Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 114

# Realisierung eines verteilten IT-Systems auf Basis von OPC UA

Thomas Pfeiffer

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer/in:** | Prof. Dr. rer. nat. Stefan Wagner,<br>Prof. Dr.-Ing. Michael Weyrich |
| **Betreuer/in:** | Dr. rer. nat. Ivan Bogicevic,<br>Andreas Zeller, M.Sc. |
| **Beginn am:** | 11. Juli 2016 |
| **Beendet am:** | 10. Januar 2017 |
| **CR-Nummer:** | I.2.1, C.2.4, C.5.0, D.2.12 |

# Kurzfassung

Der Markt für produzierende Unternehmen wird immer komplexer und erfordert eine zunehmende Flexibilität. Mit der Vision Industrie 4.0 - auch als vierte industrielle Revolution bezeichnet – wurde im Jahr 2011 auf die geänderten Anforderungen reagiert. Durch den steigenden Automatisierungsgrad sowie dem Trend zu intelligenten, effizienteren, flexibleren und stark vernetzten Produktionsanlagen wird die Informatik für die Automatisierungstechnik immer wichtiger. Ziel der vorliegenden Masterarbeit war die Realisierung eines verteilten IT-Systems auf der Basis von OPC UA. Diese Arbeit beschreibt zunächst die Grundlagen von Industrie 4.0, der Machine-to-Machine Kommunikation sowie der OPC Unified Architecture. Anhand eines Pflichtenheftes wurde das Systemmodell sowie eine Software-Systemarchitektur für das verteilte IT-System konzipiert und implementiert. Ergebnis der Masterarbeit ist ein OPC UA Netzwerk mit mehreren realen und virtuellen Produktionseinheiten (OPC UA Server) sowie einer Workstation, auf der eine Anwendung zur Visualisierung und Administration ausgeführt wird. Das entwickelte System erfüllt sämtliche gestellten Anforderungen, zeigt die vielseitige Einsetzbarkeit von OPC Unified Architecture und kann als Basis für weitere Arbeiten genutzt werden.

# Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während meines Studiums und insbesondere bei der Anfertigung dieser Masterarbeit unterstützt haben.

Besonderer Dank gilt Herrn Prof. Dr. rer. nat. Stefan Wagner und Herrn Prof. Dr.-Ing. Michael Weyrich, die mir ermöglicht haben, eine Masterarbeit zu den hochaktuellen Themen Industrie 4.0 und OPC Unified Architecture anzufertigen. Ganz herzlich bedanken möchte ich mich bei Herrn Dr. rer. nat. Ivan Bogicevic und Herrn Andreas Zeller (M.Sc.) für die hervorragende Betreuung. Vielen Dank für Verbesserungsvorschläge, Anregungen und Ihr Vertrauen.

Ein besonderes Dankeschön auch an meine Eltern, die mich während meines Studiums unterstützt haben.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

Der Markt für produzierende Unternehmen wurde in den vergangenen Jahren immer komplexer. Typische Herausforderungen dieser Entwicklung sind eine steigende Produktvielfalt, die Möglichkeit der Produktindividualisierung, abnehmende Losgrößen und kurze Lieferzeiten. Zeitnahe Produktionsanläufe, wirtschaftliche und bedarfssynchrone Produktion sowie Ressourceneffizienz sind daher für den Wettbewerb unerlässlich. Als Reaktion auf diese Entwicklung, wurde 2011 die Vision einer vierten industriellen Revolution – Industrie 4.0 - vorgestellt. Ziel dieser Vision ist die intelligente Vernetzung der Anlagen, Prozesse, Produkte und Menschen in einem cyber-physischen Produktionssystem. Die realen Entitäten des Produktionssystems werden erfasst und in eine virtuelle Welt - das Internet der Dinge - übertragen.

Das Ergebnis ist ein intelligentes Produktionsnetzwerk, in dem die Produktionssysteme und intelligente Produkte in Echtzeit erfasst, gesteuert, dokumentiert, analysiert und weltweit abrufbar sind. Die Datenerfassung der realen Zustände erfolgt mittels vieler dezentraler und vernetzter Sensoren. Für die Speicherung der Daten werden zentrale und dezentrale Clouds verwendet. Das dadurch entstehende Potenzial ist enorm. So ist beispielsweise ein echtzeitfähiges Monitoring auf Basis eines Produktionssystemabbildes möglich. Ein intelligentes Produktionsnetzwerk ermöglicht zudem einen schnellen Zugriff, den flexiblen Einsatz von Ressourcen sowie eine höhere Auslastung.

Durch die Intelligenz des Netzes ist es möglich, mehrere Basis-Dienste zu einem hochwertigeren Dienst bzw. intelligenten Produkt zusammenzufassen. Eine große Herausforderung hierbei ist die schnelle und einfache Orchestrierung von mehreren oder hochwertigeren Diensten. Denkbar sind auch unternehmensübergreifende Dienste und Anwendungen mit kooperativen Prozessen innerhalb eines Geschäftsnetzwerks (Interoperabilität).

Für die intelligente Vernetzung der Komponenten eines Produktionssystems sind industrielle Machine-to-Machine (M2M) Kommunikationsprotokolle erforderlich. OPC Unified Architecture ist eine service-orientierte Architektur (SOA) und wurde speziell für die industrielle Machine-to-Machine (M2M) Kommunikation entwickelt. OPC UA spezifiziert eine serviceorientierte Architektur für die herstellerübergreifende und plattformunabhängige industrielle M2M-Kommunikation. Die Kommunikation basiert auf standardisierten Web-Technologien und erfolgt nach dem Client-Server-Prinzip. OPC UA Server bilden die Daten von Prozessen sowie die Systemkomponenten mit Knoten und Referenzen in einem Adressraum ab. Aufgrund der Skalierbarkeit kann OPC UA auf allen Ebenen der Automatisierungspyramide eingesetzt werden.

Abbildung 1.1 visualisiert das Thema Industrie 4.0 in einer Wortwolke.


**Abbildung 1.1:** Wortwolke zum Thema Industrie 4.0

Ziel der Masterarbeit ist die Realisierung eines verteilten IT-Systems auf Basis von OPC Unified Architecture. Hierzu wird ein OPC UA Netzwerk mit mehreren realen OPC UA Servern, virtuellen OPC UA Servern und einer Workstation erstellt. Die realen OPC UA Server laufen auf einem Einplatinencomputer (Raspberry Pi) und stellen verschiedene Dienstleistungen mit Hilfe von Sensoren und Aktoren bereit. Die virtuellen OPC UA Server bieten nur fiktive Dienstleistungen an. Auf der Workstation wird ein Programm zur Visualisierung und Administration des OPC UA Netzwerks ausgeführt. Mit einer grafischen Oberfläche kann der Anwender die registrieren OPC UA Server (Produktionseinheiten) im Netzwerk auflisten, Produktionsaufträge in dem verteilten IT-System erstellen, den Status eines Produktionsauftrags abfragen, virtuelle OPC UA Server mit fiktiven Dienstleistungen generieren und virtuelle OPC UA Server administrieren. Die Vernetzung der einzelnen Geräte erfolgt mit einem Router oder Switch.

## Gliederung

Die Arbeit ist in folgender Weise gegliedert:

**Kapitel 2 – Grundlagen:** Nach der Einleitung werden in diesem Kapitel die Grundlagen zu den Themen Industrie 4.0 und der Machine-to-Machine-Kommunikation (M2M) beschrieben. Schwerpunkt der Grundlagenarbeit ist die detaillierte Betrachtung des Kommunikationsprotokolls OPC Unified Architecture.

**Kapitel 3 – Pflichtenheft:** Die Ergebnisse der Definitionsphase werden im Pflichtenheft aufgezeigt. Das Pflichtenheft definiert Ziele, Einsatzbereiche, Umgebung, funktionale und nichtfunktionale Anforderungen, Qualitätsziele, Testfälle sowie die eingesetzte Entwicklungsumgebung für die Realisierung des verteilten IT-Systems. Weiterhin wird das Vorgehensmodell zur Durchführung der Masterarbeit festgelegt.

**Kapitel 4 – Systemmodell:** Das Systemmodell bildet das Fachkonzept für die Realisierung des verteilten IT-Systems. Im Fachkonzept sind die Anwendungsfälle für das verteilte IT-System vollständig, eindeutig und konsistent beschrieben. Weiterhin wird der Aufbau und Ablauf der grafischen Anwendung mit Mockups visualisiert und beschrieben.

**Kapitel 5 – Software-Systemarchitektur:** Die Software-Systemarchitektur spezifiziert die Architekturen, Module, Softwarekomponenten, Schnittstellen und das Datenbankschema des verteilten IT-Systems. Weiterhin werden in diesem Teil der Arbeit die grundlegenden Entwurfsentscheidungen für die Implementierung der Applikationen dokumentiert.

**Kapitel 6 – Implementierung:** In diesem Kapitel werden die Ergebnisse der Implementierungsphase dokumentiert und zusammengefasst.

**Kapitel 7 – Zusammenfassung und Ausblick:** Das abschließende Kapitel fasst die Ergebnisse der Masterarbeit zusammen und zeigt die Potenziale des verteilten IT-Systems auf Basis von OPC Unified Architecture. Weiterhin werden Anknüpfungspunkte für weiterführende Arbeiten dargelegt.

# 2 Grundlagen

Im ersten Teil dieses Kapitels werden die Grundlagen zum Thema Industrie 4.0 beschrieben. Teil zwei beschäftigt sich allgemein mit der Machine-to-Machine-Kommunikation (M2M). Aufbauend auf diesen Erkenntnissen wird im nachfolgenden Hauptteil des Grundlagendokuments der Schwerpunkt auf das Kommunikationsprotokoll OPC Unified Architecture gelegt.

## 2.1 Industrie 4.0

Kapitel 2.1 beschäftigt sich mit den Grundlagen von Industrie 4.0. In der Einleitung wird der Begriff Industrie 4.0 definiert und dessen Ursprung erörtert. Anschließend wird die Entwicklung zu Industrie 4.0 mit den vier industriellen Revolutionen beschrieben. Nachfolgend werden Eigenschaften und Ziele von Industrie 4.0 in der Produktion aufgezeigt und erläutert. Am Ende des Kapitels wird auf das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0) eingegangen.

### 2.1.1 Einleitung

Industrie 4.0 ist die Vision eines cyber-physischen Produktionssystems, in welchem die Anlagen, Prozesse, Produkte und Menschen intelligent vernetzt werden (vgl. [5], [4] S. 18 u. S. 84). Das Ergebnis der intelligenten Vernetzung ist eine sogenannte Smart Factory, in welcher das Internet der Dinge und Dienste in industriellen Prozessen angewendet wird (vgl. [4] S. 18 u. S. 84, [2] S. 157, [9] S. 8).

Die Vision Industrie 4.0 bietet enorme Potentiale gegenüber bisherigen Produktionssystemen (vgl. [1], [6], [5], [4] S. 18 ff., [7], [15] Folie 6): Die Smart Factory ermöglicht eine einfache Produktindividualisierung bzgl. Design, Konfiguration und Menge. Der Hauptvorteil von Industrie 4.0 ist die Flexibilisierung des Produktionssystems. Dadurch sind unternehmensübergreifende Anlagenvernetzungen denkbar, in welchen Produktionsaufträge nach verschiedenen Kriterien wie beispielsweise Standort, Qualität, Kosten, Lieferzeit, Verfügbarkeit, Robustheit an Anlagen vergeben werden. Bei den Anlagen, Prozessen und Produkten des cyber-physischen Produktionssystems handelt es sich um autonome Einheiten, welche sich teilweise selbst organisieren und intelligent agieren. Die Verfügbarkeit aller relevanten Daten in Echtzeit ermöglicht einen einfacheren Entscheidungsprozess. Mit Industrie 4.0 soll ebenfalls das Produktionssystem in den Punkten: Produktivität, Wandelbarkeit, Ressourceneffizienz, Kosten und Qualität optimiert

werden. In Kapitel 2.1.3 sind die Eigenschaften bzw. Ziele ausführlich beschrieben.

Der Begriff Industrie 4.0 wurde erstmalig auf der Hannover Messe 2011 öffentlich verwendet (vgl. [10], [11]). Im Rahmen eines Vortrags stellten drei Vertreter (Henning Kagermann, Wolf-Dieter Lukas, Wolfgang Wahlster) aus Wirtschaft, Politik und Wissenschaft ihre Vision eines cyber-physischen Produktionssystems als vierte industrielle Revolution vor (vgl. [10], [4] S. 17). Der Arbeitskreis Industrie 4.0 von der Forschungsunion Wirtschaft – Wissenschaft untersuchte anschließend die Voraussetzungen für die vorgestellte Vision und veröffentlichte im Oktober 2012 einen Bericht mit Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0 (vgl. [12], [13]). Im April 2013 gründeten daraufhin die Verbände BITKOM, VDMA und ZEVI die Plattform Industrie 4.0, welche „unter der Leitung der Bundesministerien für Wirtschaft und Energie (BMWi) sowie Bildung und Forschung (BMBF) steht “ [13], [4] S. 82. Ziel der Plattform Industrie 4.0 ist die Erforschung, Entwicklung und Realisierung der Vision. Durch die gegründete Plattform wird deutlich, dass es sich um ein hoch aktuelles Thema von Industrie und Politik handelt.

### 2.1.2 Die Entwicklung zu Industrie 4.0

Im folgenden Abschnitt sind die vier Stufen der industriellen Revolution beschrieben. Zur Informationsbeschaffung wurden folgende Quellen herangezogen: [1] S. 5 - 13, [4] S. 17 f., [9] S. 7 und [10].

Die industrielle Revolution begann Ende des 18. Jahrhunderts durch die Einführung mechanischer Produktionsanlagen, welche mit Wasser- und Dampfkraft betrieben wurden. Ab 1870 erfolgte die zweite industrielle Revolution mit dem Einzug der elektrischen Energie in die Fertigung und einer dadurch möglichen arbeitsteiligen Massenproduktion von Gütern. Ein weiterer Fortschritt dieser Zeit war die Erfindung erster Verbrennungsmotoren. Die dritte industrielle Revolution begann Anfang der 1960er und hält bis heute an. Entscheidender Fortschritt zu den bisherigen Phasen war die Einführung der Elektronik, die eine Automatisierung der Produktionsprozesse ermöglichte. Die Automatisierung der Produktionsprozesse konnte später noch deutlich verbessert werden durch die aufstrebende Informationstechnologie.

In den vergangenen Jahren ist die Marktkomplexität für produzierende Unternehmen deutlich angestiegen (vgl. [1] S. 14). Typische Herausforderungen dieser Entwicklung sind eine steigende Produktvielfalt, die Möglichkeit der Produktindividualisierung, abnehmende Losgrößen und kurze Lieferzeiten. Zeitnahe Produktionsanläufe, wirtschaftliche und bedarfssynchrone Produktion sowie Ressourceneffizienz sind daher für den Wettbewerb unerlässlich. Als Reaktion auf diese Entwicklung, wurde 2011 die Vision einer vierten industriellen Revolution – Industrie 4.0 - vorgestellt. Ziel dieser Vision ist die intelligente Vernetzung der Anlagen, Prozesse, Produkte und Menschen in einem cyber-physischen Produktionssystem. Die realen Entitäten des Produktionssystems werden erfasst und in eine virtuelle Welt - das Internet der Dinge - übertragen.

In Abbildung 2.1 ist die historische Entwicklung der Produktion in Bezug auf Produktvolumen pro Variante und Produktvielfalt dargestellt. Zu Beginn der industriellen Revolution gab es eine große Produktvielfalt, welche nach und nach durch die erste und zweite industrielle Revolution verringert wurde. Grund hierfür ist die Mechanisierung und Massenfertigung, mit welcher das Produktvolumen pro Variante enorm anstieg. Ab der dritten industriellen Revolution kam es zu einem Wandel der bis heute anhält. Die Automatisierung der Produktionsprozesse mittels Elektronik und Informationstechnologie ermöglicht eine industrielle Serienproduktion mit größerer Produktvielfalt. Mit der Vision von Industrie 4.0 wird die Produktvielfalt durch die individuelle und flexible Produktion weiter zunehmen. Logischer weiße sinkt durch die Individualisierung das Produktvolumen pro Variante.

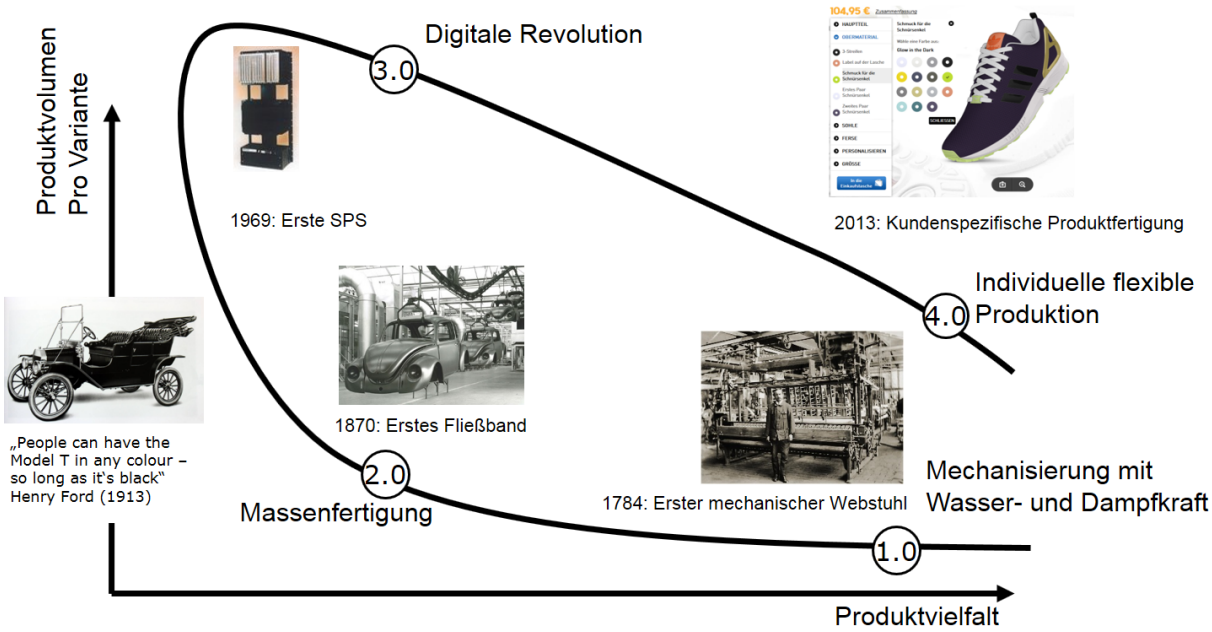


**Abbildung 2.1:** Historische Entwicklung der Produktion in Bezug auf Produktvolumen pro Variante und Produktvielfalt (Quelle: [7] Folie 229)

### 2.1.3 Eigenschaften und Ziele von Industrie 4.0 in der Produktion

Das folgende Unterkapitel beschäftigt sich mit den Eigenschaften und Zielen von Industrie 4.0 in der Produktion. Für eine bessere Übersicht wurde das Unterkapitel in fünf Abschnitte gegliedert: Vernetzung der Produktionssysteme, intelligente Fabrik, intelligente Produkte, Datenerfassung und Sicherheit.

#### 2.1.3.1 Vernetzung der Produktionssysteme

In cyber-physischen Produktionssystemen werden die Anlagen, Prozesse, Produkte und Menschen intelligent vernetzt (vgl. [5], [4] S. 17 f., [2] S. 169). Jede Entität wird dazu in der physischen Welt erfasst und in die virtuelle Welt übertragen. Durch die Verknüpfung der physischen und virtuellen Welt ist anschließend eine eindeutige und automatische Identifikation der Entitäten mit dem Internet der Dinge möglich.

Mittels horizontaler und vertikaler Integration werden die verschiedene IT-Systeme zu einem Gesamtsystem zusammengefasst (vgl. [4] S. 24f.). Die horizontale Integration verknüpft IT-technisch die unterschiedlichen Prozesse der Produktion und Unternehmensplanung zu einem durchgängigen System. Das entstehende System kann aus Partnern, Dienstleistern und Abnehmern/Kunden bestehen, die produktionsübergreifend und weltweit vernetzt sind (vgl. [2] S. 168, [1] S. 20). Bei der vertikalen Integration werden die Hierarchieebenen der Automatisierungspyramide (siehe Abbildung 2.2) durchbrochen und ein durchgängiges IT-System geschaffen (vgl. [1] S. 251f. + S. 543 f. + S. 550, [7] Folie 235). Die Topologien bzw. Hierarchiestrukturen der Produktionssysteme ergeben sich dann aus dem intelligenten Produkt und sind nicht wie bisher statisch vorgegeben. Abbildung 2.2 zeigt den Wandel von der klassischen Automatisierungspyramide zu cyber-physischen Systemen (Horizontale Integration).

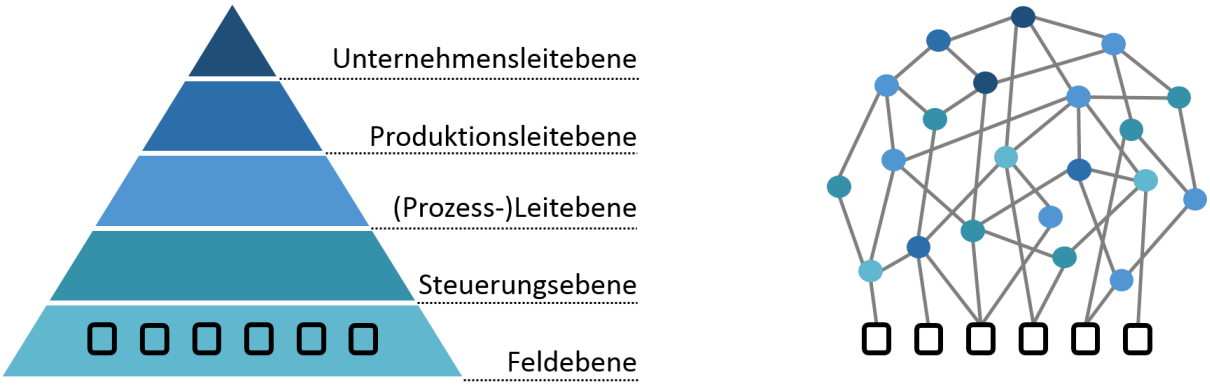


**Abbildung 2.2:** Klassische Automatisierungspyramide (links), Automatisierung basierend auf cyber-physischen Systemen (rechts) (Quelle: [7] Folie 235)

Durch die Vernetzung der einzelnen Entitäten ist das Produktionssystem hoch komplex und wird daher zur Verwaltung in drei Ebenen aufgeteilt (vgl. [1] S. 361): Eine Anlage mit seinen Komponenten befindet sich in der ersten Ebene. Die zweite Ebene besteht aus einzelnen Fabriken, in welcher sich mehrere Produktionsanlagen befinden. Bei der dritten Ebene handelt es sich um ein ganzheitliches Fabriknetzwerk, in welchem sich mehrere Fabriken befinden. Die Entitäten in dem verteilten Produktionssystem sind Ad-hoc vernetzt und werden -wenn erforderlich- rekonfiguriert (vgl. [4] S. 37, [2] S. 178). Mit vielen dezentralen und vernetzten Sensoren können die realen Zustände eines Produktionssystems in Echtzeit erfasst, analysiert

und weltweit abgerufen werden (vgl. [2] S. 157). Ein weiterer Vorteil ist der erhöhte Automatisierungsgrad, wodurch eine schnellere Reaktion auf vorhersehbare sowie unkalkulierbare Situationen möglich ist (vgl. [2] S. 157).

#### 2.1.3.2 Intelligente Fabrik

Die intelligente Fabrik (Smart Factory) ist ein vernetztes Produktionssystem, in welchem das Internet der Dinge und Dienste in industriellen Prozessen Einzug erhält (vgl. [4] S. 18, [2] S. 157). Das Ergebnis ist ein servicebasiertes Netzwerk auf Basis cyber-physischer Produktionssysteme (vgl. [4] S. 28, [1] S. 40 u. S. 586, [14] S. 84 f.). Eine Entität kann im Netzwerk einen Service bzw. Dienst anbieten, welcher von anderen Objekten abrufbar ist. Häufig gibt es in der service-orientierten Architektur (SOA) mehrere Anlagen, die denselben Dienst anbieten. Durch die Intelligenz des Netzes ist es möglich, mehrere Basis-Dienste zu einem höherwertigen Dienst bzw. intelligentem Produkt zusammenzufassen. Eine große Herausforderung hierbei ist die schnelle und einfache Orchestrierung von mehreren oder höherwertigeren Diensten (vgl. [4] S. 28 f., [2] S. 157ff.). Denkbar sind auch unternehmensübergreifende Dienste und Anwendungen mit kooperativen Prozessen innerhalb eines Geschäftsnetzwerks (Interoperabilität).

Für den Zugriff auf das Gesamtsystem gibt es verschiedene Anwendungen, mit welchen die realen Produktionsprozesse digital visualisiert, modelliert, analysiert und gesteuert werden (vgl. [2] S. 158). Aufgaben der Anwendungen sind (vgl. [1] S. 28): Einfache Steuerung und Inbetriebnahme der verteilten Produktionssysteme, Abbildung der verschiedenen Geschäftsprozesse, Sensorwerte bzw. Systemzustände in Echtzeit visualisieren, Prognosen und Analysen erstellen sowie mobile Endgeräte unterstützen.

Eine weitere Eigenschaft der intelligenten Fabrik ist die ganzheitliche Steuerung, Überwachung und Analyse der Wertschöpfungskette. Dadurch werden mit Algorithmen die Prozesse in Produktion, Logistik, Entwicklung und Materialverwendung in Echtzeit erfasst und optimiert (vgl. [4] S. 5, [14] S. 19, [15] Folie 6). Neu ist auch, dass sich die Anlagen teilweise selbst untereinander organisieren, kommunizieren und steuern (vgl. [1] S. 123).

Das Hauptziel von Industrie 4.0 ist „die Erhöhung der Produktivität bei variantenreicher und kundenindividueller Fertigung“ [2] S. 168. Für die Individualisierung des Produkts stehen dem Kunden verschiedene Design- und Konfigurationsoptionen zur Auswahl. Nachfolgend wird der Kundenauftrag erstellt und an eine intelligente Fabrik vergeben. In einem Netzwerk mit mehreren intelligenten Fabriken gibt es Algorithmen, welche nach verschiedenen Kriterien wie beispielsweise Standort, Qualität, Kosten, Lieferzeit, Verfügbarkeit, Robustheit einen Auftrag vergeben bzw. Dienste abrufen (vgl. [1] S. 147, [4]). Die Produktion der einzelnen Aufträge erfolgt bedarfssynchron (vgl. [8] S. 42, [1] S. 147). Das Ergebnis sind „entkoppelte, flexible und hoch integrierte Produktionssysteme“ [4] S. 68. In Abbildung 2.3 ist der Übergang von heutigen, getakteten Produktionen am Band zu entkoppelten, voll flexiblen und hochintegrierten Produktionssystemen dargestellt.

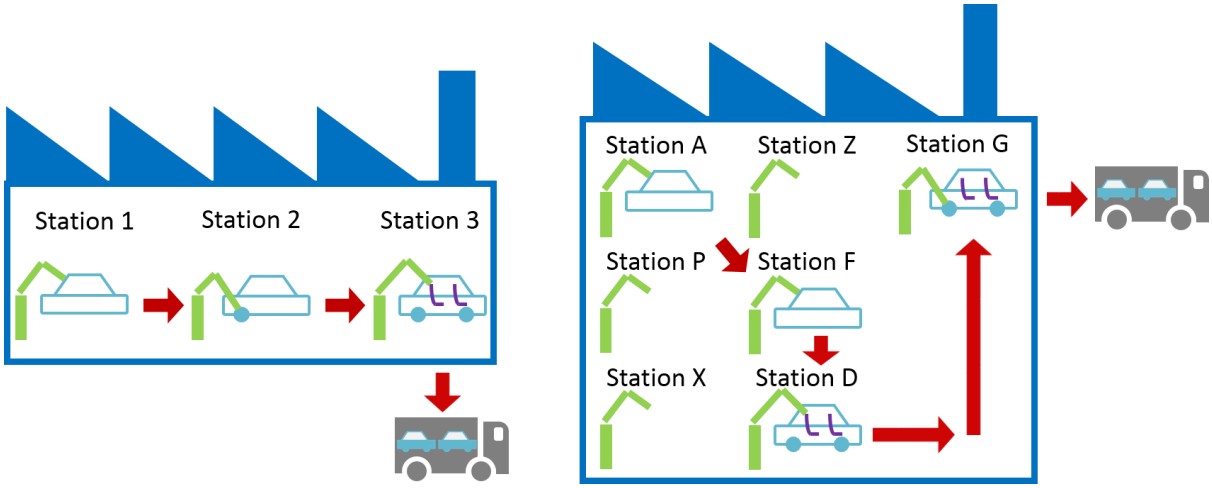


**Abbildung 2.3:** Getaktete Herstellung am Band (links), dynamisches Produktionssystem (rechts) (Quelle: [4] S. 68)

Weitere Eigenschaft der intelligenten Fabrik sind die Wandelbarkeit und geringere Störanfälligkeit. Durch einen einfachen und raschen Umbau des Produktionssystems können vorhandene bzw. verschiedene Produktionsmodule Ad-hoc in die Linie hinzugefügt oder getrennt werden (vgl. [1] S. 187f. [4], S. 105 u. S. 108, [2] S. 173). Neu hinzugefügte oder geänderte Entitäten konfigurieren sich selbst und werden in das bestehende System integriert. Es wird von einer Plug & Produce Fähigkeit gesprochen. Eine intelligente Fabrik ist aus zwei Gründen weniger störanfällig als bisherige Produktionssysteme (vgl. [2] S. 158, [4] S. 23). Zum einen können die Anlagen durch die erfassten Daten optimal gewartet werden und zum anderen bieten mehrere Anlagen gleiche Dienste an.

#### 2.1.3.3 Intelligente Produkte

Einem intelligenten Produkt wird vor Beginn der Herstellung eine eindeutige Identifizierung zugewiesen sowie ein Produktlebenszyklus angelegt, welcher mit dem Internet der Dinge und Dienste verknüpft wird (vgl. [2] S. 157, [1] S. 60 u. S. 544). Die Speicherung der Daten erfolgt in einer zentralen und dezentralen Cloud (vgl. [2] S. 159). Für die Identifizierung wird beispielsweise eine Identifikationsnummer generiert und mittels RFID-Tag oder DataMatrix-Code an dem intelligenten Produkt gespeichert (vgl. [1] S. 60 u. S. 360, [2] S. 173). Dadurch ist es jederzeit eindeutig identifizierbar und innerhalb des Produktionssystems lokalisierbar (vgl. [1] S. 122 u. S. 360, [4] S. 25). Die Produktionsparameter des intelligenten Produkts werden in dem Produktlebenszyklus abgespeichert bzw. verknüpft. In den Produktionsparametern sind beispielsweise der Produkttyp, Herstellungsprozess, die Qualitätsanforderungen, Konfigurationen, das Design sowie viele weitere Informationen gespeichert. Dadurch hat das zu erstellende Produkt Informationen zu seinem Herstellungsprozess und kann aktiv den Fertigungsprozess unterstützen bzw. steuern (vgl. [4] S. 23 ff., [1] S. 60). Die Produktionsstruktur wird daher vom

intelligenten Produkt bestimmt und ist nicht wie bisher statisch vorgegeben. In dem Produktlebenszyklus ist ebenfalls der zukünftige Einsatz bzw. die spätere Verwendung definiert. Mit der eindeutigen Identifizierung können noch kurz vor oder während der Produktion die Parameter eines intelligenten Produkts geändert werden. Durch die oben beschriebenen Eigenschaften sind die cyber-physischen Produktionssysteme deutlich dynamischer als bisherige Produktionsformen. Zudem wird eine deutlich größere Produktvielfalt möglich. Die Dokumentation des Produktlebenszyklus geht über die Aktivitäten im Produktionssystem hinaus und bringt mehrere Vorteile mit sich (vgl. [2] S. 160 f., [8] S. 92, [4] S. 7 u. 24 ff): Das intelligente Produkt wird ab der Auftragsgenerierung bis zum Recycling dokumentiert. Dadurch sind alle Prozesse transparent erfasst und im Falle eines Fehlers sind einfachere Nachforschungen möglich. Mit den erfassten Daten zum intelligenten Produkt sind außerdem Optimierungen in verschiedenen Bereichen, wie beispielsweise Produktivität, Qualität, Verfügbarkeit, Robustheit, Kosten, Lieferzeit oder Ressourceneffizienz, möglich.

#### 2.1.3.4 Datenerfassung

In vernetzten Produktionssystemen mit mehreren intelligenten Fabriken fallen enorme Datenmengen an. Die realen Entitäten des Produktionssystems werden erfasst und in eine virtuelle Welt - das Internet der Dinge - übertragen. Das Ergebnis ist ein intelligentes Produktionsnetzwerk, in dem die Produktionssysteme und intelligenten Produkte in Echtzeit erfasst, gesteuert, dokumentiert, analysiert und weltweit abrufbar sind (vgl. [4] S. 107, [1] S. 123, [2] S. 157). Die Datenerfassung der realen Zustände erfolgt mittels vieler dezentraler und vernetzter Sensoren. Für die Speicherung der Daten werden zentrale und dezentrale Clouds verwendet (vgl. [2] S. 159). Das dadurch entstehende Potenzial ist enorm. So ist beispielsweise ein echtzeitfähiges Monitoring auf Basis eines Produktionssystemabbildes möglich (vgl. [4] S. 27). Vorteile sind einfachere Entscheidungen sowie die schnellere Lokalisierbarkeit und Behebbarkeit auftretender Fehler. Die anfallenden Daten werden außerdem zur Dokumentation des Produktionssystems, der Produktionsprozesse und intelligenten Produkte verwendet. Dadurch entsteht eine durchgängige Transparenz und Nachverfolgbarkeit für jede einzelne Entität sowie im Zusammenspiel mit den anderen Entitäten (vgl. [4] S. 20). Die erfassten Daten sind sehr umfangreich (Big Data) und werden mit intelligenten Algorithmen verarbeitet. Denkbar wäre beispielsweise eine Echtzeitoptimierung der Produktionsprozesse (vgl. Abschnitt *Intelligente Fabrik*). Eine weitere Option ist die Analyse der Zuverlässigkeit und Verfügbarkeit einzelner Entitäten (vgl. [2] S. 158, [4] S. 107, [1] S. 548 f.). Hierbei wird das Verhalten einer Entität analysiert und historisches Wissen (Erfahrungswerte) verwendet. Mit den Ergebnissen der Analyse kann die Instandhaltung sowie Ersatzteilhaltung kontinuierlich verbessert werden. Durch die Bewertung der Verfügbarkeit einzelner Entitäten sind präzise Terminaussagen für intelligente Produkte bereits bei der Auftragserteilung möglich.

#### 2.1.3.5 Sicherheit

Der Arbeitskreis Industrie 4.0 schlüsselt den Begriff Sicherheit in zwei Aspekte auf: Betriebssicherheit sowie die Angriffssicherheit (vgl. [4] S. 51). Ziel der Betriebssicherheit ist ein störungsfreies Produktionssystem, von welchem keine Gefahr für den Anwender und das Umfeld ausgeht. Durch die Vernetzung der cyber-physischen Produktionssysteme ist die Sicherheit vor Missbrauch und unbefugtem Zugriff enorm wichtig (vgl. [4] S. 51, [2] S. 166, [1] S. 345 u. S. 610). Um vor Angriffen geschützt zu sein, wird eine Sicherheitsarchitektur auf Basis eines Referenzmodelles (z.B. RAMI 4.0) entworfen (vgl. [14] S. 69, [2] S. 166). In der Sicherheitsarchitektur sind Konzepte und die daraus abgeleiteten Maßnahmen definiert. Verschiedene denkbare Sicherheitskonzepte bzw. -maßnahmen wurden in der Studie „IT-Sicherheit für die Industrie 4.0“ [14] im Auftrag des Bundesministeriums für Wirtschaft und Energie (BMWI) erarbeitet (vgl. [14] S. 138 – S. 153).

Aus der Studie ergeben sich die verschiedensten Sicherheitsmaßnahmen. So erhalten Anlagen, Komponenten und Anwender individuelle Zugangsdaten für den Systemzugriff. Das Sicherheitssystem muss bei allen Anlagen und Komponenten auf dem aktuellsten Stand gehalten werden. Der Zugriff auf die Systeme darf ausschließlich über gesicherte Zugänge erfolgen. Für eine sichere Fernwartung empfiehlt das BMWI eine vorherige Gerätefreischaltung. Ein weiterer Vorschlag ist der Einsatz von kryptografischen Algorithmen zur sicheren Datenübertragung, wie zum Beispiel VPN-Lösungen. Um das Netz sicherer zu machen sind auch Netzsegmentierungen, logische Trennungen von verschiedenen Komponenten, statische Netzkonfigurationen, Firewalls sowie sichere Übertragungsprotokolle erforderlich. Mit intelligenten Programmen soll verhindert werden, dass Schadsoftware das Produktionssystem attackiert oder übernimmt. Mit Logging/Monitoring werden alle Systemzustände in Echtzeit dokumentiert und analysiert.

### 2.1.4 Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)

Das Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) ist ein dreidimensionales Schichtenmodell, welches als technisches Grundgerüst für die Strukturierung, Standardisierung und Umsetzung der Vision Industrie 4.0 verwendet wird (vgl. [17] S. 1, [3] S. 5, [15] Folie 9). Ziel der Strukturierung ist eine Aufteilung der umfangreichen und hochkomplexen Vision in überschaubare Teile (vgl. [16] S. 6 f.). In dem dreidimensionalen Schichtenmodell kann der Betrachter verschiedene Perspektiven einnehmen und Assets (Komponenten) wie beispielsweise Produktionsteile, Produkte oder Maschinen in dem Modell einordnen (vgl. [17] S. 2). Teil des RAMI4.0 ist die Spezifikation der Industrie 4.0 Komponente. Das Referenzarchitekturmodell Industrie 4.0 wurde in einem Workshop von Vertretern aus Wirtschaft, Verbänden und Wissenschaft erarbeitet und in der DIN SPEC 91345 [3] veröffentlicht.

#### 2.1.4.1 Definition Asset

Ein Asset ist eine beliebige Komponente, die von Wert für das Produktionssystem ist. Typische Assets in einem System sind beispielsweise Produktteile, Produkte, Baugruppen, Maschinen, Ersatzteile, Dokumente oder auch Menschen. In einer 2-D Matrize werden die einzelnen Assets eines Produktionssystems bezüglich ihrer Bekanntheit im Informationssystem und Kommunikationsfähigkeit eingeordnet (siehe Abbildung 2.4) (vgl. [3] S. 17, [16] S. 15). RAMI4.0 beschreibt die Einordnung der einzelnen Assets als CP-Klassifikation. Damit ein Asset zur I4.0-Komponente wird, muss dieser im Informationssystem als Entität verwaltet werden und eine passive oder aktive I4.0-konforme Kommunikationsfähigkeit besitzen (CP24-, CP34- oder CP44-Komponente) (vgl. [16] S. 15, [3] S. 17 u. S. 25).

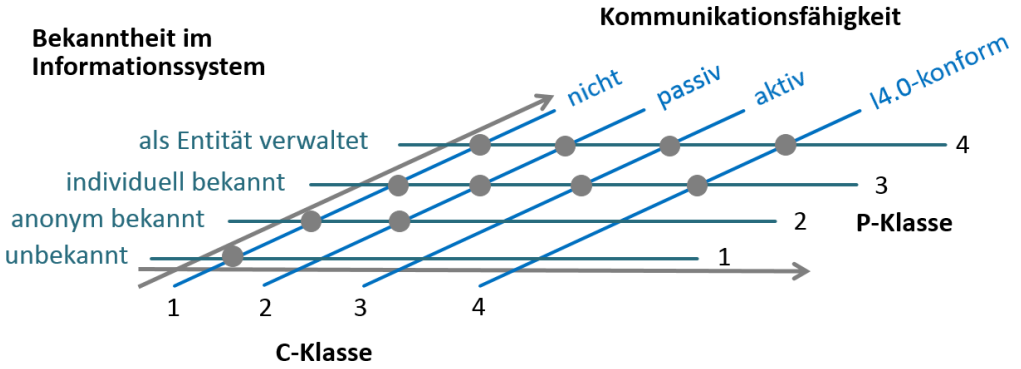


**Abbildung 2.4:** Klassifizierung einer Komponente bezüglich Bekanntheit im Informationssystem und Kommunikationsfähigkeit (Quelle: [3] S. 17)

#### 2.1.4.2 Aufbau des Referenzarchitekturmodell Industrie 4.0

Das dreidimensionale Koordinatensystem des Modells besteht aus der Architektur-Achse (Layers), Verlauf-Achse (Life Cycle & Value Stream) und Hierarchie-Achse (Hierarchy Levels). Die einzelnen Achsen sind wiederum in mehrere Ebenen bzw. Schichten unterteilt. Eine Kommunikation ist nur innerhalb einer Schicht und mit den angrenzenden Schichten möglich (vgl. [3] S. 20). In Abbildung 2.5 ist das Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) grafisch dargestellt und wird nachfolgend beschrieben.

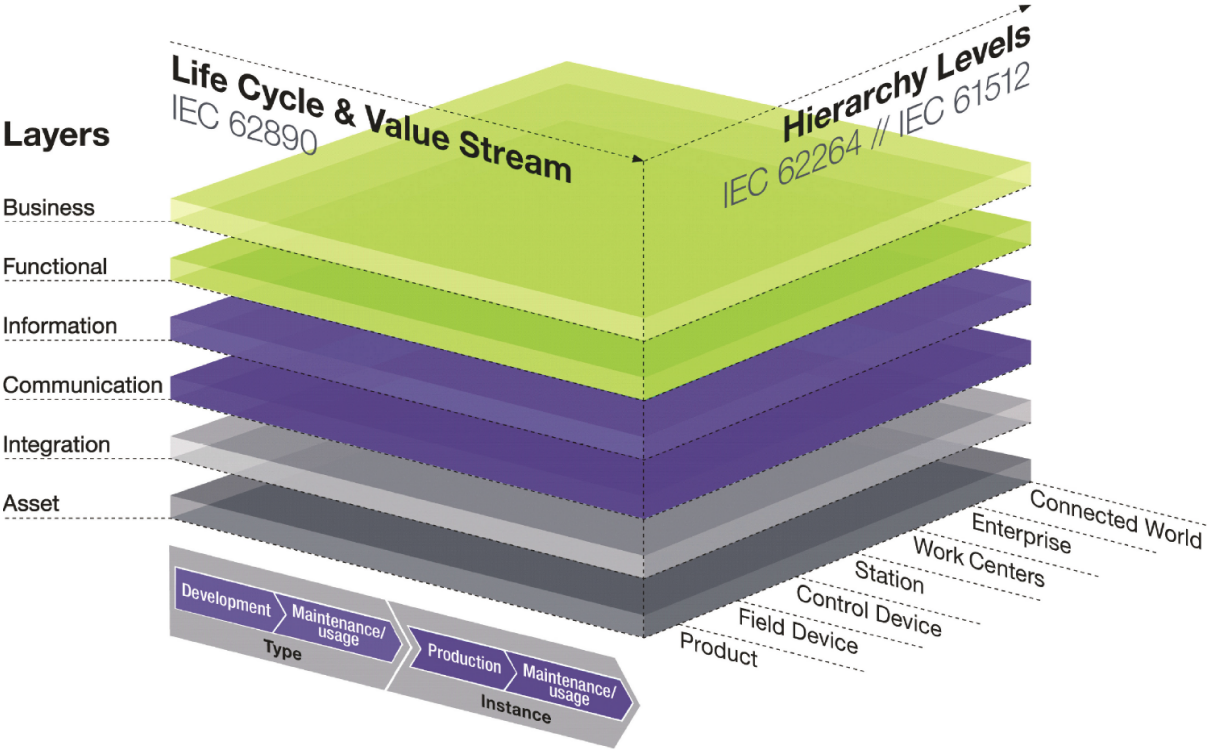


**Abbildung 2.5:** Referenzarchitekturmodell Industrie 4.0 (Quelle: [3] S. 19)

### Architektur-Achse (Layers)

Die unterste Schicht der Architektur-Achse bildet die *Gegenstandsschicht* (Asset-Layer). Mit ihr werden die realen Assets in der physischen Welt abgebildet (vgl. [16] S. 9, [3] S. 22).
Die *Integrationsschicht* (Integration) beschreibt den Übergang von der physischen Welt zur Informationswelt. Auf dieser Ebene werden die Informationen bzw. Zustände eines Assets erfasst und für die Informationsschicht mittels der Kommunikationsschicht bereitgestellt (vgl. [3] S. 22, [16] S. 9). Eine weitere Aufgabe dieser Schicht ist die Steuerung und Reglung eines technischen Prozesses. Die Steuerung und Reglung eines technischen Prozesses kann beispielsweise mit einer speicherprogrammierbaren Steuerung (SPS) erfolgen.
Zwischen der Integrations- und Informationsschicht befindet sich die *Kommunikationsschicht* (Communication). Ziel dieser Ebene ist eine standardisierte Kommunikation zwischen diesen beiden Schichten. Für die Steuerung der Integrationsebene werden verschiedene Dienste angeboten (vgl. [3] S. 21, [16] S. 9).
In der *Informationsschicht* (Information) werden Daten und Ereignisse von der Integrationsebene verarbeitet. Mittels einer Dienstschnittstelle werden anschließend die verarbeiteten Daten und Ereignisse für die Funktionsebene strukturiert bereitgestellt. Eine weitere Aufgabe der Ebene ist die „konsistente Integration verschiedener Daten und die Sicherstellung von Datenintegrität“ [16] S. 9.

Die darüberliegende *Funktionsebene* (Functional Layer) beschreibt formal die Funktionen eines Assets innerhalb des Produktionssystems (vgl. [3] S. 20). Auf dieser Schicht werden ebenfalls die Anwendungen, Dienste und Geschäftsprozesse ausgeführt sowie die unterschiedlichen Funktionen horizontal miteinander verbunden (horizontale Integration). Wartungsarbeiten am System erfolgen auf dieser Ebene.
In der *Geschäftsschicht* (Business) sind die Organisation und Geschäftsprozesse angesiedelt. Typische Aufgaben dieser Ebene sind die Organisation der Aufträge und Ressourcen sowie die Orchestrierung von Diensten in der Funktionsschicht (vgl. [16] S. 8, [3] S. 20).

#### Verlauf-Achse (Life Cycle & Value Stream)

Auf der Verlauf-Achse wird der gesamte Lebenszyklus eines Assets abgebildet (vgl. [3] S. 23). Für eine bessere Strukturierung des Lebenszyklus werden die Assets in die Klassen „Typ“ und „Instanz (Instance)“ kategorisiert (siehe Abbildung 2.5).
Ein Typ ist ein Bauplan, welcher die Eigenschaften eines Assets definiert. In der Entwicklungsphase (Development) findet die Entwicklung, Konstruktion und Simulation des Typs statt (vgl. [15] Folie 14, [16] S. 10 u. 12). Teil der Phase ist auch die Herstellung eines Prototyps. Nach der Entwicklungsphase folgt die Nutzungs- und Wartungsphase (Maintance/Usage). In dieser erhält der Typ eine Freigabe für die Generierung von Instanzen. Eine weitere Aufgabe der Phase ist die spätere Wartung des Typs.
Der Lebenszyklus einer Instanz setzt sich aus der Produktionsphase und der Nutzungs-/Wartungsphase zusammen. In der Produktionsphase wird eine Instanz auf Basis eines Typs generiert (vgl. [3] S. 14 f.). Jede Instanz erhält bei der Generierung eine eindeutige Seriennummer sowie verschiedene Daten. Die nachfolgende Phase beschreibt die Nutzung und das Wartungsmanagement einer Instanz.

#### Hierarchie-Achse (Hierarchy Levels)

Auf der Hierarchie-Achse erfolgt die funktionale Einordnung der einzelnen Assets in Hierarchiestufen (vgl. [16] S. 10, [15] Folie 12). Grundlage ist die Automatisierungspyramide der IEC 62264, welche um die Produktebene (Product) und die Ebene der vernetzen Welt erweitert wurde. In der Produktebene sind alle „kooperierende bzw. kollaborierende zu fertigende Produkte als integraler Bestandteil eines Industrie 4.0-Wertschöpfungsprozesses abgebildet“ [3] S. 23. Die Ebene der vernetzten Welt beschreibt die weltweite Vernetzung von mehreren Produktionssystemen.

### 2.1.4.3 Industrie 4.0 Komponente

Industrie 4.0 Komponenten bestehen aus einer Verwaltungsschale und mindestens einem Asset. Die Verwaltungsschale repräsentiert die einzelnen Assets einer I4.0 Komponente im Produktionssystem (vgl. [3] S. 24, [15] Folie 17, [16] S. 16 f.). Teil der Verwaltungsschale ist der Komponenten-Manager und das Manifest. In Abbildung 2.6 ist der allgemeine Aufbau einer I4.0 Komponente dargestellt.

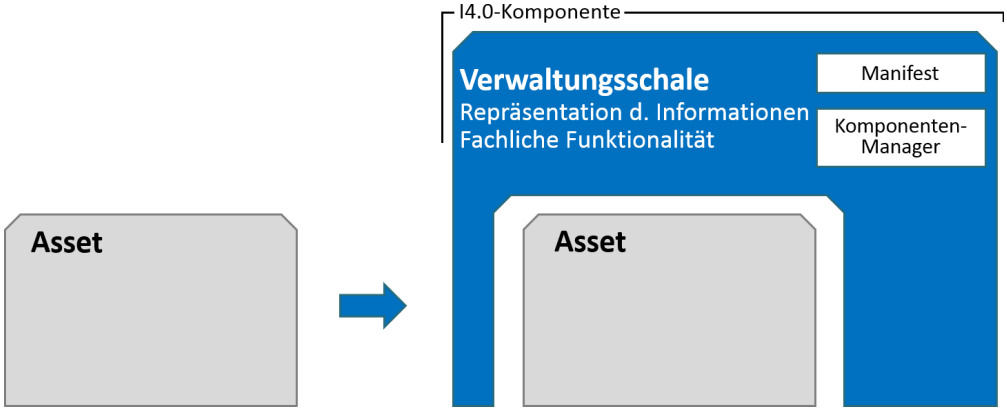


**Abbildung 2.6:** Aufbau der I4.0 Komponente (Quelle: [3] S. 24)

Auf Basis einer service-orientierten Architektur (SOA) stellt der Komponenten-Manager in der Verwaltungsschale die Informationen zu einem Asset bereit und ermöglicht einen Zugriff auf fachliche Funktionen. Für die Realisierung der service-orientierten Architektur (SOA) ist das Kommunikationsprotokoll OPC Unified Architecture vorgesehen (vgl. [16] S. 23). Zur Laufzeit erfasst der Komponenten-Manager die Daten sowie Zustände eines Assets und verarbeitet diese zu Informationen. Die weltweit eindeutige Identifikation und Adressierung der I4.0-Komponente erfolgt im Komponenten-Manager. Eine weitere Aufgabe ist die Sicherstellung der Betriebs- und Angriffssicherheit einer I4.0-Komponente.

Im Manifest sind verschiedene verpflichtende Angaben zu einer I4.0-Komponente gelistet (vgl. [3] S. 35f., [16] S. 23f.). Es handelt sich dabei um Metainformationen, welche die funktionalen und nichtfunktionalen Eigenschaften der einzelnen Assets beschreiben. Weiterhin sind im Manifest die Beziehungen zwischen den I4.0-Komponenten und Assets definiert.

