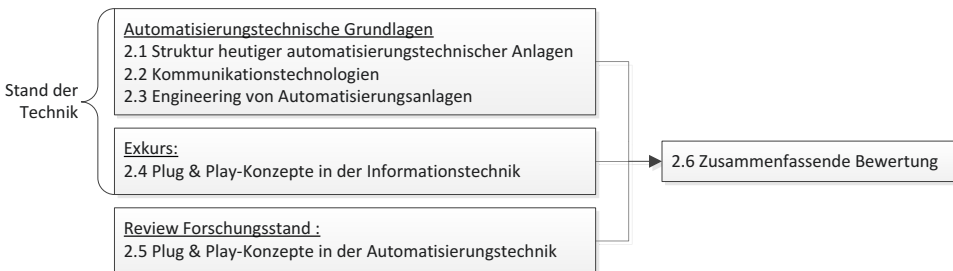


# Kapitel 2

## Stand der Technik und Forschung

Das folgende Kapitel soll einerseits einen Überblick über die technischen Grundlagen dieser Arbeit geben und andererseits in den relevanten aktuellen Forschungsstand einführen. Die Gliederung der einzelnen Abschnitte ist in Abbildung 2.1 dargestellt.



**Abbildung 2.1:** Überblick über die Gliederung von Kapitel 2

Die Abschnitte 2.1 bis 2.3 beschreiben die für diese Arbeit relevanten automatisierungstechnischen Grundlagen: Der Abschnitt 2.1 dient dazu, die Thematik dieser Arbeit in die Struktur automatisierungstechnischer Anlagen einordnen zu können. Der Schwerpunkt von Abschnitt 2.2 liegt auf einer Einführung in die Echtzeitnetzwerke, für welche in dieser Arbeit ein Verfahren zur automatischen Konfiguration entwickelt wird. In Abschnitt 2.3 wird ein Überblick über die notwendigen Engineering-Aufwände für die Inbetriebnahme automatisierungstechnischer Anlagen gegeben. Dieser Abschnitt zeigt,

dass der in der vorliegenden Arbeit behandelte Engineering-Schritt „Inbetriebnahme des Echtzeitnetzwerkes“ nur einen Teilschritt im kompletten Engineering-Vorgang einer solchen Anlage darstellt.

Zur Wiedergabe des Standes der Technik dient neben den genannten Abschnitten ebenfalls der Abschnitt 2.4. Dieser zeigt anhand zweier konkreter Technologien, wie das „Plug and Play“-Prinzip aktuell innerhalb der Informationstechnik realisiert wird. Weiterhin wird geprüft, welche für die Realisierung von „Plug and Play“ eingesetzten Methoden sich auf die Automatisierungstechnik übertragen lassen.

Welche Ansätze es zur Realisierung des „Plug and Play“-Prinzips in der Automatisierungstechnik gibt, wird in Abschnitt 2.5 dargestellt. Hier wird insbesondere die bereits in der Einleitung erwähnte Lücke im aktuellen Forschungsstand herausgearbeitet, aus welcher sich die Motivation zur vorliegenden Arbeit ergibt. Dieses Kapitel schließt in Abschnitt 2.6 mit einer Zusammenfassung der offenen Problemstellungen, welche sich aus dem Stand der Technik und Forschung ergeben.

## 2.1 Struktur heutiger automatisierungstechnischer Anlagen

Die Architektur heutiger automatisierungstechnischer Systeme lässt sich durch die sogenannte Automatisierungspyramide veranschaulichen, welche in Abbildung 2.2 dargestellt ist.

Die Grundlage eines automatisierungstechnischen Systems bildet der zu automatisierende physikalische Prozess auf der **Prozessebene**. Dabei kann es sich um fertigungstechnische Prozesse handeln, in denen geometrisch abgeschlossene, feste Körper (Stückgüter) produziert werden, oder um verfahrenstechnische Prozesse, in denen Fließgüter wie Flüssigkeiten oder Gase hergestellt werden [Weber 1999].

Auf der **Feldebene** überwachen Sensoren den Prozess bzw. greifen Aktoren in den Prozess ein. In einem engen Zusammenhang mit der Sensorik/Aktorik steht die **Steuerungsebene**, in der ein Steuerungselement –in der Regel eine SPS– die Werte der Sensoren einliest und anhand einer vordefinierten Steuerungslogik die neuen Werte für die Aktoren berechnet. Die Logik wird üblicherweise in einer Programmiersprache gemäß der Norm IEC 61131-3 [IEC 61131-3] formuliert. Der Datenaustausch zwischen Sensor, SPS und Aktor erfolgt dabei zyklisch und muss –je nach physikalischem Prozess– hohen Anforderungen z. B. an die Zykluszeit und dem Jitter genügen. Die Elemente der Feld-

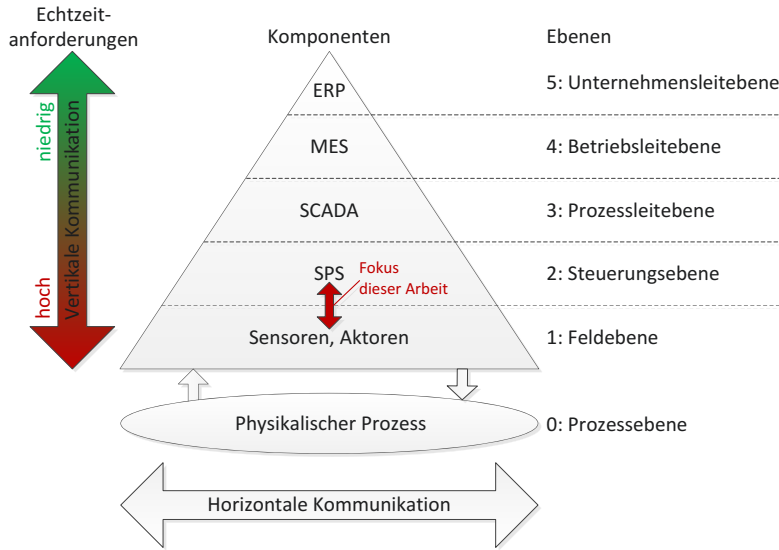


Abbildung 2.2: Automatisierungspyramide (nach [Haag 2013])

und Steuerungsebene werden auch als Prozessnahe Komponenten (PNKs) bezeichnet. PNKs unterscheiden sich von den Komponenten der folgenden Ebenen im Wesentlichen durch ihre deterministischen kurzen Reaktionszeiten [Maier 2009].

Die auf der **Prozessleitebene** angesiedelten Supervisory Control and Data Acquisition (SCADA)-Systeme beinhalten Anzeige- und Bedienkomponenten zur manuellen Überwachung und Steuerung des physikalischen Prozesses. Sie stellen die Schnittstelle zwischen dem automatisierungstechnischen System und dem Bediener der Anlage dar. Die Realisierung von SCADA-Systemen erfolgt im Gegensatz zu PNKs in der Regel als Softwarekomponente, die auf kommerziell verfügbaren PCs mit Windows oder Linux-Betriebssystem ausgeführt werden kann.

Die Manufacturing Executions Systems (MESs) der **Betriebsleitebene** stellen die Schnittstelle zwischen den technisch geprägten unteren Ebenen und der betriebswirtschaftlichen Steuerung der Produktion dar. MESs zeichnen sich für die logistische Steuerung eines lokalen Produktionsprozesses verantwortlich, in dem sie beispielsweise die Feinplanung des Materialflusses oder des Betriebsmitteleinsatzes übernehmen. Weiterhin erfassen MESs statistische produktionsbezogene Daten wie Maschinenauslastung und -verfügbarkeit.

