten dieser Art der Erfassung von Betriebs- und Prozessdaten wurden über die Jahre untersucht. Einen aktuellen Überblick hierzu veröffentlichten Teti et al. [Tet+10]. Für die Datenaggregation werden im Kontext dieser Arbeit direkte und indirekte Messmethoden mittels Sensorik betrachtet. Bei einer direkten Messung werden Parameter wie Leistungsaufnahme, Drehzahl eines Motors oder die Position erfasst. Indirekte Messungen hingegen sind dem Prozess

näher und erfassen Daten wie die auf ein Werkzeug wirkende Kraft, Temperaturen am Werkstück oder Vibrationen. Abbildung 4.1 stellt diese beiden Kategorien am Beispiel einer Werkzeugmaschine dar. Weitere von Teti et al. untersuchte Forschungsgebiete befassten sich mit Signalverarbeitung und Sensordatenfusion, der Kategorisierung von Messpunkten (Monitoring Scopes) und der Entscheidungsfindung bezüglich konkreter Reaktionen auf sich ändernde Zustände (vgl. zu diesem Absatz [Tet+10]).
Eine andere ausführliche Studie zu Arbeiten der Fertigungsprozessüberwachung wurde von Liang et al. veröffentlicht. Sie kategorisierten Forschungsarbeiten nach der Anwendung auf konkrete Anwendungsfälle (vgl. zu diesem Absatz [LHL04]).
Einen konkreten Anwendungsfall zur Signalverarbeitung untersuchten Deshpande und Pieper bei Altmaschinen. Ihr Ziel war eine nicht-invasive Methode der Echtzeitüberwachung über die Stromaufnahme. Die in Kilowatt eingehenden Verbrauchsdaten wurden durch an Bedingungen gekoppelte Schwellwerte in Status (an, aus, Leerlauf), Energieverbrauch, Werkzeugwechsel und Werkstückdurchsatz unterschieden. Via TCP und UDP konnten diese Informationen zur Weiterverarbeitung an Fremdsysteme übergeben werden. Für die Case Study und einen anschließenden Vergleich hatten Deshpande und Pieper auch moderne Maschinen mit der UPC ausgestattet (vgl. zu diesem Absatz [DP11]).
Der abstrakte Weg vom Produktionsprozess bis zur Quantifizierung des Werkzeugzustands wurde durch Ambhore et al. beschrieben. Vom physischen Prozess ausgehend erfassen Sensoren unterschiedliche Werte, die durch Signalanalysen und Klassifikation zur Beschaffenheit des Werkzeugs führen (vgl. Abbildung 4.2). Zu jedem dieser Schritte des Tool Condition Monitoring (TCM) erläutern die Autoren aktuelle Forschungsarbeiten und belegen die ökonomische Relevanz (vgl. zu diesem Absatz [Amb+15]).

Abbildung 4.2: Tool Condition Monitoring-System nach [Amb+15]

Durch einer Instanz dieses Prozesses wurden indirekte Datenaggregation, Signalverarbeitung und TCM insgesamt am Beispiel von CNC-Maschinen in der Dissertation von Lee erforscht. In seiner Arbeit untersucht und verglich er zahlreiche Sensorsysteme und analysierte durch neuronale Netzwerke und statistische Regression den Zustand des Werkzeugs (vgl. zu diesem Absatz [Lee06]).
Fusion von Sensordaten in einer Fallstudie war Schwerpunkt der Arbeit zu TCM von Downey et al. Die Verortung und Signalkorrelation von Akustikemissions-, Vibrations- und Kraftsensorik ermöglicht das Erkennen von schlechten und guten Fräsverhältnissen. Sie berücksichtigten dabei auch den aktuell ausgeführten CNC-Befehl (vgl. zu diesem Absatz [Dow+16]).

Die Konzepte und Zusammenfassungen von Teti et al. [Tet+10], Deshpande und Pieper [DP11], Ambhore et al. [Amb+15], Lee [Lee06], Liang et al. [LHL04] und Downey et al. [Dow+16] dienen der Kategorisierung von Überwachungsmöglichkeiten. Der Rahmen dieser Arbeit endet mit dem Erfassen des Sensor-Signals im Modell von Ambhore et al. (vgl. Abbildung 4.2).

4.2 Steuerung von Fertigungssystemen

Die Kontrolle von Werkzeugmaschinen geht von einem Computer aus, der CNC und interne speicherprogrammierbare Steuerungen (SPS) über eine proprietäre Schnittstelle oder Direct Numerical Control (DNC) mit Befehlen versorgt [Hir00]. Während der erste Fall kaum Schwerpunkt aktueller Forschung ist, wurden DNC-Protokolle für die Fertigungsintegration detailliert beleuchtet.
Ferrolho et al. legten den Fokus auf die Integration von Maschinen in flexible Fertigungssysteme (FFS) [FCL05]. Dazu untersuchten sie proprietäre DNC-Protokolle zweier CNC-Anlagen und konzeptionierten ein abstrahierendes Framework. Sowohl die Steuerung der CNC, als auch der über DNC verfügbare Überwachungsmechanismus, kann über ein Netzwerk und somit aus der Ferne bedient werden. Die anfallenden Daten werden auch für die Rückkopplung der Steuerung verwendet (vgl. zu diesem Absatz [FCL05]). Durch die Ethernet-basierte Kommunikation über DNC sind die Anforderungen R1 und R2 erfüllt (vgl. Abschnitt 3). Da bei der Abstraktion des DNC-Protokolls keine standardisierten Informationsmodelle und Kommunikationsmechanismen verwendet wurden, gilt R3 für dieses Konzept nicht. Die Verortung und Art der Persistenz eingehender Daten ist in der Arbeit von Ferrolho et al. nicht beschrieben, womit R4 ebenfalls nicht erfüllt ist. R5 ist nicht erfüllt, da als Integrationshardware, beziehungsweise Steuerungssysteme PCs verwendet werden. Eine Sammlung von Softwarewerkzeugen für die Steuerung von Robotern, Fräs- und Drehmaschinen entwickelten Ferrolho et al. zwei Jahre später [FC07]. Sie erkannten die Notwendigkeit von DNC-Adaptern und schufen ein erweitertes Framework für die verteilte Kontrolle dieser Produktionsmaschinen. Die Generizität der dabei entstandenen Softwarearchitektur wurde in einem Fallstudie mit fünf Anlagen verifiziert und erlaubt die Integration, unabhängig von Hersteller und verwendetem Protokoll (vgl. zu diesem Absatz [FC07]). Dennoch wurde auch in dieser Arbeit kein Standard verwendet (R3), Lokalität (R4) nicht diskutiert und kein Einplatinencomputer zur Kontrolle eingesetzt (R5), weswegen sie hier lediglich als Erweiterung des vorangegangenen Konzepts verstanden wird.

Die Standardisierung von Kommunikationsprotokollen und Informationsmodellen ist eine Forderung der Industrie 4.0 [AE15]. Somit müssen auch bei der Aufbereitung der Steuerung einer Altmaschine etablierte Standards beachtet werden.

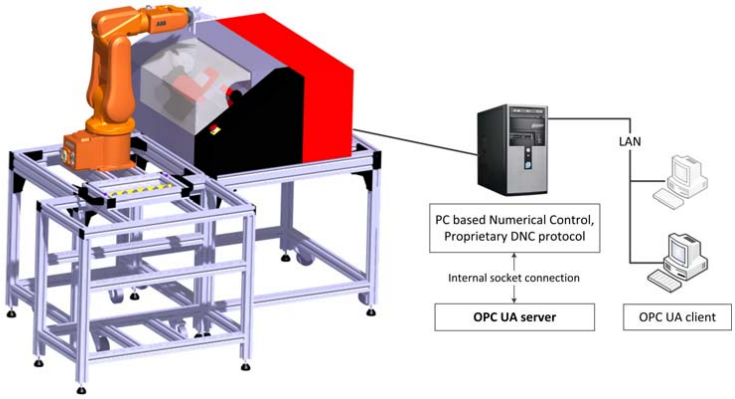


Abbildung 4.3: Flexible Fertigungszelle des IFT Wien [Aya+13]

Ayatollahi et al. entwickelten eine CNC-Steuerungsvariante auf Basis eines DNC-Protokolls mit OPC UA (vgl. Abschnitt 2.2) [Aya+13]. Für die Integration einer Drehmaschine in ein FFS, wurde ein Informationsmodell entworfen. Ein Server, verantwortlich für die Aktualisierung des Laufzeitmodells des Maschinenkontexts, hält die Verbindung zur Maschine und stellt die physikalischen Informationen und Methoden zur Kontrolle der CNC bereit. Abbildung 4.3 zeigt den konzeptuellen Aufbau des Systems am Institut für Fertigungstechnik der TU Wien. Weiterhin wurde ein OPC UA Frontend für den speziellen Fall der Maschinensteuerung entwickelt (vgl. zu diesem Absatz [Aya+13]). Die Autoren verifizierten den Anwendungsfall der Steuerung von CNC-Maschinen mit einem in der Industrie etablierten Standard [OPC14; Hop14]. Durch die Verwendung von OPC UA sind die Anforderungen R1-3 vollständig erfüllt. Das Lokalitätskriterium ist bezüglich der Erfassung von Maschinendaten erfüllt. Lediglich der Anforderung R5 wird das Konzept aufgrund der Verwendung eines PCs nicht gerecht. OPC UA beinhaltet auch die Spezifikation von Historie und beantwortet damit auch die Persistenzfrage. Analyse und Rückkopplung sind nicht Teil des Konzepts von Ayatollahi et al. und müssen durch Subsysteme implementiert werden.
Zum Einsatz von Robotern innerhalb FFS forschten Pauker et al. Sie entwarfen ein Konzept, ähnlich dem von Ayatollahi et al., für die Steuerung und Überwachung von Robotern [Pau14]. Die proprietäre Software des Roboters wird durch einen OPC UA Server gekapselt. Ein Informationsmodell bildet den Roboter mit Funktionalität und Zuständen in Form von Variablen ab. Ändert sich zur Laufzeit das vom Roboter eingesetzte Werkzeug, wird dies auch im Informationsmodell wider-

gespiegelt. Das Fehlen standardisierter Steuerungsschnittstellen wird den Autoren nach durch OPC UA ausgeglichen. Außerdem existierte damit eine semantische Beschreibung der Komponenten eines Roboters (vgl. zu diesem Abschnitt [Pau14]). Aufgrund des Fokus auf Produktionsmaschinen wird die Steuerung von Robotern in dieser Arbeit nicht weiter betrachtet. Die Weiterentwicklung des Konzepts der Steuerung beliebiger Maschinen über OPC UA bestärkt jedoch dessen Eignung für die Integration von Altmaschinen.
Auch ältere SPS profitierten von der Beschreibung von Struktur und Funktionalität durch OPC UA. Durch die Zusammenarbeit des PLCopen Konsortiums und der OPC Foundation (vgl. Abschnitt 2.1.3) wird außerdem die Programmierung von SPS mit OPC UA möglich.

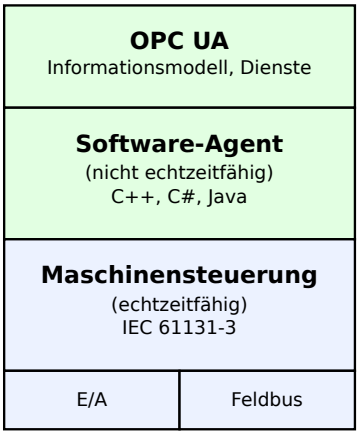


Abbildung 4.4: Service-orientierte Steuerungslogik nach [WJN15]

Auf Basis von OPC UA stellte Windmann et al. das in Abbildung 4.4 abgebildete, Service-orientierte Steuerungskonzept für die Automatisierung von Feldgeräten mit SPS vor [WJN15]. Der Fokus lag dabei auf Plug & Produce, wodurch sich das Gerät eigenständig in die Infrastruktur integriert, konfiguriert und arbeitet. Aufgaben die für eine Programmierung mit IEC 61131-3 (vgl. Abschnitt 2.1.3) zu komplex sind, werden mit einem Softwareagenten plattformunabhängig abgebildet. Die Ein- und Ausgabe sowie die Kommunikation über einen Feldbus übernimmt die Maschinensteuerung. Mit einer Fallstudie zur Bewegungssteuerung mit SPS verifizierten die Autoren das Konzept (vgl. zu diesem Absatz [WJN15]). Die vorgestellte Schichtenarchitektur (vgl. Abbildung 4.4) kann nach entsprechenden Anpassungen für das Retrofitting eines Feldgerätes im Kontext cyber-physischer Produktionssysteme (CPPS) genutzt werden. Durch die Verwendung von OPC UA sind die Anforderungen R1-3 erfüllt. Der Software-Agent als Teil der Geräteabstraktion lässt neben der lokalen Datenhaltung auch komplexere Logik für Rückkopplungsschleifen zu, wodurch R4 erfüllt ist. Durch die Verwendung eines Industrie-PCs zur Steuerung wird das Konzept der Anforderung R5 nicht gerecht.

4.3 Architektur flexibler Produktion

Überwachung und Steuerung einzelner Maschinen muss im Kontext einer Produktionsumgebung mit vielen heterogenen Feldgeräten betrachtet werden. Die Software- und Systemarchitektur des Gesamtsystems nimmt damit eine zentrale Rolle bei der Integration von Altmaschinen ein.
Wang et al. entwickelten eine offenen Architektur für die Echtzeitüberwachung und -kontrolle von im Netzwerk befindlichen CNC-Maschinen über eine grafische Schnittstelle mit 3D Repräsentation [Wan+04].
Ein Web-basierter Thin-Client des Wise-ShopFloor ermöglicht die Kontrolle und Überwachung der Maschinen über ein dreidimensionales Modell der Fertigungsstrecke. Das darunterliegende Framework basiert auf einer Client/Server-Architekturstil und verwendet seitens des Servers das MVC-Entwurfsmuster. Maschinen werden über das Fabriknetzwerk mit dem Server verbunden und sind somit vom Internet getrennt. Bei der Verwendung mehrerer Clients wird für das Routing ein Publish/Subscribe Mechanismus über HTTP-Streaming eingesetzt. Mit Hilfe dessen wird das Verhalten des auf Java 3D basierenden Visualisierungsmodells durch Sensorik an den Machinen beeinflusst. In der von Wang et al. durchgeführten Case Study wurde unter Verwendung einer CNC-Fräsmaschine die Tauglichkeit des Konzepts verifiziert. Die Schnittstelle zwischen Server und Maschine wurde durch einen Open Architecture Controller[1] bereitgestellt (vgl. zu diesem Absatz [Wan+04]).
Durch Verteilung von Steuerung und Überwachung der Maschine auf im Netzwerk befindliche Clients, sowie die browserbasierte Nutzungsschnittstelle, werden die Anforderungen R1 und R2 (vgl. Abschnitte 3.1, 3.2) vollständig erfüllt. Ein Standard wird mit der Kommunikation via HTTP verwendet, während Informationsprotokoll und -modell nicht näher erläutert wird. Damit ist R3 nur ansatzweise erfüllt. Da aber Persistenz von Prozess- und Betriebsdaten von einem dedizierten Datenserver übernommen wird, ist das Lokalitätskriterium (R4) nur teilweise erfüllt. Anforderung R5 ist nicht erfüllt, da ein PC Steuerung die Steuerung übernimmt.
Handelt es sich, wie in einem flexiblen Fertigungssystem (FFS), um einen Verbund von Maschinen, werden Rekonfigurierbarkeit und flexible Datenhaltung architektonisch relevant. Die Heterogenität und Austauschbarkeit der Feldgeräte (vgl. Abschnitt 2.1) muss zur Laufzeit berücksichtigt werden. Unter dieser Maßgabe entwarfen Pauker et al. eine Kommunikations- und Integrationsarchitektur für die Montage und Konfiguration einer Fertigungszelle auf Informationsebene [Pau+13]. Das Design beinhaltet ein Informationsmodell sowie zentrale Datenhaltung für die Komponenten einer Zelle. Abbildung 4.5 stellt den Unterschied zwischen einem herkömmlichen und dem Fertigungssystem von Pauker et al. dar.

[1] Steuerungskomponente, die Modifikationen über das API hinaus zulässt [Yon05]

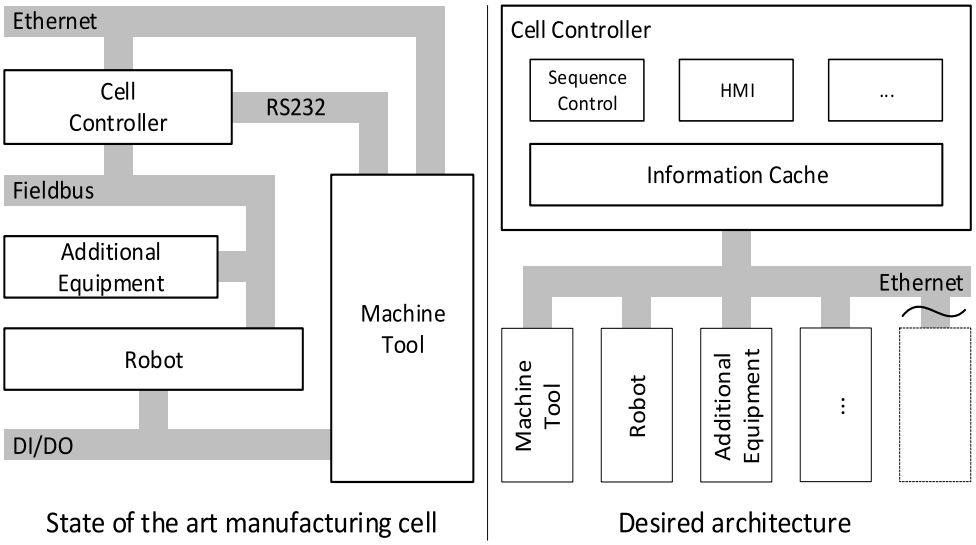


Abbildung 4.5: Vergleich der Architekturen von Fertigungszellen aus [Pau+13]

In ihrem Konzept werden die unterschiedlichen Kommunikationskanäle, wie Feldbusse, serielle und digitale Ein-/Ausgabe-Schnittstellen, durch ein TCP/IP-Protokoll auf Ethernet-Basis vereinheitlicht. Aus dem Bereich der intelligenten Systeme übernahmen die Autoren das Blackboard-Konzept (vgl. [Hay85]) für den Informationscache, respektive die zentrale Datenhaltung. Parameter für die Kommunikation, der Maschinenstatus, das aktuelle Programm sowie Informationen zu angeschlossenen Geräten werden hierfür in einer XML-Datei abgelegt. Eine Sequenzkontrollkomponente legt seine Forderungen (z.B. starte CNC-Programm) in dieser Datei ab, wodurch andere ihre Aufgaben eigenständig abholen und wahrnehmen können. Komponenten des FFS werden durch ein adaptierendes, nicht standardisiertes Protokoll angebunden. Dieser Ansatz reduziert die Komplexität der Gerätetopologie und führt zu einem geringen Konfigurationsaufwand (vgl. zu diesem Absatz [Pau+13]).
Steuerung und Überwachung können mit dem Blackboard auch aus der Ferne geschehen, womit R1 und R2 erfüllt sind. Standardisierung ist nicht Teil des Konzepts, soll aber mit einer OPC UA-Anbindung durchgesetzt werden. Gleiches gilt für die Lokalität von Daten und Logik, wodurch die Anforderungen R2 und R3 nicht erfüllt sind. Auch R5 ist durch das Konzept nicht abgedeckt. Für Retrofitting relevant ist die Arbeit im Bezug auf die zur Laufzeit mögliche Rekonfiguration der Komponenten.
Während Pauker et al. die horizontale Vereinheitlichung der Kommunikation anstreben, forschten Moctezuma et al. an vertikalem und horizontalem Informationsaustausch [Moc+12]. Dafür modernisierten die Autoren die Fastory Forschungsfertigungsstrecke. Wenn die zur Steuerung notwendige Echtzeit nahe der Maschine

bereitgestellt wird, können Web-Services die Anlagen für den abstrakteren Informationsaustausch kapseln. Die ursprünglich heterogene Feldgerätelandschaft wurde durch eine Service-orientierte Architektur (SOA) ersetzt. Ziel des Konzept ist unter anderem die Einsparung von Energie, Flexibilisierung und Rekonfigurierbarkeit, die Fähigkeit der eigenständigen Wiederherstellung nach Ausfall und Fehler (self-recovery) sowie prädiktive Wartung (predictive Maintenance). Eine zentrale Anforderung an die Schnittstellen der Gerätepräsentation ist, neben Skalierbarkeit und Homogenität, die lokale Verarbeitung von Daten. Intelligente Remote Terminal Units, wie auf der rechten Seite der Abbildung 4.6, kapseln dazu das industrielle Equipment.