Die I4.0-Komponenten in einem I4.0-System sind schachtelbar und kapselungsfähig (vgl. [3] S. 27 f., [16] S. 20 f. u. S. 23, [15] Folie 18): Schachtelbarkeit bedeutet, dass eine I4.0-Komponente sich aus mehreren I4.0-Komponenten zusammensetzen kann. Ein Beispiel für diesen Sachverhalt ist eine Fräsmaschine (I4.0-Komponente), welche sich aus den I4.0-Komponenten Elektromotor, Fräse sowie Steuerung zusammensetzt. Kapselungsfähigkeit bedeutet, dass die Assets einer I4.0-Komponenten bei Problemen oder Fehlern immer noch abgekapselte Grundfunktionen störungsfrei ausführen.

## 2.2 Machine-to-Machine Kommunikation (M2M)

Das folgende Kapitel beschäftigt sich mit den Grundlagen der Machine-to-Machine Kommunikation (M2M). In einer kurzen Einleitung wird zunächst der Grundgedanke der Machine-to-Machine Kommunikation erläutert sowie das vielfältigen Anwendungsspektrum aufgezeigt. In Kapitel 2.2.2 wird ein geeignetes Grundkonzept zur Umsetzung der M2M-Kommunikation in Produktionssystemen vorgestellt. Unterkapitel 2.2.3 liefert eine Übersicht von verschiedene Architekturen, Protokolle und Formate der M2M-Kommunikation.

### 2.2.1 Einleitung

Ziel der Machine-to-Machine Kommunikation sind intelligent vernetzte Endgeräte, die miteinander überwiegend automatisiert kommunizieren bzw. Informationen austauschen (vgl. [26] S. 10, [20] S. 2, [18]).

Durch intelligente, vernetzte Endgeräte können in der Automatisierungstechnik Produktionssysteme oder -prozesse gesteuert werden. Im M2M-System stellen die einzelnen Endpunkte zur Steuerung sowie Protokollierung der Prozesse und Systeme verschiedene Daten (z.B. Sensor, Produktion) bereit. Für die intelligente Steuerung definiert der Mensch Logiken und Regeln, nach denen die Endgeräte agieren (vgl. [26] S. 16 [25] S. 3). Häufig gibt es in einem M2M-System einen Datenintegrationspunkt (DIP), in welchem Daten der einzelnen Endgeräte zu Informationen aufbereitet und für andere Endgeräte im Netzwerk zur Verfügung gestellt werden (vgl. [21] S. 7). Typische Endgeräte im M2M-System sind beispielsweise Rechner, Steuergeräte, Feldgeräte oder auch Mobilgeräte (Laptop, Tablet-Computer, Smartphone). Die logische Vernetzung der Datenendgeräte erfolgt mittels lokaler Netzwerke sowie dem globalen Netzwerk (Internet). Die Technologie für die physikalische Datenübertragung zwischen den Endgeräten hängt vom Anwendungsfall ab (vgl. [21] S. 11). Zur physikalischen Datenübertragung sind verschiedene bewährte kabelgebundene oder kabellose Technologien denkbar. In den einzelnen Anwendungsdomänen der M2M-Kommunikation sollte ein hersteller- und dienstleisterübergreifendes Kommunikationsprotokoll verwendet werden. Die Standardisierung stellt dadurch eine größtmöglichste Integration und Interoperabilität zwischen den M2M-Komponenten sicher (vgl. [26] S. 16 [25] S. 3).

Das Anwendungsspektrum der Machine-to-Machine Kommunikation ist sehr vielseitig und geht weit über die Automatisierungsbranche hinaus. In Abbildung 2.7 ist das Anwendungsspektrum grob in die sieben Bereiche untergliedert.

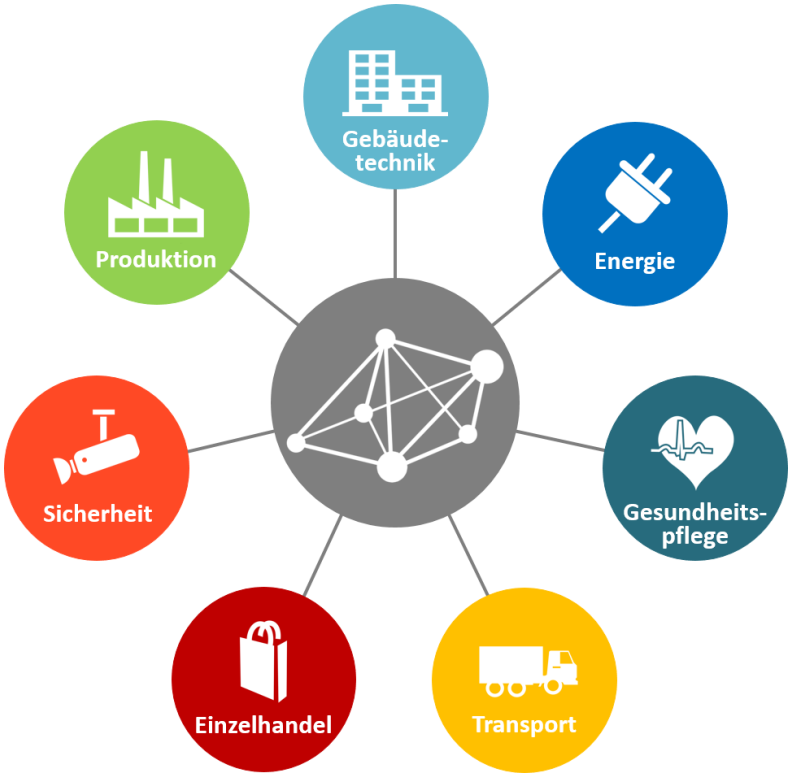


**Abbildung 2.7:** Anwendungsspektrum der Machine-to-Machine Kommunikation

M2M ermöglicht in der *Produktion* eine autonomere Steuerung sowie Datenerfassung der Produktionssysteme und -prozesse. Dadurch lassen sich die Prozesse einfach optimieren und die Produktivität steigern (vgl. [18], [25] S. 4). In der *Gebäudetechnik* können mittels M2M-Systemen Geschäfts- und Wohngebäude intelligent automatisiert werden (vgl. [22] Folie 6). Denkbar ist beispielsweise die Automatisierung der Elektrotechnik, Heizungstechnik und Klimatechnik von Gebäuden. Ein weiteres Anwendungsspektrum ist die *Energieerzeugung und -versorgung* (vgl. [21] S. 23, [22] Folie 6). In einem sogenannten intelligenten Energienetz (Smart Grid) wird mittels M2M-Kommunikation die Erzeugung, Speicherung, Übertragung und der Verbrauch von elektrischer Energie intelligent gesteuert. Die *Gesundheitspflege* kann mit M2M-Systemen automatisiert und verbessert werden. Eine Vision ist beispielsweise die Entwicklung eines intelligenten Geräts, das mit unterschiedlichen Sensoren den Gesundheitszustand eines Menschen erfasst. Die erfassten Werte werden zentral in einem Rechenzentrum überwacht. Bei der Überschreitung von Schwellenwerten wird automatisch ein Arzt zum Standort des Patienten alarmiert. Im *Transport- und Logistikwesen* konnten durch die M2M-Kommunikation enorme Fortschritte erzielt werden (vgl. [25] S. 3, [22] Folie 6). Typische Anwendungen sind beispielsweise Flottenmanagement oder Verkehrstelematik (vgl. [1]). Die M2M-Kommunikation erlaubt im *Einzelhandel* das Tracking von Gütern sowie die Automatisierung der Lieferketten (vgl. [22] Folie 6). Ein realisiertes Beispiel für die Automatisierung der Lieferketten ist der „Dash-Button“ vom Online-Versandhändler Amazon. Mit diesem kann der Kunde mit nur einem Knopfdruck Güter bestellen. Amazon möchte in Zukunft den Service soweit erweitern,

dass Geräte automatisch nach Bedarf Güter bestellen bzw. nachbestellen (Amazon Dash Replenishment) (vgl. [27]). Die zivile und militärische *Sicherheit* kann mit der M2M-Technologie erweitert und verbessert werden (vgl. [22] Folie 6, [18]). Mit der M2M-Kommunikation werden Überwachungs- und Alarmsysteme intelligent miteinander vernetzt und automatisiert.

### 2.2.2 Grundkonzept der Machine-to-Machine Kommunikation

M2M-Systeme setzten sich im Wesentlichen aus drei Grundbausteinen zusammen: Datenendpunkt (DEP), Datenintegrationspunkt (DIP) und dem Kommunikationsnetzwerk. In Abbildung 2.8 ist der Aufbau eines M2M-Systems dargestellt.

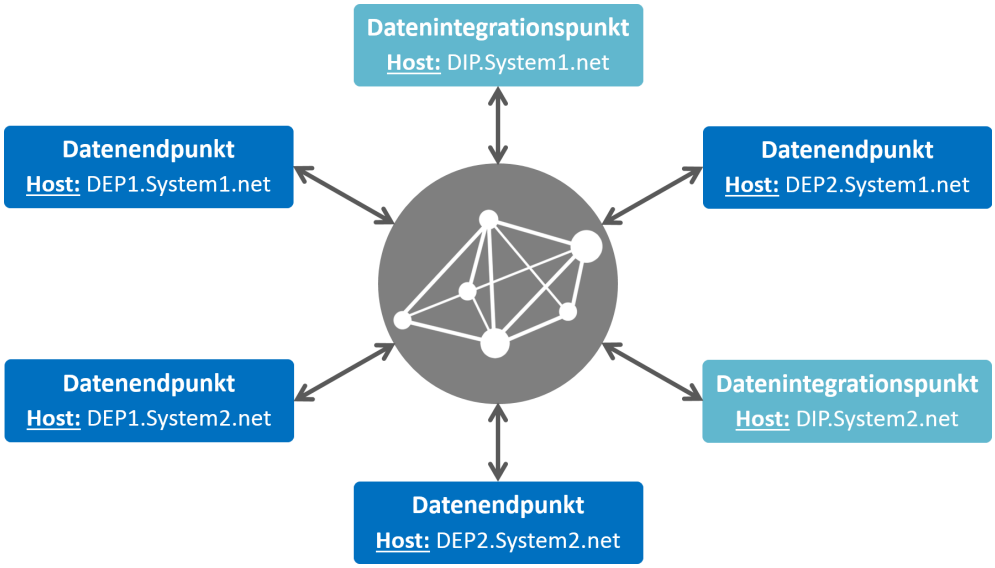


**Abbildung 2.8:** Aufbau eines M2M-Systems

Im weiteren Teil des Kapitels werden die Eigenschaften der drei Grundbausteine sowie deren Zusammenspiel miteinander beschrieben.

#### Datenendpunkt (DEP)

Datenendpunkte sind intelligente Endgeräte, die überwiegend automatisiert mit anderen Endgeräten oder dem Datenintegrationspunkt im M2M-System kommunizieren bzw. Informationen austauschen. Die Kommunikation bzw. der Informationsaustausch basiert auf einer Peer-to-Peer-Verbindung, mit welcher Endgeräte untereinander Services anbieten oder in Anspruch nehmen (vgl. [20] S. 6 ff.). Bei den angebotenen Services wird zwischen öffentlich und privat unterschieden. Für den Zugriff auf einen privaten Service ist eine Authentifizierung beim Endpunkt erforderlich (vgl. [20] S. 6 f., [21] S. 21). Denkbar ist beispielsweise eine Authentifizierung mittels Benutzername, Passwort und Zertifikat (vgl. [21] S. 23). Öffentliche

Services sind hingegen von allen Endgeräten im Netzwerk aufrufbar. Weitere Eigenschaft der Datenendpunkte sind das autonome Agieren sowie die Selbstorganisation. Für das autonome Agieren entwickelt der Mensch eine künstliche Intelligenz, mit welcher das Endgerät Entscheidungen trifft (vgl. [26] S. 16 [25] S. 3). Alle Endpunkte eines Systems sind weltweit eindeutige identifizierbar mittels einer ID oder IP (vgl. [21] S. 24, [20] S. 8). Im M2M-System können sich beliebig viele Endpunkte befinden (vgl. [21] S. 16). Mit steigender Anzahl von Endpunkten nimmt jedoch die Systemleistung ab und eine Ressourcenverwaltung wird erforderlich. Typische Endgeräte sind beispielsweise Rechner, Steuergeräte, Feldgeräte oder auch Mobilgeräte (Laptop, Tablet-Computer, Smartphone).

#### Datenintegrationspunkt (DIP)

Ein weiterer Bestandteil ist der Datenintegrationspunkt (DIP). Innerhalb eines lokalen M2M-Systems gibt es meistens nur einen zentralen DIP, der auf einem Server abgebildet wird (vgl. [26] S. 19, [18]). Es sind aber auch mehrere dezentrale DIP in einem lokalen M2M-System denkbar. Der Datenintegrationspunkt erhält zyklisch verschiedene Daten zu den einzelnen Endgeräten (DEP) oder fragt diese bei Bedarf ab. Die Daten werden anschließend aufbereitet und den anderen Endgeräten im Netzwerk zur Verfügung gestellt (vgl. [25] S. 3). Dadurch bildet der Datenintegrationspunkt ein zentrales Element im M2M-System, in welchem alle Informationen und Daten gebündelt sind. Dies bringt mehrere Vorteile mit sich. Die Endgeräte können dadurch an einer zentralen Stelle Informationen zum Systemkontext ermitteln. Ein weiterer Anwendungsbereich ist die Abfrage von Informationen eines Endgeräts, ohne mit diesem in Kontakt zu treten (vgl. [20] S. 10). Befindet sich momentan ein Endpunkt nicht im Netzwerk, können dennoch Daten zu diesem abgefragt werden. Ein weiterer Vorteil des Datenintegrationspunktes ist die Aggregation von Daten mehrerer Endpunkte zu hochwertigere Daten (vgl. [19]). Durch die hochwertigeren Daten ergeben sich wiederum neue Potentiale.

#### Kommunikationsnetzwerk

Das Kommunikationsnetzwerk verbindet die Datenendpunkte und den Datenintegrationspunkt logisch sowie physisch miteinander. Die logische Vernetzung der Datenendgeräte erfolgt mittels lokaler Netzwerke sowie dem globalen Netzwerk (Internet). Die Verbindung des Kommunikationsnetzwerks beschreibt das Medium für die Datenübertragung zwischen den Datenendpunkten und dem Datenintegrationspunkt. Es werden hierfür bewährte kabelgebundene und kabellose Technologien verwendet (vgl. [20] S. 6 u. S. 8, [25] S. 2, [23] Folie 11). Die eingesetzte Technologie hängt von Anwendungsfall ab (vgl. [25] S. 2). Für Datenendpunkte mit einem statischen Standort wird primär ein kabelgebundener Datenanschluss verwendet (vgl. [21] S. 11 ff.). Typische Endgeräte mit einem kabelgebundenen Datenanschluss sind beispielsweise Maschinen, Anlagen, Rechner oder Leitstände. Der kabellose Datenanschluss wird zur Kommunikation mit flexiblen Endgeräten eingesetzt (vgl. [21] S. 11 u. 14). Beispiele hierfür sind zu fertigende Produkte, die über einen RFID-Chip verfügen oder unterschiedliche Mobilgeräte (Messgerät, Laptop, Tablet-Computer, Smartphone). Bewährte kabelgebundene Technologien sind beispielsweise Ethernet (lokal) und DSL (WAN) (vgl. [25] S. 2 u. [21] S. 17 f.). Bei den

kabellosen Technologien wird unterschieden zwischen Datenübertragungsmedien mit kurzer und hoher Reichweite (vgl. [21] S. 17 ff., [22] Folie 14, [25] S. 2). Bekannte Technologien mit kurzer Reichweite (bis zu 500 Meter) sind beispielsweise WLAN, Bluetooth, ZigBee oder RFID. Für hohe Reichweiten werden Technologien aus dem Mobil- und Satellitenfunk verwendet. Aktuelle Mobilfunkstandards sind 2G (GSM, GPRS und Edge), 3G (UMTS) und LTE.

#### Das Grundkonzept der M2M-Kommunikation in der Automatisierungstechnik

In der Vision Industrie 4.0 werden die Assets eines Produktionssystems durch intelligente I4.0-Komponenten miteinander vernetzt. Jede I4.0-Komponente im Produktionssystem repräsentiert einen Datenendpunkt im M2M-Kontext. Im Netzwerk kommunizieren die Datenendpunkte und der Datenintegrationspunkt überwiegend automatisiert. Sensor- und Produktionsdaten sowie Steuerungsbefehle werden untereinander ausgetauscht. Der Zugriff auf die Sensor- und Produktionsdaten sowie Steuerungsbefehle erfolgt mittels einer service-orientierten Architektur (vgl. [20] S. 6 f., [22] Folie 7, [21] S. 40). Am Datenintegrationspunkt werden verschiedene Sensor- und Produktionsdaten der einzelnen Datenendpunkte gespeichert und den anderen I4.0-Komponenten zur Verfügung gestellt. Die einzelnen I4.0-Komponenten sowie der Datenintegrationspunkt agieren autonom und organisieren sich mittels der M2M-Kommunikation im Produktionssystem selbst (vgl. [23] Folie 8, [20] S. 3). Für die autonome Interaktion innerhalb des Produktionssystems entwickelt der Mensch eine künstliche Intelligenz, nach der die Komponente Entscheidungen trifft (vgl. [26] S. 16 [25] S. 3). Das Ergebnis sind intelligente Endgeräte, die miteinander mittels M2M Produktionsprozesse oder -systeme steuern und protokollieren. Die Protokollierung der Daten ermöglicht eine transparente Sicht auf die Wertschöpfungskette. Dadurch ist eine Optimierung des Produktionssystems in vielerlei Hinsicht möglich (vgl. [25] S. 4). Typische Optimierungen sind beispielsweise Produktivitätssteigerung, Qualitätssteigerung, Verkürzung von Reaktionszeiten, Synchronisation von Produktionsprozessen oder die engere Verzahnung mit dem Kunden. Vorteil von M2M-Systemen in der Automatisierungstechnik ist die gesteigerte Mobilität, wodurch die Datenendpunkte im Produktionssystem ohne Konfigurationsaufwand beliebig platzierbar sind (plug & play). Die typische Laufzeit von M2M-Komponenten beträgt im Produktionssystem teilweise über 20 Jahre (vgl. [21] S. 16). Für die M2M-Kommunikation werden in der Automatisierungstechnik hersteller- und dienstleisterübergreifende Standards definiert. Somit ist durch die Standardisierung eine größtmögliche Integration und Interoperabilität auf der Anwendungs- und Datenebene sichergestellt (vgl. [20] S. 2, [22] Folie 7, [26] S. 11 u. S. 15). Ein vorgeschlagener Standard für die M2M-Kommunikation ist die OPC Unified Architecture (OPC UA). In Kapitel 2.3 wird das M2M-Kommunikationprotokoll detailliert beschrieben.

### 2.2.3 Architekturen, Protokolle und Formate der Machine-to-Machine Kommunikation

Für die M2M-Kommunikation zwischen Datenendpunkten und dem Datenintegrationspunkt sind hersteller- und dienstleisterübergreifende Standards erforderlich. Abbildung 2.9 visualisiert in einer Übersicht verschiedene Architekturen, Protokolle und Formate der M2M-Kommunikation, die im folgenden Unterkapitel grob betrachtet werden. Als Grundlage für Betrachtung wird das OSI-Schichtenmodell verwendet.

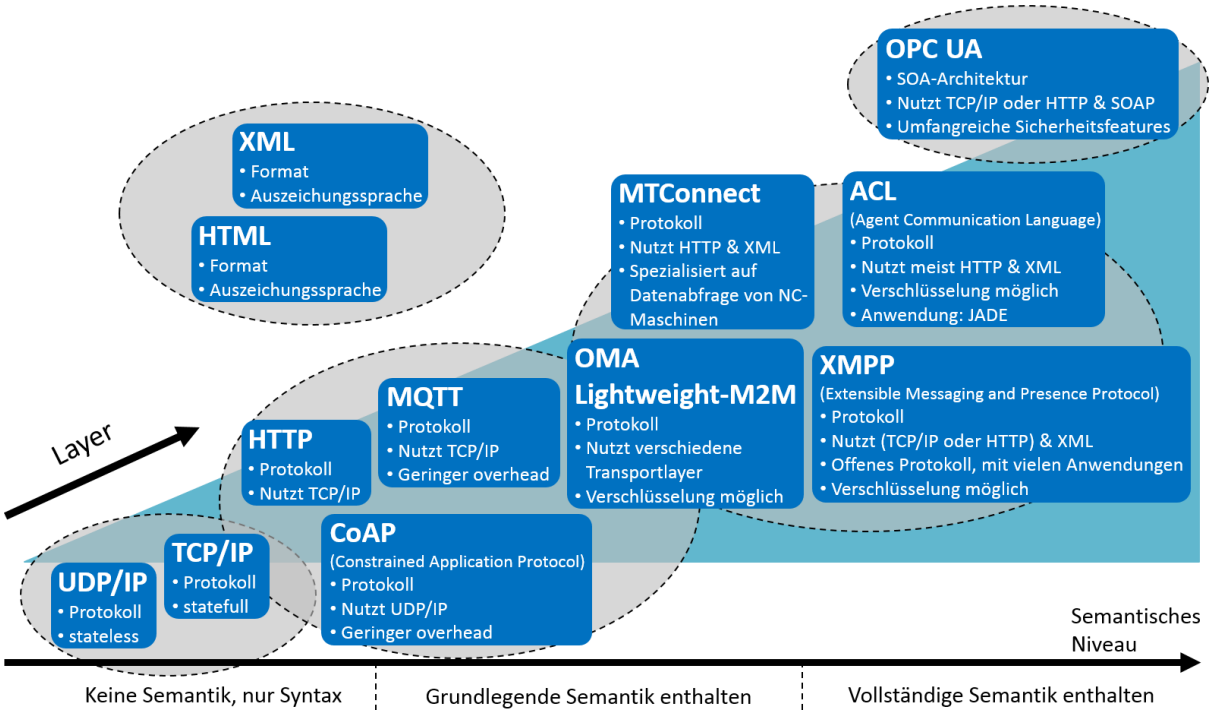


**Abbildung 2.9:** Architekturen, Protokolle und Formate der M2M-Kommunikation (Quelle: [28])

Auf der Vermittlungsschicht (Schicht 3) erfolgt die (eindeutige) Adressierung der Datenpakete mit dem Internet Protocol (IP). Die darauf aufbauende Transportschicht (Schicht 4) spezifiziert das Transportprotokoll für die Ende-zu-Ende Verbindung. Im Transportprotokoll sind die Regeln und Formate (Syntax) definiert, nach welchen der Transport und Aufbau von Datenpaketen erfolgt. Als Transportprotokoll wird in der M2M-Kommunikation TCP oder UDP verwendet. Transmission Control Protocol (TCP) ist ein verbindungsorientiertes und zuverlässiges Protokoll. Das User Datagram Protocol (UDP) ist im Gegensatz zu TCP ein verbindungsloses und unzuverlässiges Protokoll. Der Aufbau sowie die Eigenschaften der Transportprotokolle sind im Buch „Computernetzwerke“ von Andrew S. Tanenbaum und David J. Wetherall detailliert beschrieben. Auf die Transportschicht bauen die anwendungsorientierten Schichten auf. Die Sitzungsschicht (Schicht 5) verwaltet die Konversation zwischen

zwei oder mehreren Teilnehmern. Typische Dienste dieser Schicht sind der Aufbau und Abbau einer Session, die Koordination und Synchronisation der Sitzungsteilnehmer, das Versenden von Fehlermeldungen sowie die Verwaltung von Sitzungen nach Verbindungsabbrüchen (vgl. [29] Folie 7). Aufgabe der Darstellungsschicht (Schicht 6) ist die Verarbeitung und Kodierung der Daten in einem definierten Format (Syntax), die Datenkompression oder auch die Datenverschlüsselung. Für die Darstellung der Daten wird der ASCII- oder Unicode-Standard verwendet. Typische Formate in der M2M-Kommunikation sind beispielsweise HTML oder XML. Die Anwendungsschicht (Schicht 7) „stellt die Schnittstelle zwischen Netzwerkprotokollen und Anwendungen dar“ [29]. Im weiteren Teil des Kapitels werden verschiedene Protokolle und Architekturen der M2M-Kommunikation vorgestellt. Die Protokolle und Architekturen befinden sich auf den anwendungsorientierten Schichten (Schicht 5-7) und verwenden als Transportschicht TCP oder UDP.

#### HTTP

Das HyperText Transfer Protocol (HTTP) ist ein universelles, zustandsloses und objektorientiertes Protokoll zur Datenübertragung (vgl. [31], [32]). Die Datenübertragung zwischen einem Client und Server erfolgt nach dem Request/Response-Prinzip. Als Transportschicht verwendet HTTP das Transmission Control Protocol (TCP). HTTP wird hauptsächlich zur Übertragung von Daten zwischen Browser und Web-Server verwendet. Durch die universelle Struktur kann das Protokoll sehr einfach für andere Zwecke, wie beispielsweise zur Maschine-to-Machine Kommunikation eingesetzt werden (vgl. [30]). Um mit HTTP eine hersteller- und dienstleisterübergreifende M2M-Kommunikation sicherzustellen, ist wiederum ein darauf aufbauendes Protokoll bzw. ein darauf aufbauender Standard erforderlich (vgl. [30]). HTTP wird lediglich zur Datenübertragung genutzt.

#### CoAP

Das Constrained Application Protocol (CoAP) ist ein generisches Webprotokoll auf der Anwendungsschicht, welches speziell für die M2M-Kommunikation entwickelt wurde. Hauptziel von CoAP ist die Datenübertragung mit minimalem Overhead zwischen einem oder mehreren Endpunkten. Durch den minimalen Overhead ist das Protokoll auch auf ressourcenbeschränkten Geräten, wie beispielsweise einem 8-Bit Mikrocontroller, verwendbar.
Als Transportschicht verwendet CoAP das User Datagram Protocol (UDP). Um dennoch eine zuverlässige Datenübertragung zwischen den Endpunkten sicherzustellen bietet CoAP auf der Anwendungsschicht zwei verschiedene Datenübertragungsmodi an und nummeriert die Datenpakete mit einer 4-Byte ID durch. Bei der zuverlässigen Datenübertragung sendet ein Endpunkt eine sogenannte Confirmable Message (CON) zu einem anderen Endgerät und startet einen Timeout. Innerhalb des Timeouts muss der empfangende Endpunkt die Nachricht beim Sender mit einer Acknowledge Message (ACK) quittieren. Die Acknowledge Message (ACK) beinhaltet die 4-Byte ID der vorherigen Nachricht. Bei der unzuverlässigen Datenübertragung sendet der Endpunkt die Daten mittels einer Non-Confirmable Message (NON). Die Non-Confirmable Message (NON) wird nicht vom Empfänger quittiert.

Die Datenübertragung zwischen den Endpunkten erfolgt nach dem Client-Server Prinzip. Der CoAP-Client erstellt einen Request und erhält anschließend vom CoAP-Server eine Response. In CoAP kann ein Endpunkt sowohl Server als auch Client sein. Das Modell zur Datenübertragung ähnelt dem HyperText Transfer Protocol (HTTP). Weitere optionale Funktionen von CoAP ist der Discovery- und Multicast-Service. Mit dem Discovery-Service kann sich ein CoAP-Server im Netzwerk bekanntmachen. Aufgabe des Multicast-Services ist die Versendung/Adressierung von Datenpaketen an mehrere CoAP-Endpunkte. Der CoAP Multicast-Service basiert auf dem IP-Multicast.
Die Internet Engineering Task Force (IETF) veröffentlichte die Spezifikation des Protokolls als Standard unter dem Titel „RFC 7252“ [33]. Zur Erstellung des Abschnittes wurde der Standard als Quelle herangezogen.

#### MQTT

MQ Telemetry Transport (MQTT) ist ein einfaches, leichtgewichtiges und ereignisgesteuertes Protokoll für die M2M-Kommunikation. Die Kommunikation zwischen den Endpunkten erfolgt auf Basis des Publish-/Subscribe-Verfahrens.
Abbildung 2.10 zeigt die technische Umsetzung des MQTT-Protokolls. Der MQTT-Broker ist ein zentraler Messaging-Server, welcher die M2M-Kommunikation zwischen MQTT-Clients verwaltet (vgl. [34] S. 23). MQTT-Clients sind die Endpunkte eines M2M-Systems. Für die Verwaltung der Kommunikation zwischen den MQTT-Clients verwendet der MQTT-Broker sogenannte Topics (vgl. [36], [34] S. 26). Die Endgeräte können Daten zu einem Topic an den MQTT-Broker senden (publish). Aufgabe des MQTT-Brokers ist die Weiterleitung der Daten an alle Endpunkte die den Topic-String abonniert haben (vgl. [36], [34] S. 26, [35]). Ein Endpunkt wird zum Abonnenten (Subscriber) eines Topics, in dem er den Topic-String beim MQTT-Broker abonniert (subscribe).
Beispiel für Abbildung 2.10: Der Rechner und das Mobilgerät abonnieren (subscribe) Daten, die unter dem Topic-String „Temperatur“ veröffentlicht werden. Im System befindet sich ein Temperaturfühler, welcher die aktuelle Temperatur zyklisch erfasst und die Werte unter den Topicstrings „Temperatur/Büro“ an den MQTT-Broker sendet (publish). Der MQTT-Broker leitet anschließend die Werte an alle Abonnenten (Subscriber) des Topicstrings „Temperatur/Büro“ weiter.
Hauptvorteil durch den MQTT-Broker ist, das die Subscriber und Publisher sehr einfach mittels Topicstrings miteinander kommunizieren und dadurch keinerlei Informationen voneinander benötigen. MQTT ist durch das Publish-/Subscribe-Verfahren optimal geeignet für Netzwerke mit Datenlimits, geringer Bandbreite oder hoher Latenz. Eine weitere Eigenschaft von MQTT ist die Einfachheit des Protokolls (vgl. [36], [35], [34] S. 12 f.). Viele Applikationen benötigen zur M2M-Kommunikation lediglich vier Befehle, die mittels Bibliotheksmethoden einfach aufrufbar sind: *CONNECT*, *PUBLISH*, *SUBSCRIBE* und *DISCONNECT* (vgl. [35]). Als Transportschicht verwendet MQTT das Transmission Control Protocol (TCP). MQTT ist ein Open-Source-Protokoll. Auf dem Markt gibt es eine Vielzahl von kostenlosen oder kostenpflichtigen Bibliotheken, in welchen das MQTT-Protokoll implementiert ist (vgl. [34] S. 10-12).

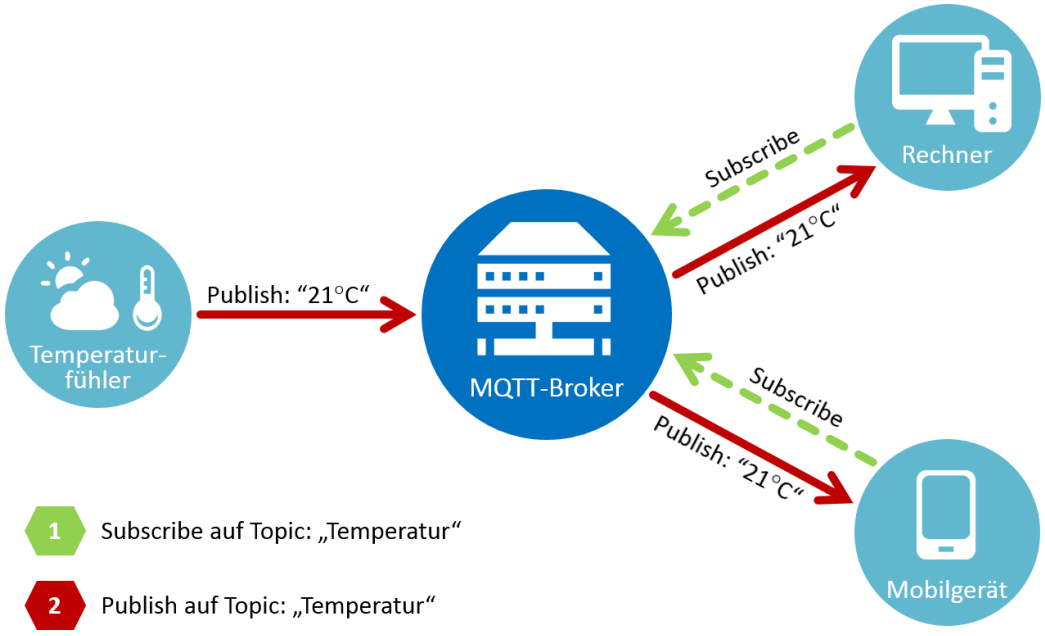


**Abbildung 2.10:** Technische Umsetzung MQ Telemetry Transport (MQTT) (Quelle: [36])

### OMA Lightweight M2M (LWM2M)

Die Open Mobile Alliance (OMA) entwickelte LWM2M als industriellen Standard für die M2M-Kommunikation. In Abbildung 2.11 ist die Architektur von OMA Lightweight M2M dargestellt. Ein LWM2M-System besteht aus beliebig vielen LWM2M-Devices und einem LWM2M-Server. Jedes LWM2M-Device setzt sich zusammen aus einem LWM2M-Client und Datenobjekten. Der LWM2M-Server hostet die Gerätemanagement-Applikation sowie verschiedene M2M-Applikationen (vgl. [37] S. 7, [40]). Für den Datenaustausch zwischen LWM2M-Server und LWM2M-Devices definiert der Standard vier logische Schnittstellen: *1.) Bootstrap*, *2.) Device Discovery und Registration*, *3.) Object und Resource Access*, *4.) Information Reporting* (vgl. [37] S. 6, [38] S. 1, [40]).
Aufgaben der *Bootstrap*-Schnittstelle sind die Zugriffskontrolle zwischen einem LWM2M-Server und den LWM2M-Devices sowie die Konfiguration der LWM2M-Devices für eine spätere Kommunikation mit dem LWM2M-Server (vgl. [37] S. 8, [40]).
Ein M2M Endpunkt verwendet die *Device Discovery und Registration*-Schnittstelle zur An- und Abmeldung beim LWM2M-Server (vgl. [39] Folie 4, [40]).
Mit der *Object und Resource Access*-Schnittstelle kann der LWM2M-Server Objekte und Ressourcen auf einem LWM2M-Client *LESEN*, *SCHREIBEN*, *AUSFÜHREN*, *ERSTELLEN* oder *LÖSCHEN* (vgl. [38] S. 1, [39] Folie 4, [40]). Ein LWM2M-Client kann beliebig viele Ressourcen besitzen. Mehrere Ressourcen werden logisch in einem Objekt verwaltet (vgl. [37] S. 8).
Zur Überwachung von Objekten eines LWM2M-Devices verwendet der LWM2M-Server die *Information Reporting*-Schnittstelle (vgl. [40], [39] Folie 4). Der LWM2M-Server setzt dazu einen Observer (Beobachter) auf ein Objekt an und erhält bei Änderungen eine Banachrichtung vom LWM2M-Client.

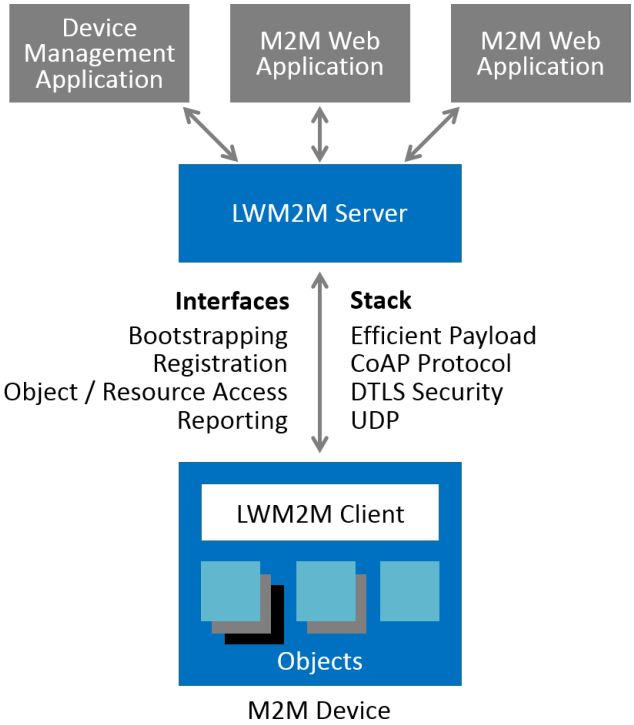


**Abbildung 2.11:** Architektur OMA Lightweight M2M (Quelle: [37] S. 7)

Bei dem oben beschriebenen LWM2M-Standard handelt es sich um ein leichtgewichtiges, offenes und sicheres Protokoll, welches auch auf ressourcenbeschränkten Geräten (Flash und RAM) implementierbar ist (vgl. [37] S. 2 u. S. 6, [38] S. 1). Der Lightweight M2M-Standard baut für die Datenübertragung auf dem Constrained Application Protocol (CoAP) auf (vgl. [39], [37] S. 2).

### MTConnect

MTConnect ist ein industrieller Standard für den Datenaustausch zwischen Applikationen und Maschinen in einem Produktionssystem. Produktionssysteme, welche mit MTConnect verbunden sind, bestehen aus fünf Grundkomponenten: Applikation, Agent, Adapter, Device und das Netzwerk. Der Aufbau der Grundkomponenten ist in Abbildung 2.12 dargestellt und wird nachfolgend erläutert (vgl. für diesen Abschnitt [41] S. 7, [44] S. 13-16, [42] S. 11).

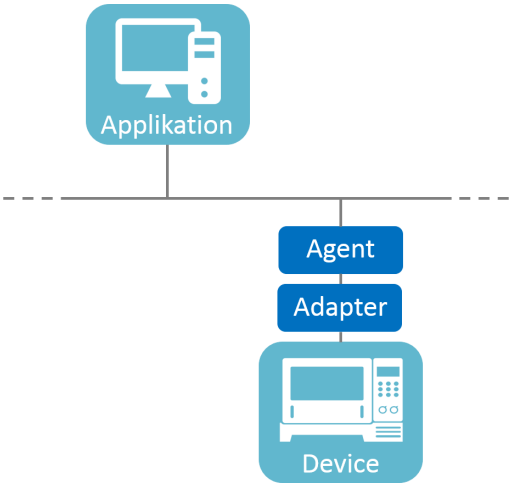


**Abbildung 2.12:** Grundkomponenten MTConnect-System (Quelle: [37] S. 7)

Applikationen des Produktionssystems fordern mittels MTConnect Daten von Maschinen an und verarbeiten diese (vgl. [42] S. 29, [41] S. 7). Verarbeitungsschritte einer Applikation sind beispielsweise die Speicherung, Bearbeitung oder Visualisierung von Daten (vgl. [41] S. 7). Das Device ist ein beliebiges Gerät in einer Fabrik. Typische Devices sind Werkzeugmaschinen, Teile von Werkzeugmaschinen oder auch Datenquellen. Der Adapter ist eine optionale Komponente, welche den Agenten mit dem Device verbindet (vgl. [44] S. 13, [41] S. 7, [42] S. 3). Aufgabe des Adapters ist die MTConnect Anbindung eines Devices an den Agenten. Der Agent sammelt, strukturiert und speichert Daten eines Devices (vgl. [42] S. 27). Erhält ein Device von einer Applikation eine Anfrage, verarbeitet und beantwortet der Agent diese für ihn (vgl. [41] S. 7). Ein Agent kann mehrere Devices im Netzwerk repräsentieren. Das Netzwerk stellt die physische Verbindung zwischen einer Applikation und einem Gerät sicher. MTConnect empfiehlt für die physische Verbindung bewährte Netzwerktechnologien (vgl. [41] S. 16+19, [44] S. 7). Agent und Adapter können auf drei Arten implementiert werden. Bei der nativen Implementierung befinden sich Adapter und Agent im Device. Geräte mit nicht nativer Implementierung sind mittels der *Translation-* oder *Connection-Unit* nachrüstbar. Mit einem Adapter kodiert die *Translation Unit* Daten eines Gerätes in MTConnect konforme Daten (vgl. [44] S. 17, [41] S. 10). Die *Connection-Unit* ist eine separate Einheit, welche Daten zu einem Device erfasst und sammelt (vgl. [41] S. 11, [44] S. 17). Durch dieses Vorgehen kann eine Applikation benötige Daten zu einem Gerät abfragen, welche das Gerät nicht selbst bereitstellt. Die Datenübertragung zwischen Applikation und Gerät basiert auf HTTP (vgl. [41] S. 6 u. S. 18, [42] S. 11). Mit einem Uniform Resource Locator (URL) kann eine Applikation Daten von einem Gerät anfordern. Die URL für eine Anfrage ist folgendermaßen aufgebaut (vgl. [43]:

*http://<AgentURL>/<DeviceName>/<Request>?path=<Parameter>*

MTConnect definiert für den Request vier grundlegende Befehle: *ASSETS*, *PROBE*, *CURRENT* und *SAMPLE*. Mit dem *ASSETS*-Befehl kann eine Anwendung alle angeschlossenen Assets eines Gerätes sowie Informationen zu den einzelnen Assets abfragen (vgl. [41] S. 14 f., [44] S. 25-27, [42] S. 46). Der *PROBE*-Befehl liefert Metadaten zu einem Gerät (vgl. [44] S. 30, [41] S. 18). In den Metadaten sind die Datenelemente und der Agent eines Gerätes beschrieben. Den aktuellen Wert eines Datenelements kann die Applikation mit dem *CURRENT*-Befehl abfragen (vgl. [42], [41] S. 24, [43] S. 31). Der *SAMPLE*-Befehl liefert eine Datenhistorie für die Datenelemente eines Geräts (vgl. [42] S. 11). Die oben beschriebenen Grundbefehle können mit Parametern in der URL für verschiedene Anforderungen anpasst werden. Eine Applikation erhält die angeforderten Daten in einem XML-File. Der Aufbau des XML-Files ist von MTConnect standardisiert (vgl. [42] S. 11, [41] S. 6).

### ACL

Ein weiterer Ansatz zur Realisierung eines M2M-Systems sind Multiagentensysteme. Die FIPA (Foundation for Intelligent Physical Agents) entwickelte für die Agentenverwaltung ein allgemeines Referenzmodell. Um eine größtmögliche Interoperabilität zwischen Agenten im Referenzmodell sicherzustellen wird die Agent Communication Language (ACL) verwendet. Es handelt sich dabei um ein standardisiertes Protokoll für die Agentenkommunikation.

Im folgenden Abschnitt wird das FIPA-Referenzmodell für die Agentenverwaltung beschrieben. Das Referenzmodell ist in Abbildung 2.13 dargestellt.
Agenten sind intelligente Softwareeinheiten die autonom agieren und mittels der Agent Communication Language (ACL) kommunizieren. Auf der Agentenplattform (AP) können Agenten verschiedene Dienste anbieten oder angebotene Dienste in Anspruch nehmen. Damit ein Agent auf der Agentenplattform identifizierbar ist, muss sich dieser beim Agentenverwaltungssystem (AMS) registrieren und erhält eine eindeutige Agentennummer (AID) zugewiesen. Ein Agent muss sich zwingend beim Agentenverwaltungssystem registrieren, andernfalls ist er nicht Teil der Agentenplattform.
Das Agentenverwaltungssystem administriert die Agenten einer Agentenplattform. Teil des Agentenverwaltungssystems ist ein Verzeichnis, in welchem alle registrierten Agenten mit Agentennummer (AID) und Adresse gelistet sind. Die Agenten können mit sogenannten White-Page-Services Informationen aus dem Verzeichnis abfragen. Das Agentenverwaltungssystem ist nur einmal pro Plattform vorhanden.
Im Dienste-Verzeichnis (Directory Facilitator) sind die angebotene Dienste einer Agentenplattform aufgelistet und beschrieben. Ein Agent kann Informationen aus dem Verzeichnis abfragen sowie seine eigenen Dienste registrieren bzw. anbieten. Auf einer Agentenplattform können mehrere Dienste-Verzeichnisse vorhanden sein. Denkbar sind auch Kooperationen zwischen den Dienste-Verzeichnissen.
Aufgabe des Nachrichtenübertragungsdienstes (Message Transport System) ist die Bereitstellung eines Dienstes, mit welchem die Agenten innerhalb einer Agentenplattform und mit Agenten anderer Agentenplattformen kommunizieren.

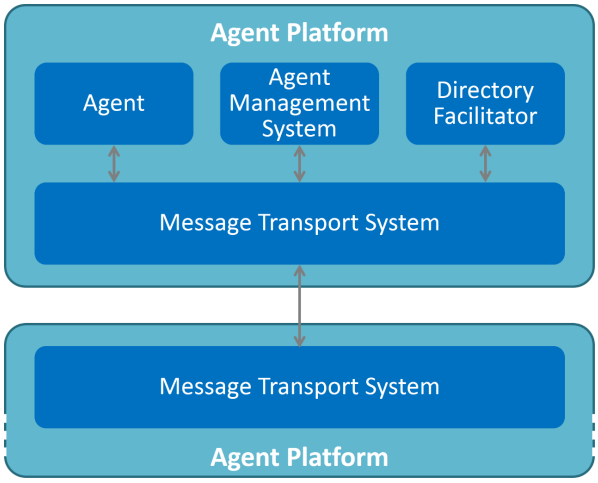


**Abbildung 2.13:** FIPA Referenzmodell (Quelle: [46] S. 5)

Für die Realisierung eines Multiagentensystems sind verschiedene Frameworks verfügbar. Ein bekanntes Framework für die Realisierung eines Multiagentensystems ist das Java Agent DEvelopment Framework (JADE). Es handelt sich dabei um ein quelloffenes Projekt, welches unter der LGPL-Lizenz veröffentlicht wurde. Zur Datenübertragung sind in dem Framework verschiedene Protokolle und Formate definiert. Hauptsächlich wird HTTP für die Datenübertragung und XML als Format verwendet (vgl. [47]). Die Informationen dieses Abschnittes basieren auf folgenden Quellen: [46], [48] S. 93-119, [45] S. 3-5 und [47].

### XMPP

XMPP (Extensible Messaging and Presence Protocol) ist ein erweiterbares Nachrichten- und Anwesenheitsprotokoll für eine echtzeitnahe Kommunikation zwischen Endpunkten. Für die Kommunikation verwendet XMPP auf der Transportschicht TCP und für den Datenaustausch den XML-Standard. Das Protokoll ist ein kostenfreier, offener und einfach verständlicher Standard für verschiedene Anwendungsbereiche. XMPP stellt für folgende Anwendungsbereiche Technologien bereit: Instant Messaging (Nachrichtensofortübertragung), Anwesenheitsanzeige, Mehrparteien Chats, Sprach- und Videoanrufe, Kollaboration[1], leichtgewichtige Middleware, Inhaltsverwaltung sowie die allgemeine Übertragung von XML-Daten (vgl. [50]).
Die IETF (Internet Engineering Task Force) veröffentlichte die Spezifikation des Standards in den RFC-Dokumenten 6120, 6121 und 6122. Zur Entwicklung einer XMPP-Applikation gibt es eine Vielzahl von Bibliotheken, in welchen die Grundfunktionen von XMPP implementiert sind (vgl. [52]).
Ein XMPP-Netzwerk besteht aus vier logischen Grundkomponenten: XMPP-Server, XMPP-Client, Gateway und das Netzwerk (vgl. [49]). In Abbildung 2.14 ist ein allgemeines XMPP-

[1] Kooperierende Zusammenarbeit mehrerer Personen innerhalb eines XMPP-Netzwerks. Dadurch können beispielsweise mehrere Personen zur gleichen Zeit an einem Dokument arbeiten.

Netzwerk dargestellt. Jeder XMPP-Server bildet mittels einer Domain ein eigenes lokales Netzwerk, in welchem mehrere XMPP-Clients gehostet sind (vgl. [50], [51] S. 11). In einem XMPP-Netzwerk werden mehrere XMPP-Server zu einem globalen Netzwerk zusammengefasst. Dadurch wird eine Kommunikation zwischen den Domains möglich. Aufgabe der XMPP-Server ist die Verbindungsverwaltung zwischen den Servern, Clients und anderen Entitäten sowie die Erfassung der Anwesenheit von Clients in einer Domain (vgl. [49], [50]). Optional kann ein XMPP-Server mit einem Gateway die Daten von XMPP in ein anderes Protokoll sowie ein anderes Protokoll in XMPP übersetzen. Dadurch ist die Kommunikation zwischen XMPP und einem Fremdsystem möglich. Im XMPP-Netzwerk in Abbildung 2.14 besitzt beispielsweise der „domain2"-XMPP-Server ein Gateway für die IRC (Internet Relay Chat) Kommunikation. Damit ein XMPP-Client Zugriff auf das XMPP-Netzwerk erhält, verbindet sich dieser mit dem XMPP-Server in seiner Domain. XMPP verwendet für die Authentifizierung der Clients und Server das Simple Authentication and Security Layer (SASL) Framework (vgl. [51] S. 13 f., [49]).

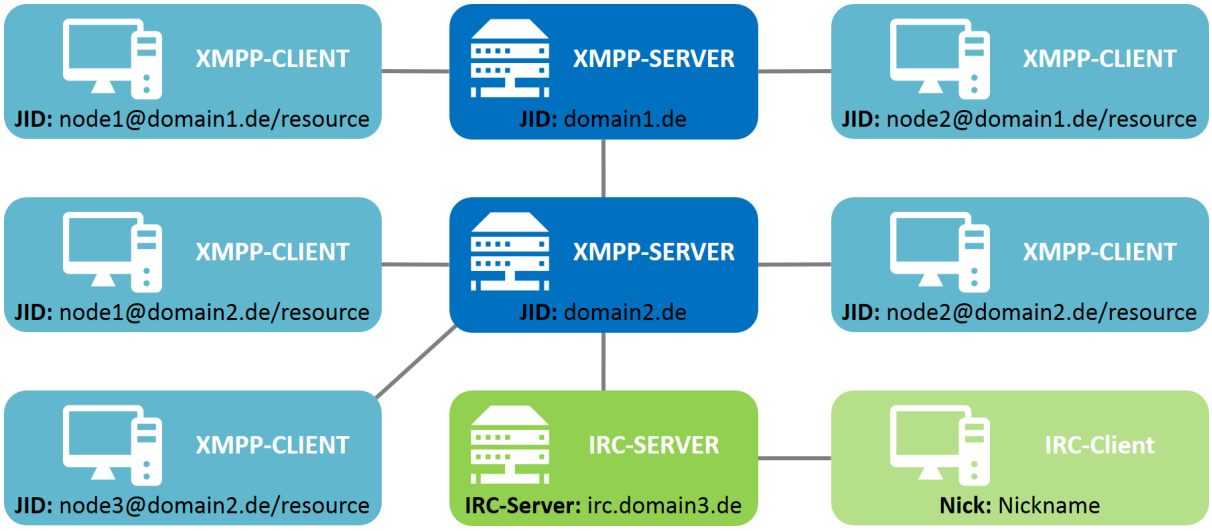


**Abbildung 2.14:** Aufbau eines XMPP-Netzwerk

Die Adressierung der einzelnen Entitäten im XMPP-Netzwerk erfolgt mittels dem Jabber-Identifier (JID). Der Jabber-Identifier ist wie folgt aufgebaut (vgl. [49]):

*node @ domain / resource*

Im XMPP-Netzwerk wird jeder XMPP-Server mittels einer eindeutigen Domain adressiert. Die Adressierung eines XMPP-Clients erfolgt mit dem Knotennamen (node) sowie der Domain des XMPP-Servers, die den Client hostet. Mit dem Resource-Teil kann ein Knoten optional Angaben zu seinem verwendeten Gerät oder Standort machen (vgl. [51] S. 14 f.). In Abbildung 2.14 ist ein Beispiel für die Adressierung mittels Jabber-Identifiers (JID) dargestellt. Die Kommunikation zwischen den Entitäten erfolgt mittels XML-Streams. Ein XML-Stream ist eine langlebige

TCP-Verbindung zwischen zwei Entitäten, über welche beliebig viele XML-Stanzas übertragen werden können (vgl. [49], [51] S. 12). XML-Stanzas sind strukturierte XML-Elemente, in welche die zu übertragenden Information eingebettet werden.