Auf der **Unternehmensleitebene** stellt die Enterprise Resource Planning (ERP)-Soft-

ware ähnliche Funktionen wie MESs bereit, allerdings wird anstelle eines lokalen Produktionsprozesses die Gesamtheit aller Produktions- und Geschäftsprozesse eines Unternehmens betrachtet. So umfasst ein ERP-System die komplette Materialwirtschaft und Produktionsplanung einer Firma. Es kann darüber hinaus auch Komponenten für das Finanz- und Rechnungswesen oder das Controlling enthalten. Ein prominentes Beispiel für eine entsprechende Software, welche hauptsächlich von Großunternehmen eingesetzt wird, ist SAP ERP (vormals SAP R/3).

Aus Sicht der Kommunikationstechnik kann zwischen horizontaler und vertikaler Kommunikation innerhalb einer automatisierungstechnischen Anlage unterschieden werden. Der Begriff Horizontal bezeichnet die Kommunikation zwischen den Elementen derselben Ebene. Aufgrund der vorherrschenden zentralistischen Struktur in der Automatisierungstechnik überwiegt jedoch die vertikale Kommunikation zwischen den Komponenten verschiedener Ebenen. Dabei ist häufig nur ein Datenaustausch zwischen zwei benachbarten Ebenen über definierte Schnittstellen möglich.

Die Anforderungen an diese Schnittstellen unterscheiden sich hauptsächlich durch die jeweiligen Echtzeitanforderungen, aus denen sich dann die jeweils eingesetzten Kommunikationstechnologien ableiten. Wie in Abbildung 2.2 dargestellt ist, nehmen die zeitlichen Anforderungen mit steigender Entfernung einer Ebene vom physikalischen Prozess ab. Die höchste Anforderung an die Kommunikationstechnik stellt die Schnittstelle zwischen Feld- und Steuerungsebene. Die Übertragung von Prozessdaten zwischen Sensor, SPS und Aktor muss bei besonders zeitkritischen Anwendungen in Zykluszeiten der Größenordnung einiger zehn Mikrosekunden unter Einhaltung eines Jitters von maximal einer Mikrosekunde erfolgen. Die dafür eingesetzten Kommunikationstechnologien, welche im Fokus dieser Arbeit stehen, werden im Abschnitt 2.2 näher beschrieben.

Die Schnittstellen zwischen den höheren Ebenen können durch den Einsatz der in der Bürokommunikation üblichen Kombination von TCP/IP und Ethernet realisiert werden. So kommunizieren SCADA-Systeme mit einer SPS häufig unter Nutzung des OPC-Standards, welcher auf Microsofts TCP/IP-basiertem Distributed Component Object Model (DCOM) beruht.

## 2.2 Kommunikationstechnologien

Wie in Abschnitt 2.1 beschrieben wurde, zeichnet sich die Schnittstelle zwischen Feld- und Steuerungsebene durch die hohen zeitlichen Anforderungen an die Kommunikation

aus, wodurch die Nutzung echtzeitfähiger Kommunikationstechnologien erforderlich wird. Aus diesem Grund wurden in der Automatisierungstechnik spezielle Feldbusse entwickelt, welche die Anbindung verschiedener Feldgeräte an eine SPS unter gemeinsamer Nutzung desselben physikalischen Übertragungsmediums erlauben. Die aufwändige Punkt-zu-Punkt Verkabelung zwischen Feldgerät und Steuerung konnte dadurch entfallen. Parallel zur Einführung der Feldbusse haben sich im IT-Bereich Ethernet und der TCP/IP-Protokollstapel als Standards für lokale Netzwerke durchgesetzt. Aufgrund einer kontinuierlichen Weiterentwicklung und der großen Verbreitung hat Ethernet in der Zwischenzeit deutliche Leistungs- und Kostenvorteile gegenüber den verschiedenen Feldbussen erreicht, wodurch es auch zunehmend ein Kandidat für die Vernetzung in der Feldebene der Automatisierungstechnik wurde. Durch die fehlende Echtzeitfähigkeit wurden jedoch Anpassungen am Ethernet-Standard erforderlich.

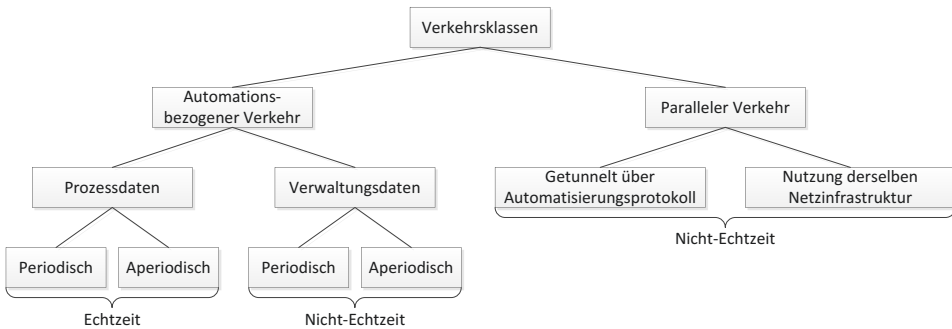
Dieser Abschnitt gibt einen Überblick über die in der Automatisierungstechnik verwendeten Kommunikationstechnologien. Die generelle Architektur industrieller Kommunikationssysteme wird in Unterabschnitt 2.2.1 beschrieben, es folgt eine kurze Einführung in Feldbusse (Unterabschnitt 2.2.2) und eine ausführlichere Darstellung der Grundlagen von industriellen Echtzeit-Ethernets in Unterabschnitt 2.2.3. Konkrete Echtzeit-Ethernet-Varianten werden in den darauf folgenden Unterabschnitten dargestellt. In dieser Arbeit wird sich dabei auf die nach [Morse 2012] fünf in der Automatisierungstechnik verbreitetsten Varianten beschränkt: Profinet RT (Unterabschnitt 2.2.4), Profinet IRT (Unterabschnitt 2.2.5), Ethernet/IP (Unterabschnitt 2.2.6), Powerlink (Unterabschnitt 2.2.7) und Ethercat (Unterabschnitt 2.2.8). Dabei liegt der Schwerpunkt auf der notwendigen manuellen Konfiguration dieser Netzwerke, welche durch die in dieser Arbeit vorgestellten Methoden möglichst automatisiert werden soll.

### **2.2.1 Grundlagen industrieller Kommunikationssysteme**

Die verschiedenen in der Automatisierungstechnik eingesetzten echtzeitfähigen Kommunikationssysteme können im Wesentlichen in die beiden Gruppen der Feldbusse und der Ethernet-basierten Technologien unterteilt werden. Aufgrund derselben zugrundeliegenden Anwendungsdomänen zeichnen sich beide Varianten durch gemeinsame Merkmale aus, von denen die für diese Arbeit relevanten Eigenschaften in diesem Unterabschnitt beschrieben werden.