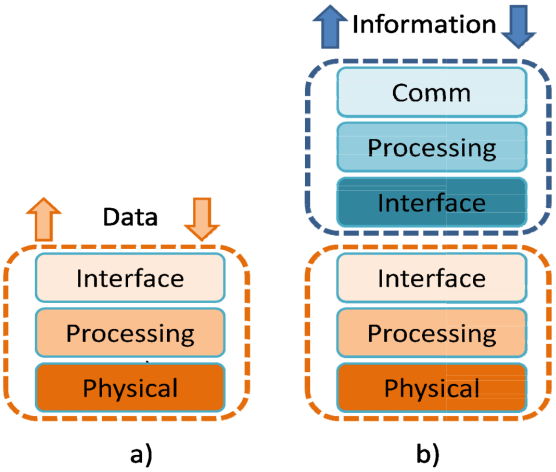


Abbildung 4.6: Equipment ohne und mit Retrofitting aus [Moc+12]

Das original-Equipment besteht aus der physischen Schicht (Physical) für Aktuatoren und Sensoren. Darüber (Processing) werden die Daten umgewandelt – bei einfachen Sensoren auch ohne Verarbeitung durchgereicht. Die Schnittstellenschicht (Interface) bindet das Gerät an ein Bussystem, digitale Ein-/Ausgabesysteme oder andere, teils proprietäre Kommunikationsmedien (vgl. Abschnitt 2.1). Mit der aufsetzenden Remote Terminal Unit wird zuerst eine, zum jeweiligen Medium kompatible Schnittstelle eingesetzt. Eine flexibel anpassbare Schicht für die lokale Datenverarbeitung und Logik stellt die Intelligenz der Remote Terminal Unit. Durch die Kommunikationsschicht werden dann Informationen, anstelle der Daten des original-Equipments weitergereicht und konsumiert. Eine Remote Terminal Unit kann außerdem mehrere funktional zusammengehörige Geräte zu einer logischen Einheit verbinden (vgl. zu diesem Absatz [Moc+12]).
Das Konzept von Moctezuma et al. erfüllt die Anforderungen R1-R4, basiert aber im Gegensatz zu Windmann et al. auf Web-Services. Lediglich R5 ist kein Teil des Konzepts.
Die Kombination aus Rekonfigurierbarkeit ([Pau+13]), Web-Service-basierter Kap-

selung von Feldgeräten ([Moc+12]) und der lokalen Informationsgewinnung aus Anlagen-Rohdaten untersuchten Dürkop et al. im Kontext von SOA [Dür+14]. Die vorgeschlagene Architektur ermöglicht Plug-and-Produce, respektive die automatische Rekonfiguration nach physischen Veränderungen auf Basis abstrakter Prozessdefinitionen. Da industrielle Automation beispielsweise Echtzeitkommunikation erfordert, ist eine reine SOA-Lösung unzureichend. Ein Vorteil von Web-Services besteht in der Möglichkeit der Kombination zu abstrakten Diensten. Der Produktionsprozess bestünde aus der Komposition mehrerer Fertigungszellen und Transportsysteme. In vorgestellten Konzept wird daher die strikte Trennung von Web-Service Modul und Echtzeit-Gerätekontrolle postuliert, dargestellt in Abbildung 4.7.

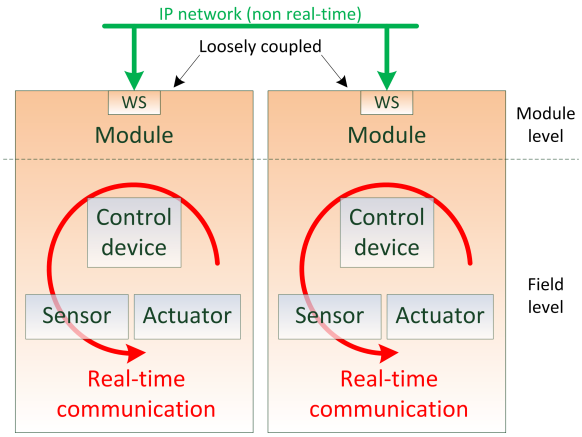


Abbildung 4.7: Aufteilung von Geräten in Automatisierungsmodule [Dür+14]

Ein Modul ist eine mechatronische Einheit, die abstrakte Funktionalität, beziehungsweise Dienste, externalisiert. Die Implementierung der Funktionen ist gekapselt und besteht aus Sensoren, Aktuatoren und einer Kontrollkomponente. Echtzeitkommunikation wird zwischen den Geräten, nicht aber zwischen den Modulen bereitgestellt. Die Schnittstelle der beiden Schichten wird durch OPC UA bereitgestellt. Dürkop et al. erläutern einen möglichen Arbeitsablauf in dem eine Prozessdefinitionssprache, wie die Business Process Execution Language (BPEL), den übergeordneten Produktionsablauf beschreibt. Anhand derer ermittelt eine Orchestrierungskomponente die für den Prozess benötigten Dienste auf Modulebene. Identifizierte Dienste werden mithilfe eines Diensteverzeichnis (Service Directory) ausgewählt (vgl. zu diesem Absatz [Dür+14]).
Überwachung und Steuerung werden von entfernten Subsystemen teilweise übernommen. Feingranulare Logik zur Kontrolle des einzelnen Geräts wird lokal verortet und übermittelt Informationen durch standardisierte Protokolle. Damit sind die Anforderungen R1-4 erfüllt, wodurch die beschriebene Architektur zum Großteil für die Modernisierung von Altanlagen tauglich ist. Anforderung R5 ist kein Teil des

Konzepts von Dürkop et al.
In der vorangegangenen Arbeit schlagen die Autoren den Einsatz von Device Profiles for Web Services[1] (DPWS) als Modulimplementierung vor. DPWS ist die Standardisierung einer generischen Web-Service-Middleware für Geräteprofile, welche Struktur und angebotene Dienste des Equipments definieren. Aufbauend auf dem SOAP-Standard kombiniert DPWS eine Auswahl von Web Service Spezifikationen, wie WS-Addressing[2] und WS-Policy[3], um das Nachrichtenmodell zu beschränken (vgl. zu diesem Absatz [Cân+10]). Aber auch im Zusammenhang mit Erweiterungen können dann eingebettete Systeme andere DPWS-fähige Geräte auffinden und unter anderem mit Ereignissen kommunizieren. Unterschiede, Gemeinsamkeiten und Interoperabilität von OPC UA und DPWS wurden durch Cândido et al. untersucht [Cân+10]. Eine teilweise Kopplung beider Standards über Complex Event Processing (CEP) erreichten Izaguirre et al. [ILL11]. Ein übergreifendes Informationsmodell und eine Middleware für die Verbindung von OPC UA und DPWS stellten Bony et al. vor [BHJ11]. Sie postulieren, dass weder OPC UA noch DPWS die Anforderungen einer SOA auf Feldebene erfüllen können und die Kombination beider vorteilhaft wäre.

Die bisher vorgestellten Arbeiten bereiten industrielles Equipment auf den Einsatz in CPPS vor, thematisieren ihn aber nicht. Für die Integration von Altmaschinen in ein CPPS wird eine dedizierte Architektur benötigt. Die fünf-Schichten CPS-Struktur (5C-Architektur) von Lee et al. liefert eine Leitfaden zu Entwicklung und Deployment von cyber-physischen Systemen (CPS) für die Produktionsdomäne [LBK15]. Die unterste Ebene (Smart Connection) dient der zuverlässigen und akkuraten Erfassung von Daten der Maschinen und Komponenten, sowie deren Ethernet-basierte Kommunikation. Darauf aufbauend (Data-to-Information Conversion) werden die Daten in Informationen gewandelt. Inferenzsysteme und intelligente Analysen dienen hier der Ausfall- und Leistungsvorhersage und führen zur Selbstwahrnehmung (self-awareness) des Equipments. Die folgende Cyber-Schicht bietet digitale Zwillinge der Maschinen und ihrer Komponenten. Sie vergleichen die Leistungsmerkmale ihres physischen Äquivalents mit denen anderer Maschinen (self-compare), führen eine Historie (Time-Machine) und stellen eine cyber-physische Schnittstelle (Cyber-Physical Interface, CPI) für die Machine-to-Machine, beziehungsweise M2M-Kommunikation. Auf einer darüber liegenden Schicht der Kognition, werten Systeme zur Entscheidungsfindung (Decision Support Systems, DSS) zur Unterstützung des Nutzers die Informationen der Cyber-Schicht aus. Auch die Simulation und Synthese weiterer Schritte zählt zu den Aufgaben der Cognition-Ebene. Die oberste Schicht (Configuration) ist als Kontrollinstanz verantwortlich für die Rückkopplung des physischen Raums in das virtuelle Modell. Das Kon-

[1] docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html (abgerufen am 15.11.2016)
[2] w3.org/2002/ws/addr (abgerufen am 15.11.2016)
[3] w3.org/2002/ws/policy (abgerufen am 15.11.2016)

zept der Time-Machine der Cyber-Ebene verwaltet Momentaufnahmen des erfassten Kontexts (Snapshot Collection) und verhilft dem System zu einer kumulierten Maschinenhistorie. Dadurch kann das aktuelle Verhalten mit bereits bekanntem verglichen werden (Similarity Identification) und ermöglicht Vorhersagen. Durch Simulation können Verhaltensmuster für Szenarien extrapoliert und optimierte Schritte synthetisiert werden (vgl. zu diesem Absatz [LBK15]).
Aufgrund der abstrakten Beschreibung einer Schichtenarchitektur werden die Anforderungen nur teilweise erfüllt. Die Time-Machine läuft nicht zwangsläufig nahe der Maschine. Dennoch erfolgen Zustandserfassung und Störfalldiagnose durch Subsysteme des CPPS, womit R1 erfüllt ist. Die Steuerung von industriellem Equipment ist kein Teil des Konzepts und erfüllt demnach R2 nicht. R3 ist nicht erfüllt, da standardisierte Informationsprotokolle und Modelle nicht integriert wurden. Die Verortung von Daten und Logik ist zweigeteilt. Anforderung R4 ist teilweise erfüllt, da Zustandsüberwachung sowie Ausfall- und Leistungsvorhersage an der Maschine geschehen, jedoch die Cyber-Ebene den zentralen Knotenpunkt für die Historie aller Geräte bildet. Die technologische Sicht wird bei dieser Architektur nicht thematisiert, wodurch Anforderung R5 nicht erfüllt wird. Insgesamt bietet der Ansatz von Lee et al. eine architektonische Basis für CPS in der Produktion.

4.4 Zusammenfassung

Die Überwachung und Steuerung von Produktionsmaschinen für Anwendungen in der Industrie 4.0 ist Teil unterschiedlicher Forschungsprojekte. Im Projekt OPC4Factory der TU Wien, wurden generische OPC UA Informationsmodelle entwickelt. Diese verbessern die Konnektivität von NC-Maschinen, Industrierobotern und anderen Komponenten innerhalb einer flexibel automatisierten Fertigungszelle. Die Orchestrierung der Fertigungsoperationen sowie die Konfiguration der Komponenten soll durch die Lösung der Schnittstellenproblematik vereinfacht werden (vgl. zu diesem Absatz [IFT]).
Die Integration bestehender Hardware in die intelligente Steuerung einer Fabrik ist Thema des RetroNet-Projekts. Das Fraunhofer IPK entwickelt mit Industriepartnern physische und logische Adapter für die Anbindung von bestehenden Anlagen an eine Steuerungsplattform. Maschinen-, Anlagen und Produktionsdaten werden zu diesem Zweck zentral erfasst und gespeichert. Weiterhin soll eine Middleware im Client-Server-Architekturstil Dienste und zugrunde liegende Teilsysteme miteinander verbinden und eine vermittelnde Rolle im Gesamtsystem einnehmen (vgl. zu diesem Absatz [Fra16]).
Forschung im Bereich Cloud-basierter Industriesteuerung wird in Zusammenarbeit von Fraunhofer, der TU Berlin und Industriepartnern betrieben. Im Projekt

pICASSO werden die Auslagerung von Steuerungsdiensten in die Cloud und Möglichkeiten einer Verteilung und Modularisierung herkömmlicher Kontrollsysteme auf CPS-Komponenten untersucht (vgl. zu diesem Absatz [ISW]).
Der Schwerpunkt eines Großteils aktueller Forschung liegt auf der Vereinheitlichung der Schnittstellen und deren Durchsetzung – meist mittels Software-Adaptern. Die Gemeinsamkeit aller vorgestellten Architekturkonzepte liegt in der Verwendung von Maschinenrepräsentanten. Auch die Aufteilung einer solchen Repräsentation in zwei Schichten ist etabliert. Der Echtzeit-Kontakt zur physischen Welt wird durch eine bereitgestellt. Intelligentes Informationsmanagement, Analyse- und Rechenfähigkeit durch die andere [Mon+16]. Das Referenzarchitekturmodell RAMI4.0 der VDI/VDE-Gesellschaft beschreibt diese Komponente als I4.0-Komponente mit virtueller Verwaltungsschale [AE15].
Nahezu alle betrachteten Forschungsarbeiten erlauben die Steuerung und Überwachung durch Nutzungsschnittstellen oder Subsysteme eines Produktionssystems. Standards werden oft gefordert, stehen jedoch meist nicht im Zentrum der wissenschaftlichen Betrachtungen. Hervorzuheben ist hierbei die Verwendung der etablierten OPC Unified Architecture (vgl. Abschnitt 2.2). Aktuelle Feldgeräte besitzen entweder einen eingebetteten OPC UA Server, sind darauf vorbereitet oder können mit zusätzlicher Peripherie[1] und Software[2] ausgestattet werden. Somit liegt die Verwendung dieser Spezifikationen für die Steuerung von Produktionskomponenten nahe und muss in ein Konzept für die Integration von Altmaschinen einfließen.

Die Gegenüberstellung von Anforderungen und bestehenden, für das folgende Konzept relevanten, Forschungsarbeiten ist in Tabelle 4.1 zusammengefasst, wobei ● die Erfüllung, ◐ die eingeschränkte oder teilweise Erfüllung und ○ die Nichterfüllung symbolisiert.

[1] opcfoundation.org/products/view/ibh-link-ua (abgerufen am 8.11.2016)
[2] inductiveautomation.com/scada-software/scada-modules/ignition-core-modules/ignitionopc (abgerufen am 8.11.2016)

Tabelle 4.1: Anforderungen bzgl. bestehender Forschungsarbeiten

	R1	**R2**	**R3**	**R4**	**R5**
[FCL05]	●	●	○	○	○
[Aya+13]	●	●	●	◐	○
[WJN15]	●	●	●	●	○
[Wan+04]	●	●	◐	◐	○
[Pau+13]	●	●	○	○	○
[Moc+12]	●	●	●	●	○
[Dür+14]	●	●	●	●	○
[LBK15]	●	○	○	◐	○

5 Konzeption

Nach der Analyse bestehender Forschungsarbeiten folgt in diesem Kapitel die Konzeption einer Lösung zu den in Abschnitt 1 beschriebenen Problemen unter Berücksichtigung der Anforderungen aus Abschnitt 3 und Konzepten aus Abschnitt 4. Ein Softwareartefakt und seine Einbettung in eine System- und Softwarearchitektur werden vorgestellt. Die verschiedenen Perspektiven auf den Entwurf sind an das 4+1 Software-Architekturmodell nach Kruchten angelehnt [Kru95]. Eine virtuelle Maschinenrepräsentation (VMR) bildet die Schnittstelle zur Altanlage, beziehungsweise zu ihren automatisierten Werkzeugkomponenten und damit den Schwerpunkt des hier vorgestellten Designs. Das Informations- und Kommunikationsmodell der VMR mit OPC UA ergänzt die Anlage um eine in der Industrie etablierte, semantische Schnittstelle. Repräsentanten der berücksichtigten Maschinen sind in Szenarien beschrieben. Die Arbeit im Kontext dieser Szenarien und die Aufteilung der Aufgaben unter den Produktionsbeteiligten wird durch Anwendungsfälle skizziert.

5.1 Kontext

5.1.1 Szenarien

S1 – Werkzeugmaschine ohne Schnittstellen. Besitzt die Altanlage keinerlei Schnittstellen, können weder CNC noch automatisierte Werkzeugkomponenten von außen beeinflusst werden. Die CNC ist fest mit der Antriebssteuerung verdrahtet und die maschineneigene SPS für automatisierte Werkzeugkomponenten ist dem Entwickler verborgen. Auch die notwendigen Daten zur Überwachung des Fertigungsprozesses können nicht durch externe Systeme bezogen werden. Ein serieller RS-232 oder Parallelport verbindet lediglich die Steuerungseinheit mit der SPS. Somit ist außer dem Lokalitätskriterium (vgl. Abschnitt 3.4) keine der

Anforderung erfüllt. Für solche Anlagen muss eine standardkonforme Schnittstelle und deren Anbindung an CNC und automatisierte Werkzeugkomponenten vollständig durch die virtuelle Maschinenrepräsentation (VMR) bereitgestellt werden.

S2 – Werkzeugmaschine mit Direct Numerical Control. Direct Numerical Control (DNC) erlaubt das sukzessive Übertragen der CNC-Befehle an die Maschine (vgl. Abschnitt 2.1.2). Trotz der damit physisch kompatiblen Datenverbindung zur Anlage, sind unterschiedliche, meist proprietäre, Kommunikationsprotokolle für DNC üblich [Alt94]. Die maschineneigene SPS ist verantwortlich für automatisierte Werkzeugkomponenten wie Türautomatik oder Kühlsystem. Dem Entwickler steht keine Schnittstelle für diese zur Verfügung. Somit muss neben Adaptern für die DNC-Protokolle eine SPS-Anbindung durch die VMR umgesetzt werden, sofern die DNC jene nicht bereits kapselt.

S3 – Speicherprogrammierbare Steuerungen. Beim Retrofitting von speicherprogrammierbaren Steuerungen (SPS) werden in dieser Arbeit drei Fälle unterschieden. Im aufwändigsten Fall besitzt die SPS keine Ethernetanbindung für Kommunikation via TCP/IP und arbeitet über einen Feldbus (S3.1). Der zweite Fall von zu modernisierenden SPS setzt eine Netzwerkanbindung voraus, veräußert jedoch weder ein Informationsmodell noch standardisierte Kommunikationsprotokolle (S3.2). Im letzten Fall verfügt die SPS bereits über ein integriertes Informationsmodell und kommuniziert durch standardisierte Protokolle. Gegebenenfalls müssen Adapter die Protokolle und Modelle zu einem, im Netzwerk einheitlichen überführen (S3.3).

Sowohl Ayatollahi et al., als auch Ferrolho et al. nutzten für die Umsetzung ihres Konzepts die Drehmaschine *EMCO Concept Turn 55*, an der auch die Anwendungsfälle S1 und S2 orientiert sind (vgl. [Aya+13; FCL05], Abschnitt 4.2). Die in dieser Anlage verbauten automatisierte Werkzeugkomponenten sind Einspann-, Luftdruck- und Kühlsystem sowie eine Türautomatik. Ein proprietäres, serielles DNC-Protokoll ermöglicht die Anbindung externer Systeme in Szenario S2.

5.1.2 Anwendungsfälle

Die unterschiedlichen Anforderungen der mit dem System interagierenden Menschen werden in Anwendungsfällen deutlich, die auszugsweise aus den Personas von Denner et al. hervorgegangen sind [Den+15].

A1 – Montagearbeiter. Ein Montagearbeiter ist neben dem Zusammensetzen einer Maschine verantwortlich für die Verwaltung und Reparatur der einzelnen Komponenten (vgl. Abbildung 5.1). Dafür muss er wissen, welche Teile verbaut werden sollen und wie diese zusammenhängen. Eine einfache Stückliste leistet das nicht und muss bei der Anlagenmodernisierung durch ein digitales Modell der Zusammensetzung – zugänglich auch für andere Teammitglieder – abgebildet werden. Bei modernen Anlagen ist der Maschinenhersteller ist für die Modellierung des Informationsmodells zur Maschinenstruktur und für die initiale Einrichtung verantwortlich.