#### OPC UA

OPC UA (OPC Unified Architecture) ist eine serviceorientierte Architektur für die industrielle M2M-Kommunikation. Im Grundlagendokument wird der Standard detailliert vorgestellt.

## 2.3 OPC Unified Architecture

Der OPC Unified Architecture Standard definiert eine serviceorientierte Architektur für die industrielle M2M-Kommunikation. Das folgende Kapitel beschäftigt sich mit den Grundlagen des OPC Unified Architecture Standards.

In einer **Einleitung** wird zunächst OPC-Klassik – der Vorgänger von OPC UA – vorgestellt sowie der technologische Übergang von OPC-Klassik zu OPC UA aufgezeigt.

**Kapitel 2.3.2** liefert einen Überblick über die Spezifikation des OPC UA Standards. Die Spezifikation umfasst dreizehn Teile, welche in drei Themenbereiche untergliedert sind.

Das Adressraummodell, die Informationsmodelle und der Adressraum sind in **Kapitel 2.3.3** erläutert. Die OPC UA Spezifikation definiert im Adressraummodell das Adressraum-Konzept sowie verschiedene Knotenklassen, Basis-Referenztypen und Basis-Datentypen. Das Adressraummodell wird für die Erstellung des Adressraums verwendet und bildet die Grundlage der Informationsmodelle. Es handelt sich dabei um ein Metamodell. Informationsmodelle erweitern das Adressraummodell um Referenz- und Datentypen sowie definierte Instanzen. Standardisierungsgremien und Hersteller können domänenspezifische Informationsmodelle auf Grundlage des Adressraummodells spezifizieren. Jeder OPC UA Server hat einen Adressraum, in welchem er Prozesse, Systemkomponenten und Systeme mit Knoten und Referenzen abbildet.

Die OPC UA Services sind in **Kapitel 2.3.4** beschrieben. In OPC UA erfolgt die Kommunikation nach dem Client-Server-Prinzip. Im Netzwerk stellt ein OPC UA Server auf der Applikationsebene eine Vielzahl von Services bereit. Möchte ein Client einen Service in Anspruch nehmen, sendet dieser einen Service Request. Der OPC UA Server nimmt den Service Request entgegen und beantwortet die Serviceanfragen mit einem Service Response. OPC UA untergliedert die spezifizierten Services in zehn Service Sets. Ein Service Set beinhaltet mehrere zusammengehörende Services. Das NodeManagement Service Set beinhaltet beispielsweise Services für das Erstellen, Lesen oder Manipulieren von Knoten im Adressraum eines OPC UA Servers.

**Kapitel 2.3.5** beschäftigt sich mit den Transportprotokollen und Datenkodierungen von OPC UA. Transportprotokolle und Datenkodierungen sind für die Übertragung der Servicenachrichten zwischen Client und Server erforderlich. Der OPC UA Standard stellt für die Übertragung der Servicenachrichten zwei Optionen bereit: OPC UA TCP Binary oder Web-Services. OPC UA TCP Binary verwendet auf der Transportebene das UA TCP Protokoll. Auf der Serialisierungsebene wird die binäre Datenkodierung - UA Binary – eingesetzt. Web-Services verwenden als Transportprotokoll HTTP oder HTTPS. Die Datenkodierung erfolgt auf der Serialisierungsebene mittels XML. Weiterhin werden die Vor- und Nachteile der einzelnen Technologien erörtert.

In **Kapitel 2.3.6** ist das Sicherheitsmodell von OPA UA beschrieben. Aufgabe des Sicherheitsmodells ist die Authentifizierung auf der Applikations- und Kommunikationsebene sowie die Sicherstellung von Vertraulichkeit und Integrität auf der Kommunikationsebene.

**Kapitel 2.3.7** erläutert die vier grundlegenden Architekturmuster (Architectural Pattern) für die Anordnung der OPC UA Clients und Server in einem OPC UA System.

Das OPC UA Discovery ist in **Kapitel 2.3.8** beschrieben. Mit dem OPC UA Discovery können Clients registrierte OPC UA Server vom Discovery Server abfragen. Damit ein OPC UA Server beim Discovery Server als verfügbar gelistet wird, muss sich dieser bei ihm registrieren. Die OPC Unified Architecture Spezifikation definiert für verschiedene Anwendungsfälle drei Arten von Discovery Servern: Lokaler Discovery Server (LDS), lokaler Discovery Server mit Multicast-Erweiterung (LDS-ME) und globaler Discovery Server (GDS).

Das **Kapitel 2.3.9** ist in zwei Teile untergliedert. Der erste Teil liefert einen Überblick über verfügbare Bibliotheken für den OPC UA Standard. Im zweiten Teil erfolgt die begründete Auswahl einer OPC UA Bibliothek. Die ausgewählte OPC UA Bibliothek wird anschließend für die Realisierung des verteilten IT-Systems auf Basis von OPC UA verwendet.

Die Ergebnisse der Nutzungsanalyse von OPC UA werden in **Kapitel 2.3.10** zusammengefasst. Im Rahmen der Nutzungsanalyse wurden potentielle Anwendungsbereiche des OPC UA Standards theoretisch analysiert.

### 2.3.1 Einleitung

In den 1990er Jahren wurden überwiegend Betriebssysteme von Microsoft in der Automatisierungstechnik eingesetzt. Für die Kommunikation zwischen Anwendungen auf einem lokalen Rechner und Anwendungen in einem Netzwerk stellte Microsoft die COM- (Component Object Model) und DCOM- (Distributed Component Object Model) Technologie mittels einer Programmschnittstelle (API) bereit. Im Jahr 1994 gründeten mehrere Hersteller aus der Automatisierungsbranche die OPC Foundation als non-profit Organisation. Ziel der Organisation war die Spezifizierung eines herstellerübergreifenden Standards für die Kommunikation zwischen Automatisierungsanlagen und Microsoft-basierten Systemen auf Basis der COM- und DCOM-Technologie.
Im August 1996 veröffentlichte die OPC Foundation mit der *OPC Data Access (OPC DA)* Spezifikation die erste Version von OPC Klassik. Die Kommunikation in OPC DA erfolgt nach dem Client-Server Modell. Der OPC DA Server bildet in einem Adressraum die Prozessdaten eines Systems mit Objekten ab. OPC DA Clients können mittels definierter Methoden durch den Adressraum eines OPC DA Servers navigieren sowie die gespeicherten Daten auslesen, schreiben und überwachen. Die Spezifikation wird hauptsächlich zur Übertragung von Echtzeitdaten von der Steuerungsebene in die Prozessleitebene verwendet (Automatisierungspyramide).

Zwei Jahre später wurde der Standard um die *OPC Alarms & Events (OPC AE)* Spezifikation erweitert. Mit der Spezifikation kann ein OPC AE Server Event- und Alarmbenachrichtigungen an einen OPC AE Client senden. Ein OPC AE Client erhält Benachrichtigungen von einem OPC AE Server in dem er diesen abonniert. Die Art der Benachrichtigungen können mit Filterparametern angepasst werden. Sobald sich überwachte Prozessdaten ändern, werden die Clients mit Alarm-Benachrichtigungen informiert. Dies erfolgt beispielsweise bei der Überschreitung von Schwellenwerten eines Prozessparameters.
Im Jahr 1999 folgte die *OPC Historical Data Access (OPC HDA)* Spezifikation. Die Spezifikation ermöglicht die Speicherung von Prozessdaten in einer Historie auf einem OPC HDA Server. Mit spezifizierten Methoden können Clients die Historie-Daten lesen und ändern. Für das Lesen der Historie-Daten stehen zwei Optionen zur Auswahl. Mit der ersten Option können Clients die Rohdaten einer oder mehrerer Variablen für einen definierten Zeitbereich oder Zeitpunkt abfragen. Eine weitere Möglichkeit ist die Aggregation von Historie-Daten. Mit Aggregationsfunktionen können die Historie-Daten von einer oder mehrerer Variablen für einen definierten Zeitbereich oder Zeitpunkt zusammengefasst werden.
Die letzte Erweiterung von OPC Klassik erfolgte 2003 mit der *OPC XML-DA* Spezifikation. Die Spezifikation hat einen ähnlichen Funktionsumfang wie der OPC DA Standard. Der Hauptunterschied ist die Kommunikation mittels Web-Services sowie die Reduzierung der Methoden. In OPC XML-DA stehen lediglich acht Methoden zur Kommunikation zwischen Client und Server zur Verfügung. Die webbasierte Kommunikation erfolgt mittels HTTP und XML-Dokumenten. Damit ist die Spezifikation erstmalig plattformunabhängig und nicht wie bisher an Microsoft Betriebssysteme durch die COM/DCOM-Technologie gebunden. Nachteile von OPC XML-DA im Vergleich zu OPC DA sind ein enormer Ressourcenverbrauch sowie die langsamere Datenübertragung.
OPC Klassik wurde ursprünglich als herstellerübergreifender Standard für die Kommunikation zwischen Automatisierungsanlagen und Microsoft-basierten Systemen auf Basis der COM- und DCOM-Technologie entwickelt. Im Laufe der Zeit wandelte sich der Standard von einer reinen Kommunikationsschnittstelle von Daten (OPC DA) zu einer Systemschnittstelle, welche weitere Dienste anbietet (OPC AE, OPC HDA, OPC XML-DA). Hauptnachteil von OPC Klassik ist die Verwendung der COM- (Component Object Model) und DCOM- (Distributed Component Object Model) Technologie. Die Spezifikationen sind dadurch an Microsoft Betriebssysteme gebunden und nicht plattformunabhängig. Weitere Nachteile der COM/DCOM-Technologie sind enorm aufwendige Konfigurationen sowie fehlende Sicherheitsmechanismen für die Automatisierungstechnik. Aus diesen Gründen veröffentlichte die OPC Foundation 2006 den OPC Unified Architecture (OPC UA) Standard als Nachfolger von OPC Klassik.
OPC Unified Architecture ist eine service-orientierte Architektur (SOA) und wurde speziell für die industrielle Machine-to-Machine (M2M) Kommunikation entwickelt. OPC UA spezifiziert eine serviceorientierte Architektur für die herstellerübergreifende und plattformunabhängige industrielle M2M-Kommunikation. Die Kommunikation basiert auf standardisierten Web-Technologien und erfolgt nach dem Client-Server-Prinzip. OPC UA Server bilden die Daten von Prozessen sowie die Systemkomponenten mit Knoten und Referenzen in einem Adressraum ab. Aufgrund der Skalierbarkeit kann OPC UA auf allen Ebenen der Automatisierungspyramide eingesetzt werden. Im Vergleich zu OPC DA können in OPC UA erstmalig

Standardisierungsgremien und Hersteller eigene Referenzen- und Datentypen sowie Instanzen mit Informationsmodellen definieren. Neu ist auch, dass der Adressraum Metadaten beinhalten kann. Für den gesicherten Datenaustausch zwischen Client und Server definiert OPC UA ein Sicherheitsmodell. Aufgabe des Sicherheitsmodells ist die Authentifizierung zwischen Client und Server sowie die Sicherstellung von Vertraulichkeit und Integrität auf der Kommunikationsebene. Aufgrund der Skalierbarkeit kann OPC UA auf allen Ebenen der Automatisierungspyramide eingesetzt werden. Die Skalierbarkeit wird durch die Spezifikation der vier OPC UA Profile „Standard“, „Embedded“, „Micro“ und „Nano“ erreicht. Das Standard-Profil stellt alle Funktionalitäten der Spezifikation bereit. Das Nano-Profil beinhaltet lediglich die minimalen Kernfunktionalitäten von OPC UA. Durch dieses Vorgehen kann OPC UA sowohl in Sensoren, Mikrocontrollern als auch auf Rechnern und Servern implementiert werden.

Zur Erstellung des Abschnittes wurden folgende Quellen verwendet: [59], [53] S. 1-8 + S. 115-120, [58] S. 1, [57] S. 1 + S. 2 + S. 6 und [54] S. 2 + S. 10-15.

### 2.3.2 Aufbau der OPC UA Spezifikation

Die OPC UA Spezifikation besteht aus einer dreizehnteiligen Spezifikation, welche in drei Themenbereiche untergliedert ist und als IEC Standard (IEC 62541) veröffentlicht wurde. In Abbildung 2.15 ist der Aufbau der OPC UA Spezifikation dargestellt. Einzelne Teile der Spezifikation bauen aufeinander auf. Die OPC-Foundation empfiehlt daher die Teile 1 bis 5 nacheinander abzuarbeiten. Zur Informationsbeschaffung wurden die folgenden Quellen verwendet: [61] S. 6 f., [53] S. 11-13 und [57] S. 33.

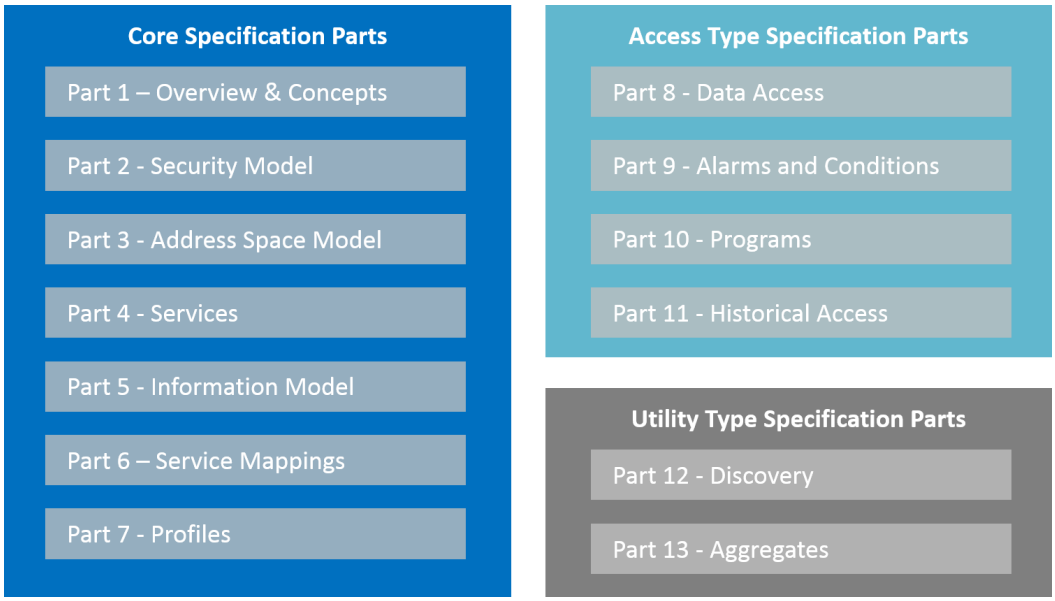


**Abbildung 2.15:** Aufbau OPC UA Spezifikation (Quelle: [61] S. 6)

#### 2.3.2.1 Basis-Spezifikation (Core Specification Parts)

In der Basis-Spezifikation (Teil 1 bis 7) sind die Kernfunktionalitäten von OPC UA beschrieben. Teil 1 liefert einen Überblick über die Spezifikation und stellt das Grundkonzept von OPC UA vor. In Teil 2 der Spezifikation ist das Sicherheitsmodell von OPC UA beschrieben. Das Sicherheitsmodell definiert die Authentifizierung zwischen Client und Server auf der Applikationsebene und stellt Vertraulichkeit und Integrität auf der Kommunikationsebene sicher. Das Adressraummodell in Teil 3 spezifiziert das Adressraum-Konzept sowie verschiedene Knotenklassen, Basis-Referenztypen und Basis-Datentypen. Es wird für die Erstellung des Adressraums verwendet und bildet die Grundlage der Informationsmodelle. Die Services für die Kommunikation zwischen Client und Server sind im vierten Teil der Spezifikation abstrakt beschrieben. Im fünften Teil sind die Richtlinien für das Erstellen von Informationsmodellen definiert. Transportprotokolle und Datenkodierungen für die Kommunikation zwischen OPC-Client und OPC-Server sind in Teil 6 (Mappings) beschrieben. Teil 7 definiert verschiedene Profile für OPC UA. Mit Hilfe der Profile wird beschrieben, welche OPC UA Funktionalitäten ein Client oder Server bereitstellt.

#### 2.3.2.2 Zugriffsmodelle (Access Type Specifications)

Die Zugriffsmodelle sind in Teil 8 bis 11 beschrieben. Teil 8 der Spezifikation beschreibt das Konzept des Datenzugriffs (Data Access). Im neunten Teil der Spezifikation definiert OPC UA ein Modell für die Überwachung von Prozessdaten und Ereignisse eines OPC UA Servers. In Teil 10 (Programs) der Spezifikation ist das Programminformationsmodell beschrieben. Mit dem Programminformationsmodell kann ein zustandsgesteuertes, komplexes und langlaufendes Programm in einer Zustandsmaschine auf dem OPC UA Server implementiert werden. Ein OPC UA Client kann mittels Services das Programm auf dem OPC UA Server ausführen und steuern. Weiterhin kann er den aktuellen Zustand sowie die Ereignisse der Zustandsmaschine abfragen. Im elften Teil (Historical Access) der Spezifikation ist der Zugriff auf historische Daten und Events beschrieben.

#### 2.3.2.3 Erweiterungen (Utility Type Specifications)

Teil 12 und Teil 13 der Spezifikation erweitern die Kernfunktionalitäten. In Teil 12 (Discovery) ist das Grundkonzept des Discovery-Prozesses beschrieben. Mit dem Discovery Prozess kann ein Client verfügbare Server in einem Netzwerk finden und Informationen zu den Servern abfragen. Teil 13 definiert typische Aggregationsfunktionen, wie beispielsweise Minimum, Maximum, Summe oder Durchschnitt, mit welchen die aktuellen oder historischen Daten eines OPC UA Server verarbeitet bzw. zusammengefasst werden können.

## 2.3.3 Adressraummodell, Informationsmodelle und Adressraum

### 2.3.3.1 Adressraummodell

Die OPC UA Spezifikation definiert im Adressraummodell das Adressraum-Konzept sowie verschiedene Knotenklassen, Basis-Referenztypen und Basis-Datentypen. Das Adressraummodell wird für die Erstellung des Adressraums verwendet und bildet die Grundlage der Informationsmodelle (vgl. [61] S. 1, [53] S. 82). Es handelt sich um ein Metamodell.

**Adressraum-Konzept**
Jeder OPC UA Server hat einen Adressraum, in welchem er Informationen strukturiert ablegt bzw. speichert. Die Strukturierung der Daten erfolgt durch Knoten, welche im Adressraum Objekte repräsentieren (vgl. [62] S. 5, [54] S. 66). Das Adressraummodell definiert acht unterschiedliche Knotenklassen. Um Knoten im Adressraum des OPC UA Servers abzubilden, werden Instanzen von den Knotenklassen erstellt. In Abbildung 2.16 ist der generelle Aufbau eines Knotens dargestellt. Knoten bestehen aus Attributen und Referenzen. Attribute sind Datenelemente, die einen Knoten beschreiben (vgl. [62] S. 6). Die Anzahl und Art der Attribute eines Knotens hängt von der instanziierten Knotenklasse ab. Referenzen werden verwendet um Knoten miteinander zu verbinden. Weiterhin wird eine Beziehung zwischen den Knoten hergestellt. Jede Referenz hat einen Quell- und Zielknoten. Durch die Referenzen zwischen den Knoten entsteht ein Adressraum mit vermaschten Strukturen (vgl. [55] S. 398).

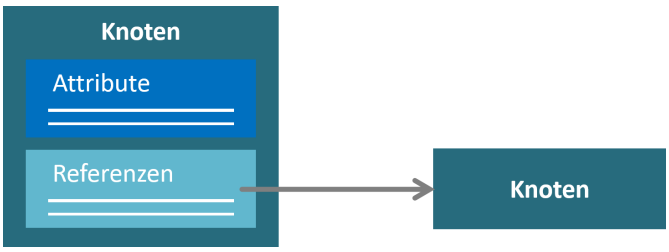


**Abbildung 2.16:** Knotenmodell (Adressraum-Konzept) (Quelle: [62] S. 5)

OPC UA Clients verwenden Services um auf den Adressraum eines OPC UA Servers zuzugreifen. In Kapitel 2.3.4 (Services) sind diese beschrieben.
Teil des Adressraummodells ist eine grafische Notation für die Modellierung der Adressräume und Informationsmodelle (vgl. [62] S. 98 f., [57] S. 400). Die grafische Notation ist an den UML-Standard angelehnt. Im Kapitel 2.3.3.2 (Adressraum) ist ein Beispiel für die Verwendung der grafischen Notation dargestellt und beschrieben.

**Knotenklassen**
Die abstrakte Basisknotenklasse spezifiziert allgemeine Attribute, welche an die Knotenklassen vererbt werden. In Tabelle 2.1 sind die allgemeinen Attribute der abstrakten Basisknotenklasse beschrieben (vgl. [53] S. 23, [62] S. 12 f., [54] S. 66):

| Attribute | Beschreibung |
|---|---|
| *NodeId* | Die Identifizierung und Adressierung von Knoten im Adressraum eines OPC UA Servers erfolgt mittels der *NodeId*. Jedem Knoten wird bei der Erstellung ein eindeutiger Identifikator vom OPC UA Server zugewiesen. |
| *NodeClass* | Das *NodeClass*-Attribut beschreibt die verwendete Knotenklasse eines Knotens. |
| *BrowseName* | OPC UA Clients können den Adressraum eines OPC UA Servers mit dem Browse Service durchstöbern. Beim Durchstöbern des Adressraums wird ein Pfad erstellt, mit welchem die durchlaufenen Knoten im Adressraum erreichbar sind. Für die Erstellung des Pfads werden die *BrowseName*-Attribute der durchlaufenen Knoten verwendet. |
| *DisplayName* | Das *DisplayName*-Attribut beinhaltet den Namen eines Knotens. OPC UA Clients verwenden die Knotennamen zur grafischen Visualisierung des Adressraums. |
| *Description* | Optional kann ein Knoten mit dem *Description*-Attribut beschrieben werden. Die Beschreibung erläutert die Bedeutung eines Knotens im Adressraum des OPC UA Servers. |
| *WriteMask* | Das *WriteMask*-Attribute ist optional und definiert die allgemeinen Schreibrechte für die Attribute eines Knotens. |
| *UserWriteMask* | Das *UserWriteMask*-Attribute ist optional und spezifiziert für den aktuell verbundenen Benutzer die Schreibrechte für die Attribute eines Knotens. |

**Tabelle 2.1:** Attribute der abstrakten Basisknotenklasse

Das Adressraummodell leitet von der abstrakten Basisknotenklasse acht Knotenklassen ab. In Tabelle 2.2 sind diese aufgelistet und beschrieben. Die OPC UA Spezifikation erlaubt keine selbstdefinierten oder erweiterten Knotenklassen (vgl. [62] S. 6, [73]). Um Knoten im Adressraum abzubilden, werden Instanzen von den Knotenklassen erstellt.

| Knotenklassen | Beschreibung |
|---|---|
| *Objekt* | Im Adressraum des OPC UA Servers werden Instanzen der *Objekt*-Knotenklasse verwendet um Prozesse, Systemkomponenten und Systeme abzubilden (vgl. [62] S. 19 + S. 21, [54] S. 70). |
| *Variable* | Instanzen der *Variablen*-Knotenklasse repräsentieren im Adressraum Datenvariablen oder beschreiben die Eigenschaften (Property) anderer Knoten (vgl. [72], [62] S. 23 f.). |

| | |
|---|---|
| *Methode* | Instanzen der *Methoden*-Knotenklasse definieren im Adressraum eines OPC UA Servers „leichtgewichtige“ Funktionen (vgl. [54] S. 70, [62] S. 12 u. S. 30 f.). Mit dem Methode Service Set (vgl. Kapitel 2.3.4.8) können Clients die gelisteten Funktionen auf dem OPC UA Server aufrufen. |
| *View* | Im Adressraum eines OPC UA Servers können beliebig viele Informationen und Daten abgebildet werden. Für einen besseren Überblick in großen und komplexen Systemen spezifiziert das Adressraummodell die *View*-Knotenklasse. Eine View beinhaltet eine Teilmenge von Knoten und Referenzen eines Adressraums (vgl. [62] S. 17, [63] S. 42, [53] S. 61 f.). Der View-Knoten bildet den Einstiegspunkt einer View (vgl. [72]). Beispiel: Ein OPC UA Server definiert einen View-Knoten für die Wartung des Systems. Die View „Wartung“ beinhaltet alle relevanten Knoten und Referenzen des Adressraums, die ein Ingenieur für die Wartung des Systems benötigt. |
| *Objekttyp* | *Objekttyp*-Knoten definieren im Adressraum den Aufbau und die Semantik von Objekten (vgl. [53] S. 37, [62] S. 21). OPC UA Clients können die Informationen und Daten eines Objektes durch den referenzierten Objekttypknoten interpretieren. |
| *Variablentyp* | *Variablentyp*-Knoten beschreiben im Adressraum die Struktur und Semantik von Variablen (vgl. [54] S. 70, [53] S. 39). Informationen und Daten einer Variable können OPC UA Clients durch den referenzierten Variablentypknoten auslegen. |
| *Referenztyp* | Knoten der *Referenztyp*-Knotenklasse definieren die Semantik von Referenzen (vgl. [53] S. 24, [72]). Hinweis: Referenzen sind Teil eines Knotens. |
| *Datentyp* | Instanzen der *Datentyp*-Knotenklasse definieren im Adressraum Datentypen (vgl. [54] S. 70, [62] S. 32). Datentypen beschreiben die Art der Daten und haben keine Semantik! Das Adressraummodell definiert bereits eine Vielzahl von Basis-Datentypen, die jedem OPC UA Server bekannt sind. Im Abschnitt „Basis-Datentypen“ werden diese beschrieben. |

**Tabelle 2.2:** Knotenklassen

**Basis-Referenztypen**

Referenzen werden verwendet um Knoten miteinander zu verbinden. Weiterhin wird eine Beziehung zwischen den Knoten hergestellt. Das Adressraummodell spezifiziert eine Vielzahl vordefinierter Basis-Referenztypen, die sich in hierarchische und nicht-hierarchische Referenztypen untergliedern ([62] S. 53 f., [72] S. 22). In Tabelle 2.3 sind die am häufigsten verwendeten Basis-Referenztypen für die Modellierung des Adressraums aufgelistet und beschrieben:

| Referenztyp | Beschreibung |
|---|---|
| *HasTypeDefinition* | Der *HasTypeDefinition*-Referenztyp wird im Adressraum eines OPC UA Servers verwendet um Objekte und Variablen mit ihrer Typendefinition zu verknüpfen (vgl. [62] S. 56, [54] S. 66). |
| *Organizes* | Dieser Referenttyp (*Organizes*) wird verwendet um zwei Knoten hierarchisch miteinander zu verbinden. Für die Beziehung zwischen den beiden Knoten wird keine weitere Semantik definiert (vgl. [53] S. 27, [62] S. 55). |
| *HasComponent* | Der *HasComponent*-Referenztyp verbindet zwei Knoten hierarchisch miteinander. Die Semantik des Referenztyps definiert, dass der Zielknoten ein Teil des Quellknotens ist (part-of-Beziehung) (vgl. [53] S. 28, [62] S. 64). Beispiel: Objekte, Datenvariablen oder Methoden die Teil eines anderen Objekts sind, werden mit diesem Referenztyp verknüpft (vgl. [62] S. 64, [54] S. 73). |
| *HasProperty* | Referenzen des *HasProperty*-Referenztyps werden verwendet um Knoten definierte Eigenschaften (Properties) zuzuweisen (vgl. [62] S. 55, [54] S. 73). Beispiel: Im Adressraum eines OPC UA Servers sind Sensorwerte in Variablenknoten abgebildet. Um die Einheit der Sensorwerte zu beschreiben, verweisen die Variablenknoten mit der HasProperty-Referenz auf Property-Variablenknoten. Property-Variablenknoten beschreiben die Werte von anderen Knoten. |

**Tabelle 2.3:** Basis-Referenztypen

**Basis-Datentypen**
Das Adressraummodell spezifiziert bereits eine Vielzahl vordefinierter Basis-Datentypen. Die vordefinierten Basis-Datentypen sind untergliedert in Standard-Datentypen und OPC UA spezifischen Datentypen (vgl. [54] S. 91, [53] S. 61 f.). Standard-Datentypen sind allgemeine und grundlegende Datentypen aus der Informatik wie beispielsweise Byte, Integer, Double, Float oder String. OPC UA spezifische Datentypen sind spezielle Datentypen für die Anwendung des OPC UA Standards wie zum Beispiel der NodeId-, NodeClass- oder DiagnosticInfo-Datentyp. OPC UA Server können in ihrem Adressraum vorhandene Basis-Datentypen erweitern oder eigene Datentypen definieren (vgl. [62] S. 32, [54] S. 68). Der OPC UA Server verwendet dazu Knoten der Datentyp-Knotenklasse.

#### 2.3.3.2 Informationsmodelle

Informationsmodelle erweitern das Adressraummodell um Referenz- und Datentypen sowie definierte Instanzen ([53] S. 82, [71] S. 58, [71] S. 3). Standardisierungsgremien und Hersteller können auf dieser Grundlage domänenspezifische Informationsmodelle erstellen (vgl. [55] S. 399, [54] S. 77). Dadurch entstehen maschinen- und herstellübergreifende Standards für die Strukturierung und Repräsentation von Objekten im Adressraum der OPC UA Server.

#### 2.3.3.3 Adressraum

Produktionssysteme und -prozesse werden im Adressraum eines OPC UA Server mit Knoten und Referenzen abgebildet. Die Implementierung des Adressraums erfolgt auf Basis des Adressraummodells und den spezifizierten Informationsmodellen (vgl. [61] S. 1, [53] S. 82). Im weiteren Teil des Abschnitts wird die Adressraummodellierung für ein einfaches Applikationsszenario beschrieben. Das Szenario besteht aus einer Workstation mit OPC UA Client, einem Mikrocontroller mit OPC UA Server und einem Förderband. Aufgabe des Mikrocontrollers ist die Steuerung des Förderbands. Das Förderband wird von einem Motor angetrieben und beinhaltet zwei Lichtschranken. Die Lichtschranken befinden sich am Anfang und Ende des Förderbands. Das Steuerungsprogramm des Mikrocontrollers hat zwei Modi: Im automatischen Modus wird die Ware transportiert, sobald die Lichtschranke am Anfang des Förderbands eine Unterbrechung signalisiert. Im manuellen Modus wird die Ware auf dem Förderband transportiert, indem ein OPC UA Client die implementierte Transport-Methode des Mikrocontrollers (OPC UA Server) aufruft. Das Förderband wird solange angetrieben bis die Ware die zweite Lichtschranke passiert. In Abbildung 2.17 ist das Applikationsszenario dargestellt.

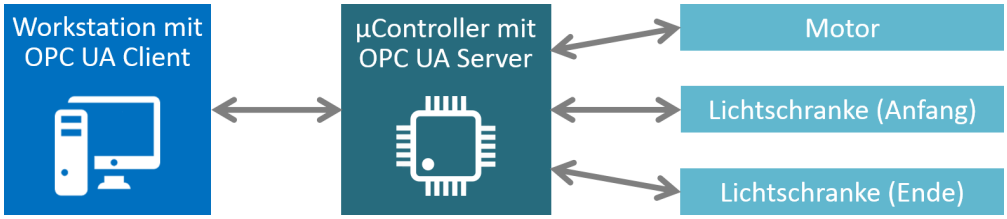


**Abbildung 2.17:** Applikationsszenario des Förderbands

Der modellierte Adressraum für das oben beschriebene Applikationsszenario ist in Abbildung 2.18 dargestellt. Der Adressraum ist auf dem OPC UA Server des Mikrocontrollers implementiert.

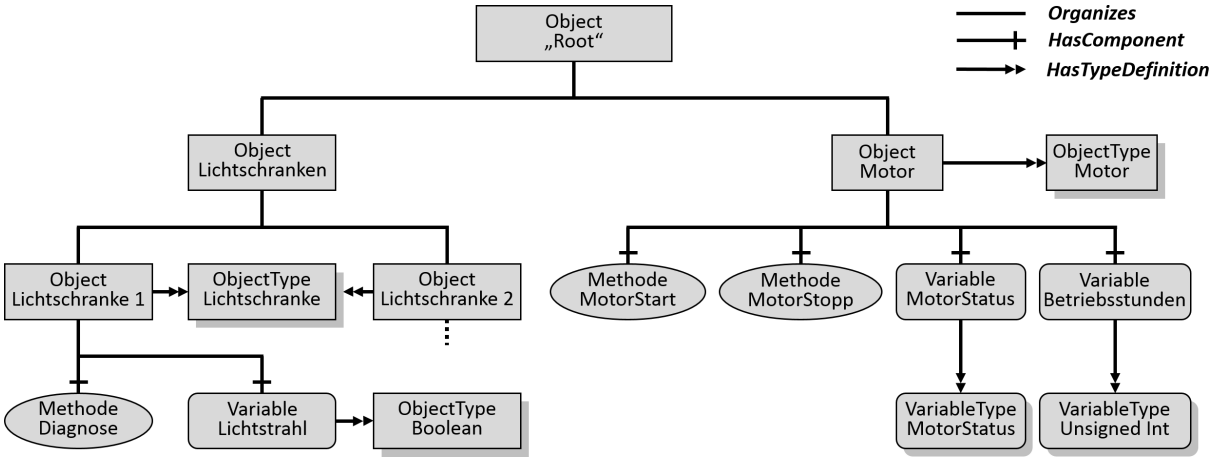


**Abbildung 2.18:** Adressraum für das Applikationsszenario

Einstiegspunkt in den Adressraum bildet der *Root*-Knoten. Der *Root*-Knoten organisiert im dargestellten Applikationsszenario das *Lichtschranken*- und *Motor*-Objekt. Im weiteren Teil wird lediglich der rechte Ast des Adressraums beschrieben (Motor). Das *Motor*-Objekt besteht aus den Komponenten: *MotorStart* (Methode), *MotorStopp* (Methode), *MotorStatus* (Variable) und *Betriebsstunden* (Variable). Als Typendefinition verwendet das *Motor*-Objekt den Objekttyp *Motor*. Die *MotorStart*-Methode ermöglicht einem OPC UA Client das Starten des Förderbands. Mit der *MotorStopp*-Methode kann das Förderband angehalten werden. In der Variable *MotorStatus* ist der aktuelle Status des Motors abgelegt. Diese ist eine Instanz vom Variablentypen *MotorStatus*. Der Variablentyp *MotorStatus* ist in diesem Anwendungsfall ein Aufzählungstyp mit drei Werten: AN, AUS, FEHLER. In der *Betriebsstunden*-Variable ist die Laufzeit des Motors gespeichert. Dieser Wert kann beispielsweise für das Wartungsintervall des Motors verwendet werden. Die *Betriebsstunden*-Variable verwendet den Variablentyp *Unsigned Int*.
Standardisierungsgremien und Hersteller könnten beispielsweise für Förderbandsysteme ein domänenspezifisches Informationsmodell definieren. Auf Basis dieses Informationsmodells kann anschließend der Adressraum für ein Förderbandsystem erstellt werden.

### 2.3.4 Services

In OPC UA erfolgt die Kommunikation nach dem Client-Server-Prinzip. Im Netzwerk stellt ein OPC UA Server auf der Applikationsebene eine Vielzahl von Services bereit. Möchte ein Client einen Service in Anspruch nehmen, sendet dieser einen Service Request. Ein Service Request kann mehrere Serviceanfragen beinhalten. Der OPC UA Server nimmt den Service Request entgegen und verarbeitet die beinhalteten Serviceanfragen. Nach der Verarbeitung der Serviceanfragen sendet der OPC UA Server einen Service Response an den Client. Der Service Response beinhaltet einen allgemeinen Statuscode für den Service Request und die Ergebnisse der Serviceanfragen.
OPC UA untergliedert die spezifizierten Services in zehn Service Sets. Ein Service Set beinhaltet mehrere zusammengehörende Services. Im weiteren Teil des Unterkapitels werden die Service Sets beschrieben. Die Informationen dieses Kapitels basieren auf den folgenden Quellen: [63] S. 3-102, [53] S. 125-189 und [54] S. 122-127 + S. 56-58.

#### 2.3.4.1 Discovery Service Set

Mit dem *Discovery Service Set* kann ein OPC UA Client registrierte OPC UA Server vom Discovery Server abfragen sowie Informationen zu den Endpunkten eines verfügbaren OPC UA Servers ermitteln. Der OPC UA Server verwendet das Service Set zur Registrierung beim Discovery Server. Das Discovery Service Set beinhaltet die folgenden Services:

| Services | Beschreibung |
|---|---|
| *FindServers* | Der *FindServers*-Service wird von Discovery Servern bereitgestellt. OPC UA Clients können mit dem *FindServers*-Service die verfügbaren bzw. registrierten OPC UA Server abfragen, die sich auf demselben Hostsystem wie der Discovery Server befinden. Dieser Service erfordert keine gesicherte Verbindung zwischen OPC UA Client und Discovery Server. |
| *FindServersOnNetwork* | Discovery Server stellen den *FindServersOnNetwork*-Service bereit. Mit dem *FindServersOnNetwork*-Service kann ein OPC UA Client alle OPC UA Server abfragen, die dem Discovery Server bekannt sind. Für die Serviceanfrage wird keine gesicherte Verbindung zwischen OPC UA Client und Discovery Server benötigt. |
| *GetEndpoints* | Der *GetEndpoints*-Service ist auf OPC UA Servern implementiert. OPC UA Clients verwendet den *GetEndpoints*-Service zur Abfrage der verfügbaren Endpunkte und den Endpunktbeschreibungen eines OPC UA Servers. Nach Aufruf des Service, erhält der OPC UA Client eine Endpunktbeschreibung für jeden verfügbaren Endpunkt. Die Endpunktbeschreibung beinhaltet die URL, das Serverzertifikat und die Sicherheitskonfiguration eines Endpunkts. Auf Basis dieser Informationen kann ein OPC Client eine Verbindung zu einem Endpunkt aufbauen. Die Serviceanfrage erfolgt auf dem Discovery Endpunkt und erfordert daher keine gesicherte Verbindung. |
| *RegisterServer* | Discovery Server stellen den *RegisterServer*-Service bereit. Damit ein OPC UA Server im Discovery Server als verfügbar gelistet wird, kann sich dieser beim Discovery Server mit dem *RegisterServer*-Service registrieren. Für den Aufruf des Service benötigt der OPC UA Server eine gesicherte Verbindung zum Discovery Server. Durch dieses Vorgehen wird sichergestellt, dass ein Discovery Server nur verifizierte OPC UA Server listet. |

**Tabelle 2.4:** Discovery Service Set

Hinweis: Kapitel 2.3.8 beschreibt die drei Discovery Server Arten sowie die Verwendung des Discovery Service Sets.

#### 2.3.4.2 SecureChannel Service Set

Das *SecureChannel Service Set* beinhaltet Services für den Aufbau und Abbau eines gesicherten Kommunikationskanals zwischen einem Client und Server. Aufgabe des gesicherten Kommunikationskanals ist die Sicherstellung von Vertraulichkeit und Integrität auf der Kommunikationsebene. Die Bereitstellung der Services erfolgt serverseitig.

| Services | Beschreibung |
|---|---|
| *OpenSecureChannel* | Ein OPC UA Client kann mit dem *OpenSecureChannel*-Service einen gesicherten Kommunikationskanal zu einem OPC UA Server erstellen. |
| *CloseSecureChannel* | Der *CloseSecureChannel*-Service wird von einem OPC UA Client verwendet, um den gesicherten Kommunikationskanal zu einem OPC UA Server zu beenden. |

**Tabelle 2.5:** SecureChannel Service Set

#### 2.3.4.3 Session Service Set

Für das Erstellen, Aktivieren, Beenden oder Abbrechen einer Session stellt das *Session Service Set* vier Services bereit. Eine Session verbindet zwei OPC UA Applikationen auf der Applikationsebene. Für das Erstellen einer Session ist ein Kommunikationskanal zwischen Client und Server erforderlich. Die Session baut auf dem Kommunikationskanal auf. Die Bereitstellung der Services erfolgt serverseitig.

| Services | Beschreibung |
|---|---|
| *CreateSession* | Ein OPC UA Client kann mit dem *CreateSession*-Service eine Session erstellen. Jede Session bekommt eine eindeutige SessionID und einen Authentifizierungstoken zugewiesen. Damit eine Session gültig wird, muss der OPC UA Client diese mit dem *ActivateSession*-Service aktivieren. |
| *ActivateSession* | Mit dem *ActivateSession*-Service kann ein OPC UA Client eine Session aktivieren oder ändern. Zum Aktivieren einer Session authentifiziert sich der Benutzer eines Clients beim Server. Die Authentifizierungsart hängt von den Sicherheitseinstellungen eines Servers ab. OPC UA definiert hierfür drei Authentifizierungsarten: Keine Authentifizierung, Authentifizierung mit Benutzername und Passwort sowie die Authentifizierung mittels X.509-Zertifikat. Der Benutzer einer Session kann mit dem *ActivateSession*-Service geändert werden. Mit dem Service ist es außerdem möglich, einer Session einen neuen Kommunikationskanal zuzuweisen. |
| *CloseSession* | Der *CloseSession*-Service wird von einem OPC UA Client verwendet, um eine Session zu beenden. Hinweis: Eine Session kann sowohl vom Client als auch vom Server beendet werden. |
| *Cancel* | Mit dem *Cancel*-Service kann ein OPC UA Client ausstehende Serviceanfragen einer Session löschen. |

**Tabelle 2.6:** Session Service Set

#### 2.3.4.4 NodeManagement Service Set

Mit dem *NodeManagement Service Set* kann ein OPC UA Client Knoten und Referenzen im Adressraum eines OPC UA Servers hinzufügen oder löschen. Zur Verwendung des *NodeManagement Service Sets* benötigt ein OPC UA Client eine aktive Session und die benötigten Zugriffsrechte im Adressraum.

| Services | Beschreibung |
|---|---|
| *AddNodes* | Der *AddNodes*-Service wird von einem OPC UA Client verwendet, um einen oder mehrere Knoten im Adressraums eines OPC UA Servers hinzuzufügen. |
| *AddReferences* | Ein OPC UA Client kann mit dem *AddReferences*-Service eine oder mehrere Referenzen zwischen Knoten im Adressraum des OPC UA Servers erstellen. |
| *DeleteNodes* | Mit dem *DeleteNodes*-Service kann ein OPC UA Client einen oder mehrere Knoten aus dem Adressraum eines OPC UA Servers löschen. |
| *DeleteReferences* | Zum Löschen einer oder mehrere Referenzen aus dem Adressraum eines OPC UA Servers verwendet der OPC UA Client den *DeleteReferences*-Service. |

**Tabelle 2.7:** NodeManagement Service Set

#### 2.3.4.5 View Service Set

Aufgabe des *View Service Sets* ist die Bereitstellung von Services, mit welchen OPC UA Clients durch den Adressraum eines OPC UA Servers navigieren können. Das *View Service Set* ist auf OPC UA Servern implementiert.

| Services | Beschreibung |
|---|---|
| *Browse* | Mit dem *Browse*-Service kann ein OPC UA Client die Referenzknoten zu einem oder mehreren definierten Startknoten aus dem Adressraum eines OPC UA Servers abfragen. Die Anzahl der zurückgelieferten Referenzknoten für eine Abfrage kann durch den Client begrenzt werden (vgl. [53] S. 141). Der Adressraum eines OPC UA Servers kann mit dem *Browse*-Service sowohl vorwärts als auch rückwärts durchlaufen werden. Filteroptionen ermöglichen das Filtern von verschiedenen Knoten- und Referenztypen. |
| *BrowseNext* | Im *Browse*-Service kann ein OPC UA Client die Anzahl der zurückgelieferten Referenzknoten für eine Abfrage begrenzen (vgl. [53] S. 141). Übersteigt eine Abfrage die Begrenzung, kann der OPC UA Client mit dem *BrowseNext*-Service die Abfrage fortsetzen. |

| | |
|---|---|
| *TranslateBrowse PathsToNodeIds* | Jeder Knoten hat im Adressraum einen eindeutigen Browse-Pfad und eine eindeutige Knoten-ID. Der *TranslateBrowsePathsToNodeIds*-Service übersetzt für einen OPC UA Client die Browse-Pfade in Knoten-IDs. |
| *RegisterNodes* | Mit dem *RegisterNodes*-Service kann ein OPC UA Client den Zugriff auf einen oder mehrere Knoten im Adressraum eines OPC UA Servers optimieren. Die Registrierung von Knoten empfiehlt sich bei mehrmaligem Lesen und Schreiben von Variablenwerten sowie für das zyklische Ausführen von Methoden. |
| *UnregisterNodes* | Ein OPC UA Client kann mit dem *UnregisterNodes*-Service den optimierten Zugriff auf einen oder mehrere Knoten entfernen. |

**Tabelle 2.8:** View Service Set

#### 2.3.4.6 Query Service Set

Im Adressraum eines OPC UA Server können beliebig viele Informationen und Daten abgebildet werden. In großen und komplexen Systemen kann es vorkommen, dass der Anwender den Überblick über den Adressraum verliert. Das *Query Service Set* definiert hierfür Services, mit welchen OPC UA Clients große und komplexe Adressräume nach relevanten Knoten filtern können. Vorteil des *Query Service Sets* ist, dass ein OPC UA Client keinerlei Informationen über den logischen Aufbau eines Adressraums benötigt, um an die für ihn relevanten Knoten zu gelangen.

| **Services** | **Beschreibung** |
|---|---|
| *QueryingView* | Mit Views kann ein OPC UA Server seinen Adressraum strukturieren. Eine View beinhaltet eine Teilmenge von Knoten und Referenzen des gesamten Adressraums (vgl. [63] S. 42). Beispiel: Ein OPC UA Server definiert eine View für die Wartung. Die View „Wartung“ beinhaltet alle relevanten Knoten und Referenzen des Adressraums, die ein Ingenieur für die Wartung des Systems benötigt. Mit dem *QueryingView*-Service kann ein OPC UA Client die Knoten für eine oder mehrere definierte Views abfragen. Die Anzahl der zurückgelieferten Knoten können vom Client begrenzt werden. |
| *QueryFirst* | Ein OPC UA Client kann mit dem *QueryFirst*-Service die für ihn relevanten Knoten aus einem großen und komplexen Adressraums eines OPC UA Servers filtern. Die Filterung der Knoten erfolgt mittels einer View, auf welcher ein Datentyp- und Content-Filter angewendet wird. Als Ergebnis erhält der Client eine Liste mit Knoten, die den definierten Filterkriterien entsprechen. Ein OPC UA Client kann die Anzahl der zurückgelieferten Knoten begrenzen. |

| | |
|---|---|
| *QueryNext* | Übersteigt eine *QueryingView*- oder *QueryFirst*-Serviceabfrage die Anzahl der maximal zurückgelieferten Knoten, kann ein OPC UA Client mit dem *QueryNext*-Service die Abfrage fortsetzen. |

**Tabelle 2.9:** Query Service Set

#### 2.3.4.7 Attribute Service Set

Im Adressraum des OPC UA Servers kann ein OPC UA Client mit dem *Attribute Service Set* die Attribute und Datenvariablen von einem oder mehreren Knoten auslesen und ändern.
Hinweis: Ein Knoten wird im Adressraum durch seine Knoten- und Daten-Attribute beschrieben. Anzahl und Art der Attribute eines Knotens hängen von der instanziierten Knotenklasse ab.

| **Services** | **Beschreibung** |
|---|---|
| *Read* | Knoten- und Daten-Attribute von einem oder mehreren Knoten kann ein OPC UA Client mit dem *Read-Service* auslesen. Für das Auslesen von Datenvariablen sind in der Regel Zugriffsrechte erforderlich. Die Zugriffsrechte für eine Datenvariable sind im AccessLevel- und UserAccessLevel-Attribut hinterlegt. Knotenattribute können ohne Zugriffsrechte ausgelesen werden. |
| *HistoryRead* | Mit dem *HistoryRead*-Service kann ein OPC UA Client Historie-Daten und Historie-Events von einem oder mehreren Knoten des Adressraum eines OPC UA Servers auslesen. Zum Auslesen der Historie-Daten und Historie-Events stellt der *HistoryRead*-Service zwei Optionen zur Auswahl: Mit der ersten Option können Clients die Rohdaten einer oder mehrerer Variablen für einen definierten Zeitbereich oder Zeitpunkt abfragen. Eine weitere Möglichkeit ist die Aggregation von Historie-Daten. Mit Aggregationsfunktionen können die Historie-Daten von einer oder mehrerer Variablen für einen definierten Zeitbereich oder Zeitpunkt zusammengefasst werden. Typische Aggregationsfunktionen sind beispielsweise Minimum, Maximum, Summe oder Durchschnitt. |
| *Write* | Ein OPC UA Client kann mit dem *Write*-Service die Attribute oder Datenvariablen von einem oder mehreren Knoten ändern. Das Ändern von Datenvariablen erfordert in der Regel Zugriffsrechte. Standardmäßig kann ein OPC UA Client die Knotenattribute eines Knotens nicht ändern. Mittels Konfigurationen am OPC UA Server ist dies aber möglich. |

| | |
|---|---|
| *HistoryUpdate* | Der *HistoryUpdate*-Service ermöglicht einem Client das Einfügen, Ersetzen und Aktualisieren von Historie Daten und Events für einen oder mehrere Knoten im Adressraums eines OPC UA Servers. Historie-Daten und Historie-Events werden üblicherweise in Datenbanken gespeichert. Zum Ausführen des *HistoryUpdate*-Services, benötigt ein Client Zugriffsrechte. |

**Tabelle 2.10:** Attribute Service Set

#### 2.3.4.8 Method Service Set

Das *Method Service Set* beinhaltet lediglich den *Call*-Service.

| Services | Beschreibung |
|---|---|
| *Call* | OPC UA Clients verwenden den *Call*-Service um bereitgestellte Methoden im Adressraum eines OPC UA Servers aufzurufen bzw. ausführen. In OPC UA sind Methoden kleine zustandslose Programme, die eine definierte Funktionalität bereitstellen. Beim Aufruf des *Call*-Service übergibt der OPC UA Client die Eingabeargumente für eine Methode. In den Ausgabeargumenten erhält der OPC UA Client das Ergebnis der aufgerufenen Methode. |

**Tabelle 2.11:** Method Service Set

#### 2.3.4.9 Subscription Service Set

Zur Erstellung einer Subskription ist eine Session zwischen OPC UA Client und OPC UA Server erforderlich. Mit dem *CreateSubscription*-Service können OPC UA Clients eine Subskription auf einem OPC UA Server erstellen. Bei der Erstellung einer Subskription definiert der OPC UA Client das Benachrichtigungsintervall und die Lebenszeit einer Subskription (vgl. [OPC4] S. 69, [OPC1] S. 159, [OPC2] S. 123). Das Benachrichtigungsintervall ist der zeitliche Abstand, in welchem der OPC UA Server einem abonnierten Client Benachrichtigungen sendet. Die Benachrichtigungen beinhalten Datenänderungen und Events von überwachten Elementen (*MonitoredItems*). Ändern sich innerhalb eines Benachrichtigungsintervalls keine zu überwachenden Elemente, sendet der OPC UA Server eine leere Benachrichtigung an den OPC UA Client der Subskription (vgl. [OPC1] S. 161). Die gesendeten Benachrichtigungen sind mit Sequenznummern protokolliert. Empfangene Benachrichtigungen muss der OPC UA Client mit dem *Publish*-Service bestätigen (vgl. [OPC4] S. 70). Ein OPC UA Server speichert gesendete Benachrichtigungen so lange, bis der OPC Client sie bestätigt ([OPC1] S. 162). Nicht bestätigte Benachrichtigungen kann ein OPC UA Client mit dem *Republish*-Service abfragen.