## Verkehrsklassen

Die zentrale Aufgabe industrieller Kommunikationssysteme in der Automatisierungstechnik ist die Vernetzung von Komponenten der Feld- (Sensoren und Aktoren) und der Steuerungsebene (SPS). Dementsprechend liegt das Hauptaugenmerk bei der Gestaltung derartiger Systeme auf der effizienten Echtzeit-Übertragung von typischerweise kleinen Paketen an Prozessdaten unter Berücksichtigung der eingeschränkten Rechenressourcen von Feldgeräten. Wie in Abbildung 2.3 gezeigt ist, gibt es neben den primären Prozessdaten noch weitere Klassen an Verkehrsdaten, welche über automatisierungstechnische Netzwerke übertragen werden müssen. [Sauter 2015] unterteilt die Daten dabei in die folgenden Klassen:



**Abbildung 2.3:** Verkehrsklassen in automatisierungstechnischen Netzwerken (nach [Sauter 2015])

- Bei der Oberklasse *Automationsbezogener Verkehr* handelt es sich um den Datenverkehr, der direkt mit dem zu steuernden physikalischen Prozess in Verbindung steht. Diese Oberklasse kann wiederum in zwei Unterklassen unterteilt werden:
  - *Prozessdaten* bezeichnen die Daten, die den aktuellen Zustand des physikalischen Prozesses repräsentieren. Der von der Sensoren erfasste Prozesszustand wird an die SPS übermittelt, welche auf Basis ihrer Steuerungslogik einen neuen Prozesszustand berechnet und diesen an die Aktoren ausgibt. Die Übermittlung der Prozessdaten über das Kommunikationssystem muss in der Regel bestimmten zeitlichen Anforderungen genügen, welche sich aus dem physikalischen Prozess ergeben. Prozessdaten können in zwei Klassen unterteilt werden:
    - \* *Periodische –bzw. zyklische– Daten* werden in gleichbleibenden Abständen

den zwischen Feld- und Steuerungsebene übertragen. Die Zykluszeit wird in der Regel vor der Inbetriebnahme des automatisierungstechnischen Systems projiziert, wobei für jede einzelne zu übertragende Information ein bestimmter Anteil an der zur Verfügung stehenden Übertragungsbandbreite reserviert werden kann.

- \* *Aperiodische –bzw. azyklische– Daten* dienen zur Übermittlung von Ereignis-basierten Informationen wie z. B. Alarmmeldungen. Damit diese innerhalb eines vorgegebenen Zeitfensters übertragen werden können, muss ggf. Bandbreite für diese Datenklasse reserviert werden.
- *Verwaltungs- bzw. Konfigurationsdaten* dienen zur Steuerung des automatisierungstechnischen Systems an sich – im Gegensatz zu Prozessdaten, deren Zweck die Steuerung des physikalischen Prozesses ist. Beispielsweise gehören Parameter zur Geräte- oder Netzwerkkonfiguration zu dieser Klasse. Solche Daten werden in der Regel aperiodisch gesendet, in bestimmten Anwendungsfällen können jedoch auch periodische Daten (z. B. regelmäßige Statusmeldungen einer automatisierungstechnischen Komponente) übermittelt werden. Die Übertragung muss dabei keinen Echtzeitanforderungen genügen.
- Der *parallele Verkehr* steht nicht in erster Linie im Zusammenhang mit dem automatisierungstechnischen Prozess und wird von anderen Prozessen erzeugt, die parallel zum automatisierungstechnischen System dieselbe Netzwerkinfrastruktur nutzen. Typischerweise handelt es sich dabei um IP-Kommunikation zwischen Büro- und automatisierungstechnischen Komponenten.

Die dominierende Rolle innerhalb dieser Klassen wird von den zyklischen Prozessdaten eingenommen [Gaj u. a. 2013], welche entsprechend im Mittelpunkt dieser Arbeit stehen.

### Kommunikationsmuster

Während die Verkehrsklassen die über das Automatisierungsnetzwerk übertragenden Daten nach ihren Eigenschaften und dem Verwendungszweck klassifizieren, beschreiben Kommunikationsmuster die Art der Kooperation zwischen miteinander kommunizierenden Komponenten. Nach [Thomesse 2005] können dabei in der Automatisierung zwei verschiedene Rollen unterschieden werden.

In einem **Client-Server Modell** werden Dienste von einem Server angeboten und von einem Client in Anspruch genommen. Jeder Kommunikationsvorgang muss von

einem Client angestoßen werden, sodass sich dieses Modell für Anwendungen eignet, bei denen Informationen bei Client-seitigen Ereignissen von einem Server abgerufen werden müssen.

Im **Publisher-Subscriber Modell** sendet eine Daten produzierende Komponente (Publisher) ihre Daten per Multicast oder Broadcast an eine oder mehrere Daten konsumierende Komponenten (Subscriber). Dementsprechend wird es auch Producer-Consumer Modell genannt. Der Vorteil des Publisher-Subscriber Modells liegt aufgrund der Multi- bzw. Broadcast-Übermittlung in der simultanen Verteilung von Informationen an alle Netzwerkteilnehmer.

### 2.2.2 Feldbusse

Die Entwicklung der heute unter dem Begriff Feldbus zusammengefassten Netzwerktechnologien begann Mitte der 1980er-Jahre unter der Motivation, die aufwändigen und verkabelungsintensiven Punkt-zu-Punkt Verbindungen zwischen digitalen oder analogen Ein-/Ausgabegeräten und den zentralen Steuerungen zu ersetzen. Durch den Einsatz eines Feldbusses können stattdessen alle Komponenten an ein gemeinsames Kommunikationsmedium angeschlossen werden. In der Norm IEC 61158 [ISO/IEC 61158 2010] wird der Begriff Feldbus wie folgt definiert:

„A fieldbus is a digital, serial, multidrop, data bus for communication with industrial control and instrumentation devices such as –but not limited to– transducers, actuators and local controllers.“

Im Gegensatz zu den universell einsetzbaren lokalen Netzwerken aus dem IT-Umfeld wie Ethernet stellen Feldbusse eine umfassende domänenspezifische Technologie dar. So beinhaltet das Lösungsangebot eines Feldbus-Herstellers oft nicht nur die Bereitstellung des eigentlichen Kommunikationsmediums, sondern auch eine Palette von Werkzeugen zur Realisierung von komplexen Automatisierungsanwendungen. Durch diese Unterstützung von aufeinander abgestimmten Soft- und Hardwarekomponenten wird einerseits der Entwicklungsprozess erleichtert, andererseits wird die Flexibilität eingeschränkt, da ein eventueller Wechsel auf ein anderes Feldbussystem den Austausch ganzer Werkzeugketten nach sich ziehen kann.

Zwei weitere Punkte, in denen sich Feldbusse aufgrund ihres Anwendungszwecks wesentlich von IT-Netzwerken unterscheiden, sind die bereits erwähnte Echtzeitfähigkeit und die Verwendung eines einfacheren Protokollstapels. Letzteres liegt zum Einen darin begründet, dass in der Feldebene oft eingebettete Systeme zum Einsatz kommen,

deren eingeschränkte Ressourcen zumindest zu Beginn der Feldbuseinführung nicht für ein komplexes Protokollhandling ausreichen. Zum anderen hat ein umfangreicher Protokollstapel ebenfalls negative Auswirkungen auf das Echtzeitverhalten (siehe auch Unterabschnitt 2.2.3).

Aufgrund der genannten unterschiedlichen Anforderungen verlief die Entwicklung von Standards für die lokale Vernetzung in den beiden Bereichen Automatisierungs- und Informationstechnik lange Zeit getrennt voneinander. Während sich in der IT Ethernet durchsetzen konnte und Konkurrenten wie Token Bus oder Token Ring fast vollständig verdrängt hat, sind in der Automatisierungstechnik bis heute eine Vielzahl unterschiedlicher, nicht zueinander kompatibler, Feldbusstandards im Einsatz.

Seit Anfang der 2000er-Jahre wurde jedoch auch Ethernet aufgrund technologischer Fortschritte zunehmend ein Kandidat für die Vernetzung auf der Feldebene. Die Entwicklung und der aktuelle Stand der Technik beim Einsatz von Ethernet in der Automatisierungstechnik wird in Unterabschnitt 2.2.3 behandelt.