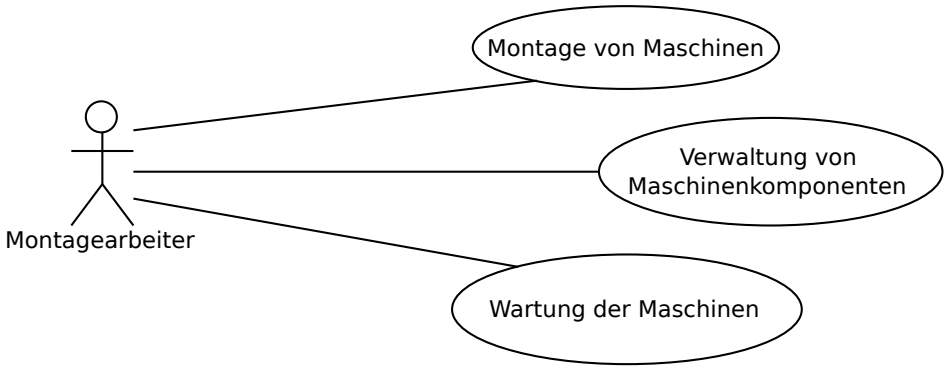


Abbildung 5.1: Anwendungsfälle eines Montagearbeiters

A2 – Produktionsleiter. Wie in Abbildung 5.2 dargestellt, ist ein Produktionsleiter hauptsächlich für die Erstellung, Überwachung und Dokumentation von Produktionsplänen zuständig. Er ist für eine reibungslos funktionierende Fertigungsstrecke verantwortlich und benötigt eine zusammenfassende Darstellung der Betriebs- und Prozessdaten um auch Wartungsaufträge delegieren zu können. Altmaschinen können diese Daten auch erfassen. Jedoch lassen sie die Einsicht aus der Ferne aufgrund fehlender Infrastrukturanbindung nicht zu.

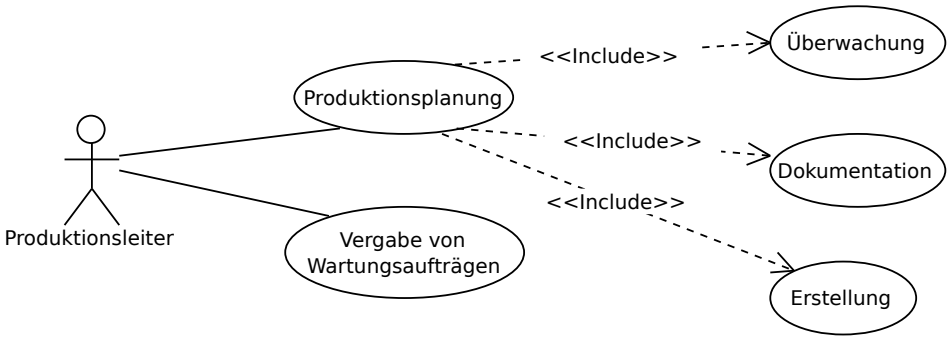


Abbildung 5.2: Anwendungsfälle eines Produktionsleiters

A3 – Maschinenbediener. Der Maschinenbediener richtet die Anlage ein (vgl. Abbildung 5.3). Er integriert sie in den Kontext der Produktionsstrecke und bindet sie an das bestehende Automatisierungssystem. Kenntnis der technischen Schnittstellen für diese Integration entnimmt er dem digitalen Modell der Maschine. Die detaillierte Darstellung des Systemzustands hilft einem Maschinenbediener bei der Überwachung des Fertigungsschritts und der Reaktion bei Störfällen. Auch maschinenspezifische Anpassungen von CNC-Programmen werden von ihm verantwortet.

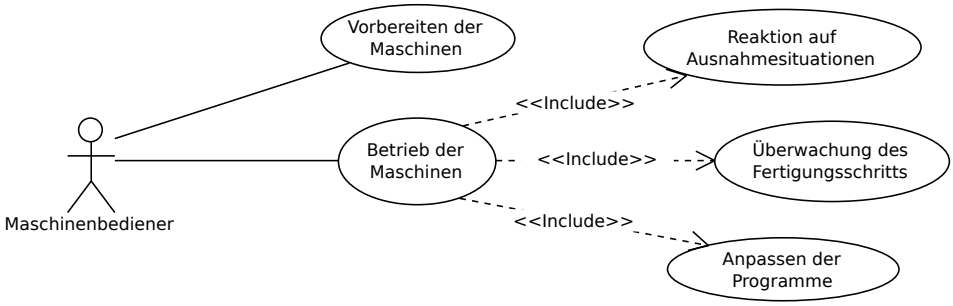


Abbildung 5.3: Anwendungsfälle eines Maschinenbedieners

5.1.3 Systemumgebung

Durch die Anwendungsfälle und Szenarien ergibt sich ein Systemkontext, dargestellt in Abbildung 5.4. In diesem interagiert der Produktionsleiter mit dem Enterprise Resource Management (ERP) und einem Produktionsplanungssystem (PPS). Aufträge und benötigte Ressourcen werden hier in die Planung der Fertigung überführt. Die Bedienung der konkreten Maschinen geschieht über die virtuelle Maschinenrepräsentation (VMR). Diese kapselt die in den Szenarien vorgestellten Altanlagen und ist weiterhin mit den Feldgeräten und der Produktionsplanung verbun-

den. Ein Maschinenbediener muss nicht direkt an der Anlage arbeiten, sondern kann die Steuerung und Überwachung von einer entfernten Nutzungsschnittstelle übernehmen. Der Montagearbeiter agiert auf Feldebene, kennt die Systemstrukturen und verwaltet und wartet das Produktionsequipment. Die Verbindung der VMR zu anderen Feldgeräten, ihre Kapselung der Altanlage und die Integration mit externen Systemen, wie einem ERP-System, wird in diesem Konzept vorgestellt.

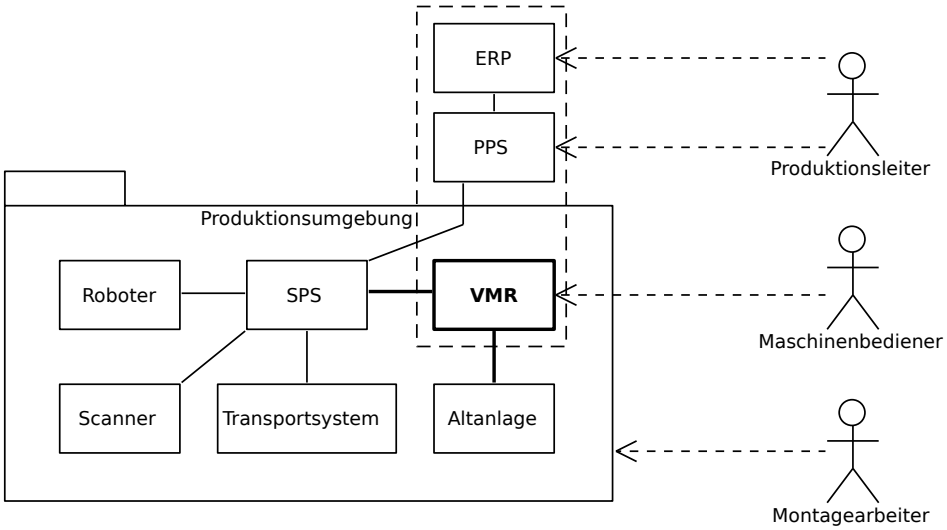


Abbildung 5.4: Kontext der zu integrierenden Altanlage

5.2 Informationsmodell

Die Forderung standardisierter Informationsmodelle und Machine-to-Machine (M2M) Kommunikation wird durch aktuelle Forschung im industriellen Umfeld gestützt [HR15]. OPC UA, im weiteren Verlauf als Unified Architecture (UA) abgekürzt, bietet die dafür geeigneten Werkzeuge [ILL11; HR15]. Echtzeit und direkte Bewegungskontrolle sind nicht möglich, weshalb eine eigenständiger Schicht in Abschnitt 5.3.1 besprochen wird [HR15]. Bei der Modernisierung von Altanlagen wird deren strukturelle Komponentenbeschreibung in einem UA-Adressraummodell hinterlegt.

5.2.1 Modellierung der Anlagenstruktur

Das Metamodell der UA bietet unter anderem typisierte Objekte, Variablen und Methoden. Mit dessen Instanzen werden die automatisierten Werkzeugkomponenten einer Maschine baumartig organisiert. Das grundständige Modell eines UA-Adressraums wurde bereits für die Integration einer Werkzeugmaschine erweitert und Modellelemente für deren automatisierte Werkzeugkomponenten und numerische Kontrolle definiert [Aya+13]. Diese Erweiterung der Data-Access Spezifikation (vgl. Abschnitt 2.2.1) von Ayatollahi et al. wird in dem vorliegenden Konzept verwendet und ist in Abbildung 5.5 dargestellt. In diesem Teilmodell sind, bis auf den `BaseObjectType` der grundlegenden UA-Spezifikation, alle Elemente aus dem Namensraum *OPC4Factory*. Variablen und Methoden sind Elemente, welche durch die `hasComponent`-Relation mit einer Maschinenkomponente verknüpft werden. Beispielsweise komponiert ein Objekte vom Typ `LoadingDoorType` sowohl die Variable `Door_Status`, als auch Methoden zum Öffnen (`Open_Door`) und Schließen (`Close_Door`) einer Ladetür.

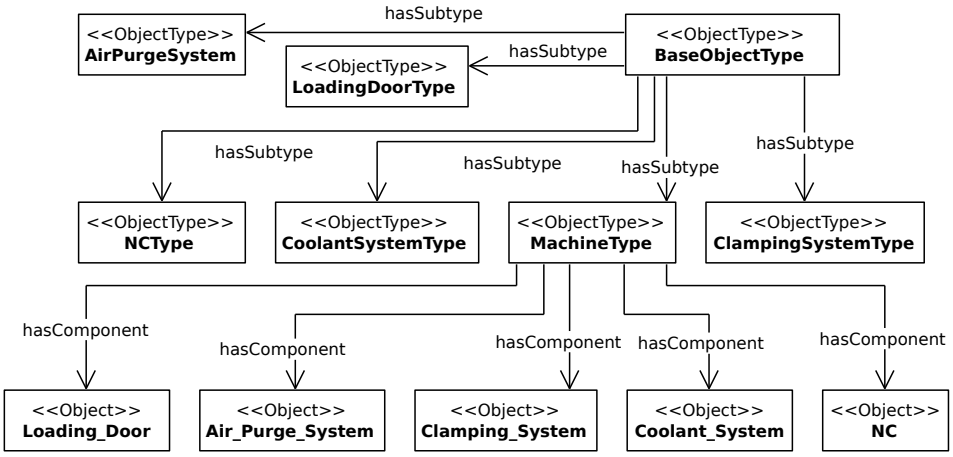


Abbildung 5.5: OPC UA Modellerweiterung nach [Aya+13]

Bei der Modellierung einer SPS wie in Szenario S3 bietet sich die Companion Specification der PLCopen an (vgl. Abschnitt 2.1.3). Durch Abbildung von Funktionsbausteinen und Ein-/Ausgabeparametern auf den UA-Adressraum können Anwendungen die Anlagenstrukturen auf gleiche Weise erfragen und manipulieren [PO]. Bei der Verwendung der IBH Link UA, ist die Informationsmodellierung implizit in der Variablendefinition enthalten und wird via Ethernet auf das Gerät übertragen[1]. Wird das Ignition OPC UA Softwaremodul verwendet, kann das Modell frei und damit nach der Erweiterung von Ayatollahi et al. gestaltet werden. Eine

[1] opcfoundation.org/products/view/ibh-link-ua (abgerufen am 12.11.2016)

Alternative zu dem vorgestellten Informationsmodell für SPS ist das von Windmann (vgl. Bild 4 in [WJN15]). Innerhalb dieses Konzepts hängt der Verwendete Ansatz vom jeweiligen Nutzungskontext ab. Da Standardisierung jedoch eine zentrale Anforderung der Anlagenmodernisierung ist, wird die Variante mit PLCopen bevorzugt.

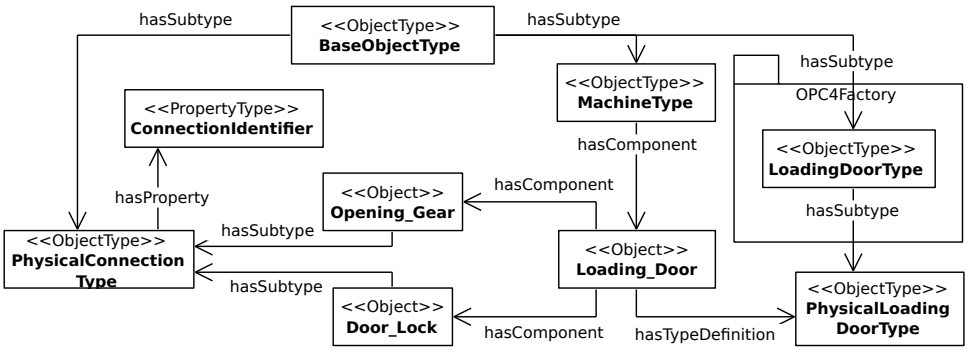


Abbildung 5.6: CPPS Erweiterung des Informationsmodells

Cyber-physische Produktionssysteme (CPPS) stehen über Aktuatoren und Sensoren mit der realen Welt in Verbindung (vgl. Abschnitt 2.3). Um sie mit der virtuellen Maschinenrepräsentation (VMR) verknüpfen zu können, sind Konfigurationsparameter, wie physische Adresse, ein Netzwerk oder Hardware-Port und andere Initialisierungswerte notwendig. Diese sollen im Informationsmodell festgelegt werden können. Dafür wird die Spezifikation von Ayatollahi et al. um physische Objekte für jede automatisierte Werkzeugkomponente ergänzt, dargestellt in Abbildung 5.6. Beispielsweise besitzt der `LoadingDoorType` aus dem OPC4Factory-Namensraum eine Unterklasse `PhysicalLoadingDoorType`. Eine Anlagendefinition vom Type `MachineType` kann dann das optionale Objekt `Loading_Door` beinhalten. Die Objekte `Opening_Gear` und `Door_Lock` sind vom Typ `PhysicalConnectionType` aus dem Namensraum CPPS. Sie sind aktive, virtuelle Teilkomponenten der Maschine und in diesem Beispiel verantwortlich für die Bewegung und einen Schließmechanismus der Anlagentür der Szenarien S1/2. Die Bewegung kann durch einen Servomotor-Aktuator und das Verschließen durch ein Relais ausgeführt werden. Eine physische Verbindung beinhaltet ein Property `ConnectionIdentifier`, welches die Konfigurationsparameter repräsentiert.
Im Anwendungsfall A1 wird bei der Montage der Maschine das Modell um die Beschreibung der physischen Verbindungen ergänzt. Auch bei der Verwaltung der Maschinenkomponenten unterstützt das Modell den Monteur. Ein Wartungsauftrag des Produktionsleiters (A2) kann im Modell mit einer solchen Verbindung verknüpft werden. Der Maschinenbediener (A3) bekommt im Fehlerfall ein UA-Ereignis mit der detaillierten Beschreibung des Ausfallgrunds an die jeweilige Nutzungsschnittstelle.

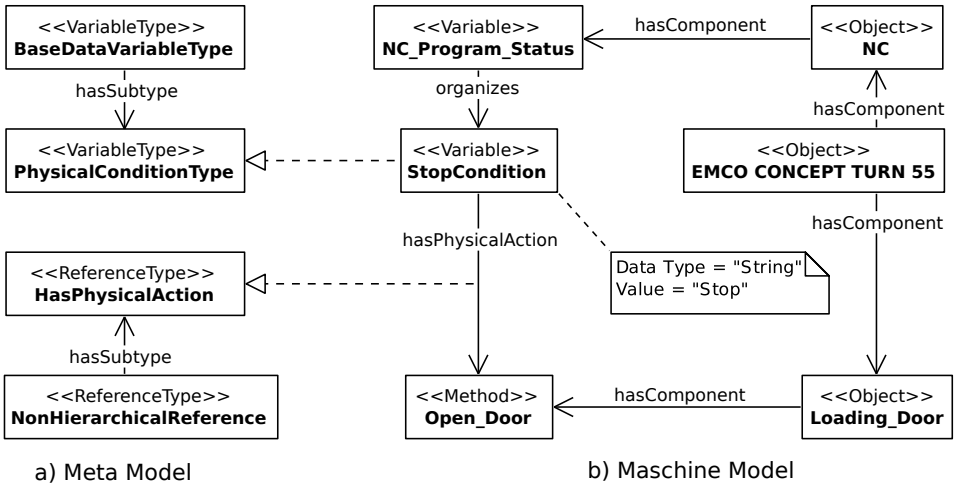


Abbildung 5.7: ECA Erweiterung des Informationsmodells

Die Regeln einer cyber-physischen Rückkopplung werden mit der Anlagenstruktur der VMR modelliert, dargestellt in Abbildung 5.7. Dafür wurde das Informationsmodell um den Variablentyp `PhysicalConditionType` und den Referenztyp `HasPhysicalActionType` erweitert (vgl. Abbildung 5.7a). Instanzen (im Beispiel `StopCondition`) des ersten sind als Teil der Variablen (`NC_Program_Status`) einer automatisierten Werkzeugkomponente (`NC`) der Maschine (`EMCO CONCEPT TURN 55`) beschrieben. Diese beinhalten den Wert einer Bedingung („Stop“) und können beliebigen Typs sein. Mit dem zweiten Typ wird die jeweilige Aktion, respektive UA-Methode (`Open_Door`) referenziert. Eine Modellierung der Ereignisse von Event-Condition-Action ist nicht notwendig (vgl. Abschnitt 5.3.4). Wird nun der Wert einer Variablen verändert und der neue entspricht der verknüpften Bedingung, folgt die physische Aktion. Im Beispiel der Abbildung 5.7b soll demnach die Ladetür der Anlage geöffnet werden, wenn die numerische Kontrolle stoppt.

Das Informationsmodell besteht demnach aus drei Schichten der Spezifikation. Allem zugrunde liegt die UA-Definition (OPC UA Part 5[1]). Darauf aufbauend wurden die Modellelemente von Ayatollahi et al. zur Steuerung und Überwachung der Altanlage übernommen (OPC4Factory, [Aya+13]). Die Verbindung mit dem physikalischen Kontext und die Modellierung der Regeln für cyber-physische Rückkopplung gehen von der in diesem Konzept entworfenen UA-Erweiterung aus (CPPS).

[1] opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model (abgerufen am 12.11.2016)

5.2.2 Laufzeitmodell

Beim Start der virtuelle Maschinenrepräsentation (VMR) wird das Informationsmodell geladen und steht für andere Maschinen und Nutzungsschnittstellen zur Verfügung. Die angeschlossenen Sensoren erfassen permanent den physischen Kontext und aktualisieren das Informationsmodell zur Laufzeit. Außerdem werden die Regeln für Rückkopplung innerhalb der VMR auch aus diesem Modell gewonnen. Damit besitzt die VMR eine Wissenbasis mit Local Context und Runtime Adaptation Model nach Wätzoldt und Giese [WG14]. Fragt eine andere Maschine oder Nutzungsschnittstelle die Struktur der Anlage hinter der VMR an, wird auch der aktuelle Kontext präsentiert.