Der OPC UA Client kann einer Subskription mehrere *MonitoredItems* zuweisen (vgl. [OPC1] S. 167). Aufgabe des OPC UA Servers ist die Überwachung aller *MonitoredItems* einer Subskription. Jedes *MonitoredItem* hat die folgenden Attribute: Abtastintervall, Monitoring-Modus, Filter und Queue (vgl. [OPC4] S. 57 f., [OPC1] S. S. 158 f.). Das Abtastintervall ist der zeitliche Abstand, in welchem der OPC UA Server den aktuellen Wert eines überwachten Elements abtastet. Mit dem Monitoring-Modus kann ein OPC UA Client Benachrichtigungen für ein überwachtes Element (*MonitoredItem*) aktivieren oder deaktivieren. Die abgetasteten Werte eines überwachten Elementes können mittels Filtern evaluiert werden. Beispielsweise sollen zu einem überwachten Element nur Benachrichtigungen erfolgen, wenn der definierte Schwellenwert überschritten wird. In einer Queue speichert das *MonitoredItem* die Benachrichtigungen für die Subskription.

Das *Subscription Service Set* beinhaltet die folgenden Services:

| **Services** | **Beschreibung** |
|---|---|
| *CreateSubscription* | Mit dem *CreateSubscription*-Service können OPC UA Clients eine Subskription auf einem OPC UA Server erstellen. Zur Erstellung einer Subskription ist eine Session zwischen OPC UA Client und OPC UA Server erforderlich. Beendet ein OPC UA Client eine Session, muss er alle Subskriptions dieser Session entfernen. |
| *ModifySubscription* | OPC UA Clients verwenden den *ModifySubscription*-Service um Einstellungen einer Subskription zu ändern. Änderbar sind beispielsweise das Benachrichtigungsintervall, die Lebenszeit oder Priorität einer Subskription. |
| *SetPublishingMode* | Eine Subskription kann aktiv oder inaktiv sein. Mit dem *SetPublishingMode*-Service kann ein OPC UA Client eine Subskription aktivieren oder deaktivieren. |
| *Publish* | Der OPC UA Client verwendet den *Publish*-Service für folgende Zwecke:<br>• Empfangene Benachrichtigungen bestätigen.<br>• Keep-Alive Anfragen beantworten. |
| *Republish* | Ein OPC UA Client kann mit dem *Republish*-Service nicht bestätigte Benachrichtigungen einer Subskription erneut abfragen.<br>Hinweis: OPC UA Server speichern Benachrichtigungen solange, bis ein OPC UA Client den Empfang mit dem *Publish*-Service bestätigt. |
| *DeleteSubscriptions* | OPC UA Clients verwenden den *DeleteSubscriptions*Service um eine oder mehrere Subskriptions auf einem OPC UA Server zu entfernen. |

**Tabelle 2.12:** Subscription Service Set

#### 2.3.4.10 MonitoredItem Service Set

Das *MonitoredItem Service Set* beinhaltet die folgenden Services:

| Services | Beschreibung |
|---|---|
| *CreateMonitoredItems* | Mit dem *CreateMonitoredItems*-Service kann ein OPC UA Client überwachte Elemente (*MonitoredItems*) erstellen und einer Subskription zuweisen. |
| *ModifyMonitoredItems* | OPC UA Clients verwenden den *ModifyMonitoredItems-Service* um Parameter von einem oder mehreren überwachten Elementen (*MonitoredItems*) zu ändern. Für ein überwachtes Element können Abtastintervall, Filter, Queue-Parameter und Monitoring-Modus geändert werden. |
| *SetMonitoringMode* | Der Monitoring-Modus eines überwachten Elements kann mit dem *SetMonitoringMode*-Service geändert werden. OPC UA definiert drei Monitoring-Modi: Disabled, Sampling und Reporting. |
| *SetTriggering* | Soll ein überwachtes Element nur in Verbindung mit einem anderen überwachten Element auslösen, kann der OPC UA Client dies mit dem *SetTriggering*-Service konfigurieren. |
| *DeleteMonitoredItems* | Mit dem *DeleteMonitoredItems*-Service kann ein OPC UA Client überwachte Elemente (*MonitoredItems*) aus einer Subskription löschen. |

**Tabelle 2.13:** MonitoredItem Service Set

### 2.3.5 Transport und Datenkodierung

Transportprotokolle und Datenkodierungen sind für die Übertragung der Servicenachrichten zwischen Client und Server erforderlich. OPC UA stellt für die Übertragung der Servicenachrichten zwei Optionen bereit: OPC UA TCP Binary oder Web-Services. Im weiteren Teil dieses Unterkapitels werden diese zwei Optionen vorgestellt. Quellen: [53] S. 191-201, [54] S. 44-49 + S. 101-103, [65] S. 5-28 + S. 40-49 und [58].

#### 2.3.5.1 OPC UA TCP Binary

OPC UA TCP Binary verwendet auf der Transportebene das UA TCP Protokoll. Auf der Serialisierungsebene wird die binäre Datenkodierung - UA Binary - eingesetzt.
UA TCP ist ein schnelles und einfaches Transportprotokoll, das auf TCP aufbaut (vgl. [65] S. 40). Die OPC Foundation entwickelte das UA TCP Protokoll aus folgenden Gründen (vgl. [53] S. 198 u. [65] S. 40): Ein OPC UA Server kann mit dem UA TCP Protokoll mehrere Endpunkte auf einer IP und einem Port parallel betreiben. Tritt auf der Transportebene ein Fehler auf, kann UA TCP darauf reagieren und anschließend die Verbindungen zwischen OPC UA Client und Endpunkt wiederherstellen. Im UA TCP Protokoll können Client und Server auf der Applikationsebene die Größe des Empfangs- und Sendepuffers konfigurieren.
Das UA TCP Datenpaket setzt sich zusammen aus dem Message Header und Message Body. Im Message Header ist der Typ und die Länge einer Nachricht beschrieben. Der Message Header besteht aus 8 Byte. Im Message Body befinden sich die kodierten Servicenachrichten. Die kodierten Servicenachrichten im Message Body können aus Sicherheitsgründen verschlüsselt sein. Das UA TCP Protokoll besteht aus den drei Nachrichtentypen *HelloMessage*, *AcknowledgeMessage* und *ErrorMessage* (vgl. [65] S. 41 f. + S. 44 f. u. [53] S. 199). Zum Aufbau einer Socket Verbindung zwischen Client und Endpunkt eines OPC UA Servers verwendet der Client die *HelloMessage*. In der *HelloMessage* übermittelt der Client seine Protokollversion, die Größe seines Empfangs- und Sendepuffers und die Endpunkt-URL, mit welcher er eine Socket-Verbindung aufbauen möchte. Der OPC UA Server beantwortet die *HelloMessage* mit einer sogenannten *AcknowledgeMessage*. In der *AcknowledgeMessage* übermittelt der Server dem Client seine Protokollversion und die Größe seines Empfangs- und Sendepuffers. Der dritte Nachrichtentyp ist die *ErrorMessage*. Bemerkt der Server einen Fehler, sendet dieser an den Client eine *ErrorMessage* und beendet den Socket. Der Client kann anschließend eine neue Socket-Verbindung zum Server herstellen. Bemerkt der Client einen Fehler, beendet dieser lediglich den Socket und sendet keine *ErrorMessage*.
Mittels der binären Datenkodierung - UA Binary - erfolgt die Serialisierung der Servicenachrichten. UA Binary kodiert die Daten einer Servicenachricht sequentiell in eine Bit-Folge. Es werden hierbei keine Informationen über die Feldnamen und Datentypen einer Servicenachricht hinzugefügt. Informationen über den Aufbau der Servicenachrichten und den Datentypen sind in der OPC UA Spezifikation definiert. In Tabelle 2.14 ist ein Beispiel für die binäre Datenkodierung der „HelloWorld“ Zeichenkette (String) dargestellt. Für die Zeichenkodierung wurde der ASCII-Standard verwendet:

| H | e | l | l | o | W | o | r | l | d |
|---|---|---|---|---|---|---|---|---|---|
| *0x48* | *0x65* | *0x6C* | *0x6C* | *0x6F* | *0x57* | *0x6F* | *0x72* | *0x6C* | *0x64* |

**Tabelle 2.14:** Binäre Datenkodierung der „HelloWorld“ Zeichenkette

Durch das oben beschriebene Verfahren können die Servicenachrichten sehr schnell, einfach und ressourcenschonend kodiert und dekodiert werden (vgl. [54] S. 103, [53] S. 193 u. S. 201). Es entsteht hierbei kein Overhead. OPC UA TCP Binary wurde primär für Systeme bzw. Geräte auf der Steuerungs- und Feldebene entwickelt (Automatisierungspyramide).
Nachteilig ist, dass OPC UA TCP Binary keine etablierten Standardtechnologien verwendet (vgl. [54] S. 103, [53] S. 201). Das Protokoll muss daher erst auf dem Client und Server implementiert werden. Für die Interpretation der Daten benötigen die Anwendungen zudem Wissen bzw. Informationen über den Aufbau der Servicenachrichten und den verwendeten Datentypen.

#### 2.3.5.2 Web-Services

Web-Services verwenden als Transportprotokoll HTTP oder HTTPS. Die Datenkodierung erfolgt auf der Serialisierungsebene mittels XML.
In Kapitel 2.2.3 ist das HTTP-Protokoll beschrieben. HTTPS ist die verschlüsselte Variante von HTTP. Für die Verschlüsselung wird das TLS (Transport Layer Security) Protokoll verwendet.
XML (Extensible Markup Language) ist eine erweiterbare Auszeichnungssprache zur Strukturierung und Beschreibung von Daten. Im Fachbuch „Einstieg in XML“ [70] von Helmut Vonhoegen sind die Grundlagen zum XML-Standard detailliert beschrieben.
Auf der Unternehmens-, Produktions- und Prozessleitebene werden hauptsächlich Web-Services eingesetzt (Automatisierungspyramide) (vgl. [53] S. 201). Folgende Vorteile ergeben sich durch die Verwendung von Web-Services (HTTP/HTTPS und XML): HTTP/HTTPS und XML sind etablierte Webtechnologie, die bereits standardmäßig auf vielen Systemen implementiert sind (vgl. [54] S. 101 f., [53] S. 199). Durch die Verwendung von XML beinhalten die Servicenachrichten Informationen über ihren Aufbau und die verwendeten Datentypen. Die Applikationen können dadurch die Servicenachrichten ohne vorheriges Wissen hinsichtlich Aufbau und verwendeten Datentypen interpretieren (vgl. [54] S.103). Im Vergleich zu OPC UA TCP Binary haben die Web-Services folgende Nachteile: XML erzeugt durch die Meta-Informationen einen enormen Overhead (vgl. [54] S. 103). Dies hat zur Folge, dass mehr Daten kodiert und übertragen werden müssen (Ressourcenverbrauch). Viele Systeme bzw. Geräte auf der Steuerungs- und Feldebene sind enorm ressourcenbeschränkt und besitzen daher nicht genug Leistung für die Implementierung von HTTP/HTTPS und XML.

Hinweis:
OPC UA Server können für unterschiedliche Aufgaben mehrere Endpunkte bereitstellen. Dadurch kann ein OPC UA Server parallel Endpunkte für OPC UA TCP Binary und Web-Services anbieten.

### 2.3.6 Sicherheitsmodell

Eine wichtige Komponente von OPC UA ist das Sicherheitsmodell. Aufgabe des Sicherheitsmodells ist die Authentifizierung auf der Applikations- und Kommunikationsebene sowie die Sicherstellung von Vertraulichkeit und Integrität auf der Kommunikationsebene. Das Sicherheitsmodell ist in Abbildung 2.19 dargestellt. Bestandteile des Sicherheitsmodells sind der gesicherte Kommunikationskanal und die Session, welche nachfolgend vorgestellt werden.

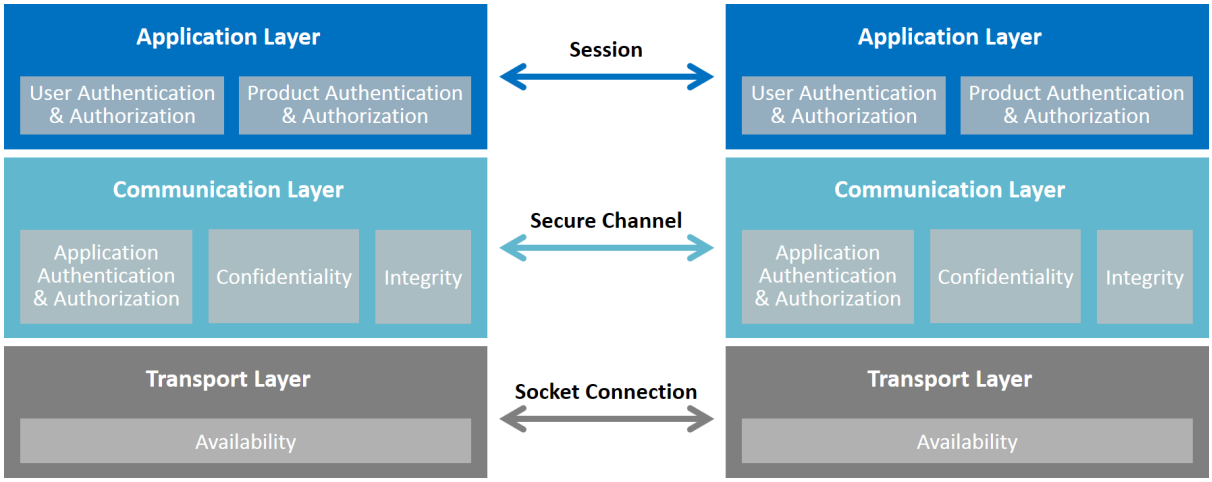


**Abbildung 2.19:** OPC UA Sicherheitsmodell (Quelle: [53] S. 211)

#### 2.3.6.1 Gesicherter Kommunikationskanal (Secure Channel)

Der Kommunikationskanal verbindet zwei Applikationen auf der Kommunikationsebene miteinander. Zur Sicherung des Kommunikationskanals stellt OPC UA die Sicherheitsmodi „None“, „Sign“ und „SignAndEncrypt“ zur Verfügung (vgl. [53] S. 221, [54] S. 111, [57] S. 21). Im Sicherheitsmodus „None“ wird der Kommunikationskanal nicht gesichert. Diese Einstellung ist nicht zu empfehlen und sollte nur in lokalen und abgeschotteten Netzwerken verwendet werden. Zur Sicherung der Integrität auf der Kommunikationsebene stellt OPC UA den Sicherheitsmodus „Sign“ bereit. In diesem Modus werden die Nachrichten eines Kommunikationskanals signiert. Für die Signierung wird das Zertifikat des OPC UA Clients verwendet. Die höchste Sicherheit garantiert der „SignAndEncrypt“ Modus. Hierbei werden alle Nachrichten eines Kommunikationskanals signiert und verschlüsselt. Der verwendete Sicherheitsmodus wird vom Endpunkt des OPC UA Servers definiert (vgl. [57] S. 20 f., [54] S. 111, [53] S. 212]).

Für die Erstellung eines gesicherten Kommunikationskanals sind vier Schritte erforderlich, welche im Sequenzdiagramm in Abbildung 2.20 dargestellt sind.

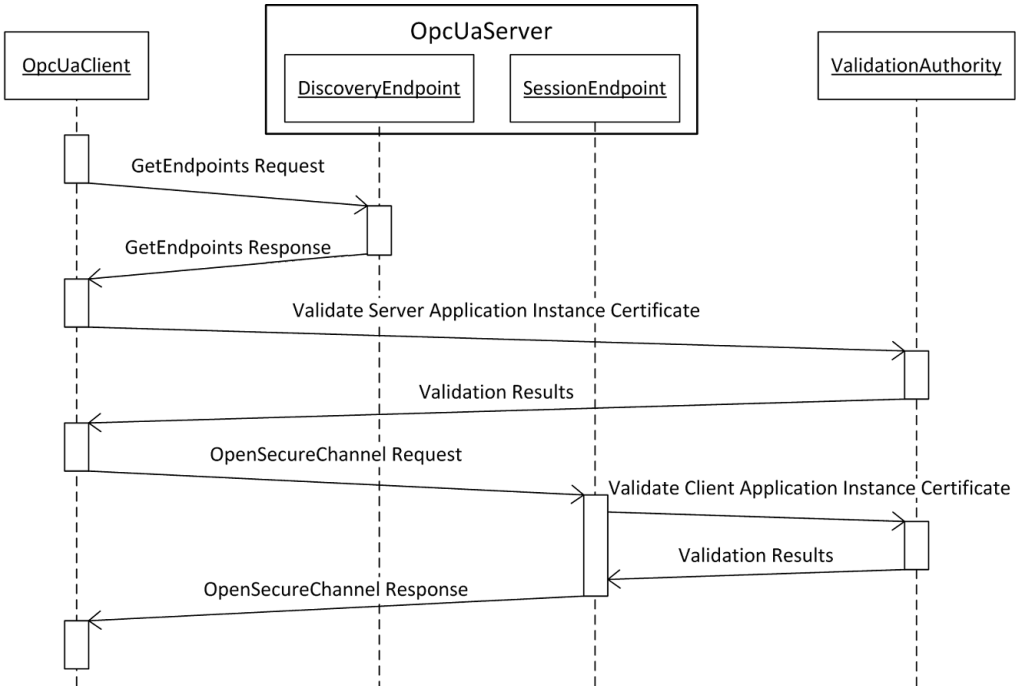


**Abbildung 2.20:** Gesicherten OPC UA Kommunikationskanal erstellen (Quelle: [53] S. 214)

Im ersten Schritt fragt der OPC UA Client mit dem GetEndpoints-Service die verfügbaren Endpunkte und Endpunktbeschreibungen eines OPC UA Servers ab. Die Endpunktbeschreibung beinhaltet die URL, das Server-Applikationszertifikat und die Sicherheitskonfiguration eines Endpunkts (vgl. [56] S. 13, [53] S. 213, [54] S. 115 f.). Auf Basis dieser Informationen kann der OPC Client anschließend eine Verbindung zum Endpunkt aufbauen. Für die Abfrage auf dem Discovery-Endpunkt ist lediglich eine Socket-Verbindung erforderlich.
Im zweiten Schritt erfolgt die Authentifizierung und Autorisierung der Applikation eines Endpunkts (vgl. [53] S. 213). Dazu sendet der OPC UA Client das empfangene Server-Applikationszertifikat des Endpunkts an die Validation-Authority (VA)[2]. Die Validation-Authority überprüft anschließend die Echtheit des Server Applikationszertifikats (vgl. [59] S. 4, [54] S. 41 u. S. 105 f., [53] S. 213).
Im dritten Schritt sendet der OPC UA Client einen OpenSecureChannel-Request an den gewünschten Endpunkt des OPC UA Servers. Mit dieser Anfrage teilt der OPC UA Client dem OPC UA Server mit, dass er einen gesicherten Kommunikationskanal zum Endpunkt aufbauen möchte (vgl. [54] S. 105, [53] S. 213, [57] S. 21, [56] S. 18f.). Die Anfrage beinhaltet unter anderem das Client-Applikationszertifikat und einen Client-Public-Key. Der Client-Public-Key wird je nach Sicherheitsmodus des Endpunkts zur Verschlüsselung des Kommunikationskanals verwendet.
Im vierten Schritt erfolgt die Authentifizierung und Autorisierung der Applikation des OPC

[2] Die Validation-Authority ist ein zentraler Server im Netzwerk, welcher Zertifikate ausstellt und die ausgestellten Zertifikate auf Echtheit prüft. Die Adresse der Validation-Authority muss den Netzwerkteilnehmern bekannt sein.

UA Clients. Der OPC UA Server sendet dazu das empfangene Client-Applikationszertifikat zur Überprüfung der Echtheit an die Validation-Authority (VA) (vgl. [53] S. 213, [54] S. 41). Bestätigt die Validation-Authority die Echtheit des Client-Applikationszertifikats, sendet der OPC UA Server eine positive OpenSecureChannel-Response Nachricht an den OPC UA Client. Anschließend kann der erstellte Kommunikationskanal verwendet werden. Der OPC UA Client kann den ersten und zweiten Schritt überspringen, wenn ihm der Endpunkt des OPC UA Servers bekannt ist. Im dritten und vierten Schritt wird der gesicherte Kommunikationskanal erstellt. Benötigt der OPC UA Client den gesicherten Kommunikationskanal nicht mehr, muss er diesen mit dem CloseSecureChannel-Service beenden.

#### 2.3.6.2 Session

Die Session ist eine anwender- oder produktspezifische Verbindung zwischen einem OPC UA Client und dem Endpunkt eines OPC UA Servers (vgl. [57] S. 16). Aufgabe der Session ist die Authentifizierung und Autorisierung von einem Benutzer oder Produkt auf der Applikationsebene (vgl. [53] S. 211, [54] S. 107). Für die Authentifizierung und Autorisierung des Benutzers oder Produkts spezifiziert OPC UA die Verfahren „Anonymous“, „UsernameAndPassword“ und „Certificate“ (vgl. [63] S. 22, [53] S. 220, [54] S. 107, [58] S. 136). Das verwendete Verfahren wird vom Session Endpunkt des OPC UA Servers vorgegeben. Beim „Anonymous“-Verfahren erfolgt keine Authentifizierung und Autorisierung des Benutzers oder Produkts. Das „UsernameAndPassword“-Verfahren verwendet für Authentifizierung und Autorisierung des Benutzers oder Produkts ein Benutzername und Passwort. Eine weitere Möglichkeit ist das „Certificate“-Verfahren. Bei diesem Verfahren wird der Benutzer oder das Produkt mit einem Zertifikat authentifiziert und autorisiert.

Die Session wird auf einem gesicherten Kommunikationskanal erstellt (vgl. [54] S. 41, [53] S. 215). Zur Erstellung und Aktivierung der Session sind fünf Schritte notwendig, die im Sequenzdiagramm in Abbildung 2.21 dargestellt sind.
Im ersten Schritt erstellt der OPC UA Client für Benutzer oder Produkte eine Session auf dem gewünschten Endpunkt des OPC UA Servers. Dazu verwendet der OPC UA Client den CreateSession-Service des OPC UA Servers. Als Ergebnis erhält der OPC UA Client die SessionID und den Authentifizierungstoken der erstellten Session sowie das Server-Softwarezertifikat des OPC UA Servers (vgl. [53] S. 215, [54] S. 119, [63] S. 22). Der Authentifizierungstoken beinhaltet Informationen über das gewählte Authentifizierungs- und Autorisierungsverfahren des Endpunkts.
Im zweiten Schritt erfolgt die Validierung des Server-Softwarezertifikats [3]. Der OPC UA Client sendet dazu das empfangene Server-Softwarezertifikat an die Validation Authority (VA) (vgl. [53] S. 215, [54] S. 120).

[3] Eine Applikation kann aus mehreren Softwareteilen bestehen. Für die Authentifizierung einer Software gibt es in OPC UA das Softwarezertifikat.

Nach der Validierung des Server-Softwarezertifikats wird im dritten bis fünften Schritt die Session aktiviert. Zur Aktivierung der Session sendet der OPC UA Client einen ActivateSession-Request an den OPC UA Server. Der ActivateSession-Request beinhaltet das Client-Softwarezertifikat und den UserIdentityToken (vgl. [63] S. 26, [53] S. 121).
Im nachfolgenden Schritt validiert der OPC UA Server mit Hilfe der Validation Authority (VA) die Echtheit des empfangenen Client-Softwarezertifikates.
Nach der Validierung des Client-Softwarezertifikates überprüft der OPC UA Server im letzten Schritt die Authentifizierungsdaten vom Benutzer oder Produkt. Die Authentifizierungsdaten befinden sich im UserIdentityToken. Zur Überprüfung sendet der OPC UA Server die Authentifizierungsdaten an die Authentication Authority (AA) (vgl. [53] S. 215). Die Authentication Authority ist ein zentraler Server im Netzwerk, welcher Authentifizierungsdaten überprüft. Nach Überprüfung der Authentifizierungsdaten, sendet der OPC UA Server einen ActivateSession-Response an den OPC UA Client. Tritt in den oben beschriebenen Schritten kein Fehler auf, wurde die Session erfolgreich erstellt.

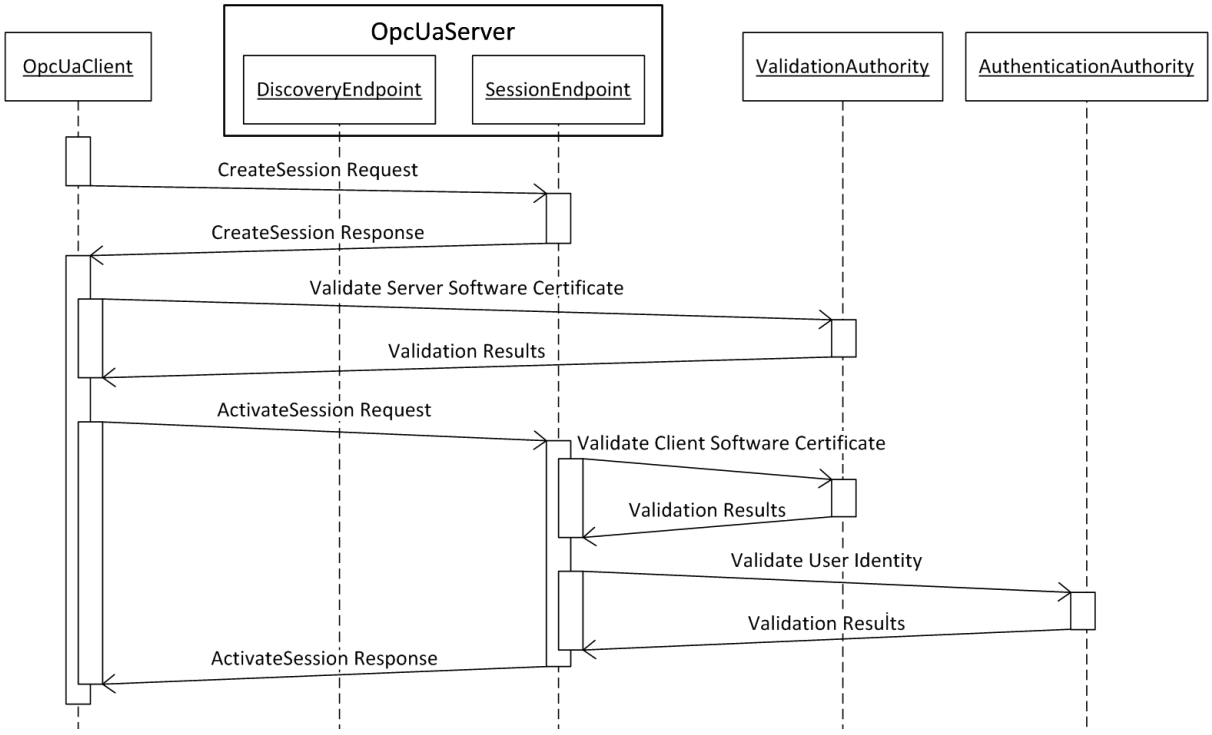


**Abbildung 2.21:** OPC UA Session erstellen (Quelle: [53] S. 215)

#### 2.3.6.3 Anwendung des Sicherheitsmodells

Die beschriebenen Mechanismen des Sicherheitsmodells sind skalierbar. Je nach Anwendungsfall kann der OPC UA Server für einen Endpunkt das benötigte Sicherheitslevel konfigurieren (vgl. [57] S. 21, [59] S. 4-5, [54] S. 107). Der OPC UA Server kann mehrere Endpunkte mit unterschiedlichen Sicherheitskonfigurationen zur Verfügung stellen (vgl. [54] S. 107).

Für die Konfiguration des gesicherten Kommunikationskanals stehen die beschriebenen Sicherheitsmodi „None“, „Sign“ und „SignAndEncrypt“ zur Verfügung (vgl. [53] S. 213, [54] S. 111). Der Austausch und die Validierung des Client- und Server-Applikationszertifikats ist optional und hängt von den Sicherheitskonfigurationen des Endpunkts ab (vgl. [54] S. 111, [57] S. 20 f.). OPC UA spezifiziert für Authentifizierung und Autorisierung des Benutzers oder Produkts die Verfahren „Anonymous“, „UsernameAndPassword“ und „Certificate“. Der Endpunkt des OPC UA Servers wählt ein Verfahren aus. Der Austausch und die Validierung des Client- und Server-Softwarezertifikats ist optional und hängt von den Sicherheitskonfigurationen des Endpunkts ab.

## 2.3.7 Systemarchitektur

Die OPC UA Systemarchitektur beschreibt die strukturelle Anordnung der OPC UA Clients und Server in einem OPC UA System. Für die Strukturierung der OPC UA Clients und Server in einem System gibt es vier grundlegende Architekturmuster (Architectural Pattern), die nachfolgend vorgestellt werden.

### 2.3.7.1 Client-Server Pattern

Das am häufigsten eingesetzte Architekturmuster ist das Client-Server Pattern (vgl. [53] S. 265). In Abbildung 2.22 ist das Client-Server Pattern dargestellt. Es besteht aus einem OPC UA Client und OPC UA Server. Um eine Aufgabe oder ein Problem zu lösen, nimmt der OPC UA Client einen bereitgestellten Service des OPC UA Servers in Anspruch. Je nach Sicherheitskonfigurationen des OPC UA Servers ist eine Authentifizierung und Autorisierung zwischen den Entitäten erforderlich.

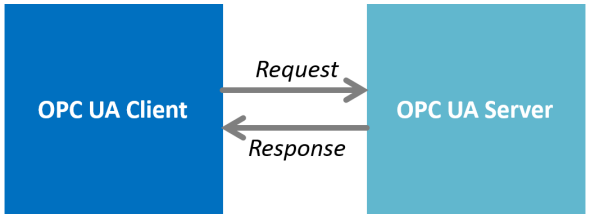


**Abbildung 2.22:** Client-Server Pattern (Quelle: [53] S. 266)

### 2.3.7.2 Server-Server Pattern

Das Server-Server Pattern ermöglicht die Kommunikation zwischen zwei OPC UA Servern. In Abbildung 2.23 ist das Server-Server Pattern dargestellt. Das Server-Server Pattern besteht aus zwei OPC UA Servern, die jeweils einen OPC UA Client einbetten. Mit dem eingebetteten OPC UA Client kann der OPC UA Server eine Verbindung zu einem anderen OPC UA Server

herstellen und Services in Anspruch nehmen. Ein OPC UA Server ohne eingebettetem OPC UA Client kann keine Verbindung zu einem anderen OPC UA Server herstellen (vgl. [61] S. 14 f., [53] S. 268)!

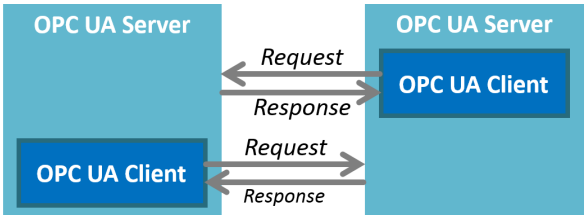


**Abbildung 2.23:** Server-Server Pattern (Quelle: [53] S. 267)

#### 2.3.7.3 Chained-Server Pattern

Das Chained-Server Pattern wird als Gateway zwischen einem OPC UA Client (*Client 1*) und OPC UA Server (*Server 2*) verwendet (vgl. [53] S. 266, [61] S. 14 f.). Eine direkte Kommunikation zwischen dem OPC UA Client (*Client 1*) und OPC UA Server (*Server 2*) ist aus verschiedenen Gründen, wie beispielsweise einer Firewall oder unterschiedlicher Transportprotokolle, nicht möglich. Um dieses Problem zu lösen, wird ein OPC UA Server (*Server 1*) mit eingebettetem OPC UA Client (*Client 2*) verwendet. Dieser leitet den Service-Request und Service-Response zwischen den zwei Entitäten unverändert weiter. In Abbildung 2.24 ist das Chained-Server Pattern dargestellt.

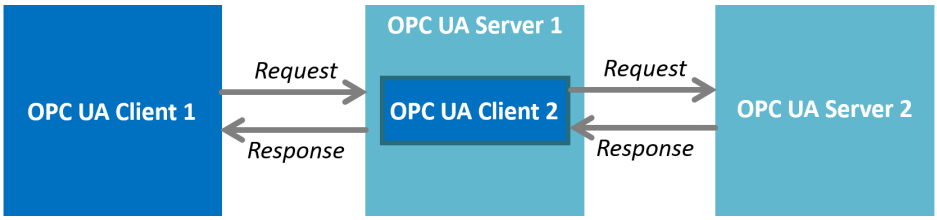


**Abbildung 2.24:** Chained-Server Pattern (Quelle: [53] S. 267)

#### 2.3.7.4 Aggregierende Server Pattern

Das aggregierende Server Pattern ermöglicht einem OPC UA Client den Zugriff auf verarbeitete bzw. aggregierte Daten mehrerer OPC UA Server.
Durch den eingebetteten OPC UA Client (*Client 1*) hat der OPC UA Server (*Server 1*) Zugriff auf mehrere OPC UA Server (im Beispiel die OPC UA *Server 2* bis *4*). Der OPC UA Server (*Server 1*) kann somit die Daten mehrerer Server abfragen, verarbeiten, aggregieren und den anderen OPC UA Clients im Netzwerk zur Verfügung stellen (vgl. [53] S. 268). In Abbildung 2.25 ist die Funktionsweise des Patterns dargestellt.

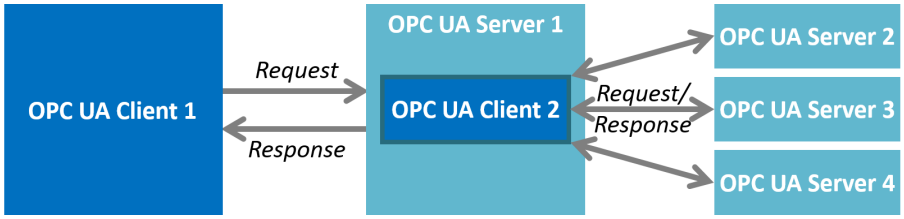


**Abbildung 2.25:** Aggregierende Server Pattern (Quelle: [53] S. 268)

Zur Erstellung des Unterkapitels wurde die folgenden Quellen verwendet: [53] S. 265-269, [61] S. 11-15

## 2.3.8 OPC UA Discovery

Mit dem *Discovery Service Set* kann ein OPC UA Client registrierte OPC UA Server vom Discovery Server abfragen sowie Informationen über die Endpunkte eines verfügbaren OPC UA Servers ermitteln. Der OPC UA Server verwendet das Service Set zur Registrierung beim Discovery Server. In Kapitel 2.3.4.1 ist das *Discovery Service Set* detailliert beschrieben. Die OPC Unified Architecture Spezifikation definiert für verschiedene Anwendungsfälle drei Arten von Discovery Servern: Lokaler Discovery Server (LDS), lokaler Discovery Server mit Multicast-Erweiterung (LDS-ME) und globaler Discovery Server (GDS). Im weiteren Teil des Unterkapitels werden diese beschrieben.

### 2.3.8.1 Lokaler Discovery Server (LDS)

Der lokale Discovery Server listet für ein Hostsystem die registrierten OPC UA Server. Damit ein OPC UA Server beim lokalen Discovery Server als verfügbar gelistet wird, muss sich dieser mit dem RegisterServer-Service registrieren (vgl. [66] S. 5, [54] S. 57). Für die Registrierung ist ein gesicherter Kommunikationskanal zwischen dem OPC UA Server und lokalen Discovery Server erforderlich (vgl. [67], [53] S. 134, [54] S. 57]). Die verfügbaren bzw. registrierten OPC UA Server eines lokalen Discovery Server (LDS) kann der OPC UA Client mit dem FindServers-Service abfragen.
Üblicherweise betreibt jeder Host einen lokalen Discovery Server. Die Adresse des lokalen Discovery Servers setzt sich zusammen aus dem Hostname und dem Port 4840 (vgl. [67], [68], [54] S. 57). Bespiel für die Adressierung des lokalen Discovery Servers:

*opc.tcp://hostname:4840*

Ein OPC UA Client kann den lokalen Discovery Server verwenden, um die registrierten OPC UA Server eines Hosts abzufragen. Für die Abfrage muss dem OPC UA Client der Host bzw. die Adresse des Hosts bekannt sein.

In Abbildung 2.26 ist das Zusammenspiel zwischen lokalem Discovery Server (LDS), OPC UA Client und OPC UA Server dargestellt.


**Abbildung 2.26:** Verwendung des lokalen Discovery Servers (Quelle: [67])

#### 2.3.8.2 Lokaler Discovery Server mit Multicast-Erweiterung (LDS-ME)

Der lokale Discovery Server mit Multicast-Erweiterung (LDS-ME) stellt dieselben Grundfunktionen wie der lokale Discovery Server (LDS) bereit und beinhaltet zusätzlich eine Multicast-Erweiterung. Die Multicast-Erweiterung basiert auf dem Multicast DNS (mDNS) Protokoll (vgl. [67], [66] S. 48). Registriert sich ein OPC UA Server bei einem lokalen Discovery Server mit Multicast-Erweiterung (LDS-ME), sendet dieser anschließend eine sogenannte MulticastAnnounce in das Subnetz. Durch die MulticastAnnounce (Multicast-Benachrichtigung) werden die anderen LDS-ME über die Registrierung des OPC UA Servers informiert und können den OPC UA Server ebenfalls in ihre Discovery-Liste hinzufügen. Damit ein LDS-ME die MulticastAnnounce Benachrichtigung erhält, muss sich dieser zwingend im selben Subnetz wie der sendende LDS-ME befinden! Durch dieses Vorgehen ist ein Ad-hoc Discovery im lokalen Subnetz möglich (vgl. [67], [68]). Wesentlicher Vorteil der Multicast-Erweiterung ist, dass der LDS-ME Informationen über OPC UA Server bereitstellt, die nicht auf seinem Host laufen. Abbildung 2.27 visualisiert das Zusammenwirken von OPC UA Server, OPC UA Client und den LDS-ME.
Mit dem FindServers- und FindServersOnNetwork-Service kann der OPC UA Client die verfügbaren bzw. registrierten OPC UA Server eines lokalen Discovery Server mit Multicast-Erweiterung (LDS-ME) abfragen. Der FindServers-Service liefert dem OPC UA Client die registrierten OPC UA Server, die sich auf demselben Hostsystem wie der LDS-ME befinden. Mit dem FindServersOnNetwork-Service kann der OPC UA Client alle OPC UA Server abfragen, die dem LDS-ME bekannt sind.
Die OPC Foundation stellt einen lokalen Discovery Server mit Multicast-Erweiterung (LDS-ME) für Microsoft-Windows-Systeme kostenfrei zur Verfügung. Dieser kann auf der OPC Foundation Webseite [69] heruntergeladen werden.

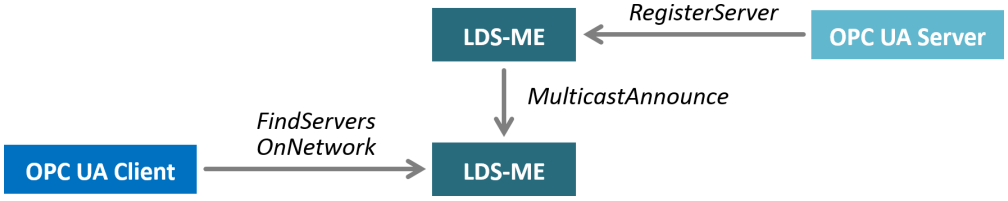


**Abbildung 2.27:** Verwendung des lokalen Discovery Servers mit Multicast-Erweiterung (Quelle: [67])

#### 2.3.8.3 Globaler Discovery Server (GDS)

Eingesetzt wird der globale Discovery Server (GDS) in großen OPC UA Systemen, die sich aus mehreren Subnetzen zusammensetzen. Der globale Discovery Server (GDS) basiert auf einem Standard OPC UA Server, welcher das globale Discovery Informationsmodell implementiert (vgl. [66] S. 13 f., [68]). Das Informationsmodell ist im zwölften Teil (Discovery) der OPC UA Spezifikation definiert.
Der globale Discovery Server (GDS) stellt für die Interaktion mit den OPC UA Servern und OPC UA Clients sechs Methoden bereit, die in Tabelle 2.15 beschrieben sind (vgl. [63] S. 15-19).

| Methoden | Beschreibung |
|---|---|
| *RegisterApplication* | Beim globalen Discovery Server können sich OPC UA Applikationen mit der *RegisterApplication*-Methode registrieren. Für die Applikationsregistrierung ist eine OPC UA Session erforderlich. Nach der Registrierung erhält die OPC UA Applikation eine eindeutige Application-ID vom globalen Discovery Server zugewiesen. |
| *UpdateApplication* | Registrierte Applikationen auf einem globalen Discovery Server (GDS) können mit der *UpdateApplication*-Methode ihren Eintrag aktualisieren. Für die Aktualisierung wird eine gesicherte OPC UA Session benötigt. |
| *UnregisterApplication* | Mit dem Aufruf der *UnregisterApplication*-Methode kann eine OPC UA Applikation seine Registrierung auf dem globalen Discovery Server (GDS) entfernen. Für den Methodenaufruf ist eine OPC UA Session erforderlich. |
| *GetApplication* | Die *GetApplication*-Methode liefert alle registrierten Applikationen, die auf dem globalen Discovery Server gelistet sind. Mit Filterparametern kann die Anzahl der zurückgelieferten Applikationen eingeschränkt werden. Der Zugriff auf die Methode erfordert keine gesicherte Verbindung. |
| *FindApplications* | Die *FindApplications*-Methode liefert für eine Applikations-Uri (Applikationsadresse) alle registrierten Applikationen. Es darf für eine Applikations-Uri lediglich eine Applikation registriert sein! Andernfalls liegt ein Fehler vor. Für den Methodenaufruf wird keine gesicherte Verbindung benötigt. Hinweis: Jede Applikation ist durch einen Applikations-Uri eindeutig im Netzwerk adressiert. |
| *QueryServers* | Mit dem Aufruf der *QueryServers*-Methode kann ein OPC UA Client alle OPC UA Server abfragen, die dem GDS im Netzwerk bekannt sind. Die Anzahl und Art der zurückgelieferten OPC UA Server kann mit Filterparametern angepasst werden. Der Aufruf der Methode erfordert keine gesicherte Verbindung. |

**Tabelle 2.15:** Methoden des globalen Discovery Servers (GDS)

Vorteil des globale Discovery Servers ist, dass dieser netzwerkweit agieren kann und nicht auf ein Subnetz begrenzt ist.
Das Unternehmen GE Automation entwickelte im Zusammenarbeit mit der OPC Foundation den ersten und bisher einzigen globalen Discovery Server (GDS) (Stand: September 2016). Mitglieder der OPC Foundation erhalten exklusiven Zugriff.


**Abbildung 2.28:** Verwendung des globalen Discovery Servers (Quelle: [67])

## 2.3.9 Bibliotheken für OPC UA

Entwickler verwenden für die einfache und schnelle Implementierung von OPC UA Anwendungen Software Development Kits (SDK). Die Software Development Kits (SDK) bauen auf dem OPC UA Stack auf und stellen die spezifizierten Funktionalitäten des OPC UA Standards kompakt bereit. (vgl. [74], [53] S. 14). Im OPC UA Stack ist das Kommunikationsprotokoll der OPC UA Spezifikation implementiert (vgl. [74], [57] S. 34). Das Kommunikationsprotokoll besteht aus der Kodierungs-, Sicherheits- und Transportebene. Für den Zugriff auf den OPC UA Stack verwendet das Software Development Kit (SDK) die bereitgestellte Programmierschnittstelle (API).
Das folgende Unterkapitel ist in zwei Teile untergliedert. Der erste Teil liefert einen Überblick über verfügbare Bibliotheken für den OPC UA Standard. Im zweiten Teil erfolgt die Auswahl einer OPC UA Bibliothek nach definierten Kriterien. Die ausgewählte OPC UA Bibliothek wird anschließend für die Realisierung des verteilten IT-Systems auf Basis von OPC UA verwendet.

### 2.3.9.1 Verfügbare OPC UA Bibliotheken

#### OPC Foundation

Die OPC Foundation stellt für die Programmiersprachen ANSI-C, Java und C# jeweils einen OPC UA Stack zur Verfügung. Für die .Net-Plattform gibt es zusätzlich ein Software Development Kit (SDK) (Client und Server). Der Quellcode der Bibliotheken steht auf GitHub zum Download bereit und ist unter GPL 2.0 lizensiert.
Der UA-AnsiC Stack ist in ANSI-C implementiert und deckt alle Funktionalitäten der aktuellen OPC UA Spezifikation ab (Version 1.03). Durch die Implementierung in ANSI-C kann der Stack sowohl auf Windows als auch auf Linux Systemen verwendet werden. Für den UA-AnsiC Stack sind keinerlei Dokumentationen verfügbar. Es gibt lediglich wenige Code-Beispiele für die Verwendung des Stacks.
Für die Programmiersprache Java entwickelte die OPC Foundation den UA-Java Stack. Zu den bisher umgesetzten Funktionalitäten des Stacks sind keine Dokumentationen und Informationen vorhanden. Daher wurde der Quellcode bzgl. Funktionalität analysiert. Die Analyse ergab, dass das benötigte Discovery Service Set nicht vollständig implementiert ist. Ansonsten ist der

Quellcode sauber strukturiert und verständlich kommentiert. Für Entwickler gibt es außerdem wenige Code-Beispiele.
Für die .Net-Plattform entwickelt die OPC UA Foundation in Zusammenarbeit mit Microsoft den UA-.NET Stack und ein darauf aufbauendes Software Development Kit (SDK) (Client und Server). Der UA-.NET Stack beinhaltet alle Funktionalitäten der OPC UA Spezifikation (Version 1.02) und ist plattformunabhängig einsetzbar. Für den .NET-Stack und das Software Development Kit (SDK) sind keine Dokumentationen oder Tutorials vorhanden. Es gibt lediglich mehrere komplexe Beispielprogramme und erfordert daher einen größeren Aufwand für die Einarbeitung.
Für die Entwicklung von Software Development Kits verwenden Hersteller häufig die Stacks der OPC Foundation.

Link: www.github.com/OPCFoundation

#### Open62541

Die Open62541 Bibliothek ist ein gemeinsames Projekt der RWTH Aachen, der TU Dresden und dem Fraunhofer IOSB. In der Bibliothek sind OPC UA Client, OPC UA Server sowie der darunterliegende OPC UA Stack implementiert. Für die Programmierung wurde die Programmiersprache ANSI-C verwendet. Dadurch ist die Bibliothek plattformunabhängig einsetzbar. Der Source-Code des Projekts wurde auf GitHub unter der LGPL-Lizenz veröffentlicht. Positiv zu erwähnen ist, dass eine ausführliche Dokumentation und ein Tutorial zur Verwendung der Bibliothek vorhanden sind. Außerdem kann die Bibliothek beliebig vom Entwickler erweitert werden. Nachteilig ist, dass das DiscoveryService Set nicht vollständig und der QueryService Set gar nicht implementiert sind.

Link: www.open62541.org

#### OpenOpcUa

Ein Konsortium aus internationalen Firmen entwickelte die OpenOPCUa Bibliothek. OpenOPCUa ist plattformunabhängig in der Programmiersprache C/C++ implementiert und beinhaltet ein Software Development Kit (SDK) für Client und Server. Laut Hersteller deckt die Bibliothek alle Funktionalitäten der aktuellen OPC UA Spezifikation (Version 1.03) ab. Veröffentlicht wurde die Bibliothek unter der CeCill-C Lizenz. Dadurch muss ein Entwickler, der die Bibliothek verwenden möchte, eine einmalige Gebühr für den Quellcode und die Dokumentation entrichten. Anschließend kann er die Bibliothek kostenfrei verwenden.

Link: www.openopcua.org

### FreeOpcUa

Eine Entwickler-Community arbeitet zusammen an dem FreeOpcUa Projekt. Ziel der Entwickler-Community ist die Bereitstellung einer plattformunabhängigen Bibliothek für den OPC UA Standard (Server und Client SDK). Der OPC UA Stack wird eigenständig entwickelt. Die FreeOpcUa Bibliothek ist in den Programmiersprachen C++ und Python verfügbar. Der Source-Code des Projekts befindet sich auf GitHub und wurde unter der LGPL-Lizenz veröffentlicht. Die Bibliothek hat mehrere Nachteile. Zum einen deckt die FreeOpcUa Bibliothek eine Vielzahl von Funktionalitäten der OPC UA Spezifikation nicht ab und zum anderen ist keine Dokumentation oder ein Tutorial vorhanden. Es gibt lediglich einen kurzen Beispielcode für die Implementierung des OPC UA Clients und OPC UA Servers.

Link: www.freeopcua.github.io

### ASNeG

Das Unternehmen ASNeG entwickelte ein kostenfreies und plattformunabhängiges Software Development Kit (inklusive Stack) für den OPC UA Standard. Programmiersprache der Bibliothek ist C/C++. Der Quellcode des Projekts steht auf der Herstellerseite kostenfrei zum Download bereit und ist unter Apache 2.0 lizensiert. Durch eine ausführliche Dokumentation und mehrere Beispiele können Entwickler die Bibliothek ohne großen Einarbeitungsaufwand verwenden. Teil der Bibliothek ist ein sogenannter Feature Plan. Der Hersteller beschreibt mit dem Feature Plan die bisher umgesetzten Funktionalitäten der OPC UA Bibliothek und welche Funktionalitäten er in Zukunft hinzufügen möchte. In der Bibliothek sind lediglich Grundfunktionen der OPC UA Spezifikation implementiert. Für einen industriellen Einsatz ist die Bibliothek daher ungeeignet.

Link: www.asneg.de/download.html

### Eclipse Milo

Milo ist ein Projekt der Eclipse Foundation. Ziel des Projekts ist die Entwicklung einer Open-Source-Bibliothek für den OPC UA Standard. Die Bibliothek beinhaltet ein Client und Server Software Development Kit (SDK). Für die Implementierung wurde die Programmiersprache Java und der Java Stack der OPC Foundation verwendet. Dadurch ist die Bibliothek plattformunabhängig einsetzbar. Unter der EPL-Lizenz veröffentlicht die Eclipse Foundation den Source-Code des Projekts auf der GitHub Plattform. Zum jetzigen Zeitpunkt (September 2016) befindet sich das Projekt noch in der Entwicklungsphase. Die erste Version ist für Ende 2016 geplant. Der bisherig veröffentlichte Quellcode macht einen strukturierten und soliden Eindruck.

Link: www.github.com/eclipse/milo

### UAF – The Unified Architecture Framework

Die belgische Universität KU Lueven entwickelt seit 2013 das Unified Architecture Framework (UAF). Das Framework verwendet das kommerzielle C++ Software Development Kit (SDK) von dem Unternehmen Unified Automation und erweitert dessen Funktionalitäten für den OPC UA Client. Der Source-Code für die Erweiterungen befindet sich auf GitHub und wurde unter der LGPL-Lizenz veröffentlicht. Das Projekt ist ausführlich dokumentiert und es werden mehrere Beispiele für die Verwendung des Frameworks bereitgestellt. Die Implementierung eines Frameworks für den OPC UA Server ist geplant.