### **2.2.3 Echtzeit-Ethernet**

Wie in der Einleitung zu Abschnitt 2.2 bereits erwähnt, hat sich parallel zur Einführung der Feldbusse Ethernet als dominierender Standard zur lokalen Vernetzung von Rechnersystemen etabliert. Die in der Informationstechnik im Vergleich zur Automatisierungstechnik kürzeren Innovationszyklen haben sich auch in der Weiterentwicklung der verschiedenen Kommunikationsstandards bemerkbar gemacht: Während in Ethernet ursprünglich nur physikalische Übertragungsschichten für eine Datenrate von 10 Mbit/s definiert waren, enthält der aktuelle Standard [IEEE 802.3 2012] auch Definitionen für Datenraten bis zu 100 Gbit/s. Ähnliche kontinuierliche Weiterentwicklungen im Feldbus-Bereich haben hingegen nicht stattgefunden, sodass beispielsweise Profibus seit der erstmaligen Standardisierung bis heute eine Bitrate von 12 Mbit/s unterstützt. Weiterhin erzielen Ethernet-Komponenten durch ihre hohe Verbreitung Kostenvorteile gegenüber Feldbus-Komponenten. Diese Gründe haben zusammen mit der Erfordernis einer einfacheren Integration der Netzwerktechnologien aus Automatisierungs- und Informationstechnik zur Einführung von Ethernet auch in der industriellen Automation geführt [Gaj u. a. 2013].

Zur Realisierung der Echtzeitfähigkeit waren jedoch Modifikationen am Ethernet-Standard notwendig, was in der Folge zur Einführung mehrerer verschiedener, zueinander nicht kompatibler Echtzeit-Ethernet (RTE) Varianten geführt hat. Die Ursachen für

die nicht vorhandene Echtzeitfähigkeit von Ethernet und mögliche Lösungsansätze sind im Folgenden dargestellt. Der Begriff RTE bezeichnet in dieser Arbeit grundsätzlich die in der Automatisierungstechnik eingesetzten Echtzeit-Ethernets.

### **Anforderungen und Lösungsansätze**

Eine entscheidende technologische Weiterentwicklung von Ethernet in Bezug auf die Eignung für den Einsatz in der Automatisierungstechnik war die Einführung von voll-duplexfähigen physikalischen Punkt-zu-Punkt Verbindungen über Switches [Wang u. a. 2002]. Die für die Verkabelung in der Automatisierungstechnik typische Linienstruktur kann dabei durch die Integration von Zwei-Port-Switches in die Feldgeräte beibehalten werden [Jasperneite u. Neumann 2001]. Durch diesen Schritt können keine zufallsabhängigen Kollisionen mehr entstehen, wie sie vorher noch durch den gleichzeitigen Sendezugriff mehrerer Stationen auf das geteilte Kommunikationsmedium entstanden sind. Dennoch kann es auch in einem geschwichteten Ethernet zu nichtdeterministischen Verzögerungen kommen: Wenn mehrere Pakete in einem Switch gleichzeitig an einen Zielport weitergeleitet werden müssen, so werden diese je nach Architektur des Switches entweder in einem Zwischenspeicher gepuffert oder verworfen. In Überlastsituation müssen auch Switches mit Sendepuffer die nicht mehr speicherbaren Pakete verwerfen. Die Echtzeitfähigkeit ist daher der entscheidende Faktor bei der Beantwortung der Frage, ob Ethernet für den Einsatz in der Automatisierungstechnik geeignet ist.

Für entsprechende Bewertungen muss zuerst definiert werden, welche Echtzeit-Anforderungen an ein industrielles Kommunikationssystem gestellt werden. Die maßgeblichen Kriterien, welche sich beide auf den zyklischen Prozessdatenaustausch zwischen Sensor, Steuerung und Aktor beziehen, sind (i) die Latenzzeit und (ii) der Jitter (siehe Abbildung 2.4). Die Latenz in Form der Eingabe/Ausgabe-Latenz beschreibt das Zeitintervall zwischen der Erfassung eines Sensor-Eingangswertes und des Setzens der jeweiligen durch die Steuerungslogik verknüpften Aktor-Ausgangswerte. Der Latenzbegriff lässt sich der Anzahl der beteiligten Übertragungsschritte entsprechend weiter verfeinern (beispielsweise durch das Übertragungsmedium verursachte Latenz für die Übermittlung eines Prozesswertes vom Sensor zur Steuerung).

Jeder zyklische Vorgang ist weiterhin zeitlichen Schwankungen unterworfen. Diese mit Jitter bezeichnete Eigenschaft beschreibt in Bezug auf die Eingabe/Ausgabe-Latenz die Varianz des Zeitintervalls zwischen Prozesswerterfassung und -ausgabe. Zur Klassifizierung der jeweiligen Echtzeit-Anforderungen gemäß den Kenngrößen Latenz und Jitter haben sich drei Klassen etabliert [Jasperneite u. Neumann 2004; Felser 2005], welche in

Tabelle 2.1 dargestellt sind.

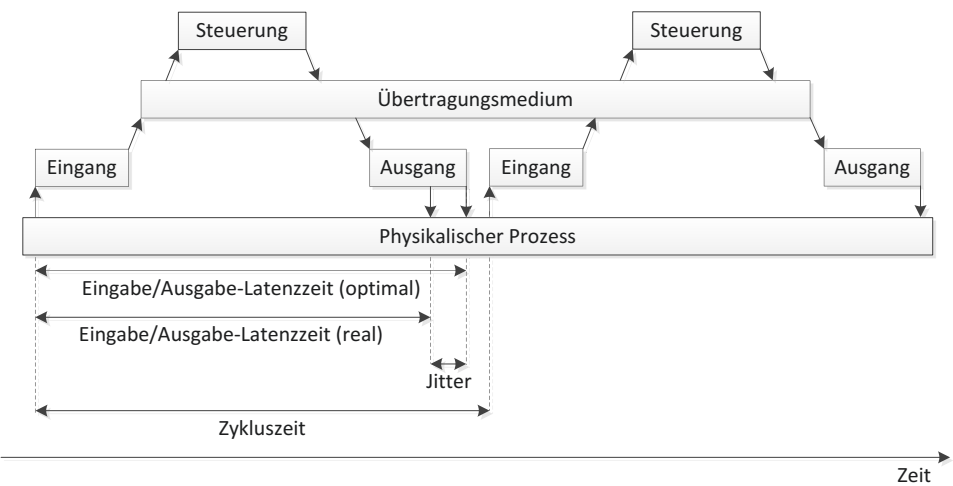


Abbildung 2.4: Latenzzeit, Jitter und Zykluszeit

Tabelle 2.1: Echtzeitklassen für industrielle Kommunikationssysteme

Echtzeit-Klasse	Anwendungen	Latenzzeit	Jitter
1	Manuelle Prozesssteuerung, Prozessüberwachung, SPS-zu-SPS Kommunikation	10 - 100 ms	-
2	Automatische Prozesssteuerung, SPS-zu-Sensor/Aktor Kommunikation	1 - 10 ms	≤ 1 ms
3	Bewegungsregelung	< 1 ms	≤ 1 μs

In die Echtzeitklasse 1 werden u. a. Anwendungen eingeteilt, die auf der Interaktion mit menschlichen Anwendern beruhen. Ein Beispiel sind SCADA-Systeme, welche über Mensch-Maschine-Schnittstellen den Zustand eines überwachten physikalischen Prozesses visualisieren. Weiterhin kann der Anwender über diese Schnittstellen selber aktiv in den Prozess eingreifen. Die Kommunikation zwischen dezentralen Peripheriegeräten wie der Sensorik/Aktorik und den Steuerungen muss in der Regel Echtzeitanforderungen der Klasse 2 genügen. Die höchsten Anforderungen an die Kommunikation werden in der Klasse 3 gestellt. Eine typische Anwendung sind hier Bewegungsregelungen, bei

denen mehrere über vernetzte Motoren angetriebene Achsen synchron angesteuert werden müssen.