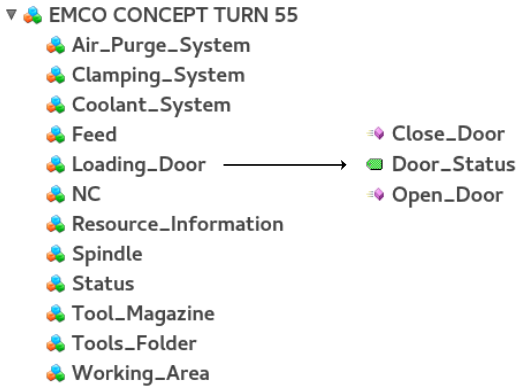


Abbildung 5.8: Laufzeitmodell der Maschine der Szenarien S1/2

In Abbildung 5.8 ist das Laufzeitmodell der Anlage aus Sicht eines Anwendungsprogramms[1] zu sehen. Durch die beispielhaften Methoden `Close_Door` und `Open_Door` sowie die Variable `Door_Status` der Ladetür, kann der physische Kontext nicht nur eingesehen, sondern auch manipuliert werden. Rückkopplungsmechanismen (vgl. Abschnitt 5.3.4) sorgen für die Konsistenz von Realität virtuellem Modell. Änderungen an der Struktur der automatisierte Werkzeugkomponenten werden im Modell reflektiert und dessen Struktur neu organisiert. In Szenario S1/2 betrifft das den Austausch des Werkzeugs der Maschine.

[1] unified-automation.com/products/development-tools/uaexpert.html (abgerufen am 12.11.2016)

5.3 Virtuelle Maschinenrepräsentation

Die virtuelle Maschinenrepräsentation (VMR) ist in der Verwaltungsschale einer I4.0-Komponente des RAMI4.0 Referenzmodells eingebettet und bietet fachliche Funktionalität (vgl. Bild 9 aus [AE15]). Durch sie können einzelne Baugruppen, Sensoren/Aktuatoren oder ganze Maschinen im Kontext eines Produktionssystems abgebildet werden. Bei der Modernisierung von Altanlagen mittels dieses Konzepts müssen allem voran die Schnittstellen durch eine VMR gekapselt werden, was in den folgenden Abschnitten detailliert beschrieben wird.
Die Architektur des Industrie 4.0 konformen Retrofittings besteht aus den Schichten und Komponenten der Arbeiten von Lee et al., Moctezuma et al. und Dürkop et al. [LBK15; Moc+12; Dür+14], dargestellt in Abbildung 5.9.

Die intelligente Remote Terminal Unit, respektive VMR, kapselt die Altanlage und bietet nahtlose M2M-Kommunikation auf Feldebene durch die Comm-Schicht. Sie wandelt mittels die Processing-Ebene die gesammelten Daten der Maschine, abgebildet durch UA-Variablen, in Informationen in Form von Fusionsvariablen und UA-Ereignissen. Fusionsvariablen entstehen durch den Schritt der Signalverarbeitung im Monitoring-Prozess nach Ambhore et al. und setzen sich aus vorverarbeiteten Sensorwerten zusammen (vgl. Abbildung 4.2 in Abschnitt 4.1) So werden die Daten der Altanlage zentral erfasst und vorverarbeitet, nicht aber persistiert, wie im Blackboard-Konzept von Pauker et al. [Pau+13]. Dennoch ist die Rekonfigurierbarkeit nach deren Konzept durch die lose gekoppelten Module gegeben. Außerdem werden historische Maschinendaten durch die Historical Access Spezifikation (OPC UA Part 11[1]) in der VMR persistiert. Die zentrale Auswertung der Informationen wird von den Mechanismen der Time-Machine (TM) auf Cyber-Ebene übernommen (vgl. [LBK15]). Diese steht in Verbindung mit der VMR, ist aber nicht direkt an der Anlage verortet, wodurch intelligente Analysealgorithmen (vgl. Time-Machine Mechanismen in Abschnitt 4.3) auf leistungsfähiger Hardware implementiert werden können. Die Verbindung zwischen Time-Machine, anderen Diensten und VMR, symbolisiert durch die gestrichelte Grenze in Abbildung 5.9, wird mit dem Web-Service-Modul nach Dürkop et al. an die Cyber- und Conversion-Schicht der 5C-Architektur gebunden [Dür+14; LBK15]. Die darüberliegenden Ebenen der Cognition und Configuration sind kein Teil dieses Konzepts.

[1] opcfoundation.org/developer-tools/specifications-unified-architecture/part-11-historical-access (abgerufen am 12.11.2016)

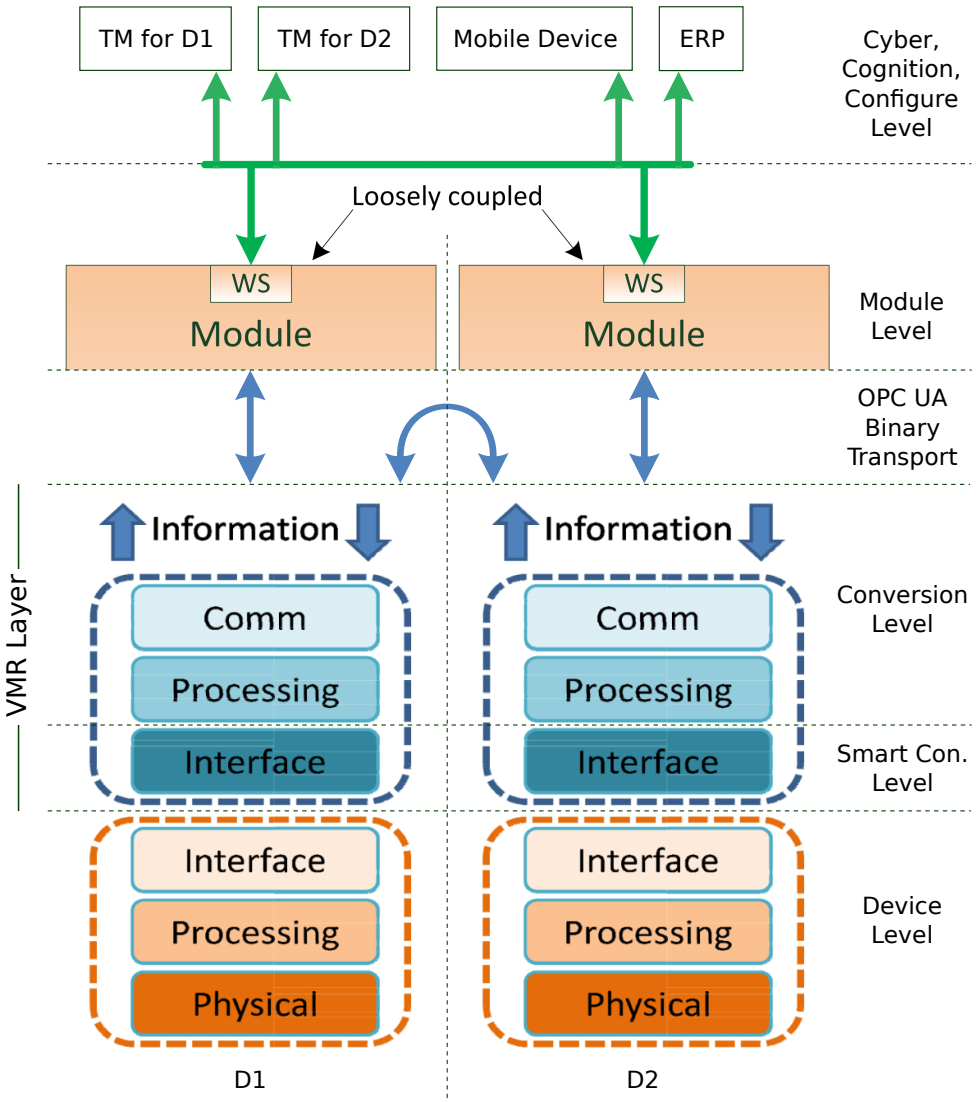


Abbildung 5.9: Konzept der virtuellen Maschinenrepräsentation

5.3.1 Anlagenanbindung

Die Remote Terminal Unit besitzt eine Schnittstelle zur Altanlage, die das physische Gerät von der VMR separiert und auf der Smart Connection Ebene implementiert ist [Moc+12; LBK15]. In den Szenario S1/2 wird diese Anbindung durch einen Einplatinencomputer auf Feldebene (Device-Level, vgl. Abbildung 5.9) realisiert. Der Einplatinencomputer fungiert als Integrationshardware zwischen den digitalen und analogen Signale von Sensoren und Aktuatoren und einem abstrakten Kommunikationsprotokoll wie zum Beispiel dem Representational State Transfer (REST) mit HTTP oder WebSockets. In Szenario S1 kapselt er die Antriebssteuerung der CNC und dessen interne SPS. Eine Software-Middleware (vgl. Abschnitt 5.4) auf dem Einplatinencomputer ist für diese Datenvermittlung verantwortlich und implementiert die Comm-, Processing und Interface-Schicht der Remote Terminal Unit (vgl. Abbildung 5.9). Eine andere Möglichkeit der physikalischen Anbindung sind Einplatinencomputer mit vorbereiteter Middleware, wie das Grove-[1] oder Wio-Link-System[2]. Die Möglichkeiten der Verortung von Sensorik und die der Signalverarbeitung zur Überwachung des Anlagenzustands werden in den Arbeiten von Teti et al., Liang et al. und Downey et al. ausführlich diskutiert und sind auf das Retrofitting mit der VMR anwendbar [Tet+10; LHL04; Dow+16]. Szenario S2 reduziert den Einsatz des Einplatinencomputers auf einen Software-Adapter für das Direct Numerical Control (DNC) Protokoll, wie bei Ferrolho et al., wobei die Echtzeitkontrolle der Anlagensteuerung selbst obliegt [FCL05]. Falls die zu modernisierenden Anlagen über COM/DCOM (vgl. Abschnitt 2.2) verfügen, können Gateways, wie das Unified Automation UaGateway[3] oder der MatrikonOPC UA Proxy[4], im Retrofitting eingesetzt werden. Im Szenario S1 kann Echtzeitfähigkeit nur über zusätzliche Steuerungshardware gewährleistet werden. So können bei der Verbindung der VMR mit der Antriebssteuerung einer CNC, ebenfalls Einplatinencomputer verwendet werden, die wiederum eine Middleware mit Schnittstelle bereitstellen [GM16].
Für die Anlagenanbindung in Szenario S3 werden drei Fälle unterschieden. Sollte keine Ethernetanbindung mit TCP/IP existieren (S3.1) muss die SPS durch Hardware entsprechen ausgestattet werden. Die SPS ist lediglich an ein Bussystem gekoppelt und wird mit einer Ethernet-Karte erweitert. Bei existierendem TCP/IP Kommunikationskanal (Szenario S3.2) repräsentiert externe Soft- und Hardware die VMR und stellt die UA-Verbindung bereit. Eine Komponente mit integriertem UA-Server ist die IBH Link UA[5] für die Steuerungen S5 und S7 von Siemens.

[1] wiki.seeed.cc/Grove_System (abgerufen am 24.11.2016)
[2] wiki.seeed.cc/Wio_Link (abgerufen am 13.11.2016)
[3] unified-automation.com/products/wrapper-and-proxy/uagateway.html (abgerufen am 15.11.2016)
[4] matrikonopc.de/opc-ua/products/ua-proxy.aspx (abgerufen am 15.11.2016)
[5] opcfoundation.org/products/view/ibh-link-ua (abgerufen am 12.11.2016)

Weiterhin existieren Softwarelösungen für SPS von Allen-Bradley oder Siemens, sowie für verschiedene proprietäre Protokollen, wie das Ignition OPC UA Module[1]. Dieses Softwaremodul ist ein UA-Server der mittels Treibern SPS, andere Geräte und Netzwerkprotokolle kapselt. Bei bestehenden Informationsmodellen (S3.3) kann die Middleware als Adapter für das Informationsmodell der SPS fungieren und eines nach Abschnitt 5.2 kommunizieren (vgl. Abschnitt 5.2.1).
Windmann et al. beschrieben einen generischen Ansatz für die Anbindung von Steuerungen, abseits industrieller Hersteller [WJN15]. Die darin vorgeschlagenen Softwareagenten implementieren das Konzept der hier vorgestellten VMR.

5.3.2 Horizontale Integration

Durch die Verbreitung der OPC Unified Architecture (UA) sind neben deren Informationsmodell (vgl. Abschnitt 5.2) die Kommunikationskonzepte geeignet für diese Arbeit. Beim Starten der Middleware auf dem Einplatinencomputer wird zuerst das Informationsmodell der virtuellen Maschinenrepräsentation (VMR) geladen und initialisiert. Danach integriert sie sich selbstständig in das Fertigungssystem um mit anderem Equipment interagieren zu können. Die Dienste der UA zum Auffinden von Geräten (OPC UA Discovery) werden für die automatische Eingliederung der VMR genutzt, dargestellt durch Abbildung 5.10.

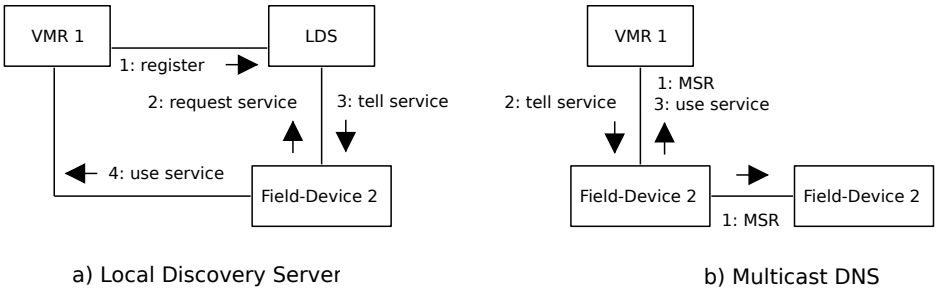


Abbildung 5.10: Horizontale VMR-Integration

Dafür wird ein Eintrag im Diensteverzeichnis der UA (Local Discovery Server, LDS) angelegt (linke Hälfte der Abbildung 5.10). Der UA-Server der VMR sendet eine Nachricht an den LDS, dessen Adresse bekannt sein muss und informiert ihn damit über seine Verfügbarkeit (Abbildung 5.10, a.1). UA-Clients können diesen Knotenpunkt nach Diensten fragen (a.2) und erhalten die jeweilige Beschreibung (a.3). Danach kann dieser Client die Dienste der VMR in Anspruch nehmen (a.4). Soll

[1] inductiveautomation.com/scada-software/scada-modules/ignition-core-modules/ignitionopc (abgerufen am 8.11.2016)

kein zentralisiertes Diensteverzeichnis verwendet werden, bietet das Multicast Domain Name System (mDNS) eine Alternative (rechte Hälfte der Abbildung 5.10). Durch Multicasts im lokalen Netzwerk der VMR können Feldgeräte ad hoc die verfügbaren Dienste erfragen (b.1). Wird solch eine Anfrage durch ein Feldgerät abgesendet, schicken ihr die umliegenden Geräte eine Beschreibung ihrer Funktionalität (b.2). Der Client entscheidet sich für die VMR und nutzt dessen Dienste (b.3). Nachteilig ist hierbei die Beschränkung auf das lokale Subnetz.
Für die eigentliche Verbindung zu den Diensten bestehender Feldgeräte mit der VMR, respektive die horizontale Integration, wird das binäre UA-Transportprotokoll verwendet (vgl. Abschnitt 2.2.2), da die Integrationshardware mit ressourcenschonender Software betrieben werden muss.

Die Anwendungsfälle eines Monteurs (A1) werden zwischen dem Module- und dem Cyber-Level des VMR-Konzepts unterstützt (vgl. Abbildung 5.9). Mit einem UA-Client hat er die Möglichkeit die Maschinenkomponenten zu verwalten und die abgeschlossene Wartung derer zu bestätigen. Über diesen Client kann auch der Maschinenbediener (A3) die Altanlage auf den Betrieb vorbereiten, indem er die physikalischen Verbindungen, also Instanzen des `PhysicalConnectionType`, konfiguriert (vgl. Abschnitt 5.2.1). Bei der Überwachung des Fertigungsschritts kann er die Sensorwerte und Aktuatorenzustände einsehen, die Situation einsehen und Ausnahmefälle antizipieren. Bevor CNC-Programme ausgeführt werden hat er die Möglichkeit sie anhand der aktuellen Situation anzupassen. Ein Beispiel wäre das Fehlen eines vom Programm geforderten Werkzeugs und dessen programmatischer Austausch im Laufzeitmodell.

5.3.3 Vertikale Integration

Durch die Anlagenanbindung (vgl. Abschnitt 5.3.1) und horizontale Integration (vgl. Abschnitt 5.3.2) kapselt die virtuelle Maschinenrepräsentation (VMR) die Funktionalität sowie den Zustand der Altanlage und ermöglicht die nahtlose Kommunikation mit anderen Feldgeräten und Nutzungsschnittstellen. Mit dem Modulkonzept von Dürkop et al. kommen Rekonfigurierbarkeit (vgl. [Pau+13]) und die Schnittstelle zu einer SOA hinzu [Dür+14]. Die UA ermöglicht zwar dem MES oder SCADA-System das Vordringen bis auf Feldebene (vgl. Abschnitt 2.1, [OPC14; BHJ11]), jedoch fehlen ihr die Konzepte für eine übergeordnete Produktionsplanung und -steuerung, die in Prozessen formalisiert beschrieben (z.B. mit BPEL) und innerhalb einer SOA ausgeführt werden kann (vgl. [Dür+14]). Mit der Spezifikation der Device Profiles for Web Services (DPWS) wird in diesem Konzept eine Modulimplementierung verwendet, die eine industrienahe, rekonfigurierbare, vertikale Integration und Prozessebene ermöglicht. Das Modul nach Dürkop et al. koppelt

die VMR mit den übergeordneten Ebenen der 5C-Architektur via DPWS und ist in Abbildung 5.11 dargestellt [Dür+14; LBK15]. Nach ihrer Initialisierung verbindet sich die VMR mit dem UA-Client ihres Modul-Pendants (Abbildung 5.11, 1) auf der Cyber-Ebene, welches Anfragen und Ereignisse zwischen den beiden Schnittstellen vermittelt. Eine Konkretisierung der Vermittlung durch das Modul geht über den Rahmen dieser Arbeit hinaus.

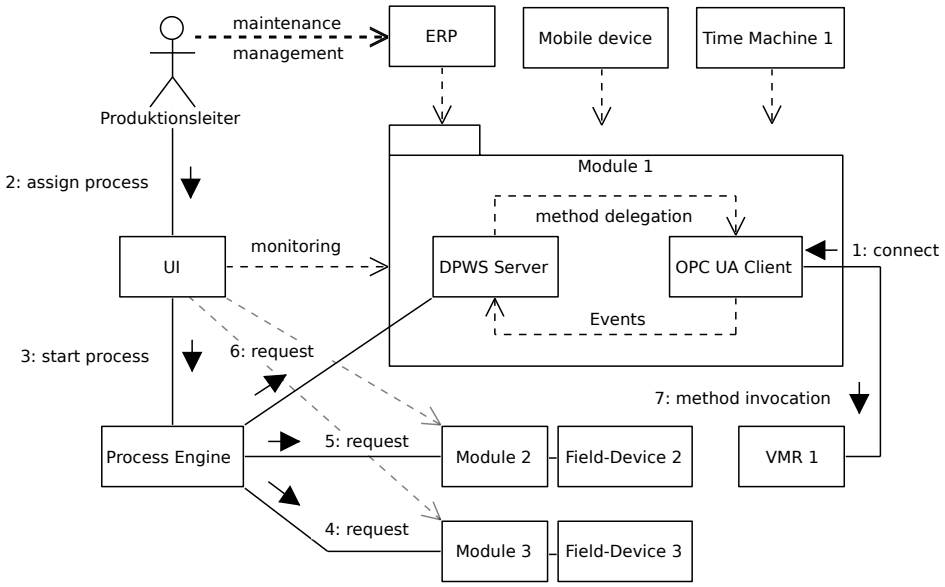


Abbildung 5.11: Vertikale Integration der VMR

Im Anwendungsfall A2 kann der Produktionsleiter einen übergeordneten Produktionsprozess inklusive der Altanlage definieren, Überwachen und anstoßen (Abbildung 5.11, 2/3). Eine Prozess-Engine ist verantwortlich für die Delegation der Fertigungsschritte und führt die in BPEL definierten Service-Aufrufe an die SOA-Module aus (vgl. Beispielarbeitsablauf in [Dür+14], Abbildung 5.11.4/5/6). Diese schicken dann die Methodenaufrufe an den UA-Server der VMR und steuern damit die Maschine (Abbildung 5.11, 7). Durch die Verwendung einer formalisierten Prozesssprache, wie der BPEL, ist die Dokumentation in Anwendungsfall A2 abgedeckt. Die gesammelten Daten der Altanlage werden permanent im Informationsmodell der VMR aktualisiert (vgl. Abschnitt 5.2.2). Das Modul regiert durch UA-Ereignisse auf Veränderungen des Modells und gibt diese durch DPWS-Ereignisse an übergeordnete Dienste, wie ein ERP, Mobilgerät oder Time-Machine, weiter. Eine geeignete Nutzungsschnittstelle (User-Interface, UI) aggregiert diese Ereignisse und bereitet sie für den Produktionsleiter zentral auf. Fällt eine Anlage aus, wird dies mittels Ereignis durchgereicht und visualisiert, so das die Vergabe von Wartungsaufträgen (A2) erfolgen kann. Der Monteur aus Anwendungsfall A1 kann diese Aufträge einsehen und entsprechend reagieren.