Link: www.github.com/uaf/uaf

### OPC Development Toolkit

Die Aktiengesellschaft Softing AG entwickelte eigenständig das OPC Development Toolkit. Das OPC Development Toolkit (Client und Server) deckt alle Funktionalitäten der OPC UA Spezifikation (Version 1.02) ab und ist in den Programmiersprachen C# und C++ verfügbar. Für die Verwendung der Bibliothek ist eine kostenpflichtige Lizenz erforderlich. Auf der Herstellerseite kann jedoch ein kostenfreies Demo- und Evaluierungspaket heruntergeladen werden. Die kostenfreie Demo- und Evaluierungsversion bietet den vollen Funktionsumfang und ist lediglich in der Laufzeit auf 90 Minuten begrenzt. Positiv zu erwähnen ist, dass das OPC Development Toolkit sehr ausführlich und verständlich dokumentiert ist. Die Dokumentation beinhaltet folgende Informationen: Einführung in OPC UA, Übersicht über das Toolkit, ausführliches Programmierhandbuch (Programmer's Guide) sowie mehrere leicht verständliche Tutorials. Das OPC UA C++ Development Toolkit ist plattformunabhängig und kann auf Windows, Linux sowie VxWorks eingesetzt werden. Die Verwendung des OPC UA .NET Development Toolkit ist auf Windows-basierten Systemen begrenzt.

Link: www.industrial.softing.com/de/produkte/software/opc-development-toolkits.html

### Unified Automation

Der Hersteller Unified Automation bietet für die Programmiersprachen ANSI-C, C++ und C# jeweils ein Software Development Kit (OPC UA Client und Server) an. Die drei Software Development Kits beinhalten alle Funktionalitäten der OPC UA Spezifikation V1.02 und verwenden die Stacks der OPC UA Foundation. Positiv zu erwähnen ist die Zertifizierung der Bibliothek mit dem Compliance-Zertifikat der OPC Foundation.
Für die Benutzung der Bibliothek ist eine kostenpflichtige Lizenz erforderlich. Zum Testen der SDK ist auf der Herstellerseite eine dreimonatige Demoversion verfügbar. Die Demoversion ist auf eine Laufzeit von 60 Minuten beschränkt und beinhaltet den vollen Funktionsumfang. Alle drei Software Development Kits (SDK) sind sehr ausführlich und verständlich dokumentiert. Die Dokumentation im HTML-Format beinhaltet folgende Informationen: Einführung in OPC UA, Übersicht über das Software Development Kit, ausführliche Beschreibung der Programmierschnittstelle sowie verschiedene Tutorials und Code-Beispiele.

Das ANSI-C Software Development Kit wurde speziell für ressourcenbeschränkte Geräte sowie zeitkritische Anwendungen entwickelt. Es kann auf den Plattformen Windows, Linux, vxWorks, QNX, EUROS und RTOS eingesetzt werden. Das C++ Software Development Kit verwendet als Basis die ANSI-C Bibliothek und unterstützt zusätzlich Multithreading. Laut Hersteller kann das C++ SDK auf Windows, Linux, vxWorks und QNX Systemen verwendet werden. Für die Implementierung einer Windows-basierten OPC UA Applikation empfiehlt Unified Automation das C# Software Development Kit. Unified Automation kooperiert mit dem Unternehmen Prosys OPC und vertreibt dessen Java Software Development Kit.

Link: www.unified-automation.com/downloads/opc-ua-development.html

**PROSYS OPC**
Das OPC UA Java Software Development Kit (OPC UA Client und Server) vom Unternehmen Prosys beinhaltet alle Funktionalitäten der OPC UA Spezifikation V1.02 und ist von der OPC Foundation mit dem Compliance-Zertifikat zertifiziert. Als OPC UA Stack verwendet Prosys den UA-Java Stack der OPC Foundation. Für die Benutzung des Software Development Kit ist eine kostenpflichtige Lizenz erforderlich. Entwickler können jedoch beim Hersteller eine kostenfreie Evaluierungsversion für einen Testzeitraum von 60 Tagen anfragen. Die Evaluierungsversion beinhaltet alle Funktionalitäten der SDK und ist in der Laufzeit auf 120 Minuten beschränkt. Für die Dokumentation und Beschreibung der Programmierschnittstelle sind eine JavaDoc-Dokumentation sowie mehrere leicht verständliche Tutorials und Code-Beispiele vorhanden. Prosys OPC vertreibt außerdem das ANSI-C, C++ und C# Software Development Kit von Unified Automation.

Link: www.prosysopc.com/products/opc-ua-java-sdk/

#### 2.3.9.2 Auswahl einer OPC UA Bibliothek

Für die Realisierung des verteilten IT-Systems auf Basis von OPC UA ist eine OPC UA Bibliothek erforderlich. In diesem Unterkapitel erfolgt die Auswahl einer OPC UA Bibliothek nach folgenden Anforderungen:

- **SDK mit objektorientierter Programmiersprache und Dokumentation**
  Für die einfache und schnelle Implementierung von OPC UA Anwendungen sind Software Development Kits (Client und Server) besser geeignet als reine OPC UA Stacks. Das verwendete Software Development Kit soll in einer objektorientierten Programmiersprache implementiert sein. Wünschenswert wäre außerdem eine ausführliche Dokumentation des Software Development Kits.
- **Plattformunabhängig**
  Die verwendete Bibliothek muss plattformunabhängig einsetzbar sein. In der Masterarbeit werden OPC UA Anwendungen auf Windows sowie Raspbian (Linux) implementiert.

- **Bereitstellung benötigter OPC UA Services**
  Für die Realisierung der Masterarbeit werden folgenden OPC UA Service Sets benötigt:
  - Discovery Service Set
  - Secure Channel Service Set
  - Session Service Set
  - Attribute Service Set
  - Node Management Service Set
  - Methode Service Set
  - MonitoredItem Service Set
  - Subscription Service Set

  Hinweis: In Kapitel 2.3.4 sind die OPC UA Service Sets ausführlich beschrieben.

- **Kostengünstige Lösung**
  Das Budget für die Lizenz der OPC UA Bibliothek ist auf 1.000 € begrenzt. Optimal wäre eine kostenfreie Bibliothek. Eventuell bieten die Hersteller eine Akademische Lizenz für die nicht kommerzielle Verwendung an.

#### Ergebnis:

Unter Berücksichtigung der gestellten Anforderungen sowie der Vor- und Nachteile wurde die OPC UA Java Bibliothek des Herstellers Prosys als am geeignetsten befunden. Insbesondere verfügt die Bibliothek über eine hervorragende Dokumentation, Tutorials sowie mehrere Beispiele. Durch die objektorientierte Programmiersprache Java ist eine schnelle und plattformunabhängige Implementierung möglich. Des Weiteren beinhaltete die OPC UA Bibliothek alle benötigten Service Sets. Hinsichtlich der Kosten wird im Rahmen dieser Arbeit vorerst die Evaluationsversion des Herstellers verwendet.

### 2.3.10 Nutzungsanalyse

#### Theoretische Thesen:

- OPC Unified Architecture spielt eine entscheidende Rolle bei der Realisierung der Vision Industrie 4.0.
- OPC Unified Architecture ermöglicht die horizontale und vertikale Integration der verschiedenen IT-Systeme zu einem gesamten und intelligenten IT-System. (Optimierungspotential der Wertschöpfungskette.)
- OPC Unified Architecture ermöglicht die intelligente Steuerung eines cyber-physischen Produktionssystems.

- OPC Unified Architecture ist ein einfacher, sicherer, herstellerübergreifender und plattformunabhängiger Standard um echtzeitnah an die Daten des cyber-physischen Produktionssystems zu gelangen.
- Der OPC Unified Architecture Standard ist skalierbar (OPC UA Profile). Durch die definierten OPC UA Profile kann der Standard selbst auf ressourcenbeschränkten Geräten eingesetzt werden. Ressourcenbeschränkte Geräte sind in diesem Kontext Sensoren und Aktoren auf der Feldebene.
- Der OPC Unified Architecture Standard ist nicht geeignet um Produktionsprozesse auf der Steuerungs- und Feldebene zu steuern oder zu überwachen.
- Im Produktionssystem können Sicherheitslücken durch die fehlerhafte Verwendung des OPC UA Standards entstehen. In sicherheitskritischen Anwendungsfällen müssen sich Entwickler detailliert mit dem Thema Sicherheit auseinandersetzen! .

In der Masterarbeit wurden lediglich theoretische Thesen für die Nutzungsanalyse aufgestellt. Eine umfangreiche Analyse übersteigt den Rahmen dieser Thesis und sollte in einer separaten Arbeit abgehandelt werden.

# 3 Pflichtenheft

## 3.1 Zielbestimmung

Ziel der Masterarbeit ist die Realisierung eines verteilten IT-Systems auf Basis von OPC UA. Hierzu wird ein OPC UA Netzwerk mit mehreren realen und virtuellen OPC UA Servern sowie einer Workstation erstellt. Die realen OPC UA Server laufen auf einem Einplatinencomputer (Raspberry Pi) und stellen verschiedene Dienstleistungen mit Hilfe von Sensoren und Aktoren bereit. Die virtuellen OPC UA Server bieten nur fiktive Dienstleistungen an. Auf der Workstation wird ein Programm zur Visualisierung und Administration des OPC UA Netzwerkes ausgeführt. Mit einer grafischen Oberfläche kann der Anwender die registrierten OPC UA Server (Produktionsanlagen) im Netzwerk auflisten (Discovery-Server), Produktionsaufträge in dem verteilten IT-System erstellen, den Status eines Produktionsauftrags abfragen, virtuelle OPC UA Server mit fiktiven Dienstleistungen generieren und virtuelle OPC UA Server administrieren. Die Vernetzung der einzelnen Geräte erfolgt mit einem Router oder Switch. Der Aufbau des verteilten Systems auf Basis von OPC UA ist in Abbildung 3.1 abstrakt dargestellt.

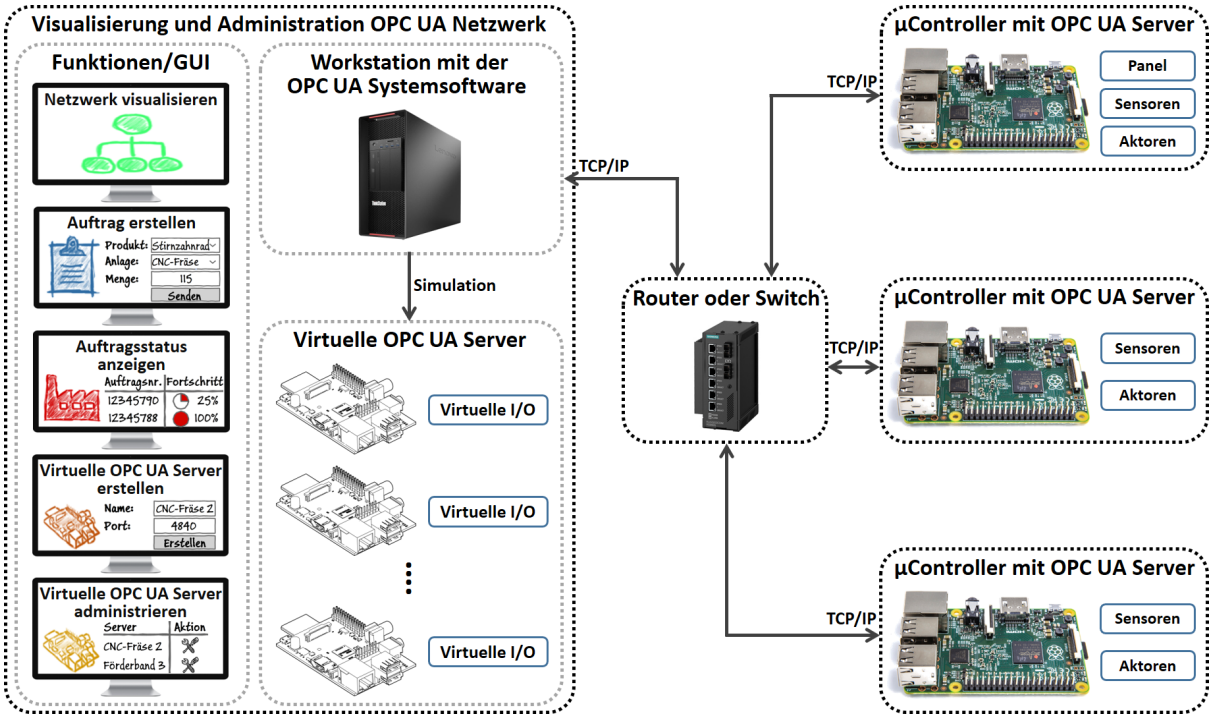


**Abbildung 3.1:** Aufbau des verteilten Systems auf Basis von OPC UA

OPC UA ist ein industrielles Machine-to-Machine (M2M) Kommunikationsprotokoll, mit dem es möglich ist, flexible und erweiterbare Produktionssysteme zu konstruieren. Diese Eigenschaft soll mit der oben beschriebenen Arbeit an einem Demonstrator veranschaulicht werden.

### 3.1.1 Musskriterien

In der Masterarbeit soll ein verteiltes IT-System auf Basis von OPC UA erstellt werden. Als OPC UA Server (Produktionseinheiten) sollen mehrere Einplatinencomputer (Raspberry Pi) und virtuelle Produktionseinheiten verwendet werden. Die virtuellen Produktionseinheiten sind auf einer Workstation zu virtualisieren. Zur Visualisierung und Administration des OPC UA Netzwerkes soll ein Programm mit grafischer Benutzeroberfläche entwickelt werden, mit dem es möglich ist, die registrierten OPC UA Server im Netzwerk aufzulisten (Discovery-Server), den Produktionsstatus eines Auftrags abzufragen, Produktionsaufträge in dem verteilten IT-System zu erstellen, virtuelle OPC UA Server mit fiktiven Dienstleistungen zu generieren und virtuelle OPC UA Server zu administrieren.
An einem Demonstrator und mit virtuellen Produktionseinheiten soll anschließend die Flexibilität und Erweiterbarkeit eines verteilten IT-Systems auf Basis von OPC UA veranschaulicht werden.

### 3.1.2 Wunschkriterien

Intention ist die Erfassung und Visualisierung des aktuellen Gesamtsystemzustands. Dem Benutzer sollen System-Informationen wie beispielsweise physikalischer Standort, Verfügbarkeit/Auslastung, aktuelle Produktionskosten oder Statusvariablen zu den einzelnen Produktionseinheiten zur Verfügung stehen.

### 3.1.3 Abgrenzungskriterien

Alle OPC UA Server (Produktionseinheit) müssen sich im lokalen Netzwerk befinden. Außerdem muss jeder OPC UA Server fünf genormte Zugriffsmethoden bereitstellen, welche vom Netzwerk aufrufbar sind. Eine Methode soll die Abfrage der angebotenen Dienstleistungen von einem OPC UA Server ermöglichen. Die Auftragsliste eines OPC UA Servers (Produktionseinheit) soll mit der zweiten Methode abrufbar sein. Mit einer weiteren Methode sollen einem OPC UA Server (Produktionseinheit) Dienstleistungsaufträge zugewiesen werden. Der aktuelle Status und Bearbeitungsfortschritt eines Dienstleistungsauftrags soll mit der vierten Methode abrufbar sein. Mit der fünften Methode kann der Anwender einen Dienstleistungsauftrag auf der Produktionseinheit stornieren.

## 3.2 Einsatz

### 3.2.1 Anwendungsbereiche

Im Rahmen dieser Arbeit wird erstmalig an der Universität Stuttgart ein größeres verteiltes IT-System auf Basis von OPC UA realisiert. Mit dem verteilten IT-System soll gezeigt werden, dass mit dem industriellen Machine-to-Machine (M2M) Kommunikationsprotokoll ein Produktionssystem flexibel und erweiterbar gestaltet werden kann. Ein weiterer Anwendungsbereich des Produkts ist die Schaffung einer Grundlage für zukünftige Forschungsprojekte und studentische Arbeiten mit dem Themenfeld „OPC Unified Architecture“.

### 3.2.2 Zielgruppen

Zielgruppe der Masterarbeit sind alle Studenten, Mitarbeiter und Industriepartner des IAS, die im Bereich „OPC Unified Architecture“ forschen und entwickeln.

## 3.3 Umgebung

### 3.3.1 Software

Das Programm zur Visualisierung und Administration des OPC UA Netzwerks soll auf einem leistungsstarken Rechner mit dem Betriebssystem Windows oder Linux ausgeführt werden.

Die Programme für die Einplatinencomputer werden auf dem Linux-basierten Betriebssystem Raspbian ausgeführt.

### 3.3.2 Hardware

Zur Realisierung des verteilten IT-Systems sind folgende Hardwarekomponenten erforderlich: Mehrere Einplatinencomputer (Raspberry Pi), ein leistungsstarker Rechner mit Monitor, ein Router oder Switch sowie der bestehende Demonstrator mit seinen Sensoren und Aktoren.

### 3.3.3 System-Schnittstellen

Bei dem zu entwickelnden Produkt handelt es sich um ein flexibles und beliebig erweiterbares System. In dem verteilten IT-System auf Basis von OPC UA können neue OPC UA Server Ad-hoc integriert werden. Die neu hinzugefügten OPC UA Server (Produktionseinheiten) müssen sich lediglich beim Discovery-Server registrieren und in zyklischen Abständen die Registrierung erneuern. Beim Registrierungsvorgang übermittelt der neu hinzugefügte OPC UA Server die angebotenen Dienstleistungen der Produktionseinheit.

## 3.4 Funktionale Anforderungen

Die funktionalen Anforderungen an das System werden in zwei Teile untergliedert.

/PFA10/ **OPC UA Netzwerk**

/PFA11/ Konzeption und Aufbau eines verteilten Systems auf Basis von OPC UA. /LA 11/

/PFA12/ Das OPC UA Netzwerk soll aus mehreren realen OPC UA Servern, virtuellen OPC UA Servern und einem leistungsstarken Rechner bestehen. /LA 12/

/PFA13/ Reale OPC UA Server sollen auf einem Einplatinencomputer (Raspberry Pi) laufen und einen bestehenden Demonstrator mittels Sensoren und Aktoren steuern. /LA15/

/RFA14/ Reale OPC UA Server sollen Dienstleistungen (Services) anbieten. /LA15/

/PFA15/ Virtuelle OPC UA Server sollen auf dem leistungsstarken Rechner visualisiert werden und mehrere fiktive Dienstleistungen (Services) anbieten. /LA14/

/PFA16/ Die fiktiven Dienstleistungen (Services) der virtuellen OPC UA Server sollen Latenzzeiten besitzen. Die Latenzzeiten sollen ungefähr den realen Bearbeitungszeiten entsprechen. /LF16/

/PFA17/ OPC UA Server sollen Ad-hoc in das System integrierbar sein. Sobald ein OPC UA Server (Produktionseinheit) ins bestehende Netzwerk hinzugefügt wird, soll eine Produktion ohne weitere Konfigurationen möglich sein. Es wird dabei von einem Plug & Produce-Prinzip gesprochen. /LA13/

/PFA18/ Es soll ein OPC UA Discovery Server entwickelt werden. Der Discovery Server soll alle registrierten OPC UA Server im Netzwerk auflisten. Zu jedem registrierten OPC UA Server soll der Servername, die Adresse und die angebotenen Dienstleistungen erfasst und bereitgestellt werden. /LF14/

/PFA19/ Wird ein OPC UA Server Ad-hoc in das System integriert, muss sich dieser beim Discovery Server registrieren. /LA 14/

/PFA20/ Wird ein OPC UA Server vom System getrennt, soll sich dieser beim Discovery Server abmelden. /LA 14/

/PFA21/ Der Discovery Server soll erkennen, ob ein OPC UA Server ohne vorherige Abmeldung getrennt wurde. /LA14/, /LNA30/

/PFA22/ Zur einfacheren Adressierung sind dem Discovery Server ein statischer Hostname und Port zuzuweisen. Die Adressierungsdaten sind allen OPC UA Servern im Netzwerk bekannt. /LF14/

/PFA23/ In dem verteilten IT-System soll es möglich sein, mehrere Dienstleistungsaufträge in einem Produktionsauftrag zu orchestrieren. Produktionsaufträge sollen so mehrere Dienstleistungsaufträge koordinieren und dadurch hochwertigere Dienstleistungen bzw. intelligente Produkte erstellen. /LF17/

/PFA24/ Ein Produktionsszenario für den Demonstrator ist zu entwickeln. In dem Szenario sollen parallel zum Demonstrator verschiedene Produktionsprozesse simuliert werden. /LA18/

/PFA25/ **Visualisierung und Administration des OPC UA-Netzwerks**

/PFA26/ Zur Visualisierung und Administration des OPC UA Netzwerks soll ein Programm entwickelt werden. /LF21/

/PFA27/ Das Programm stellt dem Benutzer mit einer grafischen Oberfläche (GUI) die folgenden fünf Funktionen bereit: Registrierte OPC UA Server im Netzwerk auflisten (Discovery-Server), Produktionsaufträge in dem verteilten IT-System erstellen, Status von einem Produktionsauftrag abfragen, virtuelle OPC UA Server mit fiktiven Dienstleistungen generieren und virtuelle OPC UA Server administrieren. /LF20/, /LF21/, /LF22/, /LF24/, /LF25/

/PFA28/ In der grafischen Anwendung sollen die registrierten OPC UA Server mit Name, Adresse, und angebotenen Dienstleistungen aufgelistet werden. Aufgabe des Discovery Servers ist die Erfassung und Bereitstellung der Registrierungsdaten.

/PFA29/ Bei der Produktionsauftragserstellung soll es dem Benutzer möglich sein, in einem Produktionsauftrag mehrere Dienstleistungen zu kombinieren und dadurch höherwertigere Dienstleistungen bzw. intelligente Produkte zu erstellen. /LF24/

/PFA30/ Der Anwender kann einem virtuellen OPC UA Server mehrere simulierten Dienstleistungen zuweisen. Es sollen dafür vordefinierte Dienstleistungen wie beispielsweise Bohren, Schrauben, Fräsen oder Transport zur Verfügung stehen. Diese Dienstleistungen sollen vereinfacht - mittels Latenzzeiten - simuliert werden. /LF16/, /LF22/

/PFA31/ Die virtuellen OPC UA Server sollen in der grafischen Oberfläche administrierbar sein. Mit der Administration kann der Benutzer einen virtuellen OPC UA Server starten, stoppen, ändern, löschen und den aktuellen Status abfragen. Verschiedene Serverparameter, wie beispielsweise Servername, Port oder die simulierten Dienstleistungen, sollen in der grafischen Oberfläche änderbar sein. /LF22/

/PFA32/ Erfassung und Visualisierung des aktuellen Gesamtsystemzustands. Dem Benutzer sollen System-Informationen, wie beispielsweise den physischen Standort, Verfügbarkeit/Auslastung, aktuelle Produktionskosten oder Statusvariablen zu den einzelnen Produktionseinheiten zur Verfügung stehen (optional). /LF26/

## 3.5 Nichtfunktionale Anforderungen

/PNA10/ Mit einer einfachen Bedienbarkeit und einem selbsterklärenden Aufbau soll eine benutzerfreundliche Anwendung erreicht werden. /LNA10/, /LNA40/

/PNA20/ Ziel ist eine hohe Zuverlässigkeit des Systems. Die Programme sollten daher keine undefinierten Zustände besitzen, auftretende Fehlerfälle abfangen und ausführlich getestet werden. Tritt dennoch ein Problem auf, wird das System in einen definierten Zustand versetzt z.B. Neustart. /LNA20/

/PNA30/ Eine einfache Änderbarkeit, Erweiterbarkeit und Wartbarkeit des Systems wird angestrebt. Zur Erreichung dieser Anforderung soll ein modularer Programmaufbau mit definierten Schnittstellen entwickelt werden. Außerdem ist der Quellcode ausreichend zu kommentieren, so dass dieser bei neuen Anforderungen wiederverwendet oder erweitert werden kann.

/PNA40/ Der Discovery-Server muss innerhalb einer Latenzzeit von drei Sekunden erkennen, ob ein OPC UA Server vom Netzwerk getrennt wurde. /LNA30/

/PNA50/ Das Programm zur Visualisierung und Administration des OPC UA Netzwerks soll in einer Windows- und Linux-basierten Umgebung lauffähig sein (hohe Übertragbarkeit).

## 3.6 Anforderungen an die Benutzungsschnittstelle

/PBA10/ Die Benutzeroberfläche zur Visualisierung und Administration des OPC UA Netzwerkes soll intuitiv und benutzerfreundlich gestaltet sein. /LNA 40/

/PBA20/ Registrierte OPC UA Server im Netzwerk auflisten (Discovery-Server), Produktionsaufträge in dem verteilten IT-System erstellen, virtuelle OPC UA Server mit fiktiven Dienstleistungen generieren und virtuelle OPC UA Server administrieren.

/PBA30/ Zur Bedienung der Benutzeroberfläche wird eine Maus verwendet. Eingaben für Textfelder erfolgen mittels Tastatur.

/PBA40/ In einem Dialogfenster soll der Benutzer über Fehler sowie deren mögliche Behebung informiert werden.

/PBA50/ Durch eine einfache Anleitung -in Papierform- sollen dem Benutzer die Funktionen sowie die Bedienung des Produkts veranschaulicht werden.

## 3.7 Qualitätszielbestimmung

| **Produktqualität** | **sehr hoch** | **hoch** | **normal** | **nicht relevant** |
|---|---|---|---|---|
| **Funktionalität** | | × | | |
| Richtigkeit | | × | | |
| Sicherheit (Security) | | | | × |
| Interoperabilität | × | | | |
| **Zuverlässigkeit** | × | | | |
| Reife | × | | | |
| Fehlertoleranz | | × | | |
| Wiederherstellbarkeit | | × | | |
| **Sicherheit (Safety)** | | | | × |
| **Benutzbarkeit** | × | | | |
| Verständlichkeit | | × | | |
| Erlernbarkeit | × | | | |
| Bedienbarkeit | × | | | |
| **Effizienz** | | | × | |
| Zeitverhalten | | | × | |
| Verbrauchsverhalten | | | × | |
| **Änderbarkeit** | × | | | |
| Analysierbarkeit | | × | | |
| Modifizierbarkeit | × | | | |
| **Übertragbarkeit** | | × | | |

## 3.8 Globale Testszenarien/Testfälle

Grundlage der Systemabnahme bildet das Pflichtenheft. Die Umsetzung der Anforderungen und Ziele sind entsprechend diesen Vorgaben zu prüfen. Weiterhin werden die Funktionalitäten des Gesamtsystems mittels plausibler Testfälle verifiziert.

## 3.9 Entwicklungsumgebung

### 3.9.1 Software

| | |
|---|---|
| **Entwicklungsumgebung:** | Eclipse IDE für Java EE (Mars 4.5) |
| **Software Development Kit:** | Java Software Development Kit 8<br>Prosys OPC UA Java SDK 2.2.4 |
| **Textverarbeitung:** | Texmaker 4.5 mit der TeX-Distribution MiKTex 2.9<br>Microsoft Office 2016 |
| **Grafikprogramm:** | Gimp 2.9.2<br>Microsoft Visio 2016<br>Microsoft Expression Studio Ultimate 4 |
| **Projektmanagement:** | Microsoft Project 2016 |
| **Zentrale Versionsverwaltung:** | Tortoise SVN 1.9<br>Apache Subversion (SVN) 1.9 |
| **Betriebssysteme:** | Windows 8<br>Raspbian Jessie<br>DiskStation Manager 6.0 |

### 3.9.2 Hardware

| | |
|---|---|
| **Entwicklungsrechner:** | Lenovo ThinkPad X1 Carbon 2014<br>Rechner im Diplomanten-Pool (IAS) |
| **Einplatinencomputer:** | Raspberry Pi 3 Model B |
| **Demonstrator:** | Sortieranlage |
| **NAS-Server:** | Synology DiskStation DS213+ |

### 3.9.3 Entwicklungsschnittstellen

Für die Realisierung des verteilten IT-Systems werden mehrere Programme entwickelt. Die Implementierung erfolgt auf den Entwicklungsrechnern.

### 3.9.4 Durchführung

Die Arbeit soll nach dem IAS-Vorgehensmodell (Modell zur Softwareentwicklung) durchgeführt werden.

In regelmäßigen Abständen (ca. alle 2 Wochen) werden die Betreuer über den aktuellen Stand informiert.

Bei der Ausarbeitung, Anfertigung und Durchführung der Arbeit werden die Richtlinien des IAS beachtet.

# 4 Systemmodell

## 4.1 Übersicht über die Anwendungsfälle

### 4.1.1 OPC UA Netzwerk visualisieren

Das Programm visualisiert für den Anwender die Informationen zum Netzwerk. In einer Übersicht werden die registrierten OPC UA Server mit Name, Adresse, Port, Standort und Anzahl angebotener Dienstleistungen aufgelistet. Der Benutzer kann einen OPC UA Server auswählen und die angebotenen Dienstleistungen mit Informationen abfragen. Die Benutzeroberfläche aktualisiert sich automatisch, sobald ein OPC UA Server hinzugefügt oder entfernt wird.

### 4.1.2 Produktionsauftrag erstellen

Zur Erstellung eines Produktionsauftrags wählt der Benutzer ein intelligentes Produkt aus und legt die gewünschte Auftragsmenge fest. Nachfolgend wird im verteilten IT-System der Produktionsauftrag erstellt und den Produktionseinheiten zugewiesen. Bei der Auftragserstellung wird für jeden Produktionsauftrag eine Auftragsnummer generiert. Produktionsaufträge bestehen aus einem oder mehreren Dienstleistungsaufträgen, deren Abarbeitungsreihenfolge über die Sequenznummer definiert ist. Besitzen mehrere Dienstleistungsaufträge dieselbe Sequenznummer, können diese parallel abgearbeitet werden.

### 4.1.3 Auftragsstatus anzeigen

Der aktuelle Status sowie der Bearbeitungsfortschritt der Produktions- und Dienstleistungsaufträge werden in der Statusübersicht visualisiert. Als Statusmeldung der Produktionsschritte sind verschiedene Zustände wie zum Beispiel *Wartend*, *Begonnen*, *Fehler* und *Beendet* definiert. Weiterhin kann der Anwender mit Schaltflächen den aktuellen Status und Bearbeitungsfortschritt aktualisieren.

### 4.1.4 Virtuellen OPC UA Server erstellen

Ein weiterer Anwendungsfall ist die Erstellung eines virtuellen OPC UA Servers. In der grafischen Oberfläche definiert der Benutzer für den zu erstellenden virtuellen OPC-Server verschiedene Serverparameter wie beispielsweise Servername, Port, Standort oder angebotene (fiktive) Dienstleistungen. Anschließend kann er diesen virtuellen OPC UA Server erstellen und starten.

### 4.1.5 Virtuellen OPC UA Server administrieren

Zur Administration werden in einer Liste alle virtuellen OPC UA Server mit Name, Port und Status aufgelistet. Der Benutzer wählt einen virtuellen OPC UA Server aus und kann diesen starten, stoppen, ändern, löschen oder den aktuellen Status abfragen.

### 4.1.6 OPC UA Server hinzufügen

Der Benutzer kann einen OPC UA Server Ad-hoc in dem verteilten IT-System hinzufügen. Mit diesem OPC UA Server (Produktionseinheit) kann ohne weitere Konfigurationen produziert werden.

### 4.1.7 OPC UA Server trennen mit Abmeldung

Um einen OPC UA Server regulär vom verteilten IT-System zu trennen, muss der Benutzer diesen beim Discovery-Server abmelden.

### 4.1.8 OPC UA Server trennen ohne Abmeldung

Der OPC UA Server wird beim Discovery-Server nicht abgemeldet, sondern irregulär vom verteilten IT-System getrennt.

## 4.2 Use-Case-Diagramm

Nachfolgend sind drei Use-Case-Diagramme dargestellt. In Abbildung 4.1 werden die Anwendungsfälle der Benutzeroberfläche abstrakt beschrieben. Abbildung 4.3 zeigt das Diagramm um eine Detail-Ebene erweitert. Die Anwendungsfälle an das physische System sind in Abbildung 4.2 beschrieben.

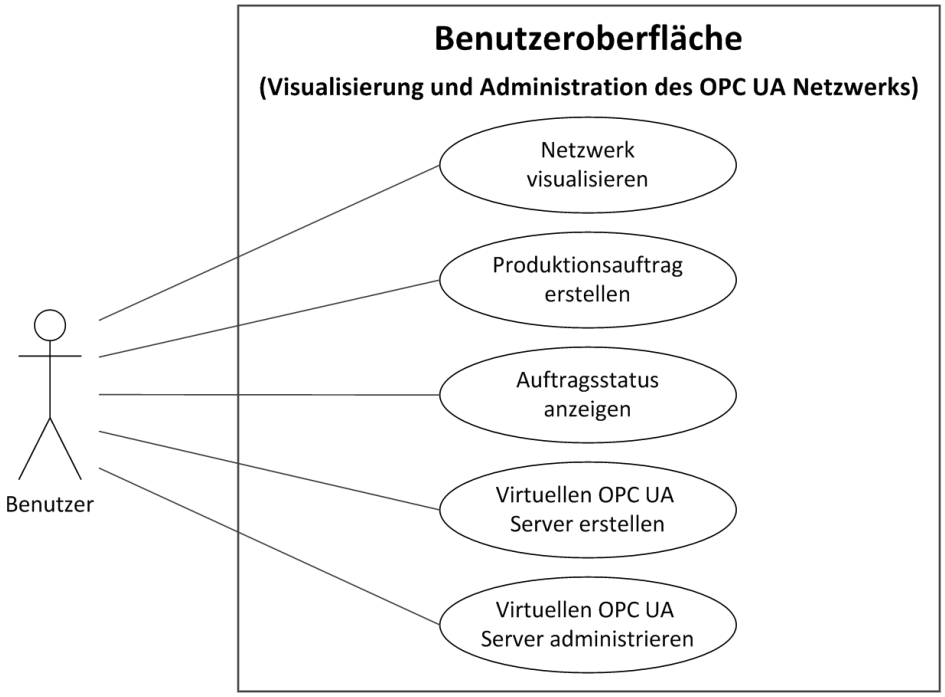


**Abbildung 4.1:** Use-Case-Diagramm Benutzeroberfläche

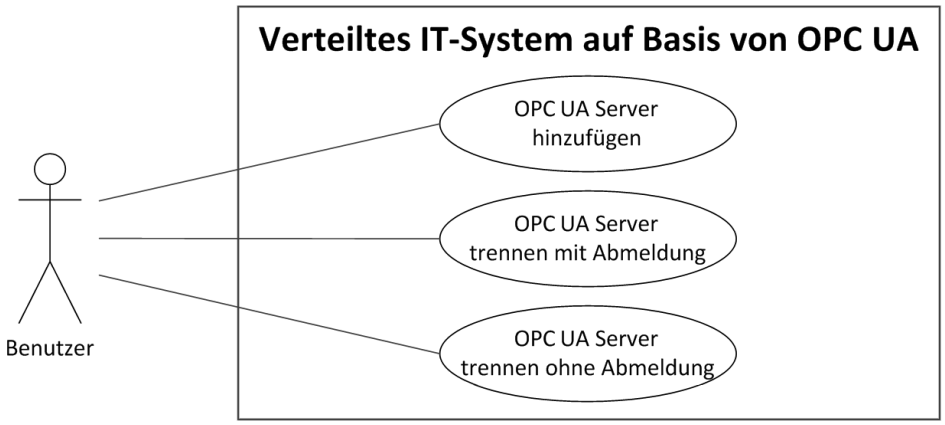


**Abbildung 4.2:** Use-Case-Diagramm physisches System

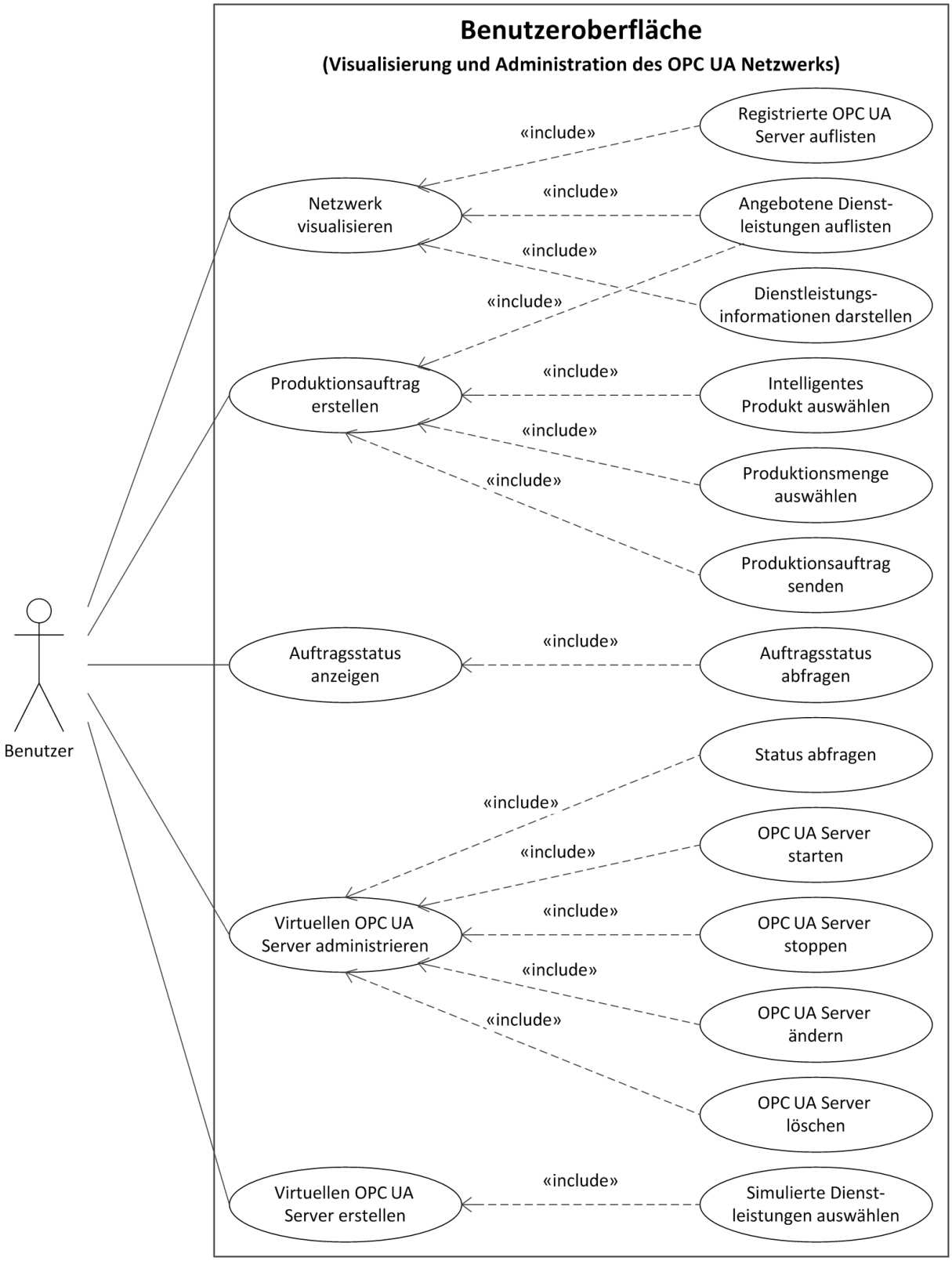


**Abbildung 4.3:** Erweitertes Use-Case-Diagramm Benutzeroberfläche

## 4.3 Beschreibung der Anwendungsfälle

### 4.3.1 OPC UA Netzwerk visualisieren

| | |
|---|---|
| *Anwendungsfall* | OPC UA Netzwerk visualisieren |
| *Ziel* | Das Programm visualisiert für den Anwender die Informationen zum Netzwerk. |
| *Kategorie* | **primär** |
| *Externe Akteure* | Benutzer, Datenbank, OPC UA Discovery Server |
| *Vorbedingung* | Die Datenbank und die OPC UA Systemsoftware müssen online sein. |
| *Nachbedingung Erfolg* | Die registrierten OPC UA Server werden mit Name, Adresse, Standort und angebotenen Dienstleistungen aufgelistet. |
| *Nachbedingung Fehlschlag* | Das Programm erkennt, dass die Datenbank oder die OPC UA Systemsoftware nicht online sind und generiert eine Fehlermeldung mit Behebungsvorschlägen. |
| *Auslösendes Ereignis* | Über die Benutzeroberfläche wird das Ereignis vom Anwender ausgelöst. |
| *Beschreibung* | **1** Über den Menüpunkt *OPC UA Netzwerk visualisieren* gelangt der Benutzer in die grafische Oberfläche zur Visualisierung des OPC UA Netzwerks.<br>**2** In der grafischen Oberfläche werden alle registrierten OPC UA Server mit Name, Adresse, Standort und Anzahl der angebotenen Dienstleistungen aufgelistet.<br>**3** Anschließend kann der Anwender einen OPC UA Server auswählen.<br>**4** Für den ausgewählten OPC UA Server werden die angebotenen Dienstleistungen mit Informationen dargestellt. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.2 Produktionsauftrag erstellen

| | |
|---|---|
| *Anwendungsfall* | Produktionsauftrag erstellen |
| *Ziel* | Der Anwender erstellt einen Produktionsauftrag, welcher aus einem oder mehreren Dienstleistungsaufträgen besteht. |
| *Kategorie* | **primär** |
| *Externe Akteure* | Benutzer, Datenbank |
| *Vorbedingung* | Die Datenbank muss online sein. |
| *Nachbedingung Erfolg* | Es wird ein Produktionsauftrag in dem verteilten IT-System erstellt. Der Anwender erhält für den Produktionsauftrag eine Auftragsnummer. |
| *Nachbedingung Fehlschlag* | Der Nutzer erhält eine Fehlermeldung, wenn die Datenbank nicht online ist. Befindet sich in dem verteilten IT-System keine Produktionseinheit für die georderte Dienstleistung, verbleibt der Dienstleistungsauftrag im Wartezustand. |
| *Auslösendes Ereignis* | Das Ereignis wird vom Anwender über die Benutzeroberfläche ausgelöst. |
| *Beschreibung* | **1** Die grafische Oberfläche zur Auftragserstellung wird geladen, wenn der Benutzer den Menüpunkt *Produktionsauftrag erstellen* auswählt.<br>**2** Zur Erstellung eines Produktionsauftrags wählt der Benutzer ein intelligentes Produkt und die gewünschte Auftragsmenge aus.<br>**3** Im Anschluss wird der Produktionsauftrag erstellt und verarbeitet.<br>**4** Bei der Auftragserstellung wird dem Produktionsauftrag eine eindeutige Auftragsnummer zugewiesen.<br>**5** Produktionsaufträge bestehen aus einem oder mehreren Dienstleistungsaufträgen, deren Abarbeitungsreihenfolge über die Sequenznummer definiert ist. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.3 Auftragsstatus anzeigen

| | |
|---|---|
| *Anwendungsfall* | Auftragsstatus anzeigen |
| *Ziel* | Den aktuellen Status und Bearbeitungsfortschritt der Produktions- und Dienstleistungsaufträge zu visualisieren. |
| *Kategorie* | **primär** |
| *Externe Akteure* | Benutzer, Datenbank, OPC UA Server (Produktionseinheit) |
| *Vorbedingung* | Die Datenbank muss online sein. |
| *Nachbedingung Erfolg* | Aktueller Status und Bearbeitungsfortschritt der Produktions- und Dienstleistungsaufträge visualisieren. |
| *Nachbedingung Fehlschlag* | Der Benutzer erhält eine Fehlermeldung, wenn in der Datenbank ein Fehler auftritt. |
| *Auslösendes Ereignis* | Über die Benutzeroberfläche wird das Ereignis vom Anwender ausgelöst. |
| *Beschreibung* | **1** Zu einem Produktionsauftrag kann der aktuelle Status und Bearbeitungsfortschritt mit dem Menüpunkt *Auftragsstatus anzeigen* abgefragt werden.<br>**2** In der grafischen Oberfläche werden alle Produktionsaufträge mit Auftragsnummer, aktuellem Status und Bearbeitungsfortschritt aufgelistet.<br>**3** Anschließend kann der Anwender aus der Liste einen Produktionsauftrag auswählen.<br>**4** Für den ausgewählten Produktionsauftrag werden die Dienstleistungsaufträge sowie der aktuelle Status und Bearbeitungsfortschritt der Dienstleistungsaufträge geladen.<br>**5** Der Anwender kann mit einem Button den Status und Fortschritt eines Produktions- und Dienstleistungsauftrags aktualisieren. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.4 Virtuellen OPC UA Server erstellen

| | |
|---|---|
| *Anwendungsfall* | Virtuellen OPC UA Server erstellen |
| *Ziel* | Ziel ist die Erstellung eines virtuellen OPC UA Servers. |
| *Kategorie* | **sekundär** |
| *Externe Akteure* | Benutzer, Datenbank |
| *Vorbedingung* | Die Datenbank muss online sein. |
| *Nachbedingung Erfolg* | Der virtuelle OPC UA Server wurde erstellt. |
| *Nachbedingung Fehlschlag* | Der virtuelle OPC UA Server konnte nicht erstellt werden. Eine Fehlermeldung wird generiert und visualisiert. |
| *Auslösendes Ereignis* | Das Ereignis wird vom Anwender über die Benutzeroberfläche ausgelöst. |
| *Beschreibung* | **1** Über den Menüpunkt *Virtuellen OPC UA Server erstellen* gelangt der Benutzer in die grafische Oberfläche zur Erstellung eines neuen virtuellen OPC UA Servers.<br>**2** In der Benutzeroberfläche kann der Anwender dem zu erstellenden OPC UA Server verschiedene Serverparameter wie beispielsweise Name, Port, Standort oder die angebotenen (fiktiven) Dienstleistungen zuweisen.<br>**3** Es stehen verschiedene fiktive Dienstleistungen zur Auswahl, die vereinfacht - mittels Latenzzeiten - simuliert werden.<br>**4** Die Serverparameter des virtuellen OPC-Server werden in die Datenbank eingetragen. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.5 Virtuellen OPC UA Server administrieren

| | |
|---|---|
| *Anwendungsfall* | Virtuellen OPC UA Server administrieren |
| *Ziel* | Der Anwender administriert die virtuellen OPC UA Server in der Benutzeroberfläche. |
| *Kategorie* | **primär** |
| *Externe Akteure* | Benutzer, Datenbank |
| *Vorbedingung* | Die Datenbank muss online sein. In der Datenbank befindet sich mindestens ein virtueller OPC UA Server für die Administration. |
| *Nachbedingung Erfolg* | In einer grafischen Übersicht kann der Anwender die virtuellen OPC UA Server administrieren. |
| *Nachbedingung Fehlschlag* | Zur Administration muss mindestens ein virtueller OPC UA Server vorhanden sein. Andernfalls wird eine Fehlermeldung angezeigt. |
| *Auslösendes Ereignis* | Das Ereignis wird vom Anwender über die Benutzeroberfläche ausgelöst. |
| *Beschreibung* | **1** Für die Administration der virtuellen OPC UA Server wählt der Benutzer den Menüpunkt *Virtuellen OPC UA Server administrieren.*<br>**2** Die Serverparameter der virtuellen OPC UA Server sind in der Datenbank abgespeichert.<br>**3** In einer Liste werden alle vorhandenen virtuellen OPC UA Server dargestellt.<br>**4** Zur Administration wählt der Anwender einen virtuellen OPC UA Server aus der Liste aus.<br>**5** Für den ausgewählten OPC UA Server werden die Serverparameter und Statuswerte visualisiert. Weiterhin kann dieser gestartet, gestoppt, geändert oder gelöscht werden. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.6 OPC UA Server hinzufügen

| | |
|---|---|
| *Anwendungsfall* | OPC UA Server hinzufügen |
| *Ziel* | Ein realer OPC UA Server wird Ad-hoc in das verteilte IT-System hinzugefügt. |
| *Kategorie* | **primär** |
| *Externe Akteure* | Benutzer |
| *Vorbedingung* | Es muss ein realer OPC UA Server vorhanden sein, welcher dem verteilten IT-System Ad-hoc hinzugefügt werden kann. |
| *Nachbedingung Erfolg* | Der reale OPC UA Server hat sich erfolgreich beim Discovery Server registriert. Der Benutzer kann mit der Produktionseinheit - ohne weitere Konfiguration - produzieren. Eine Status-LED leuchtet an dem Einplatinencomputer. |
| *Nachbedingung Fehlschlag* | Die Registrierung beim Discovery Server ist fehlgeschlagen, eine Status-LED an dem Einplatinencomputer leuchtet auf. |
| *Auslösendes Ereignis* | In das verteilte IT-System wird ein realer OPC UA Server hinzugefügt. |
| *Beschreibung* | **1** Der Benutzer verbindet einen realen OPC UA Server mit dem verteilten IT-System.<br>**2** Beim Discovery Server registriert sich der Einplatinencomputer.<br>**3** Eine Status-LED am Einplatinencomputer visualisiert den Verbindungsstatus. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.7 OPC UA Server trennen mit Abmeldung

| | |
|---|---|
| *Anwendungsfall* | OPC UA Server trennen mit Abmeldung |
| *Ziel* | Ein realer OPC UA Server wird mit vorheriger Abmeldung beim Discovery-Server vom verteilten IT-System getrennt. |
| *Kategorie* | **primär** |
| *Externe Akteure* | Benutzer |
| *Vorbedingung* | Es muss sich mindestens ein realer OPC UA Server im Netzwerk befinden. |
| *Nachbedingung Erfolg* | Der reale OPC UA Server hat sich erfolgreich beim Discovery Server abgemeldet. Der Benutzer kann anschließend die Produktionseinheit physisch vom verteilten IT-System trennen. |
| *Nachbedingung Fehlschlag* | Die Abmeldung beim Discovery Server ist fehlgeschlagen, eine Status-LED am Einplatinencomputer leuchtet auf. |
| *Auslösendes Ereignis* | Der Benutzer entscheidet einen realen OPC UA Server vom verteilten IT-System zu trennen. |
| *Beschreibung* | **1** Ein realer OPC UA Server wird beim Discovery Server abgemeldet.<br>**2** Der Anwender trennt den realen OPC UA Server physisch vom verteilten IT-System.<br>**3** Am Einplatinencomputer wird der Verbindungsstatus mit einer Status-LED visualisiert. |
| *Erweiterungen* | - |
| *Alternativen* | - |

### 4.3.8 OPC UA Server trennen ohne Abmeldung

| | |
|---|---|
| *Anwendungsfall* | OPC UA Server trennen ohne Abmeldung |
| *Ziel* | Ein realer OPC UA Server wird ohne vorherige Abmeldung beim Discovery-Server vom verteilten IT-System getrennt. |
| *Kategorie* | **sekundär** |
| *Externe Akteure* | Benutzer |
| *Vorbedingung* | Es muss sich mindestens ein realer OPC UA Server im Netzwerk befinden. |
| *Nachbedingung Erfolg* | Der reale OPC UA Server wurde ohne vorherige Abmeldung beim Discovery-Server vom verteilten IT-System getrennt. |
| *Nachbedingung Fehlschlag* | - |
| *Auslösendes Ereignis* | Verschiedene Ereignisse sind vorstellbar: Der Benutzer trennt das Ethernet-Kabel vom OPC UA Server ohne vorherige Abmeldung beim Discovery-Server. Es kommt zu einem Softwarefehler beim Mikrokontroller. Die Energieversorgung vom Einplatinencomputer wird unterbrochen. Ein Netzwerk- bzw. Übertragungsfehler tritt auf. |
| *Beschreibung* | **1** Der reale OPC UA Server wird ohne vorherige Abmeldung beim Discovers-Server vom verteilten IT-System getrennt. |
| *Erweiterungen* | - |
| *Alternativen* | - |