In Simulationsstudien wurde gezeigt, dass Ethernet in der Regel die Echtzeitanforderungen der Klasse 2 erfüllen kann, sofern bei einer Netzlast größer als 50 % ein prioritätsbasiertes Schedulingverfahren innerhalb der Switches eingesetzt wird [Jasperneite u. Neumann 2001; Jasperneite u. a. 2002]. Diese Art der Flusssteuerung kann basierend auf der Priorisierung von Ethernet-Paketen gemäß der Ethernet-Erweiterung IEEE 802.1Q [IEEE 802.1Q 2014] umgesetzt werden. Die Norm führt ein drei Bit breites *Priority code point* genanntes Feld im Ethernet-Header ein, über welches jedem Paket eine von acht Prioritätsklassen zugewiesen werden kann. Ein IEEE 802.1Q-kompatibler Switch enthält für jeden Ausgangsport entsprechend bis zu acht Ausgangspuffer. Eingehende Pakete werden in dem zu Ausgangsport und Prioritätsklasse korrespondierenden Puffer einsortiert. Vor Beginn eines Sendevorganges wird das älteste Paket des höchstpriorisierten Puffers des entsprechenden Ports als nächstes zu übertragendes Paket ausgewählt [Decotignie 2009].

Durch die Priorisierung von Paketen lassen sich zeitkritische Daten in Ethernet gut von Daten ohne entsprechende zeitlichen Anforderungen isolieren. Allerdings ist es durch dieses Verfahren nicht möglich, bestimmte zeitliche Obergrenzen für die Übertragungszeiten einzelner Pakete zu garantieren. So kann auch die Weiterleitung eines einzelnen Pakets höchster Priorität in einem Switch verzögert werden, wenn der entsprechende Ausgangsport durch die Übertragung eines Pakets mit niedrigerer Priorität belegt ist. In den genannten Simulationsstudien wurde gezeigt, dass die maximale Übertragungszeit von priorisierten Paketen stark von der Netztopologie und der Anzahl an Stationen abhängig ist. Bei den dort verwendeten Szenarien konnten die Anforderungen der Echtzeitklasse 2 ab einer Anzahl von 50 Stationen bzw. einer Netzlast von 95 % nicht mehr eingehalten werden.

Bei der Betrachtung von Latenzzeiten für die Übertragung von Daten zwischen zwei Netzwerknoten muss weiterhin berücksichtigt werden, dass nur ein geringer Teil der Latenz auf die für die physikalische Übertragung benötigte Zeit zurückzuführen ist. Zusätzlich müssen die Verzögerungen innerhalb der Knoten betrachtet werden. Insbesondere der im Zusammenhang mit Ethernet standardmäßig eingesetzte TCP/IP-Protokollstapel beansprucht durch seine Komplexität hohe Rechenkapazitäten der jeweiligen Knoten, sodass 80-90 % der Gesamtlatenz auf die Bearbeitungszeit der Datenpakete innerhalb der an der Kommunikation beteiligten Knoten zurückzuführen ist [Skeie u. a. 2006].

Die meisten etablierten RTE-Varianten setzen die Priorisierung von Paketen als Mög-

lichkeit zur Realisierung von Anforderungen der Echtzeitklasse 2 ein. Zusätzlich nutzen sie eine eigene Applikationsschicht, welche direkt mit der Ethernet-Ebene kommuniziert. Durch die Auslassung der TCP/IP-Protokolle können die erwähnten Verzögerungen innerhalb der Knoten maßgeblich reduziert werden. Die damit einhergehenden Funktionalitätseinschränkungen sind in industriellen Anwendungen in der Regel akzeptabel, beispielsweise ist ein Routing von echtzeitkritischen Prozessdaten meist nicht notwendig.

Die Anforderungen der Echtzeitklasse 1 bzw. die garantierte Einhaltung von Zeitschranken sind jedoch mit Ethernet aufgrund der nicht vorhersagbaren Verzögerungen innerhalb der Switches nicht realisierbar. Daher wurden von verschiedenen Herstellern von Automatisierungskomponenten Modifikationen am Medienzugriff durch die Einführung von Zeitmultiplexverfahren (Time Division Multiple Access (TDMA)) vorgenommen. In TDMA erfolgt der Zugriff auf das Kommunikationsmedium zeitgesteuert, wobei ein Knoten nur die ihm zugeteilten Zeitschlitz für das Senden von Daten benutzen darf. Dieses Verfahren erlaubt die Planung des gesamten Kommunikationsverhaltens (Scheduling) in einem Netzwerk vor der Inbetriebnahme und damit die Vorhersage der Übertragungszeiten. Damit sich alle Teilnehmer an den vorgegebenen Kommunikationsplan halten, müssen weiterhin Verfahren zur Zeitsynchronisation eingesetzt werden. Die Umsetzung des zeitgesteuerten Medienzugriffs unterscheidet sich in den verschiedenen RTEs und wird in den entsprechenden Abschnitten genauer beschrieben.

### **Klassifizierung von Echtzeit-Ethernets**

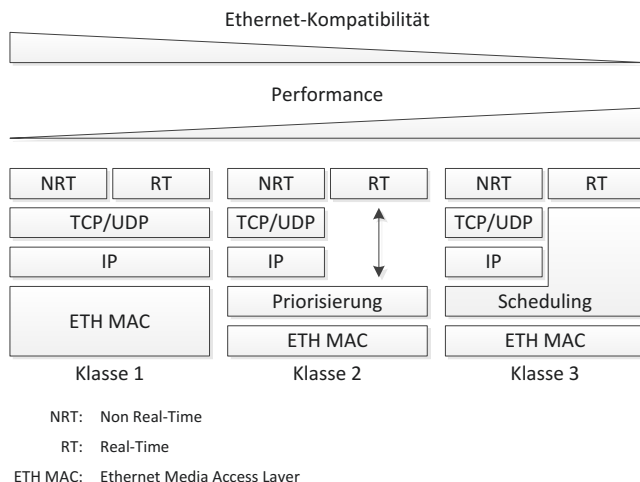
Zusammengefasst lassen sich RTEs je nach Realisierung der Echtzeitunterstützung in drei Klassen einteilen, welche in Abbildung 2.5 dargestellt sind. In Klasse 1 wird die auch in der Informationstechnik übliche Kombination von Ethernet und TCP/IP eingesetzt. RTEs dieser Klasse stellen daher keine besonderen Anforderungen an die verwendete Hard- und Software.

In der Klasse 2 wird ein getrennter Kommunikationspfad für den Echtzeit (Real-Time (RT))-Verkehr eingeführt. Für die Realisierung der Priorisierung auf Ethernet-Ebene nach IEEE 802.1Q müssen die eingesetzten Switches diesen Standard unterstützen.

Die RTEs der Klasse 3 ermöglichen einen deterministischen Datenaustausch durch eine TDMA-basierte Kommunikationsplanung. Da zu diesem Zweck Modifikationen der Medienzugriffsebene (Media Access Layer (MAC)) von Ethernet notwendig sind, müssen RTE-spezifische Hardware-Komponenten verwendet werden.