5.3.4 Cyber-physische Rückkopplung

In cyber-physischen Produktionssystemen (CPPS) wird der reale Zustand eines verwalteten Elements über Rückkopplungsschleifen (Feedback Control Loop, FLC) mit Zielen und erwartetem Verhalten verglichen und auf ihn eingewirkt (vgl. Abschnitt 2.3). Bei der Modernisierung von Fertigungsanlagen kann der Automatisierungsgrad erhöht werden, wenn die Maschine selbst als verwaltetes Element ihren Zustand teilweise kontrolliert. Eine interne FCL als Teil der virtuellen Maschinenrepräsentation (VMR) ist für diese Selbstwahrnehmung (self-aware) der Anlage zuständig. Zur eigenständigen Konfiguration (self-configure), Selbstadaptivität (self-adaptive) und Fähigkeit zum Vergleich (self-compare) im Kontext der Produktionsstrecke wird eine externe FCL genutzt, die auf dem Configure-Level der Architektur nach Lee et al. arbeitet [LBK15]. Letztere liegt außerhalb des Rahmens dieser Arbeit. Im Gegensatz zur externen FCL, ist die interne Teil des verwalteten Elements und damit Teil der VMR [WUS13]. MAPE-K bildet hier das Konzept des Adaptivitätsmechanismus und nutzt dafür Event-Condition-Action (ECA, vgl. auch [KXU11] und Abschnitt 2.3).

Eine gesonderte Verarbeitung von Ereignissen ist während des Monitorings nicht notwendig. Die VMR benötigt keinen internen Ereignismechanismus und kann in dieser Phase direkt auf die Veränderung der Variablen des Informationsmodells reagieren. Wurde beispielsweise die Variable `NC_Program_Status` aktualisiert, werden alle darunter organisierten Variablen vom Typ `PhysicalConditionType` im Modell lokalisiert. In Abbildung 5.7 aus Abschnitt 5.2.1 ist `StopCondition` die gesuchte. Stimmt der Wert der Bedingung (im Beispiel „Stop“) mit dem neuen Wert überein wird eine Veränderungsanfrage (Change Request) bezüglich der Bedingung an die Planungsphase übergeben. Die Übereinstimmung, festgestellt in der Analyse, ist nicht auf die Gleichheit der Werte beschränkt. Mit Verwendung beliebiger Variablentypen der UA-Spezifikation, können numerische Werte, Zeichenketten und strukturelle Attribute wie Wertebereiche oder Zeitstempel verglichen werden.
So kann MAPE-K beispielsweise die Temperatur eines Werkzeugs für normalen Betrieb und Störfall unterscheiden, indem die Wertebereiche (UA-Datentyp Range) in entsprechenden `PhysicalConditionType`-Variablen festgehalten werden. Eine Veränderungsanfrage beinhaltet die geltende Bedingung, durch die wiederum `HasPhysicalAction`-Referenzen auf UA-Methoden verweisen (vgl. Abschnitt 5.2.1). In der Planung werden diese Referenzen aufgelöst und sich dahinter verbergenden UA-Methoden an die Ausführungsphase übergeben. Als Wissensbasis für die einzelnen Schritte der Schleife wird das Informationsmodell mit dem Zustand der Anlage und den ECA-Regeln genutzt (vgl. Abschnitt 5.2.1).

5.4 Softwareframework

In diesem Kapitel wird das besprochene Konzept der virtuellen Maschinenrepräsentation in einem Software-Framework umgesetzt, welches die Remote Terminal Unit von Moctezuma et al. implementiert [Moc+12]. Ein Framework ist ein komplexes Software-System, welches sich Konzepten der Vererbung objektorientierter Programmierung und dynamischen Bindens von Bausteinen bedient. Es zeichnet sich durch externe Schnittstellen, Variabilität und -erweiterbarkeit aus und besteht aus abgeschlossenen und halbfertigen Elementen. Die übergeordnete Architektur, respektive die Komposition und Interaktion der Bausteine, ist vordefiniert und wird nicht verändert. Auf der anderen Seite stehen Aspekte der spezifischen Anwendungsdomäne die bei dem Entwurf einer Software nicht antizipiert werden können und an das Framework in Form von Zusatzmodulen anzubinden sind (vgl. zu diesem Absatz [Pre94]).

Zur besseren Unterscheidbarkeit werden in diesem Abschnitt Instanzen *kursiv* und Informationsmodellelemente `nichtproportional` formatiert. In den Sequenzdiagrammen sind die Instanzen von Komponenten rechteckig und Erweiterungspunkte, beziehungsweise Entitäten als Kreis dargestellt.

5.4.1 Logische Architektur

Die abgeschlossene drei-Schicht-Architektur des Frameworks der virtuellen Maschinenrepräsentation (VMR) ist in Abbildung 5.12 dargestellt. Auf der Smart Connection, beziehungsweise Interface Ebene, wird der Signalaustausch mit der Altanlage vermittelt und eine einheitliche Schnittstelle bereitgestellt (vgl. Abschnitt 5.3.1). Dies wird durch mehrere cyber-physische Adapter (CPA) bewerkstelligt. Die physischen Kontextdaten und -manipulationsbefehle werden durch die Modellkontrollkomponente konsumiert, prozessiert und produziert (vgl. Abschnitte 5.3.2, 5.3.3). Eine MAPE-K-Rückkopplungsschleife (vgl: Abschnitt 5.3.4) interagiert mit dieser. Für die Kommunikation der Informationen des Adapters bietet der Adressraum des Laufzeitmodells der VMR eine strukturelle Beschreibung der Anlage (vgl. Abschnitt 5.2). Ein UA-Server stellt dieses Modell durch das binäre Transportprotokoll bereit und vermittelt Information und Interaktion mit der Altanlage.

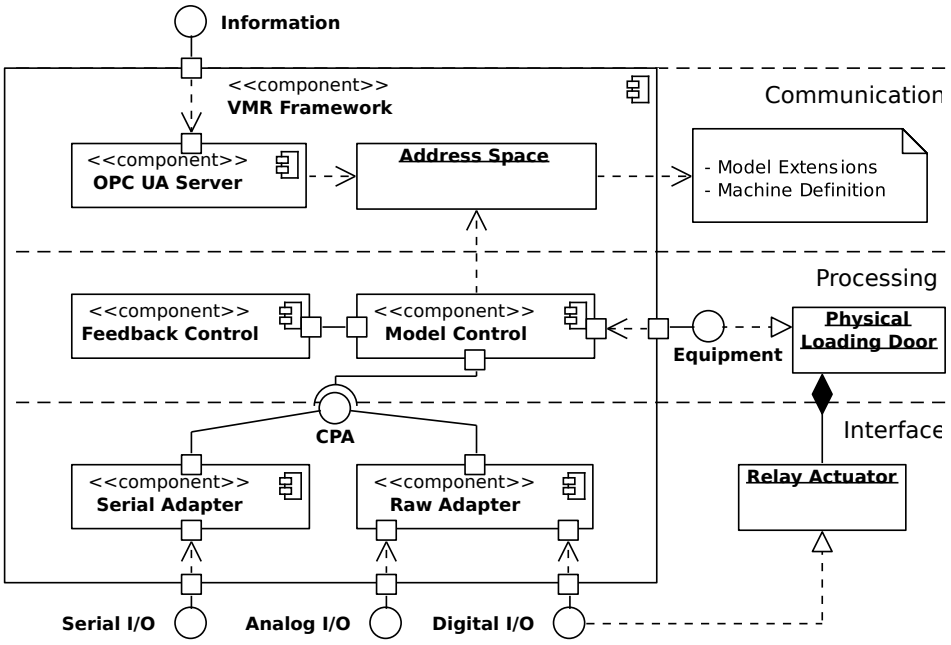


Abbildung 5.12: Framework-Schichten und -Komponenten

Interface. Auf dieser Schicht implementieren cyber-physische Adapter (CPA) die Anbindung an digitale/analoge Ein-/Ausgangssignale und serielle Schnittstellen (vgl. Abschnitt 5.3.1). Sie erfassen den physischen Kontext, vermitteln Manipulation und stellen die Daten der Maschine für die Modellkontrolle bereit. Durch die Heterogenität der Schnittstellen von Steuerungs- und Datenerfassungshardware müssen Adapter an das Framework gebunden werden können. Ob Direct Numerical Control (DNC), IO-Link oder einfach Sensorik (vgl. Abschnitt 2.1.1), wie schon bei Ferrolho et al. muss das jeweilige System gekapselt werden. [FCL05; FC07]. Diese Protokollkapselung, in Abbildung 5.12 ein *Relay Actuator*, implementieren den Erweiterungspunkt des jeweiligen Signaltyps des CPA. Neben der Software benötigt ein CPA zusätzliche Hardware und die entsprechende Anbindung durch Softwarebibliotheken der Ausführungsplattform.

Processing. Die Model Control ist zentrale Komponente dieser Schicht und verantwortet die Verwaltung des Laufzeitmodells (vgl. Abschnitt 5.2.2). Jede von den CPA kommunizierte Veränderung wird hier in das UA-Informationsmodell geschrieben. Wird eine UA-Methode aufgerufen, delegiert Model Control dies an die jeweilige Implementierung. Die Implementierung des Erweiterungspunkts Equipment erlaubt die Abbildung der Logik einer automatisierten Maschinenkomponente. Sie beschreibt deren Methoden und Variablen und besteht aus einer oder mehreren Protokollkapselungen der Interface-Ebene. Im Beispiel der Abbildung 5.12 ist *Phy-*

sical Loading Door die Instanz des Abbilds der Ladetür einer Maschine und besteht aus einem digital angebundenen Relais (*Relay Actuator*) für den Schließmechanismus `Door_Lock` (vgl. Abbildungen 5.5, 5.6 in Abschnitt 5.2.1). Die Model Control ist außerdem verantwortlich für die Initialisierung der VMR. Sie verbindet beim Start die in der Implementierung des Abbilds gekennzeichneten Variablen und Methoden mit dem Laufzeitmodell. Welche Implementierung für das Abbild geladen wird, beschreiben die Erweiterungen OPC4Factory und CPPS (vgl. Abbildung 5.5 in Abschnitt 5.2.1) im Informationsmodell. Wird im Modell der Maschine beispielsweise eine Instanz eines `PhysicalLoadingDoorType` gefunden, lädt Model Control die Implementierung dieses Typs und initialisiert sie mit den Informationen des `ConnectionIdentifier` von `PhysicalConnectionType` (vgl. Abbildung 5.6 in Abschnitt 5.2.1). Die Verknüpfung zwischen Implementierung und Modelldefinition kann durch Namenskonvention oder Annotationen in der jeweiligen Programmiersprache erfolgen.
Für die Rückkopplung ist Feedback Control verantwortlich. Beim Initialisieren des Modells werden die Variablen der Abbilder von Maschinenkomponenten an die Implementierungen von Equipment gebunden. Für jede Variable überprüft Feedback Control die Existenz von Instanzen des `PhysicalConditionType` und evaluiert die Bedingung (vgl. Abbildung 5.7 in Abschnitt 5.2.1). Jede Wertänderung löst eine Iteration der MAPE-K Kontrollschleife aus und führt schlussendlich zur Ausführung einer oder mehrerer UA-Methoden.

Communication. Beim Start der VMR wird der UA-Server mit den internen Modellen (VMR Models) aus Abschnitt 5.2.1 und den Instanzen einer externen Strukturbeschreibung der Anlage (Machine Definition) initialisiert (vgl. Abbildung 5.13). Konfigurationsparameter zur Initialisierung des Servers sind statisch und müssen nach der Auslieferung der Software festgelegt werden. Neben denen der vorausgesetzten Netzwerkanbindung wird der Middleware die Referenz auf eine Datei mit der Machine Definition übergeben (vgl. Abbildung 5.8 in Abschnitt 5.2.2). Diese, wie auch alle Modellerweiterungen, liegt im XML-Format vor und enthält die Knoten des Informationsmodells und ihre Verbindungen (UANodeSet). Für die Modellierung der Machine Definition kann die Software UaModeler[1] eingesetzt werden. Mit dem binären Transportprotokoll der UA kommuniziert der Server die von Model Control gepflegten Informationen im Laufzeitmodell an andere Maschinen und Nutzungsschnittstellen.

[1] opcfoundation.org/products/view/uamodeler (abgerufen am 20.11.2016)

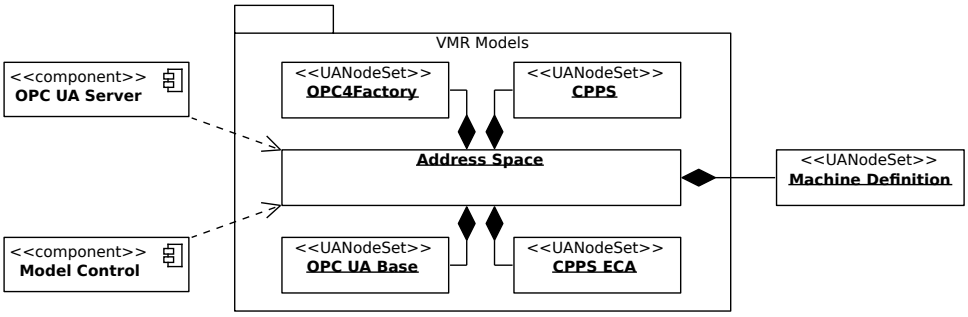


Abbildung 5.13: Zusammensetzung des Adressraums der VMR

5.4.2 Verhalten zur Laufzeit

Die logische Architektur vorausgesetzt, beschreibt dieser Abschnitt das Verhalten der virtuellen Maschinenrepräsentation (VMR) zur Laufzeit. Interaktion und Kommunikation in den auftretenden Situationen werden durch Sequenzdiagramme veranschaulicht. Es wird angenommen, dass die VMR in ein Netzwerk integriert ist und uneingeschränkt mit UA-Clients kommunizieren kann.

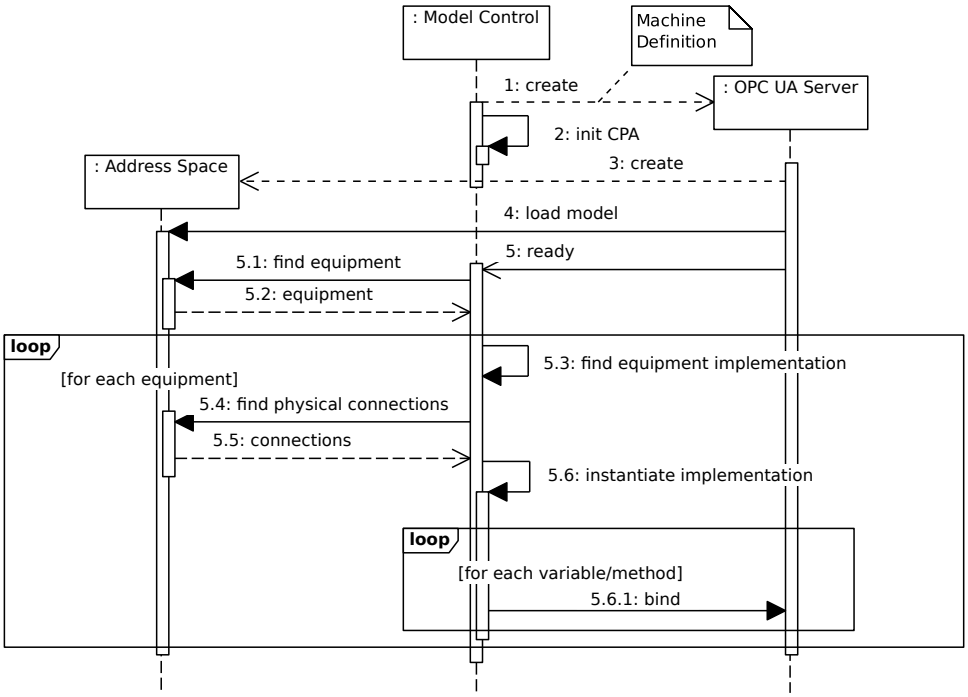


Abbildung 5.14: Initialisierung des Frameworks

Initialisierung Voraussetzung für die Initialisierung der VMR ist eine mit der Middleware ausgelieferte Machine Definition. Die Model Control startet den UA-Server mit dieser als Parameter, dargestellt in Abbildung 5.14 (1). Danach werden die cyber-physischen Adapter (CPA) instantiiert und etwaige Hardwarekomponenten für die Anbindung der Signale initialisiert (2). Der UA-Server kreiert den Adressraum (3), respektive das Laufzeitmodell der VMR, und lädt die Modelle (4). Ist das Informationsmodell vollständig geladen, sendet der Server das entsprechende Signal (5) und die Model Control sucht nach dem für die Anlage definierten Equipment (5.1). Für jede im Informationsmodell gefundene, automatisierte Werkzeugkomponente wird nun die Implementierung gesucht (5.3). Dem bereits angesprochenen Beispiel einer Ladetür (*Physical Loading Door*, vgl. Abbildung 5.12 in Abschnitt 5.4.1) ist wenigstens ein Relais (*Relay Actuator*) für den Schließmechanismus `Door_Lock` untergeordnet. Diese Ausprägungen des `PhysicalConnectionType` halten die Informationen zur Instanziierung der Implementierung in einem `ConnectionIdentifier`-Attribut und werden aus dem Modell geladen (5.4, vgl. Abschnitt 5.2.1). Nun kann *Physical Loading Door* geladen und dessen Logik, respektive Variablen und Methoden, an den UA-Server gebunden werden (5.6). Für die horizontale Integration der VMR muss sich der Server mit einem der in Abschnitt 5.3.2 vorgestellen Discovery-Mechanismen in das Produktionssystem eingliedern.

Methodendelegation. Durch die horizontale Integration, beschrieben in Abschnitt 5.3.2, kann eine Anlage oder Nutzungsschnittstelle die Methoden der VMR aufrufen. Dafür nimmt der UA-Server die Anfrage des jeweiligen Clients entgegen und leitet sie an die Model Control weiter. Das Beispiel der Ladetür mit einem Relais für den Schließmechanismus ist in Abbildung 5.15 (1) dargestellt. Daraufhin wird die mit der Methode gebundene Implementation der automatisierten Werkzeugkomponente identifiziert (1.1). Ein Zuordnung dieser Art kann durch Maps, also Listen von Schlüssel-Wert-Paaren oder das Beobachter-Muster (Observer) umgesetzt werden. Im Beispiel delegiert Model Control den Aufruf an *Physical Loading Door* – Instanz des `PhysicalLoadingDoorType` (1.2). Diese führt die Logik zum Schließen des Relais aus (1.2.1), wobei die physische Adresse des Aktuators aus dem `ConnectionIdentifier` des Informationsmodells stammt (vgl. Abschnitt 5.2.1). Ein cyber-physischer Adapter (CPA) übermittel ein digitales Signal (1.2.1.1) an die Instanz der *Relay Actuator*-Erweiterung des digital I/O, der Teil des Equipments ist (vgl. Abbildung 5.12). Mit dem Methodenaufruf der *Physical Loading Door* an die CPA lässt sich die physische Umsetzung nicht verifizieren, da der *Relay Actuator* keine Bestätigung der Aktion zurückgibt. Das Prüfen der Wirkung dieser asynchronen Methoden muss durch cyber-physische Rückkopplung geschehen (vgl. Abschnitt 5.3.4).