## 4.4 Sequenzdiagramme für die Anwendungsfälle

### 4.4.1 OPC UA Netzwerk visualisieren

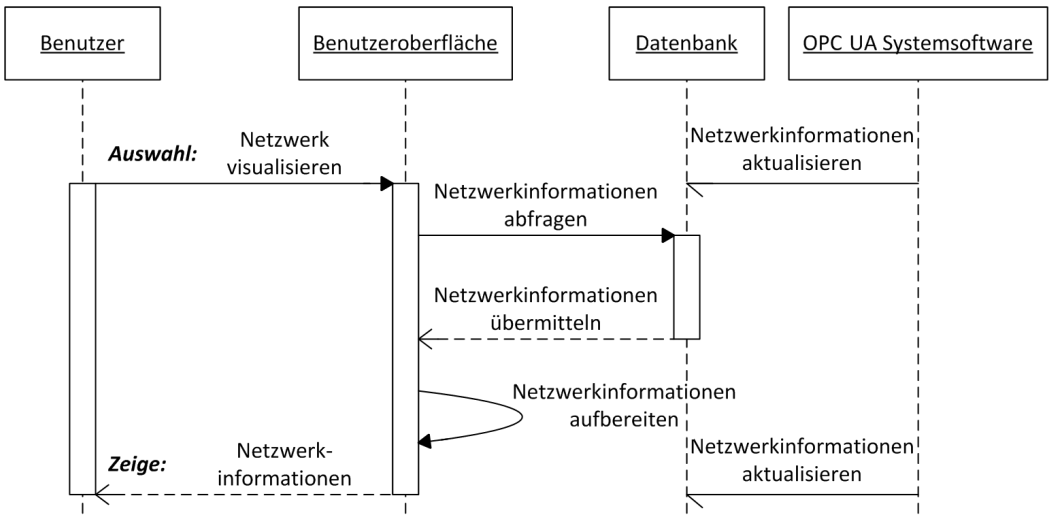


**Abbildung 4.4:** Sequenzdiagramm OPC UA Netzwerk visualisieren

### 4.4.2 Produktionsauftrag erstellen

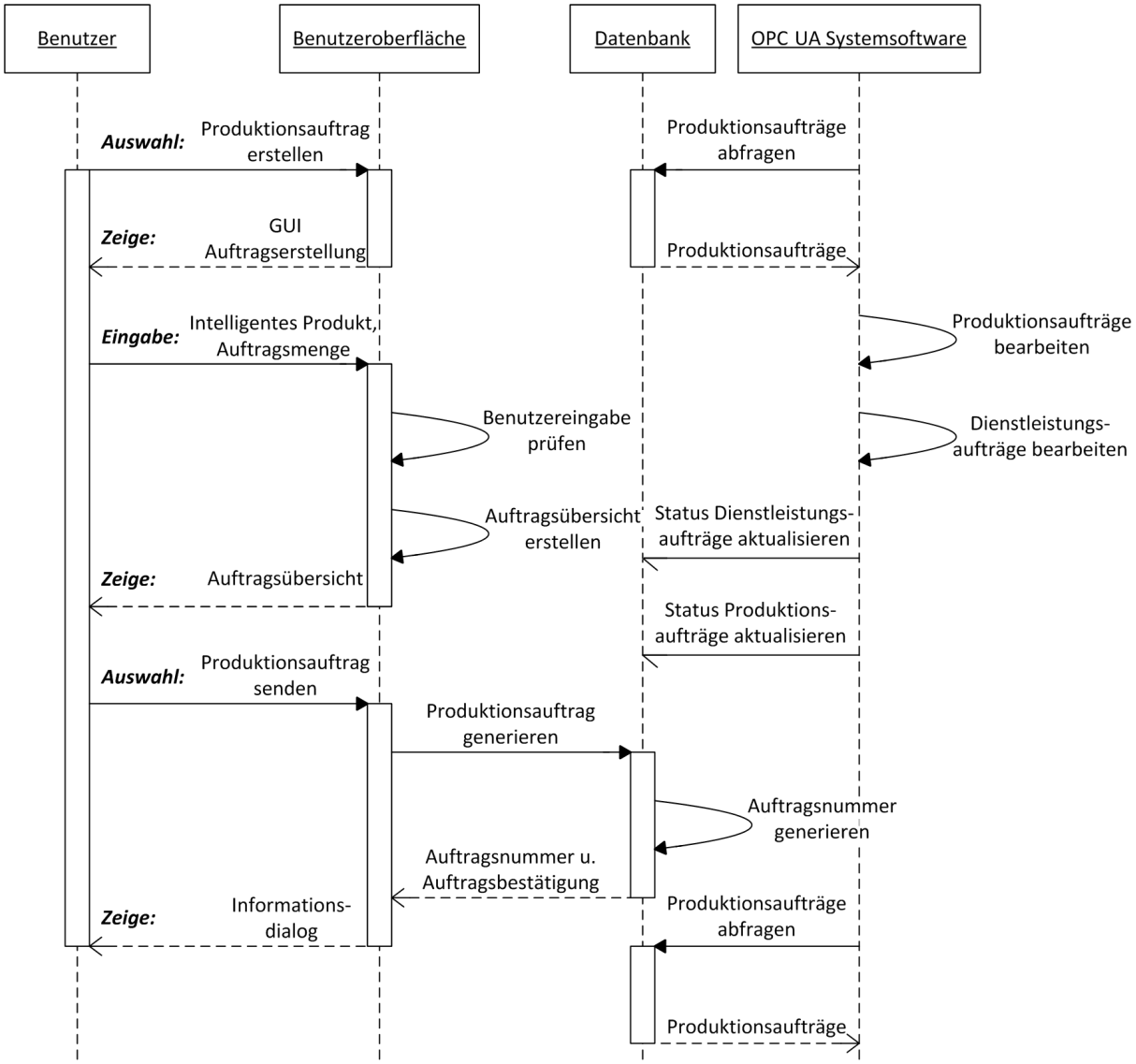


**Abbildung 4.5:** Sequenzdiagramm Produktionsauftrag erstellen

### 4.4.3 Auftragsstatus anzeigen

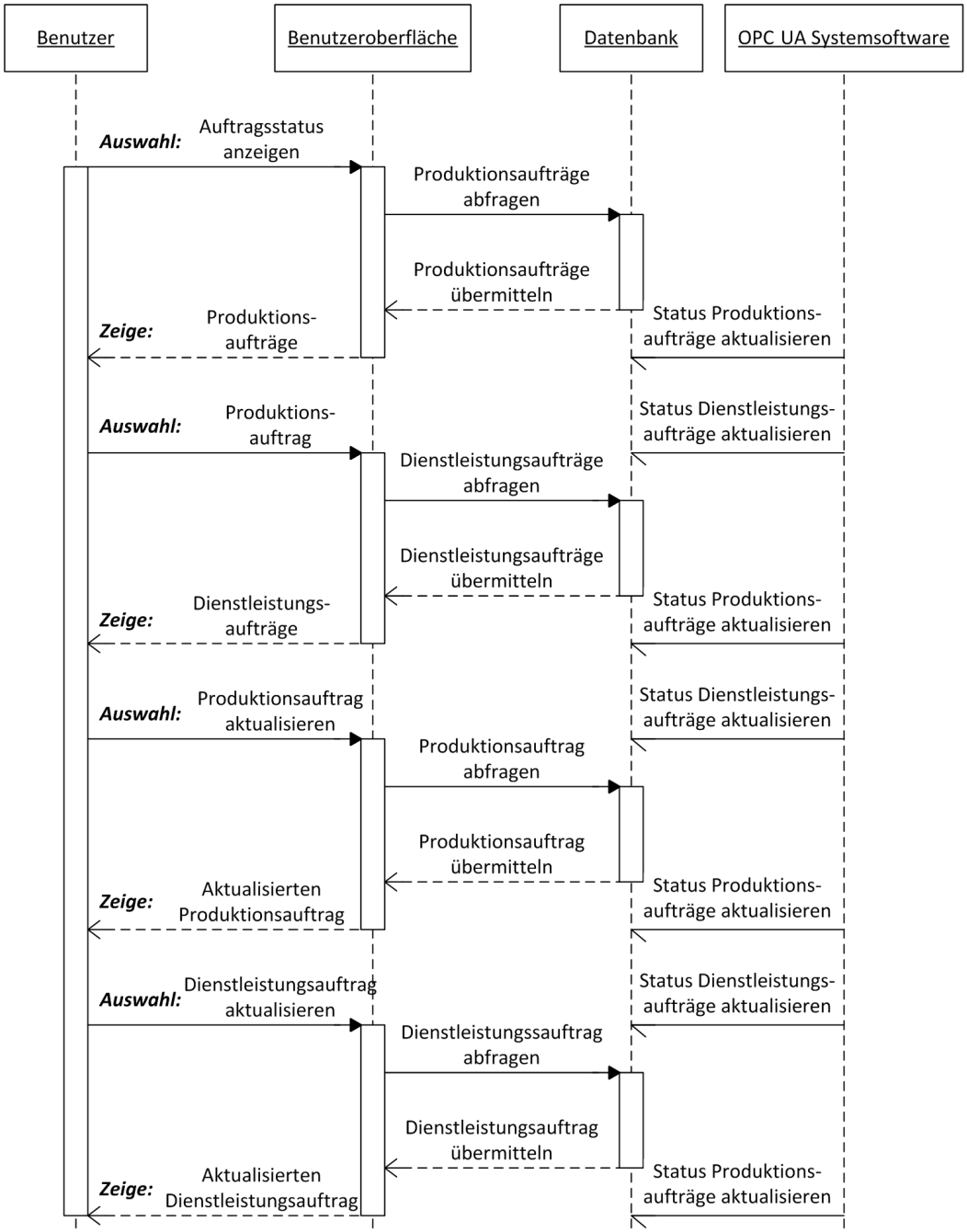


**Abbildung 4.6:** Sequenzdiagramm Auftragsstatus anzeigen

### 4.4.4 Virtuellen OPC UA Server erstellen

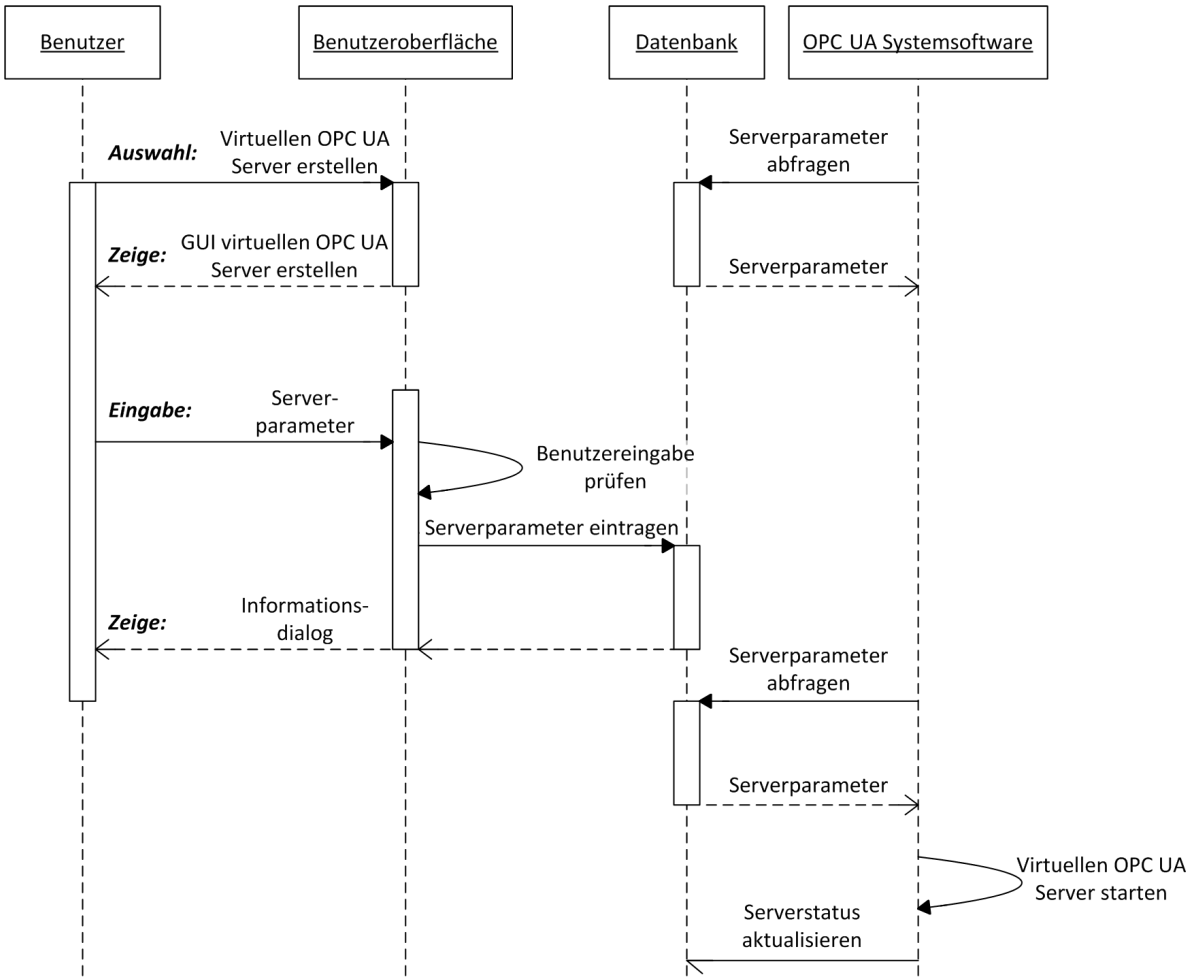


**Abbildung 4.7:** Sequenzdiagramm Virtuellen OPC UA Server erstellen

### 4.4.5 Virtuellen OPC UA Server administrieren

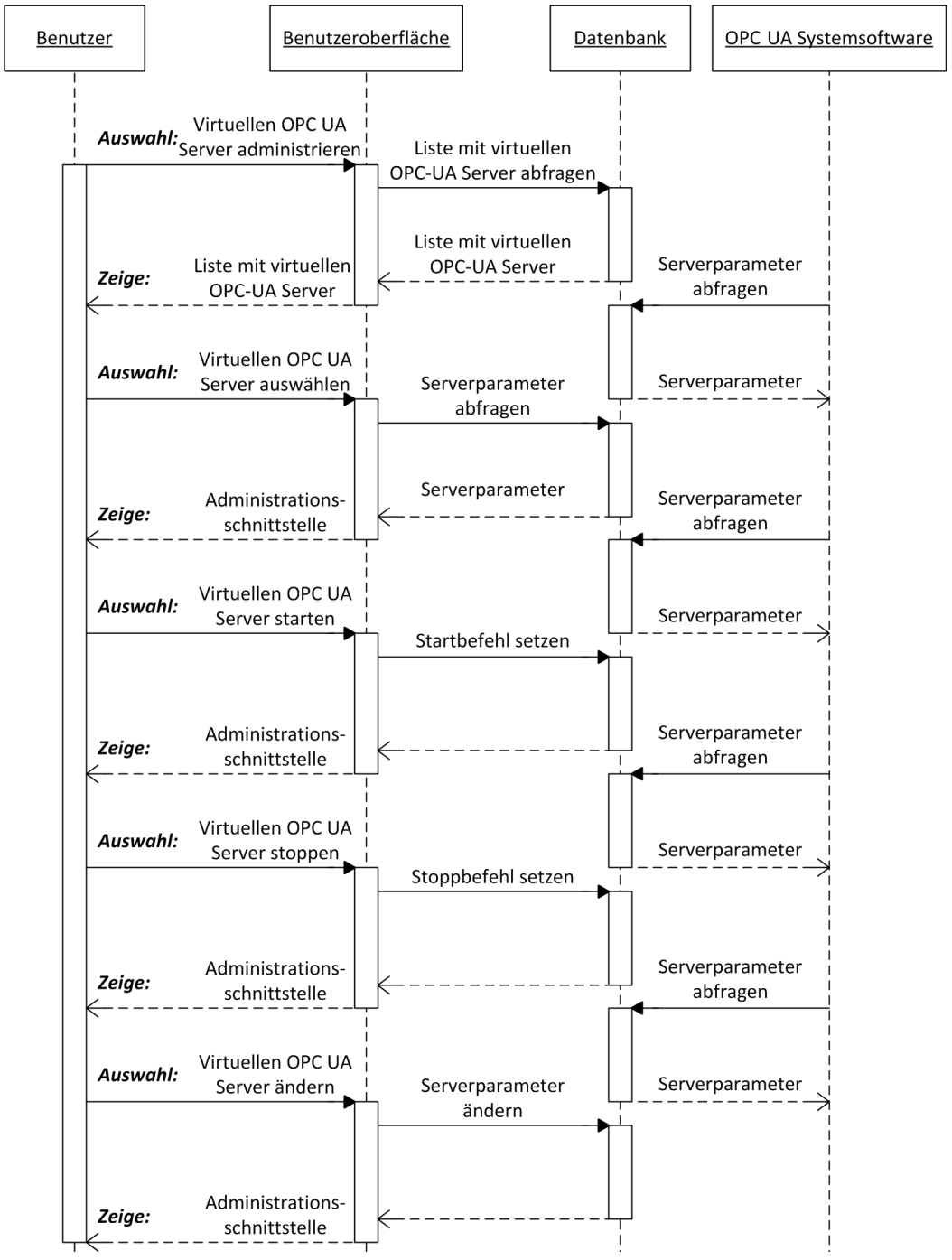


**Abbildung 4.8:** Sequenzdiagramm Virtuellen OPC UA Server administrieren

## 4.5 GUI-Konzept

### 4.5.1 Prinzipielle Bedienkonzepte

Zur Gestaltung der Benutzeroberfläche wurde ein möglichst intuitives und benutzerfreundliches Layout entworfen, welches durchgehend für alle Applikationsfenster verwendet wird. In einem übersichtlichen Menü kann der Benutzer nach dem Programmstart auf die einzelnen Grundfunktionen zugreifen. Je nach Auswahl werden die gewünschten Informationen direkt visualisiert oder ein Untermenü geladen. In allen Applikationsfenstern außerhalb des Menüs befindet sich am linken Rand eine Symbolleiste für den Schnellzugriff. Mit dem Schnellzugriff kann der Benutzer das Menü öffnen oder direkt (mit nur einem Klick) auf die einzelnen Grundfunktionen zugreifen. Tritt in der Anwendung ein Fehler auf, wird der Benutzer mit einem Dialogfenster über den Fehler sowie dessen mögliche Behebung informiert. Zur Bedienung der Benutzeroberfläche wird eine Maus verwendet. Eingaben für Textfelder erfolgen mittels Tastatur.

### 4.5.2 Oberflächen der Benutzerschnittstelle

#### 4.5.2.1 Menü/Startbildschirm

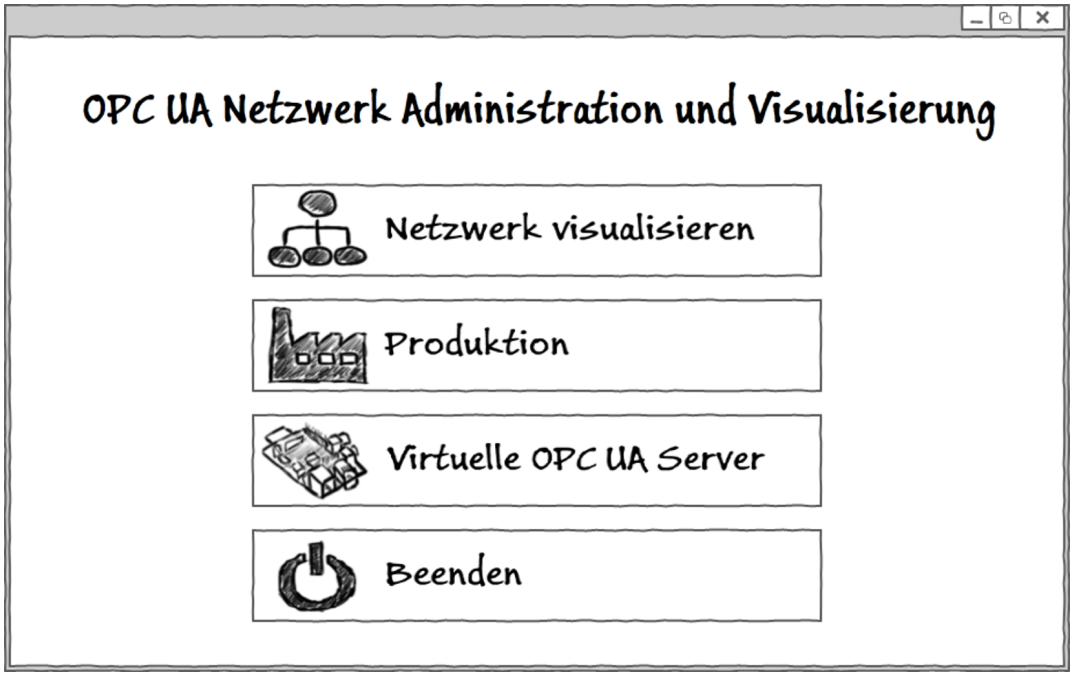


**Abbildung 4.9:** Menü/Startbildschirm

#### 4.5.2.2 Netzwerk visualisieren

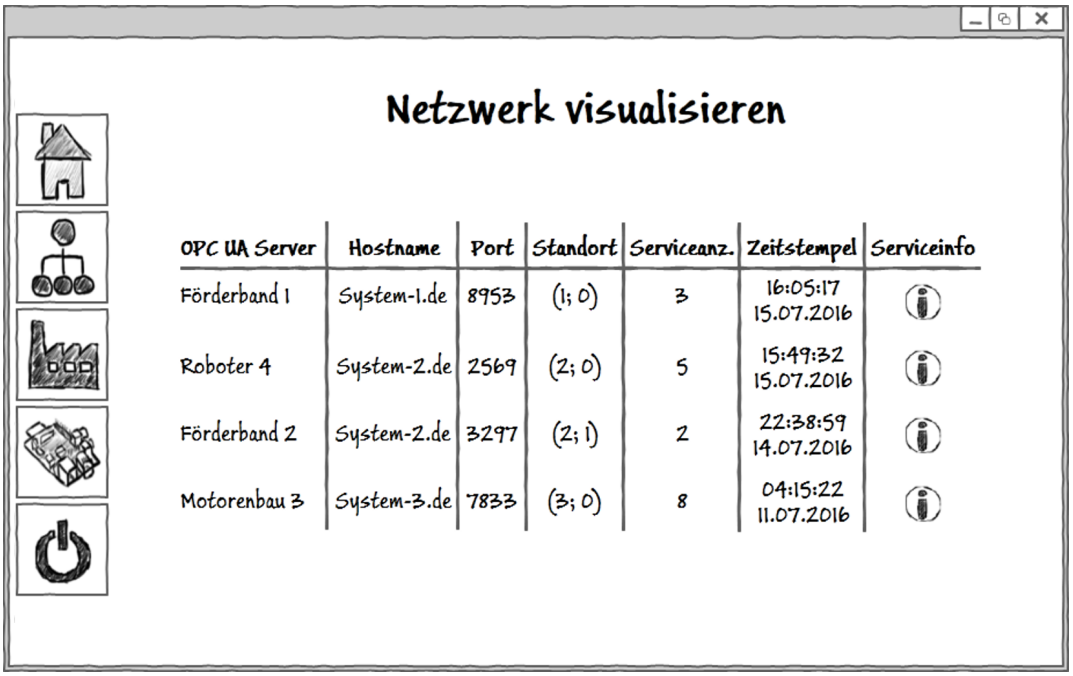


**Abbildung 4.10:** Visualisierung der Netzwerkinformationen

#### 4.5.2.3 Angebotene Dienstleistungen des OPC UA Servers

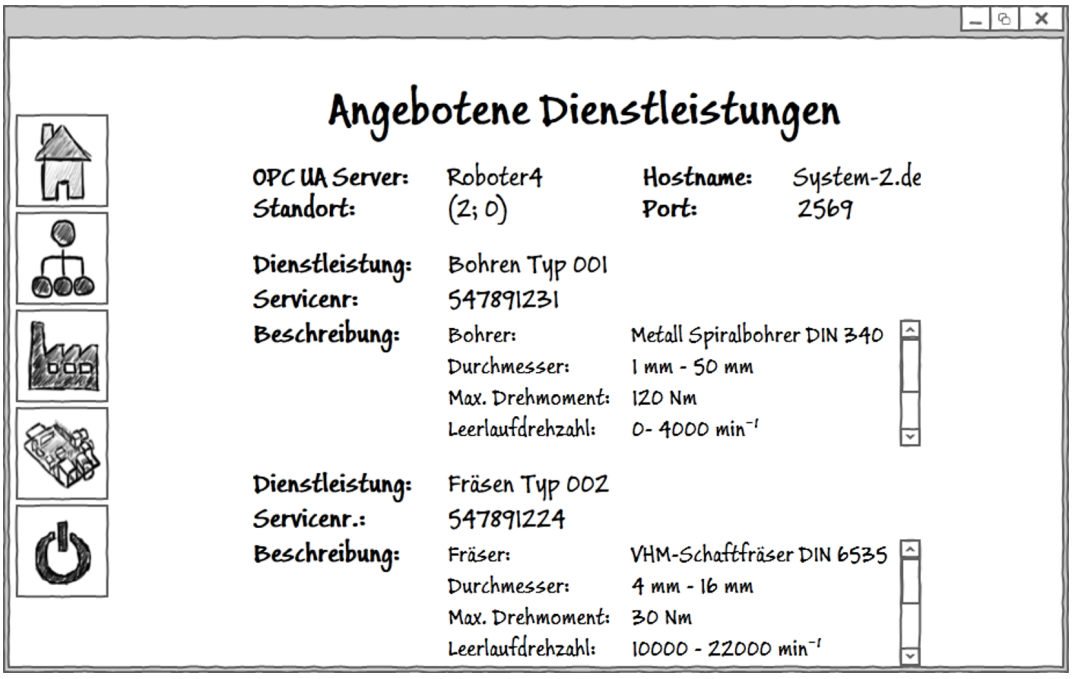


**Abbildung 4.11:** Angebotene Dienstleistungen des OPC UA Servers

#### 4.5.2.4 Produktion (Untermenü)

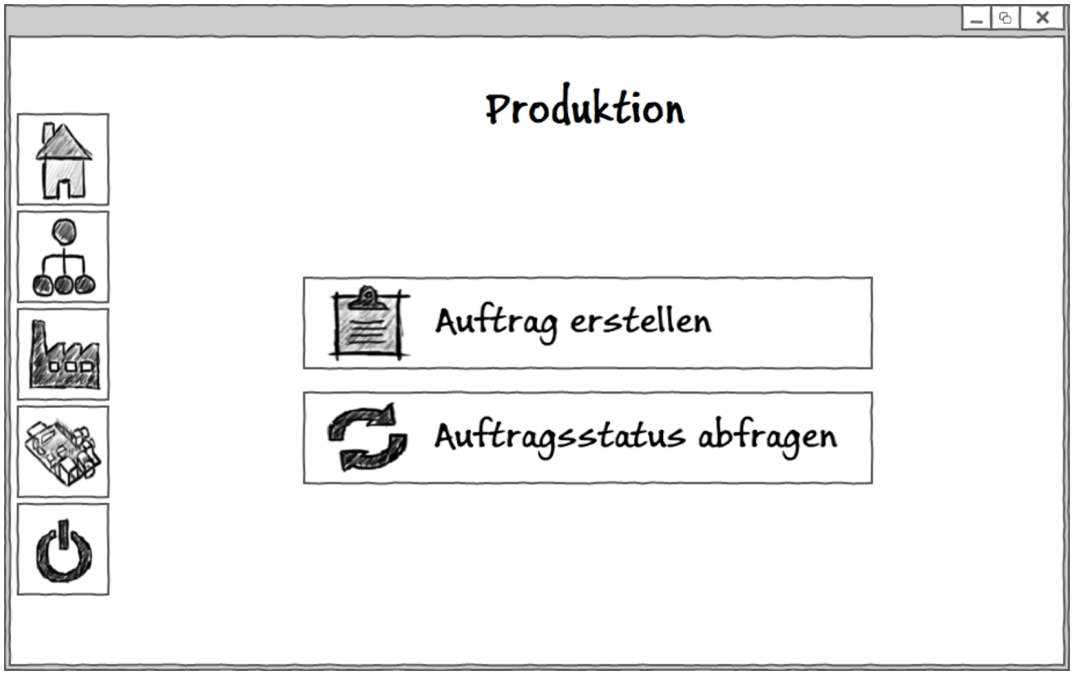


**Abbildung 4.12:** Produktion (Untermenü)

#### 4.5.2.5 Produktionsauftrag erstellen


**Abbildung 4.13:** Produktionsauftrag erstellen

#### 4.5.2.6 Produktionsauftrag Übersicht


**Abbildung 4.14:** Produktionsauftrag Übersicht

#### 4.5.2.7 Statusübersicht der Produktionsaufträge

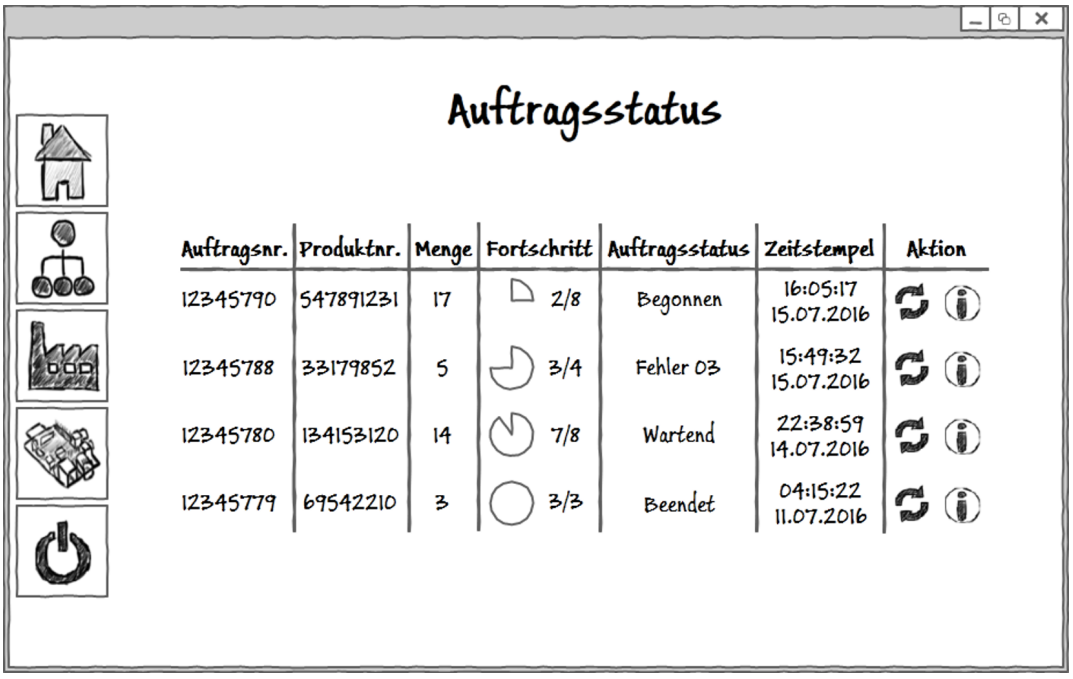


**Abbildung 4.15:** Statusübersicht der Produktionsaufträge

#### 4.5.2.8 Detaillierter Status des Produktionsauftrags

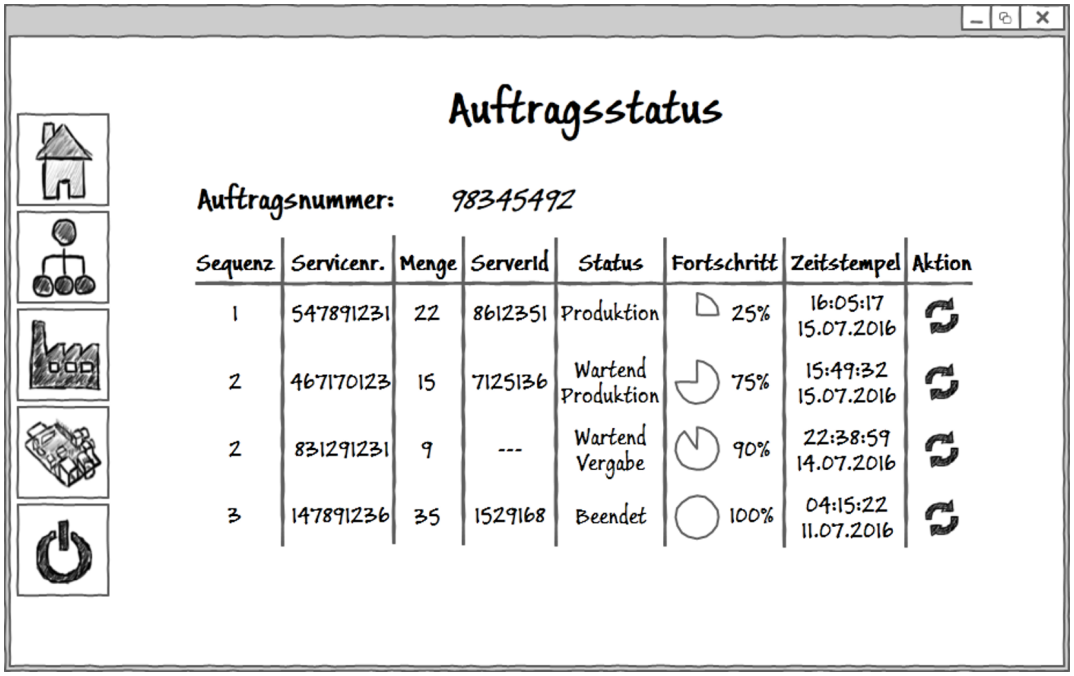


**Abbildung 4.16:** Detaillierter Status des Produktionsauftrags

#### 4.5.2.9 Virtuelle OPC UA Server (Untermenü)

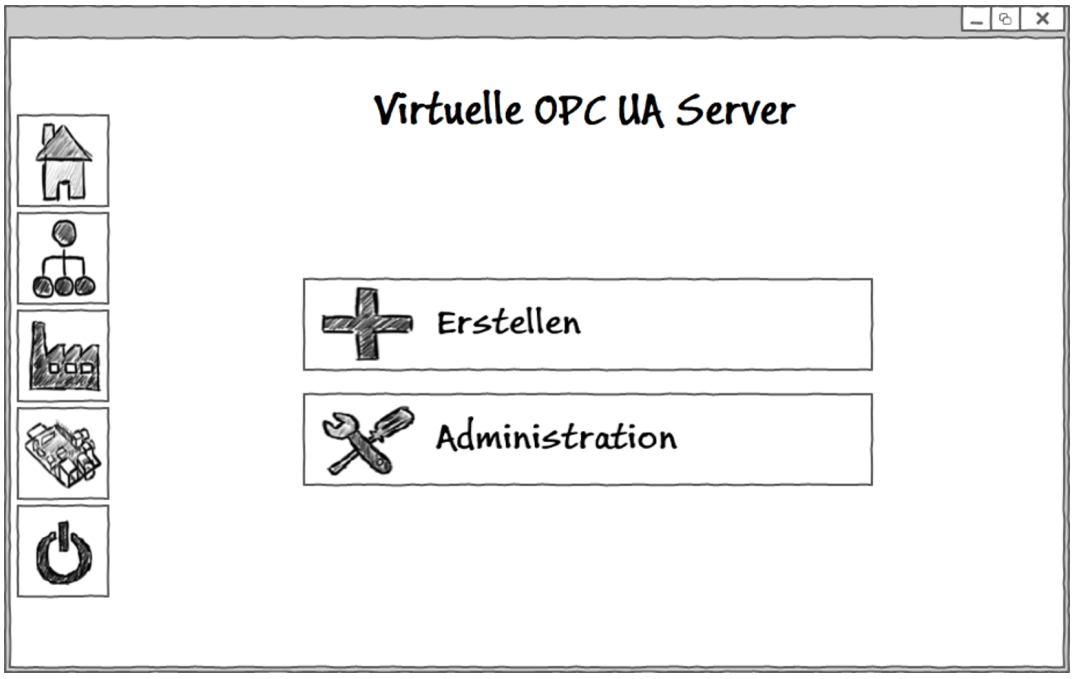


**Abbildung 4.17:** Virtuelle OPC UA Server (Untermenü)

#### 4.5.2.10 Virtuellen OPC UA Server erstellen

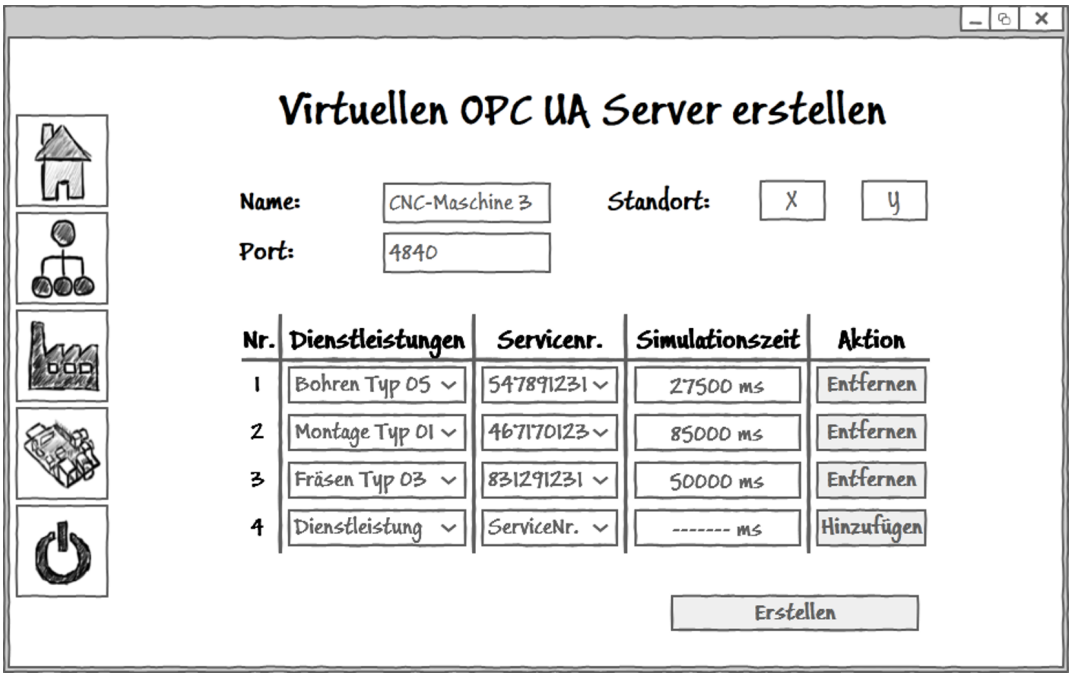


**Abbildung 4.18:** Virtuellen OPC UA Server erstellen

#### 4.5.2.11 Liste mit virtuellen OPC UA Servern (Administration)

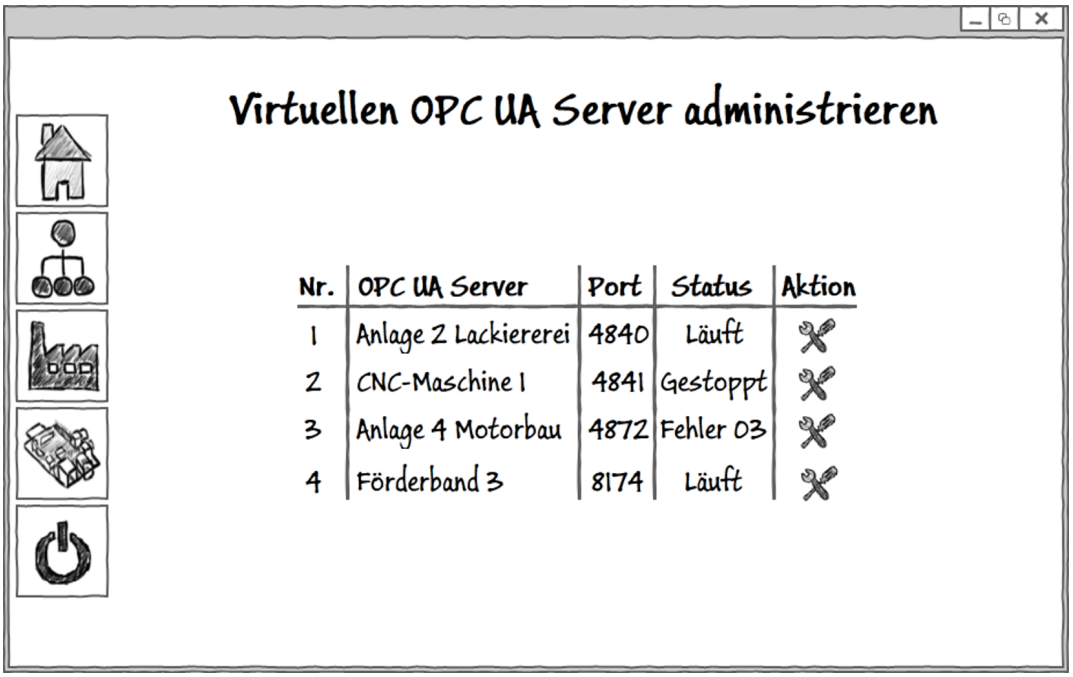


**Abbildung 4.19:** Liste mit den virtuellen OPC UA Servern

#### 4.5.2.12 Administrationsinterface für virtuellen OPC UA Server (Administration)

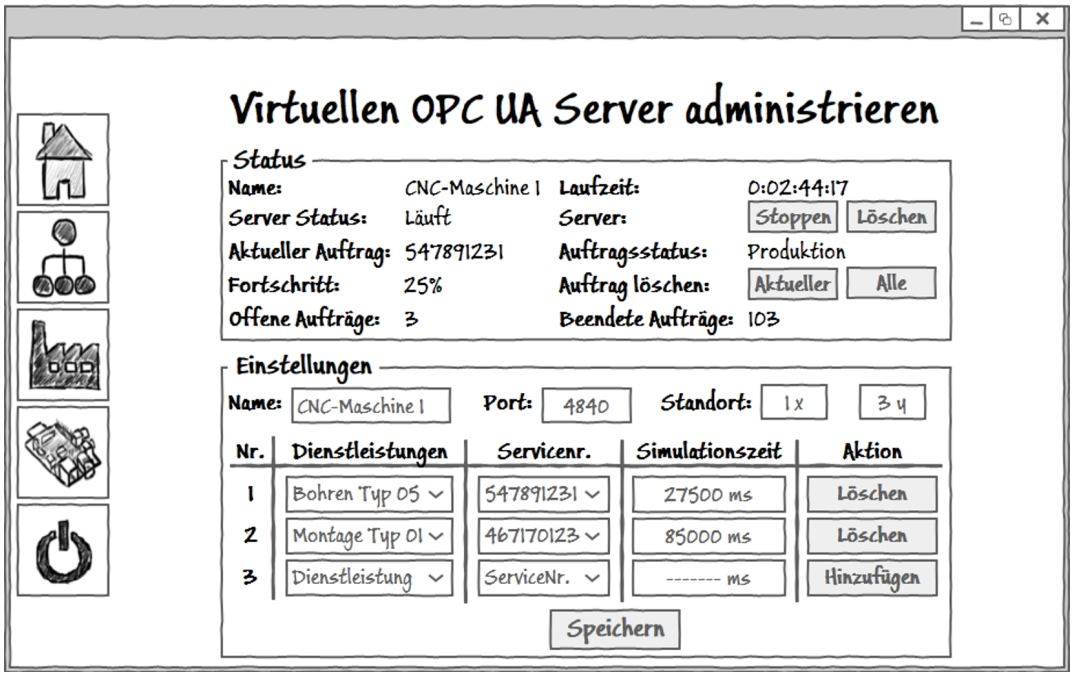


**Abbildung 4.20:** Administrationsinterface für virtuellen OPC UA Server

### 4.5.3 Aufbau der Benutzerschnittstelle

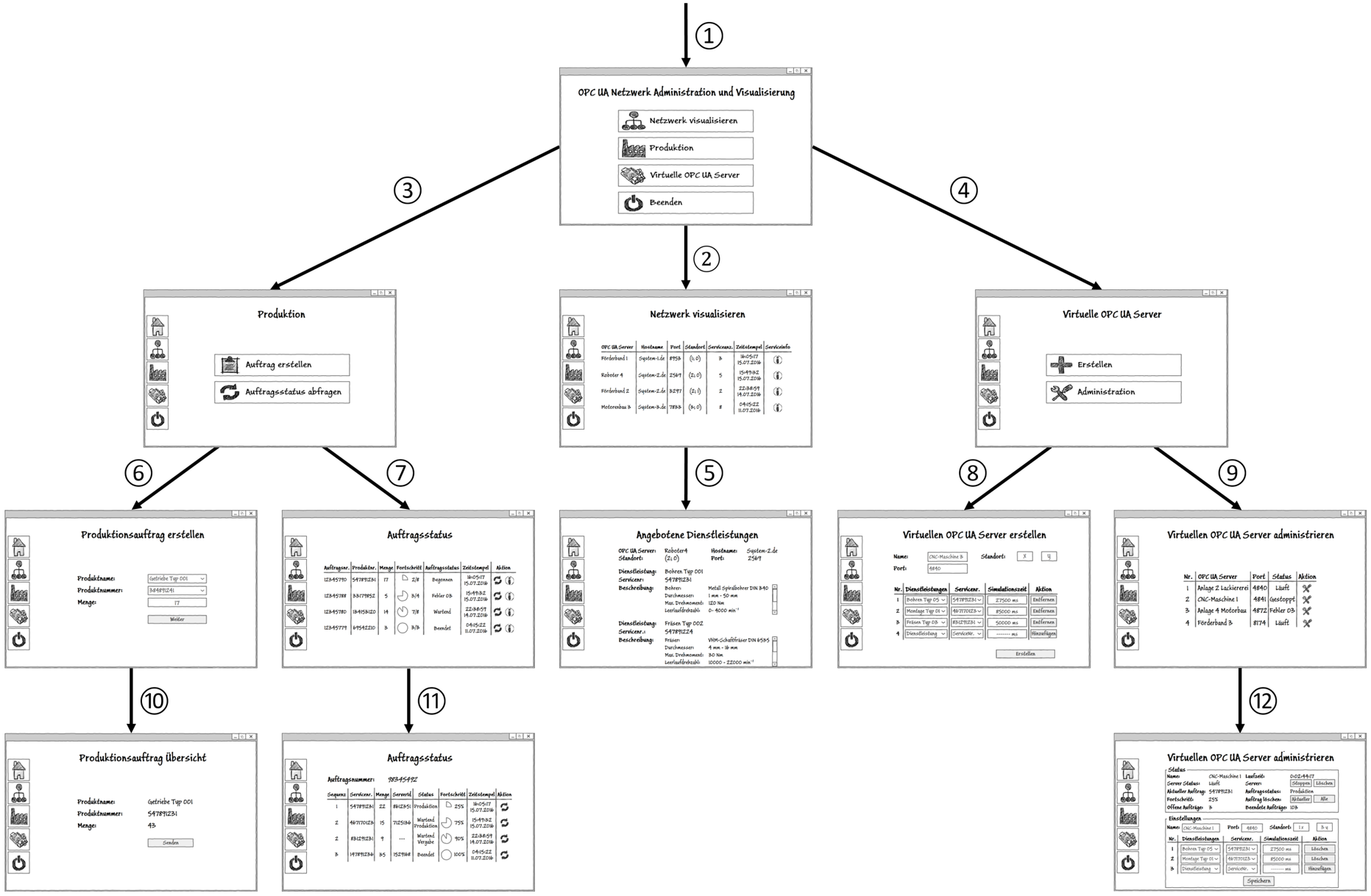


**Abbildung 4.21:** Aufbau und Ablauf der Benutzerschnittstelle

### Ablauf der Benutzerschnittstelle

Der Benutzer startet das Programm zur Visualisierung und Administration des OPC UA Netzwerks. Nach dem Programmstart wird ein übersichtliches *Menü* geladen, in welchem der Anwender auf die einzelnen Grundfunktionen zugreifen kann → ①.

Im *Menü* stehen folgende drei Grundfunktionen zur Auswahl:

- *Netzwerk visualisieren* → ②
- *Produktion* → ③
- *Virtuelle OPC UA Server* → ④

In allen Applikationsfenstern außerhalb des *Menüs* befindet sich am linken Rand eine Symbolleiste für den Schnellzugriff. Mit dem Schnellzugriff kann der Benutzer das *Menü* öffnen → ① oder direkt auf die einzelnen Grundfunktionen zugreifen → ②; → ③; → ④.

Bei der *Visualisierung des Netzwerks* ③ werden die registrierten OPC UA Server mit Name, Adresse, Port und Serviceanzahl aufgelistet. Mit einem Klick auf die *Info*-Schaltfläche können die bereitgestellten Dienstleistungen eines OPC UA Servers mit Informationen angezeigt werden → ⑤.

Die Funktionen *Auftrag erstellen* → ⑥ und *Auftragsstatus abfragen* → ⑦ befinden sich im Untermenü *Produktion* ③. In der Oberfläche zur *Erstellung eines Produktionsauftrags* ⑥ kann der Benutzer einen Auftrag für ein intelligentes Produkt erstellen. Nach der Auftragseingabe gelangt der Anwender mit dem *Weiter*-Button zur Übersicht des zu erstellenden Produktionsauftrags → ⑩. Mit dem *Senden*-Button wird anschließend der Auftrag verbindlich erstellt und eine Auftragsnummer generiert. Die Auftragsnummer wird dem Benutzer mit einem Dialogfenster visualisiert. In der Oberfläche *Auftragsstatus abfragen* ⑦ sind alle Produktionsaufträge mit Auftragsnummer, Produktnummer, Auftragsmenge, aktueller Status, Bearbeitungsschritt und Zeitstempel aufgelistet. Der Zeitstempel definiert den letzten Aktualisierungsvorgang des Produktionsauftrags. Mit dem *Refresh*-Button kann der Anwender den aktuellen Status und Bearbeitungsschritt eines Produktionsauftrags aktualisieren. Die Dienstleistungsaufträge eines Produktionsauftrags können mit dem *Info*-Button aufgerufen werden → ⑪. Zu einem Dienstleistungsauftrag werden folgende Informationen aufgelistet: Sequenznummer, Dienstleistungsnummer, Auftragsmenge, Server-ID, aktueller Status, Bearbeitungsfortschritt und Zeitstempel. Die Sequenznummer definiert die Abarbeitungsreihenfolge der Dienstleistungsaufträge. Besitzen mehrere Dienstleistungsaufträge dieselbe Sequenznummer, können diese parallel abgearbeitet werden. Die Server-ID identifiziert den OPC UA Server, der den Dienstleistungsauftrag bearbeitet. Der Zeitstempel definiert den letzten Aktualisierungsvorgang des Dienstleistungsauftrags. Mit dem *Refresh*-Button kann der Anwender den aktuellen Status und Bearbeitungsschritt eines Dienstleistungsauftrags aktualisieren.

Funktionen zur Erstellung und Administration eines virtuellen OPC UA Servers sind im Untermenü *Virtuelle OPC UA Server* ④ implementiert. Mit dem *Erstellen*-Button gelangt der Benutzer in die Oberfläche zur Erstellung eines virtuellen OPC UA Servers → ⑧. In dieser Übersicht ⑧ vergibt der Benutzer einen Namen, Port sowie Standort und wählt die fiktiven Dienstleistungen aus. Nach der Eingabe kann der virtuelle OPC UA Server mit dem Button *Erstellen* generiert werden. Zur Administration der virtuellen OPC UA Server wählt der Benutzer im Untermenü *Virtuelle OPC UA Server* ④ den Button *Administration* aus → ⑨. Anschließend werden alle virtuellen OPC UA Server mit Name, Port und Status aufgelistet ⑨. Zur Administration wählt der Benutzer mit dem *Schraubenschlüssel*-Button einen Server aus → ⑫. Für den ausgewählten Server wird eine *Administrationsinterface* ⑫ geladen. Im *Administrationsinterface* kann der Benutzer den virtuellen OPC UA Server starten, stoppen, ändern, löschen oder den aktuellen Status abfragen. Folgende Serverparameter sind änderbar: Name, Port, Standort sowie die angebotenen (fiktiven) Dienstleistungen.