Nicht echtzeitkritische (Non Real-Time (NRT)) Applikationen können in allen RTE-

Klassen weiterhin die Funktionalitäten des TCP/IP-Stacks zur Kommunikation nutzen. Das jeweils eingesetzte RTE ist dabei für die NRT-Applikation transparent.



**Abbildung 2.5:** Klassifizierung von Echtzeit-Ethernet-Protokollen [Jasperneite u. Neumann 2004]

## 2.2.4 Profinet RT

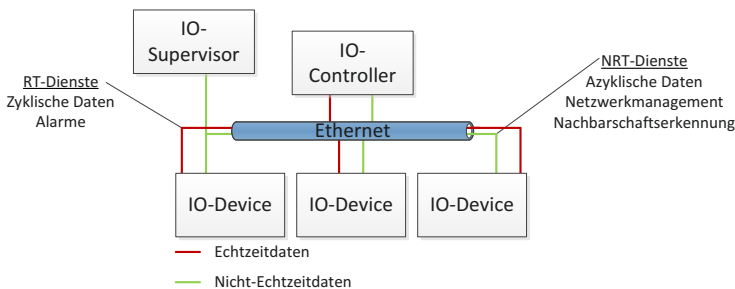
Profinet ist ein Standard, welcher maßgeblich von dem Dachverband Profibus & Profinet International (PI) entwickelt wird. Es ist der Ethernet-basierte Nachfolger des Feldbusses Profibus und lässt sich grundsätzlich in zwei verschiedene Architekturkonzepte aufteilen: Profinet Component Based Automation (CBA) sollte zur Realisierung der Kommunikation zwischen Steuerungen in verteilten automatisierungstechnischen Anlagen eingesetzt werden, was innerhalb der Automatisierungspyramide (Abbildung 2.2) einer horizontalen Kommunikation auf Ebene 2 entspricht. Allerdings konnte sich Profinet CBA nicht am Markt durchsetzen [HMS Industrial Networks 2010] und wird in dieser Arbeit nicht weiter betrachtet.

Für die vertikale Kommunikation zwischen Steuerungen und dezentraler Peripherie auf der Feldebene ist dagegen die Variante Profinet IO gedacht, bei welcher es sich um den verbreitetsten RTE-Standard handelt. Hinsichtlich der unterstützten Echtzeit-Klasse kann Profinet IO wiederum in zwei Kategorien unterteilt werden: In den von der PI festgelegten Konformitätsklassen A und B werden die Anforderungen der Klasse 2 erfüllt. Diese Ausprägung von Profinet IO wird im Folgenden als Profinet RT bezeichnet und in diesem Unterabschnitt beschrieben. Zertifizierte Geräte der Konformitätsklasse C

erfüllen Echtzeit-Anforderungen der Klasse 3. Diese zeitsynchronisierte Variante wird als Profinet Isochronous Real-Time (IRT) in Unterabschnitt 2.2.5 behandelt. Sollen gemeinsame Eigenschaften von Profinet RT und IRT beschrieben werden, wird im Folgenden der Begriff Profinet ohne Zusatzbezeichnung verwendet.

## Architektur

Profinet folgt dem Producer-Consumer Modell und definiert drei verschiedene Geräteklassen (Abbildung 2.6): Ein IO-Controller produziert Ausgangsdaten in Richtung IO-Geräte und konsumiert Eingangsdaten von den IO-Geräten. In der Regel beinhaltet ein IO-Controller eine Laufzeitumgebung für die Steuerungslogik, er ist daher meist identisch mit der SPS eines Automatisierungssystems. Auf der Feldebene sind die IO-Geräte angeordnet, welche entweder die für die Steuerung des Prozesses notwendige Sensorik und Aktorik direkt enthalten oder zumindest elektrische Schnittstellen für deren Anschluss bereit stellen. Der IO-Supervisor wird zu Inbetriebnahme-, Konfigurations- oder Überwachungszwecken eingesetzt, er ist daher nicht direkt in die Prozesssteuerung involviert.



**Abbildung 2.6:** Geräteklassen und Dienste in Profinet

Zur Übertragung von Daten werden in Profinet zwei logische Kanäle eingesetzt. Über den NRT-Kanal werden Dienste für das Netzwerkmanagement, die Nachbarschaftserkennung und für den nicht zeitkritischen, bedarfsorientierten Datenaustausch realisiert. Dazu werden auch Protokolle des TCP/IP-Stacks eingesetzt, wodurch es in Profinet möglich ist, beliebige TCP/IP-basierte Applikationsprotokolle einzusetzen. Üblicherweise enthält jedes Profinet-Gerät beispielsweise einen Web-Server, welcher eine Konfiguration des Geräts von einem PC mit Hilfe eines Browsers ermöglicht.

Parallel zu dieser nicht echtzeitfähigen Kommunikation stellt Profinet RT einen

Echtzeit-Kanal für die Übertragung von Prozessdaten und kritischen Alarmen bereit, welcher die Anforderungen der Echtzeitklasse 2 erfüllt. Dazu setzt Profinet RT die in Abschnitt 2.2.3 beschriebenen Mechanismen von RTEs der Klasse 2 ein: Daten werden unter Umgehung des TCP/IP-Stacks direkt mit der Ethernet-Ebene ausgetauscht. Während der physikalischen Übertragung werden die Pakete durch das Setzen der höchsten Prioritätsklasse gegenüber NRT-Paketen in den Switchen bevorzugt weitergeleitet.

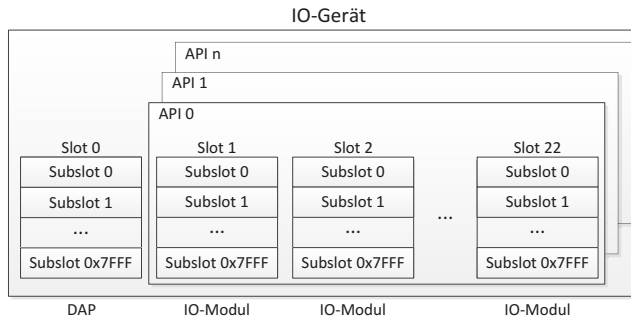
## Datenmodell

In Profinet werden zusätzlich zur Adressierung auf IP-Ebene zwei Mechanismen zur Geräte- und Datenadressierung eingesetzt: Das Profinet-spezifische Discovery and Configuration Protocol (DCP), welches zur nicht echtzeitfähigen Konfiguration und Diagnose eingesetzt wird, verwendet symbolische Gerätenamen zur eindeutigen Identifizierung. Die RT-Dienste nutzen lediglich die physikalischen Ethernet-Adressen (auch MAC-Adressen genannt) zur Geräteadressierung.