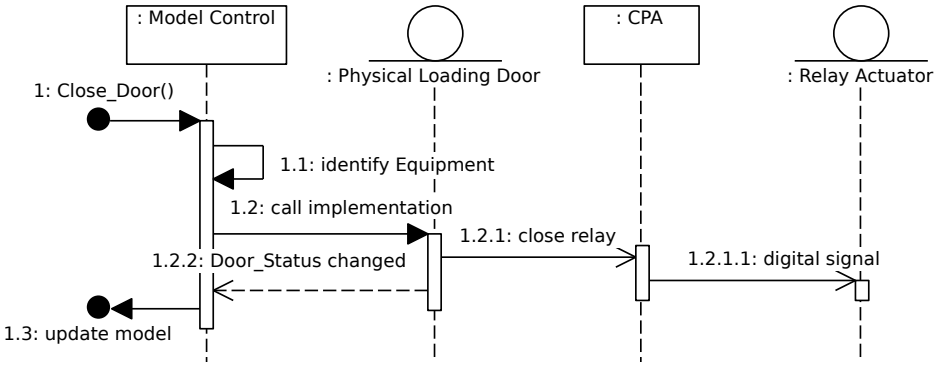


Abbildung 5.15: Methodendelegation im Framework

Kontextveränderung. Mit der VMR verbundene UA-Clients können den physischen Zustand der VMR einsehen. Dieser wird durch Variablen repräsentiert, die stetig im Laufzeitmodell aktualisiert werden (vgl. Abschnitt 5.2.2). In Abbildung 5.16 ist die Aktualisierung eines Kontaktsensors als Teil der Ladetür einer Anlage beschrieben. Der *Contact Sensor* sendet die Veränderung seines Zustands digital an die CPA (1). Sie fragt bei der Model Control nach der für die Adresse des Sensors zuständigen Implementierung (1.1) und vermittelt den neuen Status an die Instanz des Equipments (1.3). Die *Physical Loading Door* gibt die Veränderung der Variable an Model Control zurück (1.3.1), die das Model schlussendlich aktualisiert (1.3.1.1). Eine optionale Verarbeitungslogik innerhalb der *Physical Loading Door* nach 1.3 ist in diesem Diagramm nicht dargestellt.

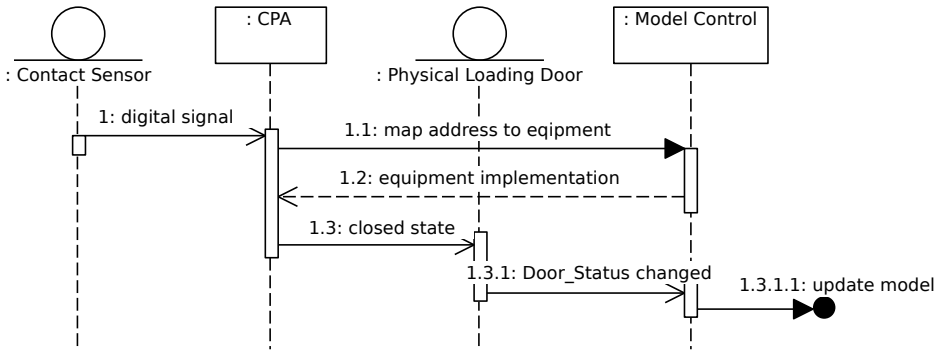


Abbildung 5.16: Kontextveränderungen im Framework

Rückkopplung. Nicht zuletzt wegen der asynchronen Befehle an Aktuatoren hinter der CPA, muss Rückkopplung gewährleistet werden (vgl. Abschnitt 5.3.4). Beispielsweise kann die Interaktion zwischen einer Ladetür und der numerischen Steuerung abgebildet werden. Dafür notwendige Regeln werden in imperativer Form im Informationsmodell hinterlegt (vgl. Abschnitt 5.2.1) und zur Laufzeit von der Feedback

Control evaluiert. Wird eine Variable wie `NC_Program_Status` geändert, zum Beispiel weil die CNC-Werkzeugmaschine den Fertigungsschritt abgeschlossen hat, beginnt die Feedback Control die Analyse der MAPE-K-Schleife (vgl. Abbildung 5.17, 1). Nach dem Einholen der Instanzen des `PhysicalConditionType` im Laufzeitmodell (1.1), wird der Wert dieser gelesen und mit dem der Variable verglichen (1.3). So würde das Ende der CNC-Operation durch den Wert „Stop“ in `NC_Program_Status` angezeigt. Dieser Wert entspricht nun dem der `StopCondition`, wodurch die Bedingung erfüllt ist (vgl. Abbildung 5.7 in Abschnitt 5.2.1). Die Feedback Control holt sich dann die Methoden hinter der `HasPhysicalAction`-Referenz im Laufzeitmodell (1.4) und überlässt der Model Control die Ausführung (1.6). Im Beispiel würde die `Open_Door`-Methode aufgerufen, womit die Ladetür nach abgeschlossener Fertigung geöffnet wird.

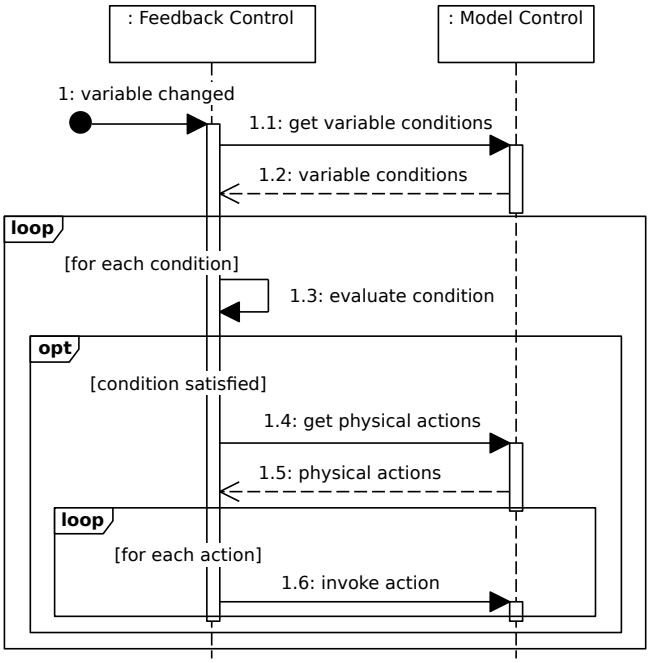


Abbildung 5.17: Cyber-physische Rückkopplung im Framework

5.4.3 Organisation

Die interne Organisation der Komponenten des Frameworks wurde in Abschnitt 5.4.1 besprochen. In diesem Abschnitt wird beschrieben, wie die Komponenten seitens der Softwarestruktur organisiert sind.

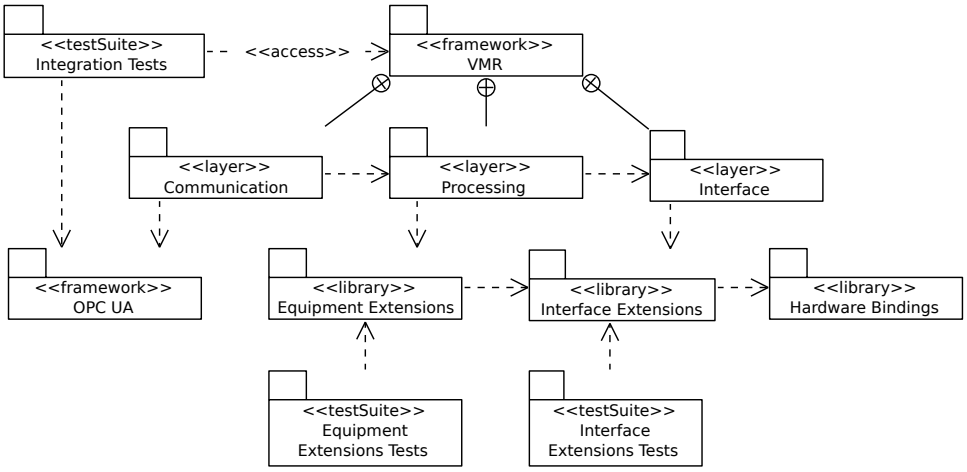


Abbildung 5.18: Organisation des Frameworks

Grundsätzlich bestehen die Komponenten aus Klassen einer objektorientierten Programmiersprache die auf mehrere Pakete aufgeteilt sind, dargestellt in Abbildung 5.18. Das Framework der virtuellen Maschinenrepräsentation (VMR), als oberstes Hierarchieelement, beinhaltet die Schichten Communication, Processing und Interface. Erstere, zuständig für die Kommunikation mit anderen Anlagen und Nutzungsschnittstellen auf Feldebene, benötigt eine Implementierung der UA-Spezifikation Data Access (OPC UA Part 8[1]) und des binären Transportprotokolls. Zur Anbindung von Variablen und Methoden ist sie abhängig von der Processing-Schicht, die deren strukturelle und logische Beschreibung, beziehungsweise Implementierung, kapselt. Für diesen Equipment-Erweiterungspunkt existiert eine dedizierte Softwarebibliothek, in Abbildung 5.18 „Equipment Extensions“ genannt, die die Implementierungen (*Physical Loading Door*) der UA-Objecttypen (`PhysicalLoadingDoorType`) beinhaltet. Das Equipment (*Physical Loading Door*) besteht aus Sensoren (*Contact Sensor*) und Aktuatoren (*Relay Actuator*), die als Interface-Erweiterungspunkte in der Bibliothek „Interface Extensions“ abgelegt werden. Da cyber-physische Adapter (CPA) zusätzliche Hardware benötigen, existiert eine Abhängigkeit zu den jeweiligen Bibliotheken im Paket „Hardware Bindings“. Die Klassen des Equipments benamen Methoden und Variablen durch Konvention nach denen des Informationsmodells. Gleiches gilt für die Namen der Klassen selbst. Beinhaltet beispielsweise die Instanz (`Loading_Door`) einer Ladetür (`PhysicalLoadingDoorType`) laut Modell die Methode `Open_Door`, besitzt die Klasse „PhysicalLoadingDoorType“ deren gleichnamige Implementierung.
Eine gesonderte Rolle spielen die Pakete des Stereotyps „testSuite“. Sie beinhalten Unit-Testfälle für die Equipment- und Interface-Implementierungen und verifizieren

[1] opcfoundation.org/developer-tools/specifications-unified-architecture/part-8-data-access (abgerufen am 22.11.2016)

deren Funktionalität. Während diese nur bestimmte Teile des Frameworks überprüfen, bieten Integrationstests einen vertikalen Durchstich. Mit der Implementierung eines UA-Clients können sie zur VMR verbinden und das korrekte Zusammenspiel von Methodenaufrufen, Kontextveränderungen und Rückkopplung im Bezug auf ein ganzheitliches Anlagenmodell testen.

5.4.4 Verteilung

Wird nun eine konkrete Anlage modernisiert, muss die physische Verteilung der Bausteine des Frameworks festgelegt werden. Es existiert ein Anlagenmodell, die entsprechenden Erweiterungen wurden implementiert und Unit- und Integrationstests verifizieren die virtuelle Maschinenrepräsentation (VMR) als als Anwendungssoftware.

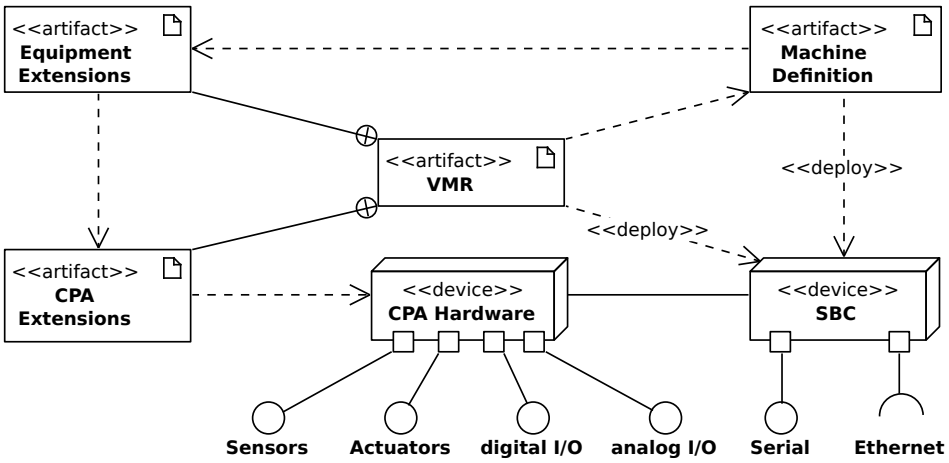


Abbildung 5.19: Verteilung des Frameworks

Abbildung 5.19 zeigt die Verteilung der Soft- und Hardwareartefakte an der Altanlage. Zentrales Element ist ein Einplatinencomputer (Single Board Computer, SBC), der eine serielle Schnittstelle bietet und einen Netzwerkanschluss benötigt. Er ist starr, z.B. via USB oder serieller Schnittstelle, mit einem oder mehreren Hardwarekomponenten für die cyber-physischen Adapter (CPA) verbunden. Diese bieten die Schnittstellen für Sensoren, Aktuatoren oder andere Geräte mit digitaler und analoger Ein- und Ausgabe. Die VMR wird auf dem Einplatinencomputer gespeichert und ist auf eine Anlagendefinition angewiesen. Für die Ausführung wird eine, der Implementierung entsprechende Laufzeitumgebung vorausgesetzt. Das An-

lagenmodell ist, je nach verwendeten Typen automatisierter Werkzeugkomponenten (`PhysicalLoadingDoorType`), abhängig von den Implementierungen des Equipments. Letzteres ist wiederum auf die jeweiligen Interface-Erweiterungen angewiesen.

6 Implementation

Der Entwurf der virtuellen Maschinenrepräsentation (VMR) wurde im vorangegangenen Kapitel vorgestellt. Nach der Design Science Research Methodology (vgl. Abschnitt 1.3) wird nun die Tauglichkeit des Konzepts anhand einer prototypischen Implementierung des Frameworks beschrieben. Neben verwendeten Technologien sind die technischen Details folgenden Beispiels im Fokus dieses Kapitels.

Die CNC-Drehmaschine aus dem Szenario S2 in Abschnitt 5.1.1 bietet eine anschauliche Grundlage für die exemplarische Anwendung der prototypischen Entwicklung. Sie besitzt eine serielle Schnittstelle für die Direct Numeric Control (DNC) der Antriebssteuerung, soll durch die VMR gekapselt und durch cyber-physische Rückkopplung zusätzlich automatisiert werden. In Abschnitt 4.1 wurden indirekte Überwachungsmethoden, auch mit Temperatursensorik, diskutiert. Im hier beschriebenen Beispiel soll die CNC angehalten und die Ladetür geöffnet werden, wenn die Temperatur des Werkzeugs der Drehmaschine einen kritischen Wert überschreitet.

6.1 Verwendete Hard- und Software

Benötigt wird ein Einplatinencomputer als Server für die VMR, eine Antriebssteuerung mit DNC sowie die Hardware für einen cyber-physischen Adapter (CPA). Temperatursensor und Relais, für den Verriegelungsmechanismus der Ladetür, müssen analoge und digitale Signale mit der CPA-Hardware austauschen können. Alle Hardwarekomponenten zur Umsetzung sind in Tabelle 6.1 zusammengestellt.

Tabelle 6.1: Verwendete Hardwarekomponenten

Komponente	**Beschreibung**
Einplatinencomputer	Raspberry Pi 3 Model B[1]
Antriebssteuerung	Smoothieboard 4XC[2]
cyber-physischer Adapter	GrovePi[3]
Temperatursensor	Grove - Temperatur- und Luftfeuchtigkeitssensor[4]
Verriegelungsrelais	Grove - Relay[5]

Der Raspberry Pi ist ein Einplatinencomputer der Ethernetanschluss, Wireless LAN Modul und USB Steckplätze, auch geeignet für serielle Kommunikation, besitzt. Er ist via USB direkt mit dem Smoothieboard verbunden. Das Smoothieboard 4XC ist ein Open Source Hardware CNC-Controller, der seriell CNC-Befehle aber auch ganze Programme entgegennimmt und für vier Schrittmotortreiber und Spindelkontrolle interpretiert (vgl. Abschnitt 2.1.2). Für die CPA-Hardware ist das GrovePi Erweiterungsboard für den Raspberry Pi geeignet. Es vereinheitlicht die Schnittstelle zu digitalen und analogen Sensoren und Aktuatoren des Grove-Systems für modulares, standardisiertes Prototyping[6]. Die Messung der Temperatur am Werkzeug der Drehmaschine wird mit einem kombinierten Temperatur- und Luftfeuchtigkeitssensor des Grove-Systems vorgenommen. Der Verriegelungsmechanismus der Ladetür wird mit einem entsprechenden Relais realisiert.

Das VMR-Framework wurde mit serverseitigem JavaScript auf Basis von „Node.js“[7] entwickelt. JavaScript ist, im Gegensatz zu C++ und Java, eine dynamisch typisierte Programmiersprache und wird primär für die Entwicklung von Nutzungsschnittstellen verwendet. Durch die serverseitige Plattform „Node.js“ etablierte sie sich auch im Backendbereich. Mit der ursprünglich von Google entwickelten Laufzeit-

[1] raspberrypi.org/products/raspberry-pi-3-model-b (abgerufen am 24.11.2016)
[2] smoothieware.org/smoothieboard (abgerufen am 24.11.2016)
[3] dexterindustries.com/grovepi (abgerufen am 24.11.2016)
[4] seeedstudio.com/Temp-p-745.html (abgerufen am 24.11.2016)
[5] seeedstudio.com/Relay-p-769.html (abgerufen am 24.11.2016)
[6] wiki.seeed.cc/Grove_System (abgerufen am 24.11.2016)
[7] nodejs.org (abgerufen am 23.11.2016)

umgebung „V8“[1] ist sie auf einem Einplatinencomputer ausreichend leistungsfähig (vgl. [KLD12]). Da JavaScript kein objektorientiertes Konzept verfolgt, wurde mit CoffeeScript[2] eine ergänzende Skriptsprache gewählt, die in JavaScript übersetzt und damit nicht zur Laufzeit interpretiert werden muss. Werkzeuge zur Modellierung des OPC UA Adressraums verwenden Codegenerierung für eine schablonenartige Struktur der Implementierung von Laufzeitlogik (z.B. der UaModeler[3] von Unified Automation). Dieser Schritt ist durch die Verwendung einer dynamisch typisierten Programmiersprache nicht notwendig und komprimiert das endgültige Softwareprodukt. Die aus dem Modellierungswerkzeug exportierte XML wird von der VMR gelesen und dynamisch mit den Implementierungen verbunden (vgl. Abschnitt 6.3). Durch das Ökosystem von „Node.js“ steht der Node Package Manager[4] mit quelloffenen Softwarebibliotheken zur Verfügung. Eine Auflistung der wichtigsten verwendeten ist in Tabelle 6.2 beschrieben[5].

Tabelle 6.2: Verwendete Softwarebibliotheken

Bibliothek	**Beschreibung**
node-opcua	Node.js OPC UA Implementierung für die OPC UA Server Komponente und das Laufzeitmodell.
node-grovepi	GrovePi-Anbindung für Sensoren und Aktuatoren des cyber-physischen Adapters.
serialport	Anbindung einer seriellen Schnittstelle für DNC und Feldgeräte mit RS-232.
watchjs	Veränderung von Objekte und Variablen überwachen für das MAPE-K Monitoring der Feedback Control.
mocha und chai	Test-Framework und Assertion-Bibliothek für Behaviour-driven Development

[1] developers.google.com/v8 (abgerufen am 23.11.2016)
[2] coffeescript.org (abgerufen am 23.11.2016)
[3] opcfoundation.org/products/view/uamodeler (abgerufen am 20.11.2016)
[4] npmjs.com (abgerufen am 24.11.2016)
[5] Die Pakete können unter npmjs.com/package/<Bibliothek> abgerufen werden.

Das Projekt „node-opcua“ implementiert den Großteil des OPC UA Stacks in JavaScript. Auf dessen GitHub-Seite[1] ist eine Tabelle mit dem Grad der Umsetzung der Spezifikationen dargestellt. Im Zuge dieser Arbeit wurde zu dem Projekt beigetragen. Zu dem GrovePi-Erweiterungsboard existiert eine Entwicklungsbibliothek für mehrere Programmiersprachen. JavaScript mit Node.js wird vollständig durch das Paket „node-grovepi“ unterstützt. Mitgeliefert werden grundsätzliche Implementierungen für analoge und digitale Sensoren und Aktuatoren, sowie das I2C-Bussystem[2] für die Kommunikation zwischen integrierten Schaltungen. Die Umsetzung des Prototypen der VMR wird weiterhin durch eine Bibliothek für die serielle Kommunikation (serialport), das Überwachen von Variablen und Objekten in JavaScript (watchjs) und Test-Bibliotheken (mocha, chai) ergänzt.