## 4.5.4 Dialoggestaltung

### 4.5.4.1 Informationsdialog

Der Informationsdialog wird in folgenden Anwendungsfällen zur Benutzerinteraktion eingesetzt:

- Nach der Erstellung eines Produktionsauftrags erhält der Benutzer eine Auftragsbestätigung und Auftragsnummer.
- Der Anwender wird nach erfolgreicher Generierung eines virtuellen OPC UA Servers informiert.
- Wenn die Einstellungen eines virtuellen OPC UA Servers erfolgreich geändert sind, wird ein Info-Dialog geladen.

In Abbildung 4.22 ist der Entwurf für den Informationsdialog dargestellt.


**Abbildung 4.22:** Entwurf Informationsdialog

#### 4.5.4.2 Dialog für Fehlermeldungen

Tritt in dem oben beschriebenen Ablauf oder Programm ein Fehler auf, wird der Benutzer in einem Dialogfenster informiert. Die Fehlermeldung umfasst den Fehlertyp, eine Beschreibung sowie mögliche Behebungsvorschläge. In Abbildung 4.23 ist der Entwurf für Fehlermeldungen dargestellt.

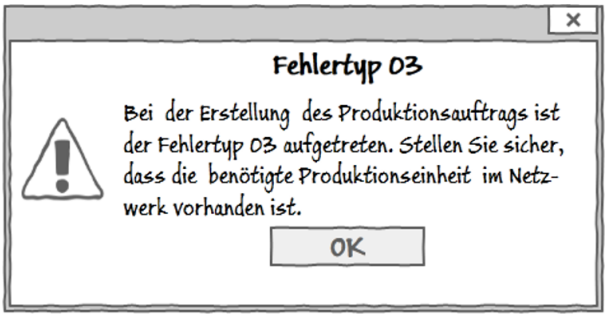


**Abbildung 4.23:** Entwurf Fehlermeldung

# 5 Software-Systemarchitektur

## 5.1 Grundlegende Entwurfsentscheidungen

### 5.1.1 Plattform

Im Rahmen der Masterarbeit wird erstmalig an der Universität Stuttgart ein größeres verteiltes IT-System auf Basis von OPC UA realisiert. Das verteilte IT-System besteht aus einer Workstation sowie mehreren Einplatinencomputern (Raspberry Pi). Die Einplatinencomputer verwenden das Betriebssystem Raspbian Jessie (Linux- Distribution). Auf der Workstation wird das Betriebssystem Microsoft Windows 10 (64-Bit) eingesetzt.

### 5.1.2 Programmiersprache

Für die Implementierung der Applikationen wird aus mehreren Aspekten die Programmiersprache Java gewählt:

- Java ist eine objektorientierte Programmiersprache (OOP). Objektorientierte Programmiersprachen ermöglichen in der Entwurfsphase eine einfache und strukturierte Planung von Applikationen. Auf Grundlage des entworfenen Objektmodells ist anschließend eine schnellere Implementierung der Applikationen möglich. Ein weiterer Vorteil von objektorientierten Programmiersprachen ist die einfache Wartbarkeit, Erweiterbarkeit und Wiederverwendbarkeit des Quellcodes.
- Java-Applikationen sind durch die Java-Laufzeitumgebung plattformunabhängig einsetzbar. Im Rahmen der Masterarbeit werden Applikationen für die Betriebssysteme Microsoft Windows 10 (64-Bit) sowie Raspbian Jessie (Linux- Distribution) entwickelt. Der Quellcode kann durch die Plattformunabhängigkeit applikationsübergreifend verwendet werden.
- Die Programmiersprache Java beinhaltet bereits eine umfangreiche Standardbibliothek. Durch die Verwendung der Standardbibliothek verringert sich der Entwicklungsaufwand. Außerdem wird auf eine erprobte Standardbibliothek zurückgegriffen, wodurch sich potentielle Fehler in der Implementierung vermeiden lassen.

Hinweis: Zur Dokumentation des Quellcodes wird Javadoc eingesetzt.

### 5.1.3 OPC UA Bibliothek

Für die Implementierung der OPC UA Funktionalitäten wird das OPC UA Java Software Development Kit (OPC UA Client und Server) des Herstellers Prosys eingesetzt. In Kapitel 2.3.9 erfolgt die Auswahl der OPC UA Bibliothek.

### 5.1.4 Grafische Benutzeroberfläche

Teil der Masterarbeit ist die Entwicklung einer grafischen Anwendung für die Administration und Visualisierung des OPC UA Netzwerkes. Für die Umsetzung der grafischen Benutzeroberfläche wird die Java-Klassenbibliothek *Swing* eingesetzt. Die Java-Klassenbibliothek *Swing* verwendet als Grundlage das *Abstract Window Toolkit (AWT)* und beinhaltet die folgenden Module:

- **Grafische Komponenten**
  Swing stellt für die Erstellung der Benutzeroberfläche verschiedene grafische Komponenten bereit. Grafische Komponenten sind beispielsweise Fenster, Container, Schaltflächen, Schriftfelder oder auch Menüs.
- **Layoutmanager**
  Aufgabe der Layoutmanager ist die Ausrichtung und Skalierung der grafischen Komponenten auf der Benutzeroberfläche. Swing beinhaltet acht unterschiedliche Layoutmanager.
- **Ereignisbehandlung**
  Grafische Komponenten agieren als Ereignisquelle und generieren bei Benutzerinteraktionen Ereignisse. Damit ein Ereignisempfänger (Event-Listener) beim Auslösen von Ereignissen aufgerufen wird, muss sich dieser bei der Ereignisquelle registrieren.

### 5.1.5 Datenhaltung

Die Datenhaltung für das verteilte IT-System erfolgt in dem relationalen Datenbanksystem *MySQL* (Version 5.7.15). Das relationale Datenbanksystem *MySQL* ermöglicht die einfache, schnelle und sichere Verwaltung von Daten in Tabellen. Aus zwei Gründen wird für die Datenhaltung *MySQL* gewählt: *MySQL* ist eines der verbreitetsten Datenbanksysteme in der Industrie. Zum anderen ist die Nutzung von *MySQL* unter der GPL-Lizenz kostenfrei. Der Zugriff auf die Datenbank erfolgt mittels JDBC (Java Database Connectivity). JDBC ist eine universelle Datenbankschnittstelle für die Java-Plattform.

Das Datenbankschema für die Datenhaltung ist in Kapitel 5.5 definiert.

## 5.2 Diagramme der Software-Systemarchitektur

Für die Realisierung des verteilten IT-Systems werden drei Applikationen entwickelt:

- Grafische Anwendung zur Administration und Visualisierung des OPC UA Netzwerks
- OPC UA Systemsoftware
- Mikrocontroller-Programme

In Abbildung 5.1 ist die Software-Systemarchitektur für das verteilte IT-System dargestellt.

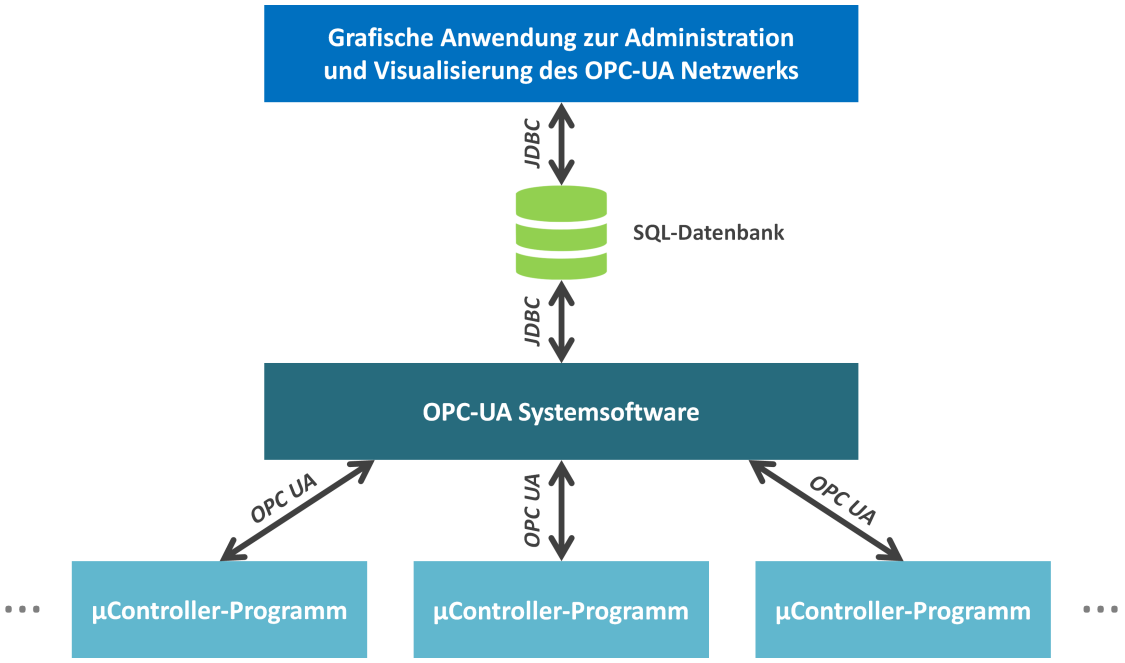


**Abbildung 5.1:** Software-Systemarchitektur für das verteilte IT-System

Im weiteren Teil werden die Applikationen beschrieben und in geeignete Module untergliedert.

### 5.2.1 Software-Systemarchitektur der grafischen Anwendung

Die grafische Anwendung ermöglicht dem Benutzer die Administration und Visualisierung des OPC UA Netzwerks. Aufbau und Ablauf der grafischen Benutzeroberfläche wird im Systemmodell (Kapitel 4.5) beschrieben.

Abbildung 5.2 visualisiert die Software-Systemarchitektur der grafischen Anwendung. Für die Umsetzung der grafischen Anwendung wird die Drei-Schichten-Architektur eingesetzt. Jede Schicht wird in einem separaten Modul realisiert.

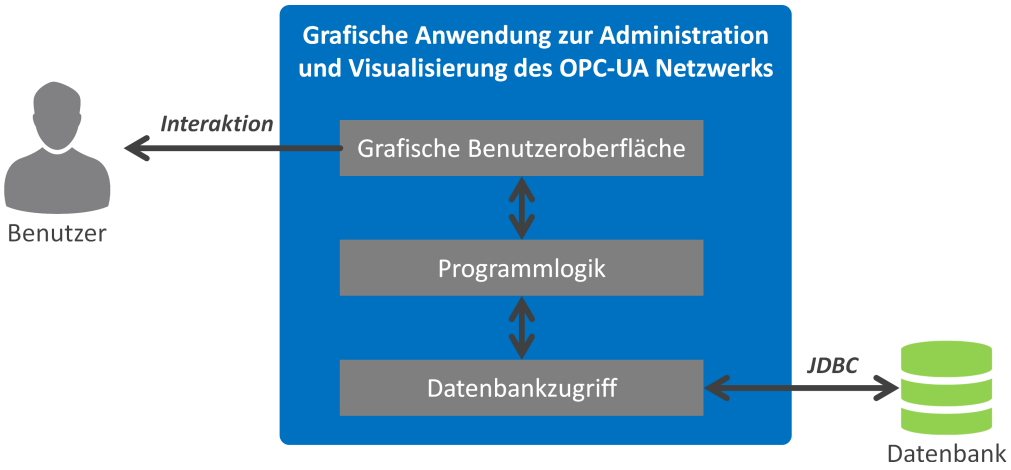


**Abbildung 5.2:** Software-Systemarchitektur der grafischen Anwendung

**Grafische Benutzeroberfläche**
Auf der oberen Ebene der Drei-Schichten-Architektur wird die grafische Benutzeroberfläche implementiert. Hauptaufgaben der Ebene sind die Visualisierung der grafischen Elemente und Daten sowie die Interaktionen mit dem Anwender.

**Programmlogik**
Die Programmlogik-Schicht befindet sich auf der mittleren Ebene der Drei-Schichten-Architektur. Auf dieser Ebene erfolgt die Verarbeitung der Benutzereingaben (Interaktionen) und die Ablaufsteuerung der grafischen Oberfläche.

**Datenbankzugriff**
Auf der unteren Ebene der Drei-Schichten-Architektur ist die Datenbankzugriffsschicht angesiedelt. Die Datenbankzugriffsschicht stellt mehrere Methoden zur Verfügung, mit der die Programmlogik auf die Datenbank des verteilten IT-Systems zugreift. Dadurch sind alle Anfragen an die Datenbank gekapselt.

### 5.2.2 Software-Systemarchitektur der OPC UA Systemsoftware

In Abbildung 5.3 ist die Software-Systemarchitektur der OPC UA Systemsoftware dargestellt.

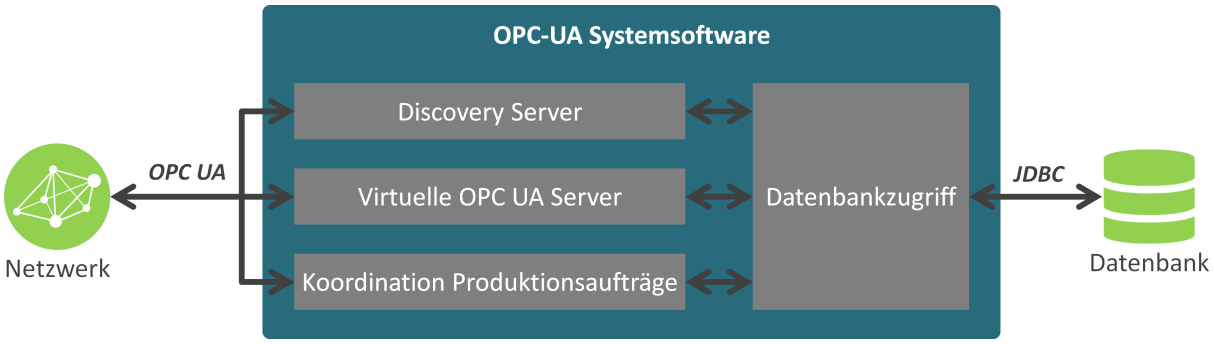


**Abbildung 5.3:** Software-Systemarchitektur der OPC UA Systemsoftware

Die OPC UA Systemsoftware besteht aus folgenden vier Modulen:

**Discovery Server**
Im *Discovery Server* Modul ist der Discovery Server für das verteilte IT-System implementiert. Der Discovery Server basiert auf einem Standard OPC UA Server, der im Adressraum mehrere Methoden für den Discovery Prozess bereitstellt. Um die Methoden auf dem Discovery Server aufzurufen, verwenden die Produktionseinheiten das *Discovery Client* Modul. In Kapitel 5.4.1 ist der Discovery Prozess beschrieben.
Der Discovery Server speichert die Daten der registrierten Produktionseinheiten in der zentralen Datenbank des verteilten IT-Systems (vgl. *5.5 Datenbankschema*). Für den Datenbankzugriff verwendet der Discovery Server das *Datenbankzugriffsmodul*.

**Virtuelle OPC UA Server**
Virtuelle OPC UA Server sind simulierte Produktionseinheiten, die fiktive Dienstleistungen (Services) im OPC UA Netzwerk anbieten. In diesem Modul werden die virtuellen OPC UA Server erstellt, initialisiert und verwaltet.
Die Parameter der virtuellen OPC UA Server sind in der Datenbank abgespeichert (vgl. *5.5 Datenbankschema*). Typische Parameter eines virtuellen OPC UA Server sind beispielsweise Servername, Port, Status, angebotene (fiktive) Dienstleistungen (Services) oder die Auftragsnummer vom aktuell bearbeiteten Dienstleistungsauftrag.
Mit der grafischen Anwendung kann der Benutzer in der Datenbank einen virtuellen OPC UA Server hinzufügen oder löschen sowie die Parameter eines vorhandenen virtuellen OPC UA Server anpassen. Die Änderungen in der Datenbank werden anschließend vom *virtuellen OPC UA Server* Modul übernommen.

**Koordination Produktionsaufträge**
In der zentralen Datenbank sind die Produktions- und Dienstleistungsaufträge für das verteilte IT-System gespeichert. Produktionsaufträge bestehen aus einem oder mehreren Dienstleistungsaufträgen, deren Abarbeitungsreihenfolge über die Sequenznummer definiert wird. Besitzen mehrere Dienstleistungsaufträge dieselbe Sequenznummer, können diese parallel abgearbeitet werden. Der Algorithmus für die Koordination der Produktions- und Dienstleistungsaufträge beinhaltet sechs Schritte:

1. **Dienstleistungsaufträge abfragen**
   Die Produktions- und Dienstleistungsaufträge für das verteilte IT-System sind in der Datenbank abgelegt (vgl. *5.5 Datenbankschema*). Im ersten Schritt werden die wartenden Dienstleistungsaufträge aus der Datenbank abgefragt.

2. **Dienstleistungsaufträge analysieren**
   Im zweiten Schritt analysiert die OPC UA Systemsoftware, welche registrierten Produktionseinheiten die benötigten Dienstleistungen bereitstellen. Der Discovery Server gibt Auskunft darüber, welche Dienstleistungen die registrierten Produktionseinheiten anbieten.

3. **Produktionseinheiten ermitteln**
   Vor der Vergabe der Dienstleistungsaufträge werden im dritten Schritt die Produktionseinheiten mit den geringsten Auslastungen ermittelt. Es werden dazu die Auftragslisten der Produktionseinheiten (OPC UA Server) abgefragt und auf Basis der Auftragslisten die Auslastung berechnet. Für die Abfrage der Auftragslisten verwendet die OPC UA Systemsoftware den Produktionsclient (OPC UA Client).

4. **Dienstleistungsaufträge vergeben**
   Im vierten Schritt werden die wartenden Dienstleistungsaufträge an Produktionseinheiten mit den geringsten Auslastungen vergeben. Für die Vergabe der Dienstleistungsaufträge verwendet die OPC UA Systemsoftware den Produktionsclient (OPC UA Client).

5. **Status der Dienstleistungsaufträge aktualisieren**
   Nach der Vergabe der Dienstleistungsaufträge wird in zyklischen Abständen der aktuelle Status und Bearbeitungsfortschritt von den Produktionseinheiten abgefragt. Die OPC UA Systemsoftware aktualisiert anschließend in der Datenbank den Status und Bearbeitungsfortschritt der Dienstleistungsaufträge.

6. **Status der Produktionsaufträge aktualisieren**
   Die OPC UA Systemsoftware aktualisiert in der Datenbank den aktuellen Status und Bearbeitungsschritt der Produktionsaufträge.

**Datenbankzugriff**
Das *Datenbankzugriffsmodul* stellt mehrere Methoden zur Verfügung, mit der die einzelnen Module der OPC UA Systemsoftware auf die Datenbank des verteilten IT-Systems zugreifen. Alle Anfragen an die Datenbank sind dadurch gekapselt.

#### 5.2.2.1 Software-Systemarchitektur der Mikrocontroller-Programme

Das OPC UA Netzwerk besteht aus mehreren realen sowie virtuellen Produktionseinheiten. Aufgabe der Mikrocontroller-Programme ist die Steuerung der realen Produktionseinheiten. Ausgeführt werden die Mikrocontroller-Programme auf den Einplatinencomputer (Raspberry Pi) des Demonstrators. In Abbildung 5.4 ist die Software-Systemarchitektur der Mikrocontroller-Programme dargestellt.

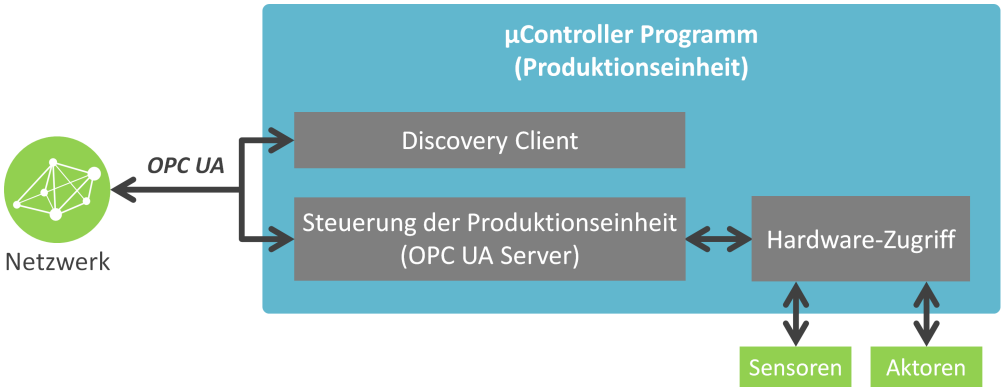


**Abbildung 5.4:** Software-Systemarchitektur der Mikrocontroller-Programme

**Discovery Client**
Aufgabe des *Discovery Client* Moduls ist die Verwaltung der Produktionseinheit beim Discovery Server. Das *Discovery Client* Modul verwendet eine OPC UA Client um die Methoden auf dem Discovery Server aufzurufen. In Kapitel 5.4.1 ist der Discovery Prozess beschrieben.

**Steuerung der Produktionseinheit (OPC UA Server)**
Das *Steuerungsmodul der Produktionseinheit* ist in zwei Teile untergliedert:

- **OPC UA Server**
  Für den Zugriff auf die Produktionseinheit stellt der OPC UA Server mehrere Methoden im Adressraum zur Verfügung. Mit diesen Methoden kann beispielsweise die OPC UA Systemsoftware einen Dienstleistungsauftrag an die Produktionseinheit vergeben oder den aktuellen Status und Bearbeitungsfortschritt eines Dienstleistungsauftrags abfragen. In Kapitel 5.4.2 sind die bereitgestellten Methoden des OPC UA Servers beschrieben. Zusätzlich werden im Adressraum des OPC UA Servers die Sensoren und Aktoren der Produktionseinheit abgebildet.

- **Auftragsbearbeitung**
  Die zugeteilten Dienstleistungsaufträge der Produktionseinheit werden im Steuerungsmodul sequentiell abgearbeitet. Das Steuerungsmodul verwendet zur Erstellung der Dienstleistungen die Sensoren und Aktoren der Produktionseinheit. Der Zugriff auf die Hardware erfolgt im *Hardwarezugriffsmodul*.

**Hardware-Zugriff**
Im *Hardwarezugriffsmodul* erfolgt die Ansteuerung der Hardwarekomponenten (Sensoren und Aktoren). Im Rahmen einer parallel laufenden Forschungsarbeit wird dieses Zugriffsmodul spezifiziert und implementiert.

## 5.3 Softwarekomponenten

### 5.3.1 Softwarekomponenten der grafischen Anwendung

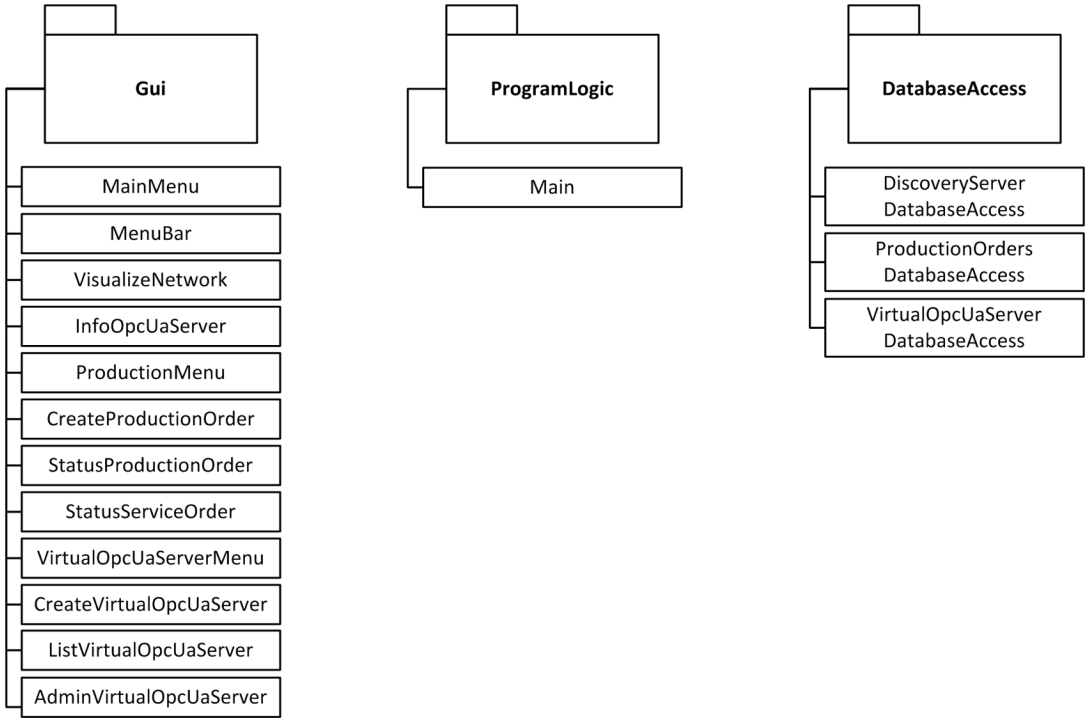


**Abbildung 5.5:** Softwarekomponenten der grafischen Anwendung

#### 5.3.1.1 Grafische Benutzeroberfläche (package *Gui*)

**MainMenu**
In der *MainMenu*-Klasse ist das Hauptmenü der grafischen Benutzeroberfläche implementiert. Der Anwender kann mit der Schaltfläche *Netzwerk visualisieren* Informationen zu den registrierten Produktionseinheiten (OPC UA Server) abfragen. Die Visualisierung der Informationen erfolgt in der *VisualizeNetwork*-Klasse. Mit der Schaltfläche *Produktion* gelangt der Benutzer in das Produktionsmenü. In der *ProductionMenu*-Klasse ist die grafische Oberfläche für das Produktionsmenü implementiert. Die Schaltfläche *Virtuelle OPC UA Server* visualisiert das

Untermenü zur Erstellung und Administration der virtuellen OPC UA Server. Implementiert ist das Untermenü in der *VirtualOpcUaServerMenu*-Klasse. Der Anwender kann die grafische Applikation mit der *Beenden*-Schaltfläche schließen.

**MenuBar**
Die Klasse *MenuBar* initialisiert und verwaltet die Symbolleiste der grafischen Benutzeroberfläche. In allen Applikationsfenstern außerhalb des Menüs befindet sich am linken Rand die Symbolleiste für den Schnellzugriff. Mit dem Schnellzugriff kann der Benutzer auf die Funktionalitäten des Hauptmenüs zugreifen (vgl. *MainMenu*).

**VisualizeNetwork**
Die *VisualizeNetwork*-Klasse visualisiert in einer Tabelle die Informationen zu den registrierten Produktionseinheiten (OPC UA Server). Aus der zentralen Datenbank des verteilten IT-Systems werden die Informationen zu den registrierten Produktionseinheiten mithilfe der *DiscoveryServerDatabaseAccess*-Klasse ausgelesen. Mit der *Serviceinfo*-Schaltfläche kann der Benutzer die angebotenen Dienstleistungen (Services) und Serviceinformationen zu einer registrierten Produktionseinheit abfragen. Für die Abfrage wird die grafische Oberfläche der *InfoOpcUaServer*-Klasse geladen.

**InfoOpcUaServer**
Die *InfoOpcUaServer*-Klasse visualisiert für eine ausgewählte Produktionseinheit (OPC UA Server) die angebotenen Dienstleistungen mit Informationen. In der zentralen Datenbank des verteilten IT-Systems sind die angebotenen Dienstleistungen (Services) der Produktionseinheit und die Dienstleistungsbeschreibungen abgespeichert.

**ProductionMenu**
In der *ProductionMenu*-Klasse ist das Untermenü zur Auftragserstellung und Auftragsstatusabfrage implementiert. Mit der Schaltfläche *Auftrag erstellen* gelangt der Benutzer in die grafische Oberfläche zur Erstellung eines Produktionsauftrags. Implementiert ist diese in der *CreateProductionOrder*-Klasse. Die Schaltfläche *Auftragsstatus abfragen* visualisiert in einer Tabelle für die generierten Produktionsaufträge den aktuellen Status und Bearbeitungsschritt. In der *StatusProductionOrder*-Klasse ist die grafische Oberfläche zur Visualisierung der Produktionsaufträge implementiert.

**CreateProductionOrder**
In der *CreateProductionOrder*-Klasse ist die Benutzeroberfläche zur Erstellung eines Produktionsauftrags implementiert. Die Informationen zu den angebotenen (intelligenten) Produkten werden aus der zentralen Datenbank mithilfe der *ProductionOrdersDatabaseAccess*-Klasse ausgelesen und in der Benutzeroberfläche visualisiert. Anschließend kann der Anwender ein (intelligentes) Produkt aus der Dropdown-Liste auswählen, die gewünschte Auftragsmenge im Textfeld festlegen und den Auftrag erstellen. Der erstellte Produktionsauftrag wird anschließend in die zentrale Datenbank des verteilten IT-System, mit der *ProductionOrdersDatabaseAccess*-Klasse, abgelegt.

**StatusProductionOrder**
Die *StatusProductionOrder*-Klasse visualisiert in einer Tabelle die Auftragsinformationen zu den generierten Produktionsaufträgen. Aus der zentralen Datenbank des verteilten IT-Systems werden die Informationen zu den generierten Produktionsaufträgen mithilfe der *ProductionOrdersDatabaseAccess*-Klasse ausgelesen. Mit der *Refresh*-Schaltfläche kann der Anwender den aktuellen Status und Bearbeitungsschritt zu einem Produktionsauftrag aktualisieren. Die Dienstleistungsaufträge eines Produktionsauftrags können mit der *Info*-Schaltfläche aufgerufen werden. In der *StatusServiceOrder*-Klasse ist die grafische Oberfläche zur Visualisierung der Dienstleistungsaufträge implementiert.

**StatusServiceOrder**
Die *StatusServiceOrder*-Klasse visualisiert in einer Tabelle die Dienstleistungsaufträge eines Produktionsauftrags. Aus der zentralen Datenbank des verteilten IT-Systems werden die Informationen zu den Dienstleistungsaufträgen mithilfe der *ProductionOrdersDatabaseAccess*-Klasse ausgelesen. Mit der *Refresh*-Schaltfläche kann der Anwender den aktuellen Status und Bearbeitungsfortschritt zu einem Dienstleistungsauftrag aktualisieren.

**VirtualOpcUaServerMenu**
In der *VirtualOpcUaServerMenu*-Klasse ist das Untermenü zur Erstellung und Administration der virtuellen OPC UA Server implementiert. Mit der Schaltfläche *Erstellen* gelangt der Benutzer in die grafische Oberfläche zur Erstellung eines neuen virtuellen OPC UA Servers. Implementiert ist die grafische Oberfläche in der *CreateVirtualOpcUaServer*-Klasse. Die grafische Oberfläche zur Administration der virtuellen OPC UA Server kann mit der Schaltfläche *Administration* aufgerufen werden. In der *ListVirtualOpcUaServer*-Klasse ist die grafische Oberfläche implementiert.

**CreateVirtualOpcUaServer**
In der *CreateVirtualOpcUaServer*-Klasse ist die Benutzeroberfläche zur Erstellung eines virtuellen OPC UA Servers implementiert. In der grafischen Benutzeroberfläche definiert der Anwender die folgenden Parameter für den zu erstellenden virtuellen OPC UA Server: Servername, Portnummer, Standort sowie die angebotenen (fiktiven) Dienstleistungen. Mit der *Erstellen*-Schaltfläche kann der Anwender den virtuellen OPC UA Server erstellen. Der erstellte virtuelle OPC UA Server wird anschließend in die zentrale Datenbank mit der *VirtualOpcUaServerDatabaseAccess*-Klasse abgelegt.

**ListVirtualOpcUaServer**
Die *ListVirtualOpcUaServer*-Klasse visualisiert in einer Tabelle zu jedem virtuellen OPC UA Server den Servernamen, Port, Status und die *Werkzeugschlüssel*-Symbolschaltfläche. Aus der zentralen Datenbank des verteilten IT-Systems werden die Informationen zu den virtuellen OPC UA Server mithilfe der *VirtualOpcUaServerDatabaseAccess*-Klasse ausgelesen. Mit der *Werkzeugschlüssel*-Symbolschaltfläche gelangt der Anwender für einen ausgewählten virtuellen OPC UA Server in die Administrationsoberfläche. Die Administrationsoberfläche ist in der *AdminVirtualOpcUaServer*-Klasse implementiert.

**AdminVirtualOpcUaServer**
Die *AdminVirtualOpcUaServer*-Klasse visualisiert für einen ausgewählten virtuellen OPC UA Server die Administrationsoberfläche. Aus der zentralen Datenbank des verteilten IT-Systems werden die Parameter zum ausgewählten virtuellen OPC UA Server ausgelesen und in der Administrationsoberfläche visualisiert. In der Administrationsoberfläche kann der Anwender den Servernamen, Portnummer, Standort sowie die angebotenen (fiktiven) Dienstleistungen der virtuellen Produktionseinheit anpassen. Weiterhin kann er den virtuellen OPC UA Server starten, stoppen oder löschen. Mit der Schaltfläche *Speichern* werden anschließend die Änderungen in der zentralen Datenbank übernommen.

#### 5.3.1.2 Programmlogik (package *ProgramLogic*)

**Main**
Aufgabe der *Main*-Klasse ist die Initialisierung und Ablaufsteuerung der grafischen Oberfläche sowie die Verarbeitung der Benutzereingaben (Interaktionen).

#### 5.3.1.3 Datenbankzugriff (package *DatabaseAccess*)

**DiscoveryServerDatabaseAccess**
In der *DiscoveryServerDatabaseAccess*-Klasse werden die Informationen zu den registrierten Produktionseinheiten aus der Datenbank ausgelesen. (Datenbankzugriff für die *VisualizeNetwork*- und *InfoOpcUaServer*-Klasse).

**ProductionOrdersDatabaseAccess**
In der *ProductionOrdersDatabaseAccess*-Klasse werden neu erstellte Produktionsaufträge in der Datenbank eingetragen sowie die Informationen zu den generierten Produktions- und Dienstleistungsaufträgen ausgelesen. (Datenbankzugriff für die *CreateProductionOrder*- und *StatusProductionOrder*-Klasse)

**VirtualOpcUaServerDatabaseAccess**
In der *VirtualOpcUaServerDatabaseAccess*-Klasse können virtuelle Produktionseinheiten in der Datenbank angelegt sowie Parameter vorhandener Produktionseinheiten ausgelesen und bei Bedarf aktualisiert werden. (Datenbankzugriff für die *CreateVirtualOpcUaServer*-, *ListVirtualOpcUaServer*- und *AdminVirtualOpcUaServer*-Klasse)

### 5.3.2 Softwarekomponenten der OPC UA Systemsoftware

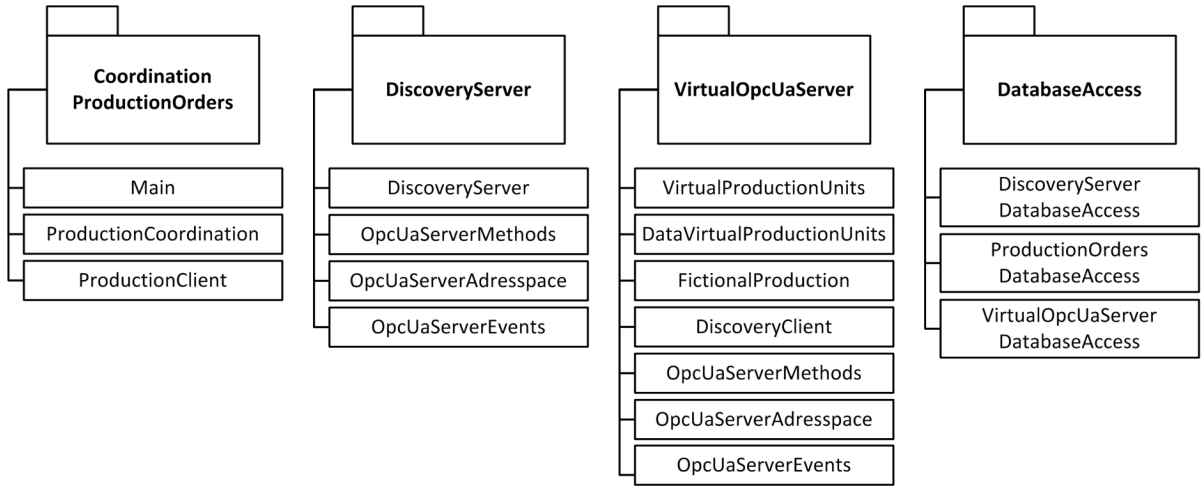


**Abbildung 5.6:** Softwarekomponenten der OPC UA Systemsoftware

#### 5.3.2.1 Koordination Produktionsaufträge (package *CoordinationProductionOrders*)

**Main**
In der *Main*-Klasse werden die einzelnen Module der OPC UA Systemsoftware initialisiert und ausgeführt.

**ProductionCoordination**
In der *ProductionCoordination*-Klasse werden die Produktionsaufträge für das verteilte IT-System koordiniert. Der Algorithmus für die Koordinierung der Produktionsaufträge ist in Kapitel 5.2.2 beschrieben. Der Datenbankzugriff erfolgt in der *ProductionOrdersDatabaseAccess*-Klasse.

**ProductionClient**
In der *ProductionClient*-Klasse erfolgt der Zugriff auf die Produktionseinheiten. Die *ProductionClient*-Klasse implementiert einen OPC UA Client um die definierten Methoden auf den Produktionseinheiten (OPC UA Servern) aufzurufen. In Kapitel 5.4.2 sind Methoden für den Zugriff auf die Produktionseinheiten spezifiziert und beschrieben.

#### 5.3.2.2 Discovery Server (package *DiscoveryServer*)

**DiscoveryServer**
In der *DiscoveryServer*-Klasse ist der Discovery Server für das verteilte IT-System implementiert. Der Discovery Server basiert auf einem Standard OPC UA Server, der im Adressraum mehrere Methoden für den Discovery Prozess bereitstellt.

Die bereitgestellten Methoden des Discovery Servers sind in der *OpcUaServerMethods*-Klasse implementiert. In der *OpcUaServerAdresspace*-Klasse wird der Adressraum für den Discovery Server erstellt und verwaltet.
Zu einer registrierten Produktionseinheit werden folgende Daten erfasst: Servername, Hostname, Port, Standort, Serviceanzahl, angebotene Dienstleistungen (Services) sowie der Zeitstempel der letzten Registrierung. Die Daten der registrierten Produktionseinheiten werden in der zentralen Datenbank mit der *DiscoveryServerDatabaseAccess*-Klasse abgelegt.

**OpcUaServerMethods**
In der *OpcUaServerMethods*-Klasse sind die Methoden des Discovery Servers implementiert. Für die Methodenaufrufe verwenden die Produktionseinheiten einen OPC UA Client. In Kapitel 5.4.1 sind Methoden für den Discovery Prozess spezifiziert und beschrieben.

**OpcUaServerAdresspace**
Die *OpcUaServerAdresspace*-Klasse erstellt und verwaltet den Adressraum des Discovery Servers. Im Adressraum werden die Methoden für den Discovery Prozess abgebildet (vgl. *OpcUaServerMethods*).

**OpcUaServerEvents**
Die *OpcUaServerEvents*-Klasse generiert und verwaltet die Ereignisse (Events) des Discovery Servers. Im Rahmen der Masterarbeit wird diese Klasse nicht weiter spezifiziert und implementiert.

#### 5.3.2.3 Virtuelle OPC UA Server (package *VirtualOpcUaServer*)

**VirtualProductionUnits**
Die *VirtualProductionUnits*-Klasse erstellt, initialisiert und verwaltet die virtuellen Produktionseinheiten. Jede virtuelle Produktionseinheit verwendet für die Kommunikation und Ressourcenverwaltung einen OPC UA Server. Im Adressraum des OPC UA Servers werden die Zugriffsmethoden und simulierten Produktionsprozesse der virtuellen Produktionseinheiten abgebildet. In der *OpcUaServerMethods*-Klasse sind die Zugriffsmethoden implementiert. Die Adressräume der OPC UA Server werden in der *OpcUaServerAdresspace*-Klasse erstellt und verwaltet. In der *FictionalProduction*-Klasse erfolgt die Auftragsbearbeitung.

**DataVirtualProductionUnits**
In der *DataVirtualProductionUnits-Klasse* werden die Daten und Variablen der virtuellen Produktionseinheiten (OPC UA Server) gespeichert und verwaltet. Der Zugriff auf die Daten und Variablen erfolgt mittels Getter- und Setter-Methoden.

**FictionalProduction**
In der *FictionalProduction*-Klasse werden die Dienstleistungsaufträge der virtuellen Produktionseinheit sequentiell abgearbeitet. Für die Erstellung der fiktiven Dienstleistungen (Services)

verwendet die *FictionalProduction*-Klasse simulierte Produktionsprozesse. Die Produktionsprozesse werden mittels Latenzzeiten vereinfacht simuliert.

**DiscoveryClient**
Aufgabe der *DiscoveryClient*-Klasse ist die Verwaltung der virtuellen Produktionseinheiten beim Discovery Server. Die *DiscoveryClient*-Klasse verwendet eine OPC UA Client um die Methoden auf dem Discovery Server aufzurufen. In Kapitel 5.4.1 ist der Discovery Prozess beschrieben.

**OpcUaServerMethods**
Für den Zugriff auf die virtuelle Produktionseinheit implementiert die *OpcUaServerMethods*-Klasse mehrere Methoden im Adressraum des OPC UA Servers. In Kapitel 5.4.2 sind Methoden für den Zugriff auf die Produktionseinheit spezifiziert und beschrieben.

**OpcUaServerAdresspace**
Die *OpcUaServerAdresspace*-Klasse erstellt und verwaltet die Adressräume der virtuellen Produktionseinheiten (OPC UA Server). Im Adressraum des OPC UA Servers werden die Zugriffsmethoden (vgl. *OpcUaServerMethods*) und simulierten Produktionsprozesse der virtuellen Produktionseinheiten (OPC UA Server) abgebildet.

**OpcUaServerEvents**
Die *OpcUaServerEvents*-Klasse generiert und verwaltet die Ereignisse (Events) der virtuellen Produktionseinheiten (OPC UA Server). Im Rahmen der Masterarbeit wird diese Klasse nicht weiter spezifiziert und implementiert.

#### 5.3.2.4 Datenbankzugriff (package *DatabaseAccess*)

**DiscoveryServerDatabaseAccess**
In der *DiscoveryServerDatabaseAccess*-Klasse werden die Informationen zu den registrierten Produktionseinheiten in der Datenbank abgelegt. (Datenbankzugriff für das *Discovery Server* Modul.)

**ProductionOrdersDatabaseAccess**
In der *ProductionOrdersDatabaseAccess*-Klasse werden die Produktions- und Dienstleistungsaufträge aus der Datenbank abgefragt. Weiterhin wird in der Datenbank der aktuelle Status und Bearbeitungsfortschritt der Produktions- und Dienstleistungsaufträge aktualisiert. (Datenbankzugriff für die *Koordinierung der Produktionsaufträge*.)

**VirtualOpcUaServerDatabaseAccess**
In der *VirtualOpcUaServerDatabaseAccess*-Klasse werden die Parameter der virtuellen Produktionseinheiten aus der Datenbank abgefragt und bei Bedarf aktualisiert. (Datenbankzugriff für das *virtuelle OPC UA Server* Modul.)

### 5.3.3 Softwarekomponenten der Mikrocontroller-Programme

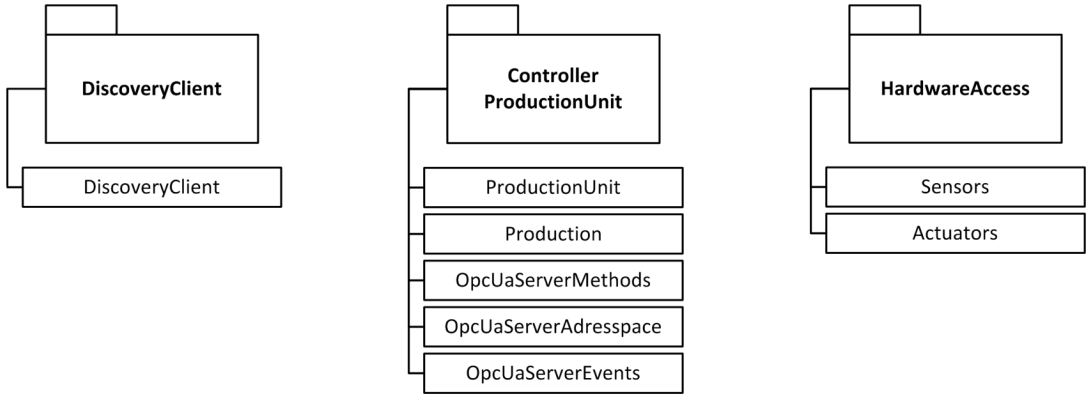


**Abbildung 5.7:** Softwarekomponenten der Mikrocontroller-Programme

#### 5.3.3.1 Discovery Client (package *DiscoveryClient*)

**DiscoveryClient**
Aufgabe der *DiscoveryClient*-Klasse ist die Verwaltung der virtuellen Produktionseinheiten beim Discovery Server. Die *DiscoveryClient*-Klasse verwendet eine OPC UA Client um die Methoden auf dem Discovery Server aufzurufen. In Kapitel 5.4.1 ist der Discovery Prozess beschrieben.

#### 5.3.3.2 Steuerung der Produktionseinheit (package *ControllerProductionUnit*)

**ProductionUnit**
In der *ProductionUnit*-Klasse ist die Steuerung der Produktionseinheit implementiert. Die Steuerung der Produktionseinheit verwendet für die Kommunikation und Ressourcenverwaltung einen OPC UA Server. Im Adressraum des OPC UA Servers werden die Zugriffsmethoden sowie Hardwarekomponenten (Sensoren und Aktoren) der Produktionseinheit abgebildet. In der *OpcUaServerMethods*-Klasse sind die Zugriffsmethoden implementiert. Der Adressraum wird in der *OpcUaServerAdresspace*-Klasse erstellt und verwaltet. In der *Production*-Klasse erfolgt die Auftragsbearbeitung der Produktionseinheit.

**Production**
In der *Production*-Klasse werden die Dienstleistungsaufträge der Produktionseinheit sequentiell abgearbeitet. Für die Erstellung der Dienstleistungen (Services) verwendet die *Production*-Klasse die Hardwarekomponenten (Sensoren und Aktoren) der Produktionseinheit. Die Ansteuerung der Hardwarekomponenten erfolgt in der *Sensors*- und *Actuators*-Klasse. Weiterhin wird in zyklischen Abständen der aktuelle Status und Bearbeitungsfortschritt der bearbeiteten Dienstleistungsaufträge aktualisiert.

**OpcUaServerMethods**
Für den Zugriff auf die Produktionseinheit implementiert die *OpcUaServerMethods*-Klasse mehrere Methoden im Adressraum des OPC UA Servers. In Kapitel 5.4.2 sind Methoden für den Zugriff auf die Produktionseinheit spezifiziert und beschrieben.

**OpcUaServerAdresspace**
Die *OpcUaServerAdresspace*-Klasse erstellt und verwaltet den Adressraum der Produktionseinheit (OPC UA Server). Im Adressraum des OPC UA Servers werden die Zugriffsmethoden (vgl. *OpcUaServerMethods*) sowie die Hardwarekomponenten (Sensoren und Aktoren) der Produktionseinheit (OPC UA Server) abgebildet.

**OpcUaServerEvents**
Die *OpcUaServerEvents*-Klasse generiert und verwaltet die Ereignisse (Events) der Produktionseinheit (OPC UA Server). Im Rahmen der Masterarbeit wird diese Klasse nicht weiter spezifiziert und implementiert.

#### 5.3.3.3 Hardware-Zugriff (package *HardwareAccess*)

**Sensors**
Aufgabe der *Sensors*-Klasse ist das Auslesen und Bereitstellen der Sensordaten. Am Produktionssystem des Demonstrators werden verschiedene Sensoren für die Erfassung der Prozessgrößen eingesetzt.

**Actuators**
In der *Actuators*-Klasse erfolgt die Ansteuerung der Aktoren. Aufgabe der Aktoren sind Stelleingriffe in dem Produktionssystem des Demonstrators.

Hinweis: Das *Hardware-Zugriffsmodul* wird im Rahmen einer Forschungsarbeit spezifiziert und implementiert.

## 5.4 Schnittstellendefinitionen

### 5.4.1 Discovery Prozess

Der Discovery Server für das verteilte IT-System ist in der OPC UA Systemsoftware implementiert. Als Basis wird ein Standard OPC UA Server verwendet, der im Adressraum folgende Methoden zur Verfügung stellt:

**Registrierung**

| **Methode** | *int registerOpcUaServer ( vgl. Parameterliste )* | |
|---|---|---|
| **Beschreibung** | Die Methode registriert eine Produktionseinheit beim Discovery Server. Bei der Registrierung übermittelt der Discovery Client mehrere Informationen über die Produktionseinheit. | |
| **Parameter** | *servername* | Servername der zu registrierenden Produktionseinheit. |
| | *hostname* | Hostname der zu registrierenden Produktionseinheit. |
| | *port* | Portnummer der zu registrierenden Produktionseinheit. |
| | *serviceNr[]* | Angebotene Dienstleistungen (Services) der zu registrierenden Produktionseinheit. |
| | *location[]* | Standort der zu registrierenden Produktionseinheit (2-D). |
| **Rückgabewert** | *serverId* | Identifikator der registrierten Produktionseinheit. Der Wert ist negativ, wenn beim Registrierungsvorgang ein Fehler auftritt. |

**Abmeldung**

| **Methode** | *int deregisterOpcUaServer (int serverId)* | |
|---|---|---|
| **Beschreibung** | Die Methode wird verwendet, um eine registrierte Produktionseinheit (OPC UA Server) beim Discovery Server abzumelden. | |
| **Parameter** | *serverId* | Identifikator der registrierten Produktionseinheit. |
| **Rückgabewert** | *status* | Der Rückgabewert ist negativ, wenn ein Fehler beim Abmeldevorgang auftritt. |

**Keep-Alive-Mechanismus**

| **Methode** | *int keepAliveOpcUaServer (int serverId)* | |
|---|---|---|
| **Beschreibung** | Die Keep-Alive Methode überprüft die Erreichbarkeit der registrierten Produktionseinheiten (OPC UA Server). In einem Zeitintervall von 3 Sekunden, muss jede registrierte Produktionseinheit die Keep-Alive Methode aufrufen. Andernfalls wird die Registrierung beim Discovery Server entfernt. | |
| **Parameter** | *serverId* | Identifikator der registrierten Produktionseinheit. |
| **Rückgabewert** | *status* | Der Rückgabewert ist negativ, wenn in der Methode ein Fehler auftritt. |

**Aktualisierung der angebotenen Dienstleistungen**

| **Methode** | *int updateOfferedServices (int serverId, int serviceNr [])* |
|---|---|
| **Beschreibung** | Die Methode aktualisiert im Discovery Server die angebotenen Dienstleistungen einer registrierten Produktionseinheit (OPC UA Server). |

| | | |
|---|---|---|
| **Parameter** | *serverId* | Identifikator der registrierten Produktionseinheit. |
| | *serviceNr[]* | Dienstleistungen (Services), die eine Produktionseinheit anbietet. |
| **Rückgabewert** | *status* | Der Rückgabewert ist negativ, wenn in der Methode ein Fehler auftritt. |

**Aktualisierung des Standorts**

| | | |
|---|---|---|
| **Methode** | *int updateLocation (int serverId, int location [])* | |
| **Beschreibung** | Die Methode aktualisiert im Discovery Server den Standort einer registrierten Produktionseinheit (OPC UA Server). | |
| **Parameter** | *serverId* | Identifikator der registrierten Produktionseinheit. |
| | *location[]* | Standort der registrierenden Produktionseinheit (2-D). |
| **Rückgabewert** | *status* | Der Rückgabewert ist negativ, wenn in der Methode ein Fehler auftritt. |

Die Produktionseinheiten verwenden das *Discovery Client* Modul um die Methoden auf dem Discovery Server aufzurufen. Das *Discovery Client* Modul beinhaltet einen OPC UA Client.