Innerhalb eines IO-Gerätes sieht das Profinet Gerätemodell vier verschiedene Adressierungsstufen vor, welche vom IO-Controller genutzt werden, um die einzelnen Ein- bzw. Ausgänge eines IO-Geräts anzusprechen. Das Adressierungsschema orientiert sich dabei an dem üblichen Aufbau eines IO-Geräts, welches aus mehreren Modulen bestehen kann. Der obligatorische Device Access Point (DAP) ist dasjenige Modul, welches die Ethernet-Schnittstelle enthält. Des Weiteren enthalten modulare IO-Geräte Steckplätze für das Hinzufügen weiterer, funktional eigenständiger, Module. Bei kompakten IO-Geräten (auch als Block IO-Geräte bezeichnet) ist die Unterteilung des Gerätes in logische bzw. physikalische Funktionseinheiten hingegen vorgegeben und nicht änderbar. Die einzelnen adressierbaren Elemente eines IO-Geräts sind (siehe Abbildung 2.7):

- Über die Angabe des **Slots** wird eine logische bzw. physikalische Funktionseinheit des IO-Geräts adressiert – beispielsweise der Steckplatz, in dem das anzusprechende Modul eingesetzt ist.
- Der **Subslot** dient zur Adressierung der konkreten Prozessdaten des dem jeweiligen Slot zugeordneten Moduls. So kann durch den Subslot z. B. angegeben werden, welche Eingangs- oder Ausgangsdaten des Moduls zyklisch an den IO-Controller übertragen werden sollen.
- Der **Index** wird genutzt, um innerhalb eines Slots oder Subslots azyklisch auf Daten zugreifen zu können, beispielsweise auf Diagnose-Informationen.

- Über den **Application Process Identifier (API)** kann eine bestimmte auf dem IO-Gerät laufende Applikation adressiert werden, wenn dieses die Ausführung mehrerer gleichzeitiger Prozesse unterstützt.



**Abbildung 2.7:** Profinet-Gerätemodell für die Adressierung von Ein-/Ausgabedaten

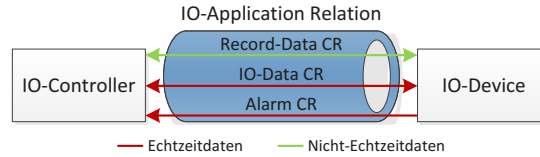
## Kommunikationsbeziehungen

Um zwischen den Profinet-Applikationsprozessen zweier Geräte Daten austauschen zu können, müssen entsprechende Kommunikationsbeziehungen eingerichtet werden. Der Aufbau dieser in Profinet Application Relationship (AR) genannten Verbindungen wird beim Systemhochlauf vom IO-Controller vorgenommen. Die entsprechenden Informationen über die einzurichtenden ARs erhält der IO-Controller von einem Engineering-Tool (siehe auch Abschnitt Inbetriebnahme). Innerhalb einer IO-AR, welche zwischen IO-Controller und -Device eingerichtet wird, können drei verschiedene Communication Relations (CRs) realisiert werden (siehe auch Abbildung 2.8):

- Die **Record-Data CR** wird für den Austausch von Konfigurationsdaten über den NRT-Kanal verwendet.
- Die **IO-Data CR** dient dem zyklischen Austausch von Prozessdaten über den RT Kanal.
- Über die **Alarm CR** werden echtzeitkritische Alarmmeldungen vom IO-Gerät an den -Controller gesendet.

Neben der IO-AR gibt es weiterhin eine IOS-AR für die Kommunikation zwischen IO-Supervisor und -Device. Über diese AR kann der IO-Supervisor auch auf Prozessdaten

zugreifen, allerdings steht ihm dafür nur der NRT-Kanal zur Verfügung. Die Implicit AR dient dem lesenden Zugriff von IO-Controller bzw. IO-Supervisor auf Daten des IO-Geräts. Diese Form der AR benötigt keinen expliziten Verbindungsaufbau.



**Abbildung 2.8:** Applikations- und Kommunikationsbeziehungen zwischen IO-Controller und -Device

### Echtzeitverhalten

Die Prozessdaten einer IO-Data CR werden zusammen in einem Ethernet-Frame in festen zeitlichen Abständen zwischen IO-Controller und -Device ausgetauscht. Diese Zykluszeit, welche während des Engineerings festgelegt wird, kann dabei für jeden Subslot einzeln vorgegeben werden. Zusätzlich zur Zykluszeit wird die zeitliche Positionierung der Frames aller IO-Data CRs relativ zueinander festgelegt. Dazu wird die zur Verfügung stehende Übertragungszeit in gleich lange Phasen unterteilt und die Positionierung der Frames innerhalb der Phasen bestimmt. Die Festlegung erfolgt über vier Parameter, die für jede IO-Data CR festgelegt werden müssen:

- Über den Parameter **SendClockFactor** wird die SendClock für das gesamte Profinet-System festgelegt. Die SendClock gibt die Dauer einer Phase an und berechnet sich wie folgt:

$$SendClock = SendClockFactor \cdot 31,25\mu s \quad (2.1)$$

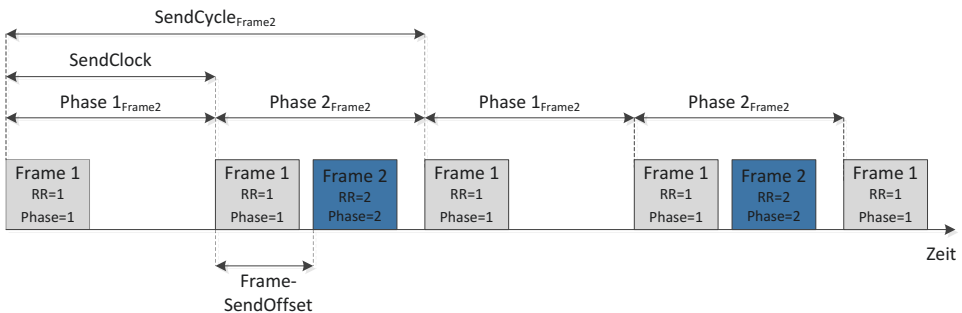
- Die Zykluszeit SendCycle für die Frames einer IO-Data CR ergibt sich aus der SendClock und dem Frame-spezifische Parameter **ReductionRatio**:

$$SendCycle = SendClock \cdot ReductionRatio \quad (2.2)$$

- Sei  $r$  der Wert des Parameters ReductionRatio. Dann unterteilt sich die Zykluszeit des entsprechenden Frames in  $r$  gleich lange Phasen. Der Parameter **Phase** gibt an, in welcher dieser Phasen der Frame gesendet wird.

- Über den Parameter **FrameSendOffset** wird angegeben, zu welchem Zeitpunkt nach Phasenbeginn der jeweilige Frame gesendet werden soll.

Die Zusammenhänge zwischen den einzelnen Parameter sind in Abbildung 2.9 dargestellt. In diesem Beispiel werden die Frames zweier IO-Data CRs übertragen, wobei für die Frames der ersten CR ein ReductionRatio von 1 und für die anderen Frames ein Wert von 2 eingestellt ist.



**Abbildung 2.9:** Zeitliche Positionierung von Frames in Profinet

In der Variante Profinet RT muss berücksichtigt werden, dass keine Synchronisierung stattfindet, der Beginn eines Sendezyklus ist daher in verschiedenen Geräten nicht aufeinander abgestimmt. Eine zeitliche Positionierung von Frames verschiedener Geräte relativ zueinander ist daher nicht möglich.

## Inbetriebnahme

Der Prozess zur Inbetriebnahme eines Profinet-Netzwerkes ist –wie bei allen folgenden RTEs– von wesentlicher Bedeutung, da in dieser Arbeit letztendlich Verfahren für dessen Automatisierung entwickelt werden sollen. In herkömmlichen Profinet-Systemen lässt sich die Inbetriebnahme in die drei Phasen Projektierung, Download der Konfigurationsdatei und System-Hochlauf unterscheiden, welche im Folgenden beschrieben werden:

Während der **Projektierung**, welche in der Regel in einem Engineering-Tool des Herstellers des IO-Controllers durchgeführt wird, werden die Informationen festgelegt, die für den späteren Betrieb des Profinet-Netzwerkes benötigt werden. Insbesondere müssen diese Festlegungen getroffen werden:

- Definition im Netzwerk vorhandenen Profinet-Geräte

- Konfiguration der Geräte: Welche Module sind welchen Slots und welche Submodule sind welchen Subslots zugeordnet? Welche Daten muss das Gerät mit dem IO-Controller austauschen? Die genauen Eigenschaften eines Geräts wie Anzahl der Slots und die kompatiblen Module mit ihren Ein- und Ausgabemöglichkeiten sind in Gerätebeschreibungsdateien festgelegt, welche in Profinet General Station Description (GSD)-Dateien genannt werden. Zur Konfiguration werden die GSD-Dateien der entsprechenden Geräte in das Engineering-Tool geladen.
- Definition der Zykluszeiten
- Gerätetaufe: Jedem Profinet-Gerät müssen eine IP-Adresse und ein Profinet-spezifischer Geräteiname zugewiesen werden.

Anschließend findet der **Download der Konfigurationsdatei** vom Engineering-Tool in den IO-Controller statt. Enthält der IO-Controller zusätzlich eine Laufzeitumgebung für die Steuerungslogik und übernimmt entsprechend die Rolle einer SPS, so muss die Konfigurationsdatei ebenfalls die Logik in einer für den Controller lesbaren Form enthalten. Weiterhin müssen die Kommunikationsobjekte der Steuerungslogik (z. B. Variablen) mit den zugehörigen Ein-/Ausgabedaten der IO-Geräte verknüpft sein.

Den letzten Schritt der Inbetriebnahme bildet der **System-Hochlauf**, in dem der IO-Controller anhand der in der Konfigurationsdatei vorhandenen Projektierungsdaten die einzelnen IO-Geräte konfiguriert und die ARs aufbaut. Anschließend nehmen alle Geräte ihre Funktion auf, insbesondere der zyklische Prozessdatenaustausch startet zu diesem Zeitpunkt.

## 2.2.5 Profinet IRT

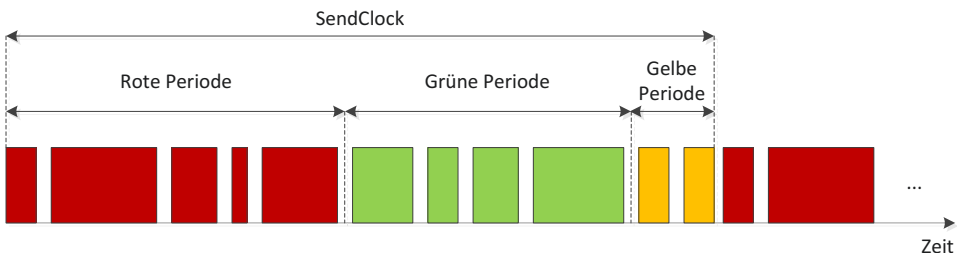
Profinet IRT ist eine Weiterentwicklung von Profinet RT in Hinblick auf deterministisches Übertragungsverhalten. Da sich die Grundprinzipien beider Varianten –insbesondere in den Punkten Architektur, Gerätemodell, Adressierung und Kommunikationsbeziehungen– sehr ähneln, werden in diesem Unterabschnitt nur die wesentlichen Unterschiede im Vergleich zu Profinet RT behandelt.

### Echtzeitverhalten

Das Einsatzgebiet von Profinet IRT ist die Steuerung von Prozessen der Echtzeitklasse 3. Im Gegensatz zu Profinet RT muss es daher wesentlich geringere Übertragungsverzögerungen bei minimalen Jitter ermöglichen (siehe Tabelle 2.1). Dabei nutzt Profinet IRT

dieselben vier Parameter für die Bestimmung der Sendezeitpunkte wie die RT-Variante (siehe Abbildung 2.9). Allerdings werden zusätzlich die Uhren aller IRT-Geräte –auch die der Switche– mit einer Genauigkeit von bis zu einer Mikrosekunde synchronisiert, sodass alle Teilnehmer ein gemeinsames Verständnis über den Beginn eines Sendezyklus haben und damit die vom IO-Controller vorgegebenen Sendezeitpunkte genau einhalten können. Durch die Festlegung eines systemweiten Kommunikationsfahrplanes werden Wartezeiten bei der Paketweiterleitung innerhalb eines Switches vermieden. Die Synchronisierung erfolgt über das Precision Transparent Clock Protocol (PTCP) [ISO/IEC 61158 2010], welches zur Einhaltung der geforderten Genauigkeit Modifikationen auf Hardware-Ebene erfordert. Standard-Ethernet Komponenten können daher in Profinet IRT nicht eingesetzt werden.

Gleichzeitig ist es in Profinet IRT möglich, TCP/IP-basierte Applikationen parallel zu den Echtzeitprozessen zu betreiben. Um gegenseitige Beeinflussungen von RT- und NRT-Frames und die daraus resultierenden nichtdeterministischen Verzögerungen auszuschließen, werden die einzelnen Phasen bei Profinet IRT in drei Perioden unterteilt (siehe auch Beispiel in Abbildung 2.10):



**Abbildung 2.10:** Unterteilung der Sendezeit in verschiedene Echtzeit-Perioden bei Profinet IRT

- Die **rote Periode** ist reserviert für die isochrone Kommunikation, jeder Frame wird hier zu genau spezifizierten Zeiten gesendet. Zur Reduzierung von Übertragungsverzögerungen wird der normalerweise auf MAC-Adressen beruhende Weiterleitungsmechanismus innerhalb der Switches in dieser Periode durch eine zeitgesteuerte Weiterleitung ersetzt. Dazu besitzt jeder Switch eine Tabelle mit den Zeitpunkten, zu denen bestimmte Frames eingehen und an welchen Port diese weitergeleitet werden müssen. Die Weiterleitungstabelle wird während der Projektierung des IRT-Systems erzeugt, dazu muss die vollständige Topologie des Netzes bekannt sein. Sollten Nicht-IRT-Frames in der roten Periode eingehen,

werden diese vom Switch bis zum Beginn der grünen Periode gepuffert.

- In der **grünen Periode** können RT- und NRT-Frames gesendet werden, wobei eine Priorisierung wie bei Profinet RT erfolgen kann.
- Die **gelbe Periode** stellt den Übergang zur roten Periode der folgenden Phase dar. Um eine Störung der IRT-Kommunikation zu verhindern, leiten Switches Frames in der gelben Periode nur dann weiter, wenn der Sendevorgang innerhalb dieser Phase abgeschlossen werden kann – ansonsten wird der Frame zwischengespeichert.

### Inbetriebnahme

Der wesentliche Unterschied in der Inbetriebnahme eines IRT-Systems im Vergleich zu Profinet RT liegt in der Erfordernis eines Kommunikationsfahrplans (Schedule), welcher die Sendezeitpunkte für jeden einzelnen IRT-Frame festlegt. Zu diesem Zweck wird ein Planungsalgorithmus benötigt, welcher für jeden Frame die folgenden Eigenschaften bestimmt:

- Die Einordnung des Frames in den Profinet-Sendezyklus
- Den Empfangs- und Sendepunkt eines Frames in einem Switch
- Den Empfangs- und Sendezeitpunkt eines Frames in einem Switch

Zur Berechnung dieser Daten müssen dem Algorithmus während der Projektierung die folgenden Informationen bekannt gemacht werden:

- Die Netztopologie sowie die Verzögerungszeiten für die Datenübertragung innerhalb eines Geräts sowie die Übertragungszeit zwischen zwei Geräten
- Der Quell- und Zielknoten eines jeden Frames
- Die innerhalb