6.2 Laufzeitmodell und Erweiterungspunkte

Für die Umsetzung des Beispiels vom Anfang dieses Kapitels wurde ein OPC UA Modell der CNC-Werkzeugmaschine erstellt. Die für den Anwendungsfall relevanten Komponenten, Variablen und Methoden sind in Abbildung 6.1 dargestellt.

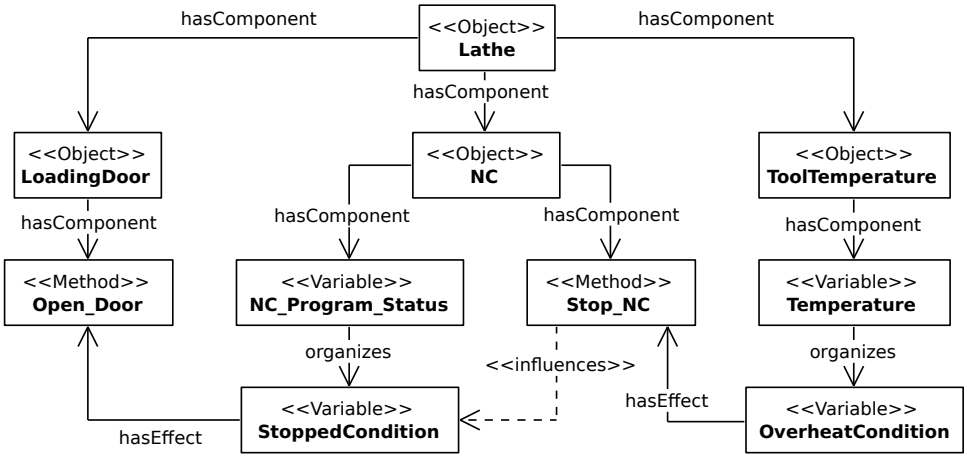


Abbildung 6.1: Laufzeitmodell des Prototypen

Im Modell besitzt die Drehmaschine eine Ladetür, eine NC-Steuerung und einen Sensor für die Temperatur des Werkzeugs. Letztere ist mit der Implementierung des Equipment-Erweiterungspunkts *PhysicalToolTemperatureType* verknüpft, die eine OPC UA Variable für die tatsächliche Temperatur am Werkzeug beschreibt. Unter

[1] github.com/node-opcua/node-opcua#supported-features (abgerufen am 24.11.2016)
[2] mikrocontroller.net/articles/I2C (abgerufen am 24.11.2016)

dieser Variable organisiert ist mit `OverheatCondition` ein Bereich für diesen Wert definiert der Überhitzung symbolisiert. Da „node-opcua“ noch keine Unterstützung für spezialisierte Referenzen bietet, wird `HasEffect`, aus der Spezifikation von OPC UA, als Synonym zu `HasPhysicalAction` verwendet (vgl. Abschnitt 5.2.1). Feedback Control reagiert auf die Veränderung der Temperatur und löst `Stop_NC` aus, sobald die Bedingung zutrifft (vgl. Abschnitt 6.3). Nach der Methodenausführung ändert sich die Variable `NC_Program_Status` durch die Implementierung des `PhysicalNCType`. Mit der daraufhin zutreffenden `StoppedCondition` wird aufgrund der `HasEffect`-Referenz `Open_Door` ausgeführt. Die Instanz der `PhysicalLoadingDoor` schließt dann das Relais über den `RelayActuator` – eine abgeleitete Klasse des `DigitalSensor` der GrovePi-Bibliothek.

Um einen Equipment-Erweiterungspunkt zu laden, liest die Model Control alle Objekte des Pakets „Processing Extensions“ (vgl. Abschnitt 5.4.3) und hält sie für die Bindung zu Modellelementen bereit. Exemplarisch ist der „PhysicalLoadingDoorType“ in Listing 6.1 beschrieben. Damit die Model Control OPC UA Variablen und Methoden binden kann, werden diese mit einem Dollar-Zeichen am Anfang des Bezeichners markiert. Die Ladetür besitzt im Informationsmodell nach Ayatollahi et al. (vgl. Abschnitt 5.2.1) eine Variable `Door_Status` sowie die Methoden `Close_Door` und `Open_Door`, die auch die Implementierung aus Listing 6.1 deklariert (Zeilen 3, 12 und 16). Sie beinhaltet den „RelayActuator“ aus dem Paket „Interface Extensions“ (vgl. Abschnitt 5.4.3), der mit den Parametern des `ConnectionIdentifier` (vgl. Abbildung 5.6 in Abschnitt 5.2.1) initialisiert wird (Zeile 6[1]). In Zeile 8 wird die Ladetür mit dem Öffnen des Relais initialisiert. Die Methode „$Close_Door“ – von der Model Control aufgerufen – delegiert die Anweisung an den „RelayActuator“ und aktualisiert den Wert der Variablen „$Door_Status“.

Listing 6.1: Implementierung des Processing-Erweiterungspunkts

```
1  class PhysicalLoadingDoorType
2
3    $Door_Status: null
4
5    constructor: (connectionIdentifier) ->
6      @DoorLock = new RelayActuator
           connectionIdentifier.DoorLock.pin
7
8      @onState  = 'closed'
9      @offState = 'open'
10
11     @$Open_Door()
12
13   $Close_Door: =>
```

[1] In CoffeeScript hat das „@“ am Anfang eines Bezeichners die gleiche Semantik wie „this“ in Java und referenziert die Instanz der umschließenden Klasse.

```
14      @DoorLock.on()
15      @$Door_Status = @onState
16
17    $Open_Door: =>
18      @DoorLock.off()
19      @$Door_Status = @offState
```

Der Equipment-Erweiterungspunkt wird durch Definition im Modell und Namenskonvention im Paket „Equipment Extensions“ an das Framework gebunden. Die „Interface Extension“ hingegen durch Aggregation im jeweiligen Equipment und das Erben von Klassen der GrovePi-Bibliothek. In Listing 6.2 solch eine Spezialisierung von „DigitalSensor“ dargestellt. Der „RelayActuator“ ist eine Komponente der „PhysicalLoadingDoor“ aus Listing 6.1. Zeile 4 deklariert den physischen Kommunikationskanal als ausgehend mit der Pin-Identifikation aus dem `ConnectionIdentifier`-Wert (vgl. Listing 6.1 Zeile 6). In den Zeilen 6 und 7 sind die Möglichkeiten des Aktuators beschrieben. Für das Öffnen und Schließen übergeben sie ein digitales Signal an die GrovePi-Implementierung.

Listing 6.2: Implementierung des Interface-Erweiterungspunkts

```
1  class RelayActuator extends DigitalSensor
2
3    constructor: (@pin) ->
4      @board.pinMode @pin, @board.OUTPUT
5
6    on:  => @write 1
7    off: => @write 0
```

6.3 Umsetzung der Komponenten

Die Implementierungen der Erweiterungspunkte müssen nun an das Modell der Maschine gebunden werden. Model Control initialisiert, ähnlich wie in Abbildung 5.14 aus Abschnitt 5.4.2, das Modell. Das Binden einer automatisierten Werkzeugkomponente, zum Beispiel der Ladetür (`LoadingDoor`), mit seiner Equipment-Implementierung ist, bis auf Ausnahme- und Fehlerbehandlung, in Listing 6.3 dargestellt. Die bereits bekannten `PhysicalConnections` werden mit denen des Objekts abgeglichen (Zeilen 4 ff.) und der in JSON definierte `ConnectionIdentifier`-Wert geladen (Zeile 8). Im Beispiel ist `{"pin":7}` der Wert für die Deklaration der Nummer der Steckverbindung des GrovePi-Erweiterungsboards für das Relais. Der „typeNode“ aus Zeile 10 beschreibt den Equipmenttyp des Laufzeitmodells, dessen Name die Implementierung (`PhysicalLoadingDoorType`) identifiziert. Diese wird in

Zeile 13 instanziiert und durch das Objekt referenziert. In den Zeilen 15-19 werden nun die OPC UA Methoden und Variablen mit deren Implementierung verknüpft (vgl. Abschnitt 6.2). Für die Rückkopplung wird beim Binden der Variablen, zum Beispiel `NC_Program_Status` as Abbildung 6.1, „listenTo“ der Feedback Control mit einer Referenz auf die Variable aufgerufen.

Listing 6.3: Binden einer Werkzeugkomponente mit der Implementierung

```
1  bind: (object) =>
2    conns = {}
3
4    for conn in @physicalConnections
5      if object.nodeId == conn.parent.nodeId
6        connName = conn.browseName.name
7        connIdentifier = @valueOf conn.connectionIdentifier
8        conns[connName] = JSON.parse connIdentifier
9
10   typeNode = @addressSpace.findNode object.typeDefinition
11   typeName = typeNode.browseName.name
12
13   object.instance = new equipment[typeName](conns)
14
15   if not @methodsOf(object.instance).isEmpty()
16     @bindMethodsTo object
17
18   if not @variablesOf(object.instance).isEmpty()
19     @bindVariablesTo object
```

Ein Auszug der Feedback Control in Listing 6.4 beschreibt die Implementierung der Phasen von MAPE-K (vgl. Abschnitt 5.3.4). Für das Monitoring werden zuerst alle Variablenzustände (`PhysicalCondition`) identifiziert. Besitzt die Variable mindestens einen zu verarbeitenden Zustand, wird sie mittels „watchjs“ überwacht (Zeile 5). Der letzte Parameter der Methode „watch“ ist ein Callback für den Fall der Veränderung. Innerhalb dessen iteriert Feedback Control über die möglichen Variablenzustände und überprüft (Analyze-Phase) ihre Qualifikation für den nächsten Schritt (Zeilen 6-7). In der Plan-Phase werden über die `HasEffect`-Referenz verbundene OPC UA Methoden gesucht und schlussendlich ausgeführt (Execute, Zeile 8). „hasBeenMet“ führt eine Fallunterscheidung nach dem Typ der `PhysicalCondition` durch. Bei booleschem Typ, Zeichenketten und anderen direkt Werten wird ein einfacher Abgleich mit dem tatsächlichen Zustand durchgeführt – Wert der Bedingung entspricht dem Wert der Variablen. Im Beispiel dieses Kapitels wird der Wert „Stop“ der Variable `NC_Program_Status` mit dem Wert der `StoppedCondition` vergleichen und gegebenenfalls die Methode `Open_Door` der `LoadingDoor` aufgerufen (vgl.

Abbildung 6.1). Der andere Ausgang der Fallunterscheidung von „hasBeenMet“ ist durch den kritischen Temperaturbereich des OPC UA Typs „Range“ abgedeckt. In diesem Fall wird überprüft, ob der tatsächliche Zustand (`Temperature`) innerhalb dieses Bereichs (`OverheatCondition`) liegt.

Listing 6.4: Rückkopplung durch Feedback Control

```
1 listenTo: (addressSpaceVariable, variable, equipment) =>
2   # [...] find physical conditions
3
4   if not physicalConditions.isEmpty()
5     watch equipment, variable, () =>
6       for condition in physicalConditions
7         if @hasBeenMet(condition, addressSpaceVariable)
8           @findAndExecuteActionFor condition
```

Die Implementierung der cyber-physischen Adapter und des OPC UA Servers sind delegierende Fassaden (Wrapper) um die Bibliotheken „node-grovepi“ und „node-opcua“ und können im Quellcode des Prototypen[1] nachgeschlagen werden.

6.4 Softwaretests für Erweiterungspunkte

Die Implementierung des Framework kann durch Unit- und Integrationstests überprüft werden. Da sie sich als Kern der virtuellen Maschinenrepräsentation (VMR) kaum verändert, werden diese Tests einmalig definiert. Interessanter sind die Erweiterungspunkte, die stetig wachsend, einem kontinuierlichen Testzyklus unterliegen müssen. Um das logische Verhalten der Implementierung von Equipment vor dem operativen Einsatz zu verifizieren, werden Unit-Tests wie in Listing 6.5 eingesetzt. Mit „mocha“ als JavaScript Test-Framework und „chai“ für die Assertions (Annahmen) kann das Verhalten des Equipments beschrieben werden. Die „describe“-Funktionen der Zeilen 1-2 definieren einen Kontext für die eigentlichen Tests, welche mit „it“ (Zeilen 4, 7 und 11) und einem passenden Bezeichner definiert werden.
Im ersten Testfall (Zeilen 4-5) des „PhysicalNCType“ soll der Start der numerischen Kontrolle in einem OPC UA Status-Code für ungültige Zustände münden, falls kein Programm für die Werkzeugmaschine festgelegt wurde. Die Variable „its“ hält dabei eine Instanz der zu testenden Implementierung. Im zweiten Fall (Zeilen 7-9) schlägt der Start aufgrund einer bereits laufenden Programmausführung fehl. Dafür wird die Variable „$NC_Program_Status“ auf „Active“ gesetzt, „$Start_NC“ aufgerufen und der ungültige Zustand festgestellt. Da „PhysicalNCType“ vom „Smoothie-

[1] github.com/phdd/diplom/tree/vmr (abgerufen am 25.11.2016)

boardActuator“ der „Interface Extensions“ abhängt, müssen Hilfskonstruktionen zur Überwachung der internen Methodenaufrufe an andere Objekte verwendet werden. In Zeile 12 wird dafür ein „Spion“ auf die Methode „start“ des „SmoothieboardActuator“ angesetzt und der aktuelle Zustand festgelegt (Zeile 13). Die Ausführung von „$Start_NC“ (Zeile 14) muss dann in einem Aufruf der Startfunktion des Aktuators münden.

Listing 6.5: Unit-Test des PhysicalNCType

```
1  describe 'PhysicalNCType', ->
2    describe 'operation start', ->
3
4      it 'should fail without a program assigned', ->
5        its.$Start_NC().statusCode.should.equal
             status.BadInvalidState
6
7      it 'should fail when already active', ->
8        its.$NC_Program_Status = 'Active'
9        its.$Start_NC().statusCode.should.equal
             status.BadInvalidState
10
11     it 'should succeed if inactive and assigned program', ->
12       start = chai.spy.on its.SmoothieboardActuator, 'start'
13       its.$NC_Program = 'test'
14       its.$Start_NC()
15       start.should.have.been.called.with 'test'
```

Dasselbe Prinzip kann auf die Implementierungen des „Interface Extension“ Pakets angewendet werden. Da die GrovePi-Bibliothek getestete Implementierungen für die meisten Sensoren und Aktuatoren bereitstellt, reduziert sich der Aufwand und wird hier nicht näher beschrieben. Spezielle Erweiterungen wie der „SmoothieboardActuator“ oder dessen Abhängigkeit vom „SerialPort“ müssen ebenfalls validiert werden, was kein Teil der prototypischen Umsetzung ist.
Integrations- sind den Unit-Tests ähnlich, greifen aber nicht auf die konkrete Implementierung zurück, sondern validieren das Verhalten der VMR mit der Perspektive eines OPC UA Clients.

7 Evaluation

These/Behauptung?

- Steigerung des Automatisierungsgrads durch Feedback Loop
- physische Anwesenheit des Werkers technisch überwinden (Remote-Control/-Programming)
  - „Echtzeitanalyse“ durch Werker auch entfernt mgl.
- Laufzeitmodell für online-Monitoring

Umsetzung?

- Proof of concept
- Case-Study mgl.?
- HIL-Simulation?
- Argumentative Evaluation des Artefakts nach DSRM i.O.
- Messung der Rückkopplungsgeschwindigkeit => node profiling (https://nodejs.org/en/docs/guides/si profiling/)
- Tabelle mit Komponenten und deren konzeptueller und implementeller Umsetzungsgrad
- Implementierung der PhysicalCondition bzgl. konstanter Variablen unvollständig; bisher String, Bool => node-opcua
- Conditions durch UA-Datentype ausdrucksstark (Range), aber weder logisch kombinierbar noch zeitlich einzuordnen
- Logik ist in der VMR, dennoch keine Echtzeit => Ausblick ECA RT Abbildung (RT OS + statische C/C++ Code Generierung für die Regeln beim Deployment)
  - Not-Aus etc. verbindungsorientiert anbinden
- Konflikte in ECA-Regeln
- der Action fehlen die Parameter der Methode
- Redundanz der PhysicalConditions (Closed für Relay-Aktuator/Contact-Sensor)

- imperative Regeln (NC Stopped => Open_Door()) vs. deklarative Systembeschreibung
  - Door_Status == „Closed“ impliziert Contact_Status == „Closed“
    - Was, wenn nicht => Kompensationsstrategie?
  - (RelayActuatorClosed)-[HasPhysicalEffect]->(ContactSensorClosed)->?
  - Imperativ beschreibt wie Feedback Control reagiert => alle Möglichkeiten müssen beschrieben werden
  - Deklarativ beschreibt was bei einem bestimmten Systemzustand erwartet wird und wie auf eine unerwartete Situation zu reagieren ist
- Test-Suite auf Unit-Tests beschränkt => Integrationstests?

Blocking Factors/mögliche Kritik?

- Leistung von embedded computing devices => siehe [GM16]
- Pi hat Grenzen bei CNC => Smoothieboard
- Smoothieboard als Maschinen-Adapter
  - Nachteil: Beobachten des Prozessfortschritts langsam (progress) => kann nicht in online-FB einbezogen werden
- Industriekomponenten nicht mit Smoothieboard vergleichbar
- Energieverbrauch?
- RAMI4.0-Konformität
- ECA für MAPE-K FCL
  - nur einfache Conditions im Informationsmodell abbildbar (bisher)
  - conflicts between policies can arise that are hard to detect [HM08]
  - Conditions können alle OPC UA Typen annehmen (auch Ranges)
- Integrationskonfiguration Teil des Informationsmodells => zentrale Modellierung aller Details mit etablierten Tools

7.1 Anforderungsabdeckung

7.2 Leistungsfähigkeit

QUANTITATIV

.

7.3 Diskussion

KRITIK

- keine quantitative Evaluation

8 Zusammenfassung

8.1 Schlussfolgerung

- Forschungsfragen aufgreifen!

8.2 Ausblick

- Eignung anderer SBC und Vergleich mit dieser Impl.
- Regelbasierte Rückkopplung durch intelligentere ersetzen [Sei+16]
- externe FCL (Cyber- und Configuration-Level nach [LBK15])
- Prozess-Engine nach [SHS15]
- Modulimplementierung (Dürkop) bzgl. [BHJ11] oder [ILL11] konkretisieren
- TSN Ethernet trägt RT vertikal nach oben
- Nutzungsschnittstellen (Unified Automation Android App)
- Wise-ShopFloor mit OPC UA horizontal integrieren
- Steuerungsalternative OPC UA Programs ([OPC14])
- Fog mit OPC UA und WS (vgl. [Bon+12; AH16])
- MDSD mit [PFK16]
- CNC ersetzen durch STEP-NC? [Suh+03; XN06; Xu06; XLY06]
- AutomationML und OPC UA (vgl. [OPC14])
- Welcher G-Code Befehl korrelliert auf welche Weise mit welchen gemessenen Werten? [Dow+16]
- Case-Study!!!

A Anhang

Abbildungen

Programmcode

Tabellen

Literatur

[AE15] Peter Adolphs und Ulrich Epple. *Statusreport: Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)*. Technischer Bericht April. 2015 (siehe Seiten 32, 40, 52).

[AH16] Mohammad Aazam und Eui-Nam Huh. „Fog Computing: The Cloud-IoT/IoE Middleware Paradigm“. In: *IEEE Potentials* 35.3 (2016), Seiten 40–44. DOI: 10.1109/MPOT.2015.2456213 (siehe Seite 83).

[Alt94] Leo Alting. *Manufacturing Engineering Processes*. 1994. ISBN: 9780130743169 (siehe Seite 44).

[Amb+15] Nitin Ambhore, Dinesh Kamble, Satish Chinchanikar u. a. „Tool Condition Monitoring System: A Review“. In: *Materials Today: Proceedings* 2.4-5 (2015), Seiten 3419–3428. DOI: 10.1016/j.matpr.2015.07.317 (siehe Seiten 25, 30, 31).

[Aya+13] Iman Ayatollahi, Burkhard Kittl, Florian Pauker u. a. „Prototype OPC UA Server for Remote Control of Machine Tools“. In: *International Conference on Innovative Technologies*. Band 1009. 12. 2013, Seiten 73–76 (siehe Seiten 17, 26, 27, 32, 41, 44, 48, 50).

[Ber15] Simon Bergweiler. „Intelligent Manufacturing based on Self-Monitoring Cyber-Physical Systems“. In: *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies* c (2015), Seiten 108–113. DOI: 10.13140/RG.2.1.2762.9281 (siehe Seiten 5, 23).

[BHJ11] Bernard Bony, Michael Harnischfeger und Francois Jammes. „Convergence of OPC UA and DPWS with a cross-domain data model“. In: *IEEE International Conference on Industrial Informatics (INDIN)* (2011), Seiten 187–192. DOI: 10.1109/INDIN.2011.6034860 (siehe Seiten 38, 56, 83).

[Bon+12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu u. a. „Fog Computing and Its Role in the Internet of Things“. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012, Seiten 13–16. DOI: 10.1145/2342509.2342513 (siehe Seiten 27, 28, 83).

[Bor+14] Theodor Borangiu, Silviu Raileanu, Damien Trentesaux u. a. „Distributed manufacturing control with extended CNP interaction of intelligent products“. In: *Journal of Intelligent Manufacturing* 25.5 (2014), Seiten 1065–1075. DOI: 10.1007/s10845-013-0740-3 (siehe Seite 24).

[Cân+10] Gonçalo Cândido, François Jammesy, José Barata De Oliveira u. a. „SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications“. In: *IEEE International Conference on Industrial Informatics (INDIN)* (2010), Seiten 598–603. DOI: 10.1109/INDIN.2010.5549676 (siehe Seite 38).

[Den+15] Jonathan Denner, Peter Heinrich, Constantin Heldman u. a. *Project Deliverable 1.2 - First version of requirements of workers and organisations*. Technischer Bericht. 2015. URL: http://www.facts4workers.eu (siehe Seite 45).

[Dow+16] Jonathan Downey, Denis O'Sullivan, Miroslaw Nejmen u. a. „Real Time Monitoring of the CNC Process in a Production Environment- the Data Collection & Analysis Phase“. In: *Procedia CIRP* 41 (2016), Seiten 920–926. DOI: 10.1016/j.procir.2015.12.008 (siehe Seiten 30, 31, 54, 83).

[DP11] Amit Deshpande und Ron Pieper. „Legacy Machine Monitoring Using Power Signal Analysis“. In: *ASME 2011 International Manufacturing Science and Engineering Conference, Volume 2*. ASME, 2011, Seiten 207–214. DOI: 10.1115/MSEC2011-50019 (siehe Seiten 2, 25, 30, 31).

[Dür+14] Lars Dürkop, Henning Trsek, Jens Otto u. a. „A field level architecture for reconfigurable real-time automation systems“. In: *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS* (2014). DOI: 10.1109/WFCS.2014.6837601 (siehe Seiten 11, 37, 41, 52, 56, 57).

[FC07] Antnio Ferrolho und Manuel Crisostomo. „Intelligent Control and Integration Software for Flexible Manufacturing Cells“. In: *IEEE Transactions on Industrial Informatics* 3.1 (2007), Seiten 3–11. DOI: 10.1109/TII.2006.890529 (siehe Seiten 2, 27, 31, 60).

[FCL05] António Ferrolho, Manuel Crisostomo und Mario Lima. „Intelligent control software for industrial CNC machines“. In: *2005 IEEE International Conference on Intelligent Engineering Systems, 2005. INES '05.* IEEE, 2005, Seiten 267–272. DOI: 10.1109/INES.2005.1555171 (siehe Seiten 31, 41, 44, 54, 60).

[Fra16] Fraunhofer IPK. „RetroNet - Praxisnahe Brücke in die Industrie 4.0“. In: *FUTUR* (2016), Seite 8. URL: https://issuu.com/claudiaengel/docs/futur_1_2016 (siehe Seiten 1, 39).

[Gau+14] Jürgen Gausemeier, Roman Dumitrescu, Jürgen Jasperneite u. a. *Auf dem Weg zu Industrie 4.0: Lösungen aus dem Spitzencluster it's OWL*. Technischer Bericht. Paderborn: it's OWL Clustermanagement GmbH, 2014. URL: http://www.its-owl.de/fileadmin/PDF/Industrie_4.0/Auf_dem_Weg_zu_Industrie_4.0_-_Loesungen_aus_dem_Spitzencluster_its_OWL_RGB.pdf (siehe Seite 1).

[Gee11] Guido L. Geerts. „A design science research methodology and its application to accounting information systems research". In: *International Journal of Accounting Information Systems* 12.2 (2011), Seiten 142–151. DOI: 10.1016/j.accinf.2011.02.004 (siehe Seite 5).

[GM16] Sergej N. Grigoriev und Georgi M. Martinov. „An ARM-based Multichannel CNC Solution for Multi-tasking Turning and Milling Machines". In: *Procedia CIRP* 46 (2016), Seiten 525–528. DOI: 10.1016/j.procir.2016.04.036 (siehe Seiten 4, 28, 54, 80).

[GML00] a. Gunasekaran, H. B. Marri und B. Lee. „Design and Implementation of Computer Integrated Manufacturing in Small and Medium-Sized Enterprises: A Case Study". In: *International Journal of Advanced Manufacturing Technology* 16.1 (2000), Seiten 46–54. DOI: 10.1007/PL00013131 (siehe Seite 2).

[Gro08] Mikell P. Groover. *Automation, production systems, and computer-integrated manufacturing*. Prentice Hall, 2008, Seite 815. ISBN: 0132393212 (siehe Seite 3).

[GSL14] Dominic Gorecky, Mathias Schmitt und Matthias Loskyll. „Mensch-Maschine-Interaktion im Industrie 4.0-Zeitalter". In: *Industrie 4.0 in Produktion, Automatisierung und Logistik* (2014), Seiten 525–542. DOI: 10.1007/978-3-658-04682-8_26 (siehe Seiten 25–27).

[Hay85] Barbara Hayes-Roth. „A Blackboard Architecture for Control". In: *Artificial Intelligence* 26.3 (1985), Seiten 251–321. ISSN: 09574174. DOI: 10.1016/0004-3702(85)90063-3 (siehe Seite 35).

[Hir00] Andreas Hirsch. *Werkzeugmaschinen Grundlagen: Lehr- und Übungsbuch*. Vieweg, 2000. ISBN: 978-3528049508 (siehe Seiten 12–14, 26, 31).

[HLG15] Berthold Heinrich, Petra Linke und Michael Glöckler. *Grundlagen Automatisierung*. Springer, 2015. DOI: 10.1007/978-3-658-05961-3 (siehe Seiten 14–16, 26).

[HM08] Markus C. Huebscher und Julie A. McCann. „A survey of autonomic computing - degrees, models, and applications". In: *ACM Computing Surveys* 40.3 (2008), Seiten 1–28. DOI: 10.1145/1380584.1380585. URL: http://portal.acm.org/citation.cfm?doid=1380584.1380585 (siehe Seite 80).

[Hop14] Stefan Hoppe. „Standardisierte horizontale und vertikale Kommunikation: Status und Ausblick“. In: *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, Seiten 325–341. DOI: 10.1007/978-3-658-04682-8_16 (siehe Seiten 2, 20, 32).

[HR15] Veit Hammerstingl und Gunther Reinhart. „Unified Plug&Produce architecture for automatic integration of field devices in industrial environments“. In: *Proceedings of the IEEE International Conference on Industrial Technology* 2015-June.June (2015), Seiten 1956–1963. DOI: 10.1109/ICIT.2015.7125383 (siehe Seiten 10, 11, 47).

[IBM06] IBM. *An architectural blueprint for autonomic computing*. Technischer Bericht June. 2006. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.150.1011 (siehe Seite 22).

[IFT] IFT TU Wien. *OPC4Factory*. URL: https://www.ift.at/forschung/foschungsprojekte/opc4factory/ (besucht am 28. 08. 2016) (siehe Seite 39).

[ILL11] M. Jorge A Garcia Izaguirre, Andrei Lobov und Jose L Martinez Lastra. „OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing“. In: *IEEE International Conference on Industrial Informatics (INDIN)* (2011), Seiten 205–210. ISSN: 19354576. DOI: 10.1109/INDIN.2011.6034874 (siehe Seiten 38, 47, 83).

[ISW] ISW Universität Stuttgart. *piCASSO*. URL: http://www.projekt-picasso.de/ (besucht am 27. 08. 2016) (siehe Seite 40).

[KLD12] Matthias Kovatsch, Martin Lanter und Simon Duquennoy. „Actinium: A RESTful runtime container for scriptable internet of things applications“. In: *Proceedings of 2012 International Conference on the Internet of Things, IOT 2012* (2012), Seiten 135–142. DOI: 10.1109/IOT.2012.6402315 (siehe Seite 71).

[Kru95] Philippe Kruchten. „Architectural Blueprints - The ”4+1”View Model of Software Architecture“. In: *IEEE Software* 12.6 (1995), Seiten 42–50. DOI: 10.1145/216591.216611 (siehe Seite 43).

[KWH13] Henning Kargermann, Wolfgang Wahlster und Johannes Helbig. *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0*. Technischer Bericht April. 2013. URL: https://www.bmbf.de/files/Umsetzungsempfehlungen_Industrie4_0.pdf (siehe Seiten 16, 28).

[KXU11] Rüdiger Klein, Jingquan Xie und Andrij Usov. „Complex events and actions to control cyber-physical systems“. In: *Proceedings of the 5th ACM international conference on Distributed event-based system - DEBS ’11* (2011), Seite 29. DOI: 10.1145/2002259.2002265. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-80051938493%7B%5C&%7DpartnerID=tZOtx3y1%7B%5C%%7D5Cnhttp:

//dl.acm.org/citation.cfm?id=2002265%7B%5C%%7D5Cnpapers2://publication/uuid/B3B9E284-9877-4B04-B8D0-ECF61DC4829C%7B%5C%%7D5Cnhttp://portal.acm.org/citation.cfm?doid=2002259.2002265 (siehe Seite 58).

[LBK15] Jay Lee, Behrad Bagheri und Hung An Kao. „A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems“. In: *Manufacturing Letters* 3.September 2016 (2015), Seiten 18–23. DOI: 10.1016/j.mfglet.2014.12.001 (siehe Seiten 24, 38, 39, 41, 52, 54, 57, 58, 83).

[Lee06] Soo-Yen Lee. „In-process tool condition monitoring systems in CNC turning operations“. Dissertation. Iowa State University, 2006 (siehe Seiten 30, 31).

[Lee08] Edward A. Lee. „Cyber Physical Systems: Design Challenges“. In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)* (2008), Seiten 363–369. DOI: 10.1109/ISORC.2008.25 (siehe Seiten 21, 25, 28).

[Lee99] Tina Y. Lee. „Information modeling: From design to implementation“. In: *Proceedings of the second world manufacturing congress*. 1999, Seiten 315–321 (siehe Seite 17).

[LHL04] Steven Y. Liang, Rogelio L. Hecker und Robert G. Landers. „Machining Process Monitoring and Control: The State-of-the-Art“. In: *Journal of Manufacturing Science and Engineering* 126.2 (2004), Seite 297. DOI: 10.1115/1.1707035 (siehe Seiten 1, 30, 31, 54).

[Lin15] Petra Linke. „Grundlagen zur Automatisierung“. In: *Grundlagen Automatisierung*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, Seiten 1–28. DOI: 10.1007/978-3-658-05961-3_1 (siehe Seiten 7–9).

[LS01] Gunter Lay und Elna Schirrmeister. *Sackgasse Hochautomatisierung? Praxis des Abbaus von Overengineering in der Produktion*. Technischer Bericht. Karlsruhe: Fraunhofer-Institut für System- und Innovationsforschung, 2001. URL: http://hdl.handle.net/10419/29534 (siehe Seite 2).

[Mah03] Nitaigour Premchand Mahalik, Herausgeber. *Fieldbus Technology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. DOI: 10.1007/978-3-662-07219-6 (siehe Seite 11).

[Mil14] Joachim Milberg. *Trends in der fabrik*. Technischer Bericht. Institut für Produktion und Logistik, 2014. URL: http://www.ifpconsulting.de/media/pdf/ifp_ku_trends_2014sm_abstract.pdf (siehe Seite 1).

[Moc+12] Luis E Gonzalez Moctezuma, Jani Jokinen, Corina Postelnicu u. a. „Retrofitting a factory automation system to address market needs and societal changes“. In: *IEEE International Conference on Industrial Informatics (INDIN)* (2012), Seiten 413–418. ISSN: 19354576. DOI: 10.1109/INDIN.2012.6301202 (siehe Seiten 35–37, 41, 52, 54, 59).

[Mon+16] L. Monostori, B. Kádár, T. Bauernhansl u. a. „Cyber-physical systems in manufacturing“. In: *CIRP Annals - Manufacturing Technology* 65 (2016), Seiten 621–641. ISSN: 00078506. DOI: 10.1016/j.cirp.2016.06.005 (siehe Seite 40).

[OPC14] OPC Foundation. *OPC Unified Architecture - Wegbereiter der 4. industriellen (R)Evolution*. Technischer Bericht. 2014. URL: https://opcfoundation.org/wp-content/uploads/2014/03/OPC_UA_I_4.0_Wegbereiter_DE_v2.pdf (siehe Seiten 16–18, 26, 32, 56, 83).

[Pau+13] Florian Pauker, T Weiler, I Ayatollahi u. a. „Information Architecture for Reconfigurable production systems“. In: *DAAAM International Scientific Book 2013* January (2013), Seiten 873–886. DOI: 10.2507/daaam.scibook.2013.53 (siehe Seiten 11, 34–36, 41, 52, 56).

[Pau14] Florian Pauker. „OPC UA for machine tending industrial robots - Prototypic development of an OPC UA server for ABB industrial robots“. In: October (2014). DOI: 10.15224/978-1-63248-031-6-155 (siehe Seiten 32, 33).

[PFK16] Florian Pauker, Thomas Frühwirth und Burkhard Kittl. „A systematic approach to OPC UA information model design“. In: *49th CIRP Conference on Manufacturing Systems*. May. 2016 (siehe Seite 83).

[PO] PLCopen und OPC Foundation. *Introduction in the PLCopen and OPC UA Communications Model*. Technischer Bericht. URL: http://www.plcopen.org/pages/tc4_communication/downloads/intro_plcopen_opc_v3.pdf (siehe Seite 48).

[PO10] PLCopen und OPC Foundation. *OPC UA Information Model for IEC 61131-3*. 2010. URL: http://www.plcopen.org/pages/tc4_communication/downloads/plcopen_opcua_information_model_%20v100.pdf (siehe Seite 16).

[Pre94] Wolfgang Pree. „Meta Patterns - A Means For Capturing the Essentials of Reusable Object-Oriented Design“. In: *Proceedings of the 8th European Conference on Object-Oriented Programming*. Springer Verlag, 1994, Seiten 150–162. DOI: 10.1007/BFb0052181 (siehe Seite 59).

[Sei+16] Ronny Seiger, Steffen Huber, Peter Heisig u. a. „Enabling Self-adaptive Workflows for Cyber-physical Systems“. In: *Enterprise, Business-Process and Information Systems Modeling*. Springer International Publishing, 2016, Seiten 3–17. DOI: 10.1007/978-3-319-39429-9_1 (siehe Seiten 23, 83).

[SG16] David Siepmann und Norbert Graef. „Industrie 4.0 – Grundlagen und Gesamtzusammenhang“. In: *Einführung und Umsetzung von Industrie 4.0*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, Seiten 17–82. DOI: 10.1007/978-3-662-48505-7_2 (siehe Seiten 1, 27).

[SHS15] Ronny Seiger, Steffen Huber und Thomas Schlegel. „PROtEUS: An Integrated System for Process Execution in Cyber-Physical Systems“. In: *Enterprise, Business-Process and Information Systems Modeling*. 214. Springer, 2015, Seiten 265–280. DOI: 10.1007/978-3-319-19237-6_17 (siehe Seiten 23, 83).

[Smi08] Peter Smid. *CNC programming Handbook*. Industrial Press, 2008. ISBN: 9780831133474 (siehe Seite 13).

[SSG13] Jochen Schlick, Peter Stephan und Thomas Greiner. „Kontext, Dienste und Cloud Computing - Eigenschaften und Anwendungen cyber-physischer Systeme“. In: *atp edition* 55.4 (2013), Seiten 32–41 (siehe Seite 23).

[Suh+03] S.H. Suh, B.E. Lee, D.H. Chung u. a. „Architecture and implementation of a shop-floor programming system for STEP-compliant CNC“. In: *Computer-Aided Design* 35.12 (2003), Seiten 1069–1083. DOI: 10.1016/S0010-4485(02)00179-3 (siehe Seite 83).

[Tet+10] R. Teti, K. Jemielniak, G. O'Donnell u. a. „Advanced monitoring of machining operations“. In: *CIRP Annals - Manufacturing Technology* 59.2 (2010), Seiten 717–739. DOI: 10.1016/j.cirp.2010.05.010 (siehe Seiten 29–31, 54).

[TGP08] Ying Tan, Steve Goddard und Lance C. Pérez. „A prototype architecture for cyber-physical systems“. In: *ACM SIGBED Review* 5.1 (2008), Seiten 1–2. DOI: 10.1145/1366283.1366309. URL: http://www.cs.virginia.edu/sigbed/archives/2008-01/Tan.pdf (siehe Seite 22).

[Ver13] Verein Deutscher Ingenieure e.V. *Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation*. Technischer Bericht. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, 2013. URL: https://www.vdi.de/technik/fachthemen/mess-und-automatisierungstechnik/fachbereiche/thesen-und-handlungsfelder/ (siehe Seiten 23, 24).

[Wan+04] Lihui Wang, Peter Orban, Andrew Cunningham u. a. „Remote real-time CNC machining for web-based manufacturing“. In: *Robotics and Computer-Integrated Manufacturing* 20.6 (2004), Seiten 563–571. DOI: 10.1016/j.rcim.2004.07.007 (siehe Seiten 1, 2, 25, 27, 34, 41).

[WG14] Sebastian Wätzoldt und Holger Giese. „Classifying distributed self-∗ systems based on runtime models and their coupling“. In: *CEUR Workshop Proceedings* 1270 (2014), Seiten 11–20. ISSN: 16130073 (siehe Seiten 23, 51).

[WJN15] S. Windmann, F. Jungbluth und O. Niggemann. „Ansätze zur Erhöhung der Flexibilität und Vernetzbarkeit industrieller Steuerungen“. In: *Branchentreff der Mess- und Automatisierungstechnik (Automation)*. 2015 (siehe Seiten 3, 33, 41, 49, 55).

[WUS13] Danny Weyns, M. Usman Iftikhar und Joakim Soderlund. „Do external feedback loops improve the design of self-adaptive systems? A controlled experiment“. In: *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (2013), Seiten 3–12. ISSN: 21572305. DOI: 10.1109/SEAMS.2013.6595487 (siehe Seite 58).

[XLY06] X.W. Xu, Lihui Lihui Wang und Yiming Yiming Rong. „STEP-NC and function blocks for interoperable manufacturing“. In: *IEEE Transactions on Automation Science and Engineering* 3.3 (2006), Seiten 297–308. DOI: 10.1109/TASE.2005.862147 (siehe Seiten 13, 83).

[XN06] X.W. Xu und S.T. Newman. „Making CNC machine tools more open, interoperable and intelligent - a review of the technologies“. In: *Computers in Industry* 57.2 (2006), Seiten 141–152. DOI: 10.1016/j.compind.2005.06.002 (siehe Seite 83).

[Xu06] X.W. Xu. „Realization of STEP-NC enabled machining“. In: *Robotics and Computer-Integrated Manufacturing* 22.2 (2006), Seiten 144–153. DOI: 10.1016/j.rcim.2005.02.009 (siehe Seite 83).

[Yon05] Chi Yonglin. „An evaluation space for open architecture controllers“. In: *The International Journal of Advanced Manufacturing Technology* 26.4 (2005), Seiten 351–358. DOI: 10.1007/s00170-004-2111-x (siehe Seite 34).