Der Discovery Server speichert die Daten der registrierten Produktionseinheiten in der zentralen Datenbank des verteilten IT-Systems (vgl. *5.5 Datenbankschema*).

### 5.4.2 Produktionseinheit

Für die Kommunikation und Ressourcenverwaltung verwenden die Produktionseinheiten einen OPC UA Server. Der OPC UA Server muss die folgenden Methoden im Adressraum bereitstellen:

**Auftragsvergabe**

| | | |
|---|---|---|
| **Methode** | *int serviceOrderPlacement (int orderNr, int serviceNr, int quantity)* | |
| **Beschreibung** | Die Methode wird verwendet, um der Produktionseinheit einen Auftrag zuzuweisen. Produktionseinheiten sind berechtigt einen Auftrag abzulehnen. | |
| **Parameter** | *orderNr* | Identifikationsnummer des Dienstleistungsauftrags. |
| | *serviceNr* | Servicenummer der georderten Dienstleistung (Service). |
| | *quantity* | Auftragsmenge der georderten Dienstleistung (Service). |
| **Rückgabewert** | *status* | Der Rückgabewert ist negativ, wenn die Produktionseinheit den Dienstleistungsauftrag ablehnt oder in der Methode ein Fehler auftritt. |

**Auftragsstatus**

| **Methode** | *int [] getServiceOrderStatus (int orderNr)* | |
|---|---|---|
| **Beschreibung** | Die Methode liefert zu einem Dienstleistungsauftrag den aktuellen Status und Bearbeitungsfortschritt. | |
| **Parameter** | *orderNr* | Identifikationsnummer des Dienstleistungsauftrags. |
| **Rückgabewert** | *status[]* | Der Rückgabewert beinhaltet den aktuellen Status und Bearbeitungsfortschritt des Dienstleistungsauftrags. Im Fehlerfall wird ein negativer Wert übergeben. |

**Auftragsstornierung**

| **Methode** | *int cancelServiceOrder (int orderNr)* | |
|---|---|---|
| **Beschreibung** | Die Methode storniert den gewünschten Dienstleistungsauftrag auf der Produktionseinheit. | |
| **Parameter** | *orderNr* | Identifikationsnummer des Dienstleistungsauftrags. |
| **Rückgabewert** | *status* | Der Rückgabewert ist negativ, wenn in der Methode ein Fehler auftritt. |

**Auftragsliste**

| **Methode** | *int [] getOrderList ()* | |
|---|---|---|
| **Beschreibung** | Die Methode liefert die Auftragsliste der Produktionseinheit. | |
| **Rückgabewert** | *orderNr []* | Auftragsliste der Produktionseinheit. |

**Angebotene Dienstleistungen**

| **Methode** | *int [] getOfferedServices ()* | |
|---|---|---|
| **Beschreibung** | Angebotene Dienstleistungen (Services) der Produktionseinheit (OPC UA Server) abfragen. | |
| **Rückgabewert** | *serviceNr[]* | Dienstleistungen (Services), die der OPC UA Server anbietet. |

**Standort der Produktionseinheit**

| **Methode** | *int [] getLocation ()* | |
|---|---|---|
| **Beschreibung** | Die Methode liefert den Standort der Produktionseinheit (2-D). | |
| **Rückgabewert** | *location[]* | Standort der Produktionseinheit (2-D). Im Fehlerfall wird ein negativer Wert übergeben. |

Zusätzlich werden im Adressraum des OPC UA Servers die Hardware-Komponenten (Sensoren und Aktoren) der Produktionseinheit abgebildet.

Die OPC UA Systemsoftware verwendet die Methoden der Produktionseinheit um die Produktionsaufträge zu koordinieren.

## 5.5 Datenbankschema

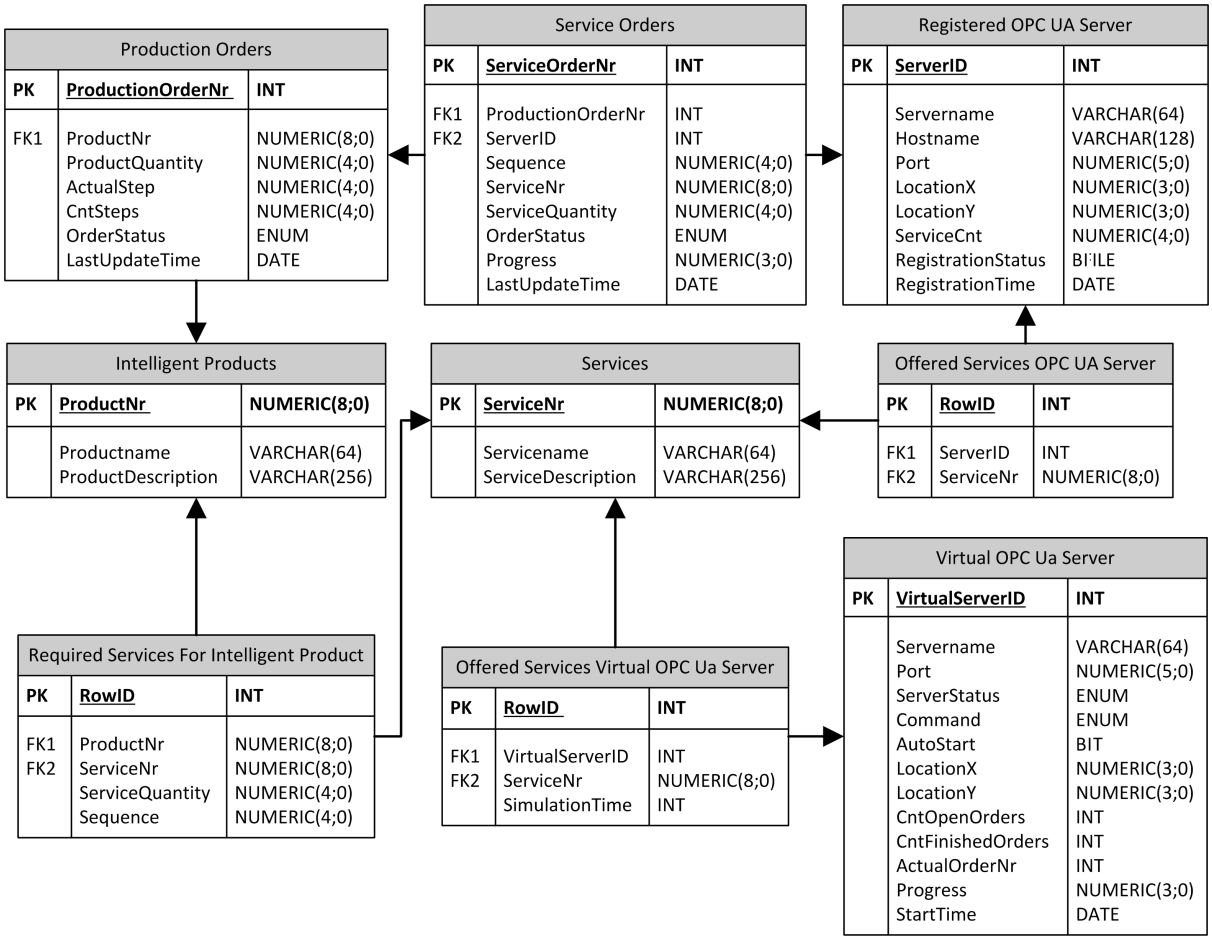


**Abbildung 5.8:** Datenbankschema

### Beschreibung der Tabellen

**Services**
In der Tabelle *Services* werden die Dienstleistungen definiert. Die definierten Dienstleistungen können anschließend von den registrierten OPC UA Servern im Netzwerk angeboten werden. Folgende Informationen werden zu jeder Dienstleistung definiert:

| Spalte | Beschreibung |
|---|---|
| *ServiceNr* | Eindeutige Dienstleistungsnummer (*ServiceNr*) zur Identifizierung der Dienstleistungen. Die Spalte *ServiceNr* wird als Primärschlüssel (PK) verwendet. |
| *Servicename* | In der Spalte *Servicename* sind die Namen der Dienstleistungen gespeichert. |
| *ServiceDescription* | Die textuelle Beschreibung der Dienstleistungen (Services) sind in der *ServiceDescription*-Spalte abgelegt. |

**Intelligent Products**
In der Tabelle *Intelligent Products* definiert die OPC UA Systemsoftware intelligente Produkte. Folgende Informationen werden zu jedem intelligenten Produkt hinterlegt:

| Spalte | Beschreibung |
|---|---|
| *ProductNr* | Eindeutige und einmalige Produktnummer (*ProductNr*) zur Identifizierung der intelligenten Produkte. Die Spalte *ProductNr* wird als Primärschlüssel (PK) verwendet. |
| *Productname* | In der Spalte *Productname* sind die Namen der intelligenten Produkte gespeichert. |
| *ProductDescription* | Die textuelle Beschreibung für die intelligenten Produkte sind in der Spalte *ProductDescription* abgelegt. |

**Required Services For Intelligent Product**
Die OPC UA Systemsoftware definiert in der Tabelle, welche Dienstleistungen für die Erstellung eines intelligenten Produkts erforderlich sind. Für jede benötigte Dienstleistung (Service) wird in der Tabelle eine neue Reihe generiert. In der Reihe wird das intelligente Produkt (*ProductNr*) mit der benötigten Dienstleistung (*ServiceNr*) verknüpft sowie die entsprechende Sequenznummer gesetzt (*Sequence*).

| Spalte | Beschreibung |
|---|---|
| *RowID* | Die *RowID*-Spalte wird zur Durchnummerierung und Identifizierung der Tabelleneinträge verwendet (Primärschlüssel). |
| *ProductNr* | Fremdschlüssel von der Tabelle *Intelligent Products*. |
| *ServiceNr* | Fremdschlüssel von der Tabelle *Services*. |
| *ServiceQuantity* | In der *ServiceQuantity*-Spalte wird die Dienstleistungsmenge festgelegt. |
| *Sequence* | Die *Sequence*-Spalte definiert für das intelligente Produkt die sequentielle Abarbeitungsreihenfolge der erforderlichen Dienstleistungen (Services). Besitzen mehrere erforderliche Dienstleistungen dieselbe Sequenznummer, können diese parallel abgearbeitet werden. |

**Registered OPC UA Server**
Die OPC UA Systemsoftware speichert in der Tabelle die Informationen zu den registrierten OPC UA Servern. Folgende Informationen werden zu jedem registrierten OPC UA Server gespeichert:

| Spalte | Beschreibung |
|---|---|
| *ServerID* | Jeder registrierte OPC UA Server erhält von der Datenbank eine fortlaufende Identifikationsnummer zugewiesen. Die Spalte *ServerID* wird als Primärschlüssel (PK) verwendet. |
| *Servername* | Die Spalte *Servername* speichert die Namen der registrierten OPC UA Server. |
| *Hostname* | In der Spalte *Hostname* werden die Hostnamen der registrierten OPC UA Server abgelegt. |
| *Port* | In der Spalte *Port* werden die Port-Nummern der registrierten OPC UA Server gespeichert. |
| *LocationX* | Die *LocationX*-Spalte speichert die x-Koordinate der registrierten OPC UA Server (Produktionseinheiten) |
| *LocationY* | Die *LocationY*-Spalte speichert die y-Koordinate der registrierten OPC UA Server (Produktionseinheiten) |
| *ServiceCnt* | Die Spalte *ServiceCnt* gibt Auskunft über die Anzahl der angebotenen Dienstleistungen eines registrierten OPC UA Servers. |
| *RegistrationStatus* | In der *RegistrierungsStatus*-Spalte wird der Registrierungsstatus der OPC UA Server gespeichert. Die Registrierung eines OPC UA Servers ist valide, wenn der Boolesche Wert *true* ist. Andernfalls ist die Registrierung abgelaufen. |
| *RegistrationTime* | Der Zeitstempel speichert den letzten Registrierungsvorgang des OPC UA Servers. |

**Offered Services OPC UA Server**
Die OPC UA Systemsoftware speichert in der Tabelle die angebotenen Dienstleistungen der registrierten OPC UA Server. Für jede angebotene Dienstleistung wird in der Tabelle eine neue Reihe generiert. In der Reihe ist hinterlegt, welche Dienstleistung (*ServiceNr*) der registrierte OPC UA Server (*ServerID*) anbietet. Ein registrierter OPC UA Server kann mehrere Dienstleistungen (Services) anbieten.

| Spalte | Beschreibung |
|---|---|
| *RowID* | Die *RowID*-Spalte wird zur Durchnummerierung und Identifizierung der Tabelleneinträge verwendet (Primärschlüssel). |
| *ServerID* | Fremdschlüssel von der Tabelle *Registered OPC UA Server*. |
| *ServiceNr* | Fremdschlüssel von der Tabelle *Services*. |

**Production Orders**
In der Tabelle *Production Orders* sind die Produktionsaufträge für das verteilte IT-System abgelegt. Für jeden Produktionsauftrag wird in der Tabelle eine neue Reihe generiert. Folgende Daten werden zu einem Produktionsauftrag erfasst:

| Spalte | Beschreibung |
|---|---|
| *ProductionOrderNr* | Jeder erstellte Produktionsauftrag erhält von der Datenbank eine fortlaufende Identifikationsnummer zugewiesen. Die Spalte *ProductionOrderNr* wird als Primärschlüssel (PK) verwendet. |
| *ProductNr* | Die Spalte *ProductNr* definiert, welches intelligente Produkt in dem Produktionsauftrag erstellt werden soll. (Fremdschlüssel von der Tabelle *Intelligent Products*.) |
| *ProductQuantity* | In der *ProductQuantity*-Spalte wird die Auftragsmenge des Produktionsauftrags gesetzt. |
| *ActualStep* | Die *ActualStep*-Spalte speichert den aktuellen Bearbeitungsschritt des Produktionsauftrags. |
| *CntSteps* | Anzahl der Produktionsschritte für die Erstellung des Produktionsauftrags. |
| *OrderStatus* | Der aktuelle Status des Produktionsauftrags ist in der *OrderStatus*-Spalte abgelegt. |
| *LastUpdateTime* | Der Zeitstempel speichert den letzten Aktualisierungsvorgang des Produktionsauftrags. |

**Service Orders**
In der Tabelle *Service Orders* sind die Dienstleistungsaufträge der Produktionsaufträge abgelegt. Produktionsaufträge bestehen aus einem oder mehreren Dienstleistungsaufträgen, deren Abarbeitungsreihenfolge über die Sequenznummer definiert wird. Besitzen mehrere Dienstleistungsaufträge dieselbe Sequenznummer, können diese parallel abgearbeitet werden. Für jeden Dienstleistungsauftrag wird in der Tabelle eine neue Reihe generiert. Folgende Daten werden zu einem Dienstleistungsauftrag erfasst:

| Spalte | Beschreibung |
|---|---|
| *ServiceOrderNr* | Jeder generierte Dienstleistungsauftrag erhält von der Datenbank eine fortlaufende Identifikationsnummer zugewiesen. Die Spalte *ServiceOrderNr* wird als Primärschlüssel (PK) verwendet. |
| *ProductionOrderNr* | Die Spalte *ProductionOrderNr* verknüpft den Dienstleistungsauftrag mit einem Produktionsauftrag. (Fremdschlüssel von der Tabelle *Production Orders*.) |
| *ServerID* | In der *ServerID*-Spalte setzt die OPC UA Systemsoftware den OPC UA Server der den Dienstleistungsauftrag bearbeitet. (Fremdschlüssel von der Tabelle *Registered OPC UA Server*.) |

| | |
|---|---|
| *ServiceNr* | Die Spalte *ServiceNr* definiert, welche Dienstleistung in dem Dienstleistungsauftrag in Anspruch genommen wird. (Fremdschlüssel von der Tabelle *Services*.) |
| *Sequence* | Die *Sequence*-Spalte legt für einen Produktionsauftrag die sequentielle Abarbeitungsreihenfolge der dazugehörigen Dienstleistungsaufträge fest. Besitzen mehrere Dienstleistungsaufträge dieselbe Sequenznummer, können diese parallel abgearbeitet werden. |
| *ServiceQuantity* | In der *ServiceQuantity*-Spalte wird die Auftragsmenge des Dienstleistungsauftrags gesetzt. |
| *OrderStatus* | Der aktuelle Status des Dienstleistungsauftrags ist in der *OrderStatus*-Spalte abgelegt. |
| *Progress* | In der *Progress*-Spalte wird der Bearbeitungsfortschritt für den Dienstleistungsauftrag gespeichert (Prozentwert). |
| *LastUpdateTime* | Der Zeitstempel speichert den letzten Aktualisierungsvorgang des Dienstleistungsauftrags. |

**Virtual OPC UA Server**
In der Tabelle *Virtual OPC UA Server* sind die Parameter der virtuellen OPC UA Server gespeichert. Jede Reihe speichert die Parameter für einen virtuellen OPC UA Server.

| Spalte | Beschreibung |
|---|---|
| *VirtualServerID* | Jeder virtuelle OPC UA Server erhält von der Datenbank eine fortlaufende Identifikationsnummer zugewiesen. Die Spalte *VirtualServerID* wird als Primärschlüssel (PK) verwendet. |
| *Servername* | Die Spalte *Servername* speichert den Namen der virtuellen OPC UA Server. |
| *Port* | Die Port-Nummer der virtuellen OPC UA Server sind in der Spalte *Port* gespeichert. |
| *ServerStatus* | Liefert Informationen über den aktuellen Status der virtuellen OPC UA Server. |
| *Command* | Mit der *Command*-Spalte können virtuelle OPC UA Server in der OPC UA Systemsoftware gestartet oder gestoppt werden. In zyklischen Abständen verarbeitet die OPC UA Systemsoftware die gesetzten Befehle. Nach deren Ausführung werden die Werte zurückgesetzt. |
| *Autostart* | Bei der Initialisierung der OPC UA Systemsoftware werden alle virtuellen OPC UA Server mit gesetztem *Autostart*-Flag automatisch gestartet. |
| *LocationX* | In der Spalte *LocationX* wird die x-Koordinate des virtuellen OPC UA Server gespeichert. |
| *LocationY* | In der Spalte *LocationY* wird die y-Koordinate des virtuellen OPC UA Server gespeichert. |

| | |
|---|---|
| *CntOpenOrders* | Speichert für einen virtuellen OPC UA Server die Anzahl der offenen Dienstleistungsaufträge. |
| *CntFinishedOrders* | Speichert für einen virtuellen OPC UA Server die Anzahl der abgeschlossenen Dienstleistungsaufträge. |
| *ActualOrderNr* | In der *ActualOrderNr* Spalte ist die Auftragsnummer vom aktuell bearbeiteten Dienstleistungsauftrag abgelegt. |
| *Progress* | Der Fortschritt (Prozentwert) vom aktuell bearbeiteten Dienstleistungsauftrag ist in der *Progress* Spalte gespeichert. |
| *StartTime* | Auf Basis der Startzeit (*StartTime*) kann die Laufzeit eines virtuellen OPC UA Servers ermittelt werden. |

Die OPC UA Systemsoftware verwendet die Daten der Tabelle zur Erstellung und Administration der virtuellen OPC UA Server. Mit der grafischen Anwendung kann der Benutzer einen neuen virtuellen OPC UA Server der Tabelle hinzufügen oder die Parameter von einem vorhandenen virtuellen OPC UA Server anpassen.

**Offered Services Virtual OPC Ua Server**
In dieser Tabelle sind die angebotenen Dienstleistungen (Services) der virtuellen OPC UA Server gespeichert. Für jede angebotene Dienstleistung wird in der Tabelle eine neue Reihe generiert. In der Reihe ist hinterlegt, welche Dienstleistung (*ServiceNr*) der virtuelle OPC UA Server (*VirtualServerID*) anbietet. Ein virtueller OPC UA Server kann mehrere Dienstleistungen (Services) anbieten.

| Spalte | Beschreibung |
|---|---|
| *RowID* | Die *RowID*-Spalte wird zur Durchnummerierung und Identifizierung der Tabelleneinträge verwendet (Primärschlüssel). |
| *VirtualServerID* | Fremdschlüssel von der Tabelle *Virtual OPC Ua Server*. |
| *ServiceNr* | Fremdschlüssel von der Tabelle *Services*. |
| *SimulationTime* | Simulationszeit für die Erstellung der Dienstleistung. |

# 6 Implementierung

## 6.1 Implementierung der Anwendungen

### 6.1.1 Grafische Anwendung

Die grafische Anwendung zur Administration und Visualisierung des OPC UA Netzwerks konnte wie im Systemmodell (Entwurf) vorgesehen, umgesetzt werden. In drei Punkten wurde die grafische Anwendung in der Implementierungsphase erweitert:

- **Strukturierung und Gestaltung des Hauptmenüs**
  In Abbildung 6.1 ist das erweiterte Hauptmenü dargestellt. Für die Strukturierung und Gestaltung des Hauptmenüs wurde das Kacheldesign gewählt. In der linken Spalte sind die Schaltflächen der Hauptmenüpunkte angeordnet. In der mittleren Spalte visualisiert die grafische Anwendung die Statusinformationen des verteilten IT-Systems. Folgende Informationen werden in der Statusbox aufgelistet: Anzahl der registrierten OPC UA Server, Anzahl der inaktiven OPC UA Server, Anzahl der verfügbaren Dienstleistungen, Anzahl der wartenden, begonnen und beendeten Produktions- und Dienstleistungsaufträge sowie die Anzahl der aktiven und inaktiven virtuellen OPC UA Server. In zyklischen Abständen aktualisiert die grafische Anwendung die Informationen der Statusbox. In der rechten Spalte befinden sich die Untermenüpunkte des Hauptmenüs sowie die Schaltflächen der Benutzeranmeldung und für das Beenden der grafischen Anwendung.

- **Farbliche Gestaltung der Anwendung**
  Für die intuitive Benutzerführung und die Strukturierung der Anwendungsfunktionen wurde ein Farbschema entwickelt. In der grafischen Anwendung werden die Anwendungsfunktionen mittels Farbwerten strukturiert und thematisch zusammengefasst. In Abbildung 6.1 ist ein Beispiel für farbliche Gestaltung der Anwendung dargestellt.

- **Autostart der virtuellen Produktionseinheiten (OPC UA Server)**
  Bei der Erstellung und Administration der virtuellen Produktionseinheiten (OPC UA Server) kann der Anwender den Autostart setzen. Beim Initialisierungsvorgang der OPC-UA Systemsoftware werden anschließend alle virtuellen Produktionseinheiten (OPC UA Server) mit gesetztem Autostart automatisch ausgeführt.

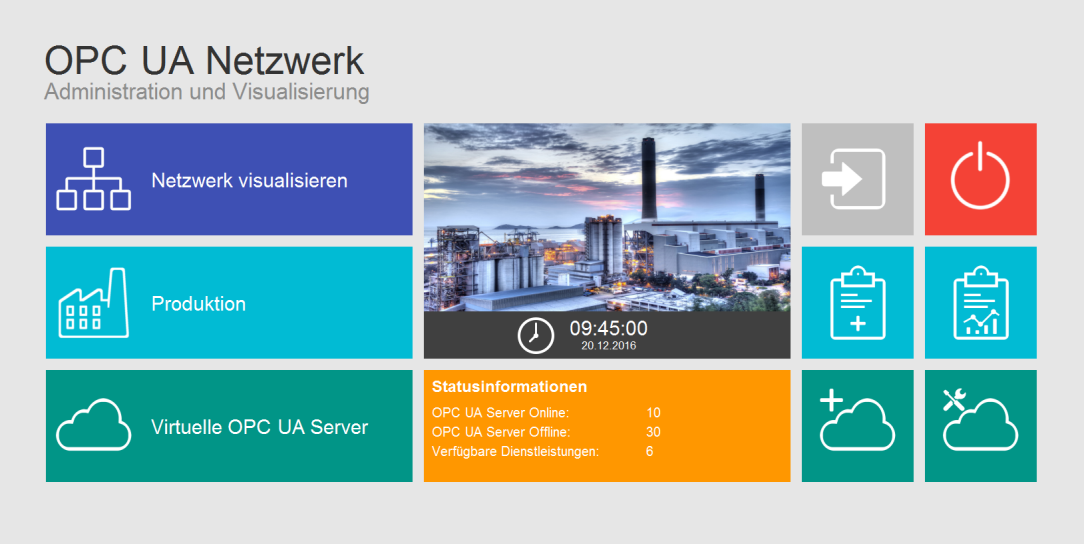


**Abbildung 6.1:** Hauptmenü der grafischen Anwendung.

### 6.1.2 OPC UA Systemsoftware

Die Implementierung der OPC UA Systemsoftware erfolgte nach dem Entwurf der Software-Systemarchitektur. Es sind keine nennenswerten Probleme oder Schwierigkeiten bei der Realisierung des Entwurfs aufgetreten.
In der Initialisierungsphase der OPC UA Systemsoftware werden der Discovery Server, die virtuellen Produktionseinheiten (OPC UA Server) sowie das Modul zur Koordination der Produktionsaufträge initialisiert und ausgeführt. Der Initialisierungsvorgang der einzelnen Module erfolgt sequentiell. Das Starten einer virtuellen Produktionseinheit benötigt -je nach Hardware- bis zu drei Sekunden. In der OPC UA Systemsoftware können -theoretisch- beliebig viele virtuelle Produktionseinheiten (OPC UA Server) erstellt und ausgeführt werden. Im Modul zur Koordination der Produktionsaufträge werden in zyklischen Abständen von den Produktionseinheiten der aktuelle Status und Bearbeitungsfortschritt der vergebenen Dienstleistungsaufträge abgefragt und in der Datenbank aktualisiert. Die Zeitspanne der Statusabfrage variiert je nach Anzahl der vergebenen Dienstleistungsaufträge. Im Informationsbereich des Betriebssystems wird der Anwender über den aktuellen Status der OPC UA Systemsoftware informiert.

### 6.1.3 Mikrocontroller Programme

Die Software-Systemarchitektur der Mikrocontroller Programme besteht aus drei Modulen. Der Discovery Client und das Steuerungsmodul der Produktionseinheit konnten entsprechend dem Entwurf vollständig umgesetzt werden. Das Hardwarezugriffsmodul sollte im Rahmen einer parallel ablaufenden Forschungsarbeit eines Kommilitonen spezifiziert und implementiert

werden. Aufgrund einer zeitlichen Verzögerung dieser Arbeit konnte das Hardwarezugriffsmodul leider nicht rechtzeitig fertigstellt werden. Um dennoch die Funktionalität der realen Produktionseinheiten zu veranschaulichen, wurden im Rahmen dieser Masterarbeit Leuchtdioden mit Multifarben (RGB-LED) verwendet. Den angebotenen Dienstleistungen der realen Produktionseinheit werden definierte Farbwerte zugewiesen. Produziert die reale Produktionseinheit einen Dienstleistungsauftrag, blinkt die Leuchtdiode mit dem definierten Farbwert der Dienstleistung.
Der Mikrocontroller (Raspberry Pi) hat im Vergleich zur Workstation nur einen Bruchteil der Rechenleistung. Aufgrund der begrenzten Rechenleistung benötigt der Initialisierungsvorgang der Mikrocontroller-Programme bis zu 20 Sekunden. Der Registrierungsvorgang der realen Produktionseinheit (OPC UA Server) erfolgt -wie vorgesehen- alle drei Sekunden beim Discovery Server.

## 6.2 Quelldateien und Datenbankschema

Die Quelldateien der implementierten Anwendungen werden auf dem Subversion-Server (SVN) der Universität Stuttgart verwaltet. In folgenden Projektverzeichnissen befinden sich die Quelldateien der Applikationen.

| | |
|---|---|
| **Grafische Anwendung:** | *\Masterarbeit_PFEIFFER\impl\GrafischeAnwendung* |
| **OPC-UA Systemsoftware:** | *\Masterarbeit_PFEIFFER\impl\OpcUaSystemsoftware* |
| **Mikrocontroller Programme:** | *\Masterarbeit_PFEIFFER\impl\MikrocontrollerProgramme* |
| **Datenbankschema:** | *\Masterarbeit_PFEIFFER\impl\Datenbankschema* |

## 6.3 Dokumentation

Das Systemmodell beschreibt die Anwendungsfälle sowie den Aufbau und Ablauf der grafischen Anwendung. Im Kapitel Software-Systemarchitektur sind die grundlegenden Entwurfsentscheidungen, der modulare Aufbau, die Softwarekomponenten und Schnittstellen des verteilten IT-Systems sowie das Datenbankschema der Datenhaltung dokumentiert.
Zur Dokumentation der implementierten Module und Softwarekomponenten wurde außerdem das Software-Dokumentationswerkzeug JavaDoc eingesetzt. Weiterhin erfolgte eine ausgiebige Kommentierung des Quellcodes. Bei der Umsetzung des Projekts wurde insbesondere auf eine strukturierte, selbsterklärende und saubere Programmierung Wert gelegt.
Eine Anleitung für die Installation und Inbetriebnahme des verteilten IT-Systems ist nicht Teil des Pflichtenhefts, wird jedoch nach Abschluss der Masterarbeit zusätzlich erstellt.

## 6.4 Systemabnahme

Grundlage der Systemabnahme bildet das Pflichtenheft. Die Überprüfung hat ergeben, dass das implementierte IT-System die spezifizierten Anforderungen und Ziele vollständig erfüllt. Weiterhin wurden die spezifizierten Funktionalitäten des Gesamtsystems von mehreren Anwendern ausgiebig getestet und mittels plausibler Testfälle verifiziert. Es konnten dabei keine undefinierten Fehler oder ein unerwünschtes Verhalten festgestellt werden. Definierte Fehler wurden erfolgreich von den implementierten Fehlermechanismen abgefangen, verarbeitet und die betroffenen Komponente des verteilten IT-Systems in einen sicheren Zustand überführt.

# 7 Zusammenfassung und Ausblick

Ziel der vorliegenden Masterarbeit war die Realisierung eines verteilten IT-System auf Basis von OPC UA. Nach einem Einstieg in die Vision Industrie 4.0 und die Machine-to-Machine Kommunikation war die detaillierte Betrachtung des Kommunikationsprotokolls OPC Unified Architecture Schwerpunkt der Grundlagenarbeit.

Zunächst wurden die Ziele, Einsatzbereiche, Umgebung, funktionale und nichtfunktionale Anforderungen, Testfälle sowie die eingesetzte Entwicklungsumgebung für die Realisierung des verteilten IT-Systems definiert und in einem Pflichtenheft dokumentiert.

In der nachfolgenden Entwurfsphase wurden ein Systemmodell sowie die Software-Systemarchitektur entworfen, auf deren Basis die Entwicklung und Umsetzung der grafischen Anwendung, der OPC UA Systemsoftware sowie der Mikrocontroller Programme erfolgte.

Der Anwender kann mit der grafischen Anwendung die registrierten Produktionseinheiten auflisten, Produktionsaufträge erstellen, der Status eines Produktionsauftrags visualisieren, virtuelle OPC UA Server mit fiktiven Dienstleistungen generieren oder virtuelle OPC UA Server administrieren. Die entwickelte OPC UA Systemsoftware besteht aus mehreren Modulen und bildet das Kernelement des verteilten IT-Systems. Der Discovery Server basiert auf einem Standard OPC UA Server, der im Adressraum mehrere Methoden für den Discovery Prozess bereitstellt. In einem weiteren Modul werden die virtuellen OPC UA Server für das verteilte IT-System erstellt, initialisiert und verwaltet. Weiterhin werden in der OPC UA Systemsoftware die Produktions- und Dienstleistungsaufträge für das verteilte IT-System mit intelligenten Algorithmen im Produktionssystem koordiniert und überwacht. Aufgabe der erstellten Mikrocontroller-Programme ist die Steuerung der realen Produktionseinheiten. Ausgeführt werden die Mikrocontroller-Programme auf den Einplatinencomputer (Raspberry Pi) des Demonstrators. Die Datenhaltung des verteilten IT-Systems erfolgt zentral in einer Datenbank. Für die einfache und schnelle Implementierung der OPC UA Funktionalitäten wurde die OPC UA Java Bibliothek des Herstellers Prosys eingesetzt.

Nennenswerte Probleme oder Schwierigkeiten sind bei der Realisierung des verteilten IT-Systems nicht aufgetreten. Das entwickelte System erfüllt sämtliche gestellten Anforderungen und zeigt die vielseitige Einsetzbarkeit von OPC Unifed Architecture.

# Ausblick

OPC Unified Architecture wird bei der Umsetzung der Vision Industrie 4.0 eine Schlüsselrolle einnehmen. Die Masterarbeit und das entwickelte IT-System bilden eine solide Grundlage für weiterführende Arbeiten im Themenfeld OPC Unified Architecture.

Für die Visualisierung und Administration des OPC UA Netzwerks wird aktuell eine Desktop-Applikation eingesetzt. Es wird empfohlen, diese Desktop-Applikation in eine Webapplikation zu überführen. Benutzeranfragen an das verteilte IT-System könnten zentral verarbeitet werden. Zudem sind keine Installationen und Softwareverteilungen auf den Endgeräten erforderlich. Die Applikation kann mit einem beliebigen Webbrowser plattformunabhängig aufgerufen werden. Weiterhin kommt es zu keinen veralteten Softwareversionen auf den Endgeräten der Anwender. Durch ein responsives Webdesign ist die Applikation zudem auf verschiedenen Endgeräten wie beispielsweise Arbeitsplatzrechner, Laptop, Tablet oder Smartphone nutzbar.

Die Logistik für den Material- und Warenfluss im Produktionssystem war nicht Teil der Masterarbeit. Intelligente Algorithmen für die Organisation, Steuerung sowie Durchführung des Material- und Warenflusses sollten für das verteilte IT-System entwickelt und implementiert werden.

Eine mögliche Erweiterung ist die Visualisierung des Produktionssystems. Ziel wäre ein möglichst realistisches Abbild der Gesamtstruktur mit den dazugehörigen Produktionsprozessen.

Aktuell werden von der entwickelten Software die Dienstleistungsaufträge an die Produktionseinheiten mit der geringsten Auslastung zugewiesen. Überlegenswert wäre ein Algorithmus, welcher die Dienstleistungsaufträge unter Berücksichtigung verschiedener Kriterien vergeben kann, wie beispielsweise Standort, Qualität, Kosten oder Umweltfaktoren.

# Literaturverzeichnis


[1] T. Bauernhansl, M. Hompel, B. Vogel-Heuser. *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Wiesbaden: Springer Fachmedien, 2014. ISBN: 978-3-658-04681-1.

[2] E. Westkämper, C. Löffler. *Strategien der Produktion - Technologien, Konzepte und Wege in die Praxis*. Heidelberg: Springer-Verlag, 2016. ISBN: 978-3-662-48913-0.

[3] Deutsches Institut für Normung e. V. (DIN). *Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) - DIN SPEC 91345 2016-04*. Berlin: Beuth Verlag GmbH, 2016.

[4] Forschungsunion Wirtschaft und Wissenschaft. *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0 (2013)*. [Zugriff am 19.07.2016]. URL: www.bmbf.de/files/Umsetzungsempfehlungen_Industrie4_0.pdf.

[5] F. Behrendt. *Produktion und Dienstleistung – Industrie 4.0 (2016)*. [Zugriff am 19.07.2016]. URL: www.fraunhofer.de/de/forschung/forschungsfelder/produktion-dienstleistung/industrie-4-0.html.

[6] O. Bendel. *Industrie 4.0 (2016)*. [Zugriff am 19.07.2016]. URL: www.wirtschaftslexikon.gabler.de/Definition/industrie-4-0.html.

[7] M. Weyrich. *Automatisierungstechnik II (Vorlesungsskript)*. Stuttgart: Universität Stuttgart - Institut für Automatisierungstechnik und Softwaresysteme, 2015.

[8] Frauenhofer IAO. *Produktionsarbeit der Zukunft - Industrie 4.0*. Stuttgart: Frauenhofer Verlag, 2013. ISBN: 978-3-8396-0570-7.

[9] C. Ramsauer. *Industrie 4.0 – Die Produktion der Zukunft (2013)*. [Zugriff am 19.07.2016]. URL: www.forschungsnetzwerk.at/downloadpub/7521_0_DieProduktionderZukunft_ChristianRamsauer.pdf.

[10] H. Kagermann, L. Wolf-Dieter. *Industrie 4.0 - Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution (2011)*. [Zugriff am 19.07.2016]. URL: www.vdi-nachrichten.com/Technik-Gesellschaft/Industrie-40-Mit-Internet-Dinge-Weg-4-industriellen-Revolution.

[11] J. Jasperneite. *Internet und Automation: Was hinter Begriffen wie Industrie 4.0 steckt (2012)*. [Zugriff am 19.07.2016]. URL: www.computer-automation.de/steuerungsebene/steuern-regeln/artikel/93559/.

[12] Bundesministerium für Wirtschaft und Energie. *Hintergrund zur Plattform Industrie 4.0 (2016)*. [Zugriff am 19.07.2016]. URL: www.plattform-i40.de/I40/Navigation/DE/Plattform/Plattform-Industrie-40/plattform-industrie-40.html.

[13] Wikipedia - Die freie Enzyklopädie. *Industrie 4.0 (2016).* [Zugriff am 20.07.2016]. URL: www.de.wikipedia.org/wiki/Industrie_4.0.

[14] Bundesministerium für Wirtschaft und Energie. *IT-Sicherheit für die Industrie 4.0.* Berlin: Bundesministerium für Wirtschaft und Energie - Öffentlichkeitsarbeit, 2016.

[15] Plattform Industrie 4.0. *Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0) - Eine Einführung (2016).* [Zugriff am 26.07.2016]. URL: www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/rami40-eine-einfuehrung.pdf.

[16] Verein Deutscher Ingenieure (VDI). *Statusreport - Referenzarchitekturmodell Industrie 4.0 (RAMI4.0).* Düsseldorf: VDI Verein Deutscher Ingenieure e.V., 2015.

[17] M. Hankel. *Industrie 4.0: Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0) - Eine Einführung (2015).* [Zugriff am 26.07.2016]. URL: www.zvei.org/Downloads/Automation/ZVEI-Faktenblatt-Industrie4_0-RAMI-4_0.pdf.

[18] Wikipedia - Die freie Enzyklopädie. *Machine to Machine (2016).* [Zugriff am 28.07.2016]. URL: www.de.wikipedia.org/wiki/Machine_to_Machine.

[19] P. Schnabel. *M2M - Maschine-zu-Maschine-Kommunikation (2016).* [Zugriff am 28.07.2016]. URL: www.elektronik-kompendium.de/sites/kom/1502061.htm.

[20] Bundesministerium für Wirtschaft und Energie. *Machine-to-Machine Kommunikation – eine Chance für die deutsche Industrie.* Berlin: Bundesministerium für Wirtschaft und Energie - Öffentlichkeitsarbeit, 2012.

[21] Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (OECD). *Machine-to-Machine Communications - Connecting Billions of Devices (2012).* [Zugriff am 28.07.2016]. URL: www.oecd-ilibrary.org/docserver/download/5k9gsh2gp043.pdf.

[22] European Telecommunications Standards Institute (ETSI). *Machine to Machine Communication – ETSI M2M Overview (2011).* [Zugriff am 28.07.2016]. URL: www.ietf.org/mail-archive/web/smartobjectdir/current/pdfiZ5zTqvuNV.pdf.

[23] M. Dohler. *Machine-to-Machine Technologies - Vision, Standards and Applications (2013).* [Zugriff am 28.07.2016]. URL: www . itu . int / dms _ pub / itu - r / oth / 0a / 06 / R0A060000570003PDFE.pdf.

[24] LEER. *LEER.* [Zugriff am 28.07.2016]. URL: LEER.

[25] T. Zerbach, M. Pfeifer. *Machine-to-Machine (M2M) - Kommunikation und der Impact auf die CRM Strategie von Unternehmen (2015).* [Zugriff am 28.07.2016]. URL: www.cintellic.com/download/Cintellic_M2M_Kommunikation.html.

[26] A. Glanz, M. Büsgen. *Machine-to-Machine-Kommunikation.* Frankfurt am Main: Campus Verlag, 2013. ISBN: 978-3-593-39896-9.

[27] P. Beuth. *Amazon Dash-Button (2015).* [Zugriff am 04.08.2016]. URL: www.zeit.de/digital/internet/2015-04/amazon-dash-button.

[28] M. Weyrich. *Kommunikations-Architekturen, -Protokolle und -Formate für die M2M-Kommunikation*. Stuttgart: Universität Stuttgart - Institut für Automatisierungstechnik und Softwaresysteme, 2014.

[29] G. Carle. *Grundlagen Rechnernetze und Verteilte Systeme - Kapitel 5: Sitzungs-, Darstellungs- und Anwendungsschicht (2012)*. [Zugriff am 09.08.2016]. URL: www.net.in.tum.de/fileadmin/TUM/teaching/grnvs/ss12/slides_chap5.pdf.

[30] M. Wojcieszak. *Das Netz der Zukunft: Protokollstandardisierung in der Sackgasse? (2016)*. [Zugriff am 09.08.2016]. URL: www.informatik-aktuell.de/betrieb/netzwerke/das-netz-der-zukunft-protokollstandardisierung-in-der-sackgasse.html.

[31] J. Ousterhout. *HTTP and HTTPS (2010)*. [Zugriff am 09.08.2016]. URL: www.web.stanford.edu/~ouster/cgi-bin/cs142-fall10/lecture.php?topic=http.

[32] K. Lipinski. *HTTP (HyperText transfer protocol) (2016)*. [Zugriff am 09.08.2016]. URL: www.itwissen.info/definition/lexikon/hypertext-transfer-protocol-HTTP-HTTP-Protokoll.html.

[33] Z. Shelby, K. Hartke, C. Bormann. *The Constrained Application Protocol (CoAP) - RFC 7252 (2014)*. [Zugriff am 09.08.2016]. URL: www.tools.ietf.org/html/rfc7252.

[34] International Business Machines Corporation (IBM). *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. Armonk: IBM Redbooks, 2012. ISBN: 9780738437088.

[35] International Business Machines Corporation (IBM). *MQTT V3.1 Protocol Specification (2010)*. [Zugriff am 10.08.2016]. URL: www.public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html.

[36] C. Götz. *MQTT - Protokoll für das Internet der Dinge (2014)*. [Zugriff am 10.08.2016]. URL: www.heise.de/developer/artikel/MQTT-Protokoll-fuer-das-Internet-der-Dinge-2168152.html.

[37] Open Mobile Alliance (OMA). *Lightweight M2M – Enabling Device Management and Applications for the Internet of Things (2014)*. [Zugriff am 11.08.2016]. URL: www.cdn2.hubspot.net/hub/183757/file-610591431-pdf/docs/OMA_Whitepaper_Lightweight_M2M_3-14[5].

[38] J. Prado. *OMA Lightweight M2M Resource Model (2014)*. [Zugriff am 11.08.2016]. URL: www.iab.org/wp-content/IAB-uploads/2016/03/OMA_LightweighM2M_Resource_Model_Summary.pdf.

[39] H. Tschofenig. *LWM2M Introduction (2016)*. [Zugriff am 11.08.2016]. URL: www.slideshare.net/OpenMobileAlliance/lwm2m-introduction-edinburgh-2016-workshop-with-arm.

[40] Open Mobile Alliance (OMA). *Lightweight Machine to Machine Technical Specification (2014)*. [Zugriff am 11.08.2016]. URL: www.dev_devtoolkit.openmobilealliance.org/IoT/LWM2M10/doc/TS/index.html.

[41] Association for Manufacturing Technology (AMT). *Getting Started with MTConnect - Connectivity Guide (2011).* [Zugriff am 11.08.2016]. URL: www.mtconnect.org/s/Getting-Started-with-MTConnect-FINAL.pdf.

[42] Association for Manufacturing Technology (AMT). *MTConnect Standard Part 1 - Overview and Protocol (Version 1.3.0) (2014).* [Zugriff am 11.08.2016]. URL: www.mtconnect.org/s/mtc_part_1_overview_v13.pdf.

[43] A. Dugenske. *MTConnect Overview (2009).* [Zugriff am 11.08.2016]. URL: www.ume.gatech.edu/mechatronics_course/IntroMech/FIS_Dugenske.pdf.

[44] D. Edstrom. *Getting started with MTConnect - Writing Client Applications (2013).* [Zugriff am 11.08.2016]. URL: www.automation.com/pdf_articles/gettingstartedwithmtconnectwriting clientsrevisionjuly2013.pdf.

[45] T. Kluge. *Agentenkommunikation - Einsatz von FIPA/ACL zur Modellierung der Kommunikation zwischen Dialogmodell und Applikation (2005).* [Zugriff am 13.08.2016]. URL: www.isl.anthropomatik.kit.edu/cmu-kit/downloads/tobias_kluge.pdf.

[46] Foundation for Intelligent Physical Agents (FIPA). *FIPA Agent Management Specification (2002).* [Zugriff am 13.08.2016]. URL: www.fipa.org/specs/fipa00023/SC00023K.pdf.

[47] JADE Open Source Community. *JAVA Agent DEvelopment Framework (2015).* [Zugriff am 13.08.2016]. URL: www.jade.tilab.com.

[48] S. Erkayhan. *Ein Vorgehensmodell zur automatischen Kopplung von Services am Beispiel der Integration von Standardsoftwaresystemen.* Karlsruhe: KIT Scientific Publishing, 2011. ISBN: 978-3-86644-697-7.

[49] Jabber Software Foundation. *Extensible Messaging and Presence Protocol (XMPP) - RFC 3920 (2004).* [Zugriff am 14.08.2016]. URL: www.xmpp.org/rfcs/rfc3920.html.

[50] XMPP Standards Foundation. *An Overview of XMPP (2016).* [Zugriff am 14.08.2016]. URL: www.xmpp.org/about/technology-overview.html.

[51] United States Department of Defense. *Unified Capabilities Extensible Messaging and Presence Protocol (2013).* [Zugriff am 14.08.2016]. URL: www.disa.mil/~/media/Files/DISA/Services/UCCO/UCR2013/UC_XMPP_2013.pdf.

[52] XMPP Standards Foundation. *XMPP Libraries (2016).* [Zugriff am 14.08.2016]. URL: www.xmpp.org/software/libraries.html.

[53] W. Mahnke, S.-H. Leitner, M. Damm. *OPC Unified Architecture.* Heidelberg: Springer-Verlag Berlin Heidelberg, 2009. ISBN: 978-3-540-68898-3.

[54] J. Rinaldi. *OPC Unified Architecture - The Everyman's Guide to the Most Important Information Technology in Industrial Automation.* Pewaukee: CreateSpace Independent Publishing Platform (Amazon), 2016. ISBN: 978-1530505111.

[55] U. Enste, W. Mahnke. „OPC Unified Architecture - Die nächste Stufe der Interoperabilität". In: *at-Automatisierungstechnik* Volume 59 (2011), S. 394–405.

[56] S. Hoppe. *Internet of Things (IoT) - Mit OPC-UA vom Sensor bis in die IT-Cloud (2014).* [Zugriff am 28.08.2016]. URL: www.nik-nbg.de/uploads/tx_seminars/Hoppe-Interoperability-Vom-Sensor-bis-in-die_Cloud.pdf.

[57] OPC Foundation. *OPC Unified Architecture - Wegbereiter der 4. industriellen (R)Evolution (2014).* [Zugriff am 28.08.2016]. URL: www.opcfoundation.org/wp-content/uploads/2014/03/OPC_UA_I_4.0_Wegbereiter_DE_v2.pdf.

[58] T. Rössel. „Sicherheit für OPC“. In: *SPS-MAGAZIN* (6/2015), S. 135–137.

[59] E. Murphy. *Die OPC Unified Architecture Security - Haben Sie Vorbehalte? (2010).* [Zugriff am 28.08.2016]. URL: www.funkschau.de/fileadmin/media/whitepaper/files/WP_OPC-UA-Security_DE_final_072010.pdf.

[60] OPC Foundation. *History (2016).* [Zugriff am 01.09.2016]. URL: www.opcfoundation.org/about/opc-foundation/history/.

[61] OPC Foundation. *Part 1 - Overview and Concepts (Version 1.03) (2015).* [Zugriff am 30.08.2016]. URL: www.opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/.

[62] OPC Foundation. *Part 3 - Address Space Model (Version 1.03) (2015).* [Zugriff am 30.08.2016]. URL: www.opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model.

[63] OPC Foundation. *Part 4 - Services (Version 1.03) (2015).* [Zugriff am 30.08.2016]. URL: www.opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-services.

[64] OPC Foundation. *Part 5 - Information Model (Version 1.03) (2015).* [Zugriff am 30.08.2016]. URL: www.opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model.

[65] OPC Foundation. *Part 6 - Mappings (Version 1.03) (2015).* [Zugriff am 30.08.2016]. URL: www.opcfoundation.org/developer-tools/specifications-unified-architecture/part-6-mappings.

[66] OPC Foundation. *Part 12 - Discovery (Version 1.03) (2015).* [Zugriff am 30.08.2016]. URL: www.opcfoundation.org/developer-tools/specifications-unified-architecture/part-12-discovery.

[67] Unified Automation GmbH. *Discovery and Security Configuration (2016).* [Zugriff am 10.09.2016]. URL: www.documentation.unified-automation.com/uasdkcpp/1.5.3/html/L2UaDiscoveryConnect.html.

[68] OPC Foundation. *Server Discovery (2015).* [Zugriff am 10.09.2016]. URL: www.wiki.opcfoundation.org/index.php/Server_Discovery.

[69] OPC Foundation. *Local Discovery Server (LDS) (2016).* [Zugriff am 10.09.2016]. URL: www.opcfoundation.org/developer-tools/developer-kits-unified-architecture/local-discovery-server-lds/.

[70] H. Vonhoegen. *Einstieg in XML - Grundlagen, Praxis, Referenz*. Bonn: Galileo Computing, 2011. ISBN: 978-3-8362-2620-2.

[71] W. Mahnke, S.-H. Leitner. *OPC Unified Architecture - Der zukünftige Standard für Kommunikation und Informationsmodellierung in der Automatisierung (2009)*. [Zugriff am 18.09.2016]. URL: www.library.e.abb.com/public/43fd4ef093cb19c0c125762f006cbd96/56-61%203M903_GER72dpi.pdf.

[72] Siemens Aktiengesellschaft. *Simatic NET OPC UA Server (2011)*. [Zugriff am 19.09.2016]. URL: www.cache.industry.siemens.com/dl/files/467/26548467/att_15741/v1/26548467_opcua_ac_doku_v10_d.pdf.

[73] Unified Automation GmbH. *Address Space Concepts (2016)*. [Zugriff am 10.09.2016]. URL: www . documentation . unified - automation . com / uasdkcpp / 1 . 5 . 2 / html / L2UaAddressSpaceConcepts.htm.

[74] Prosys PMS Ltd. *What is OPC and OPC UA?* [Zugriff am 10.09.2016]. URL: www.prosysopc.com/opc/.

**Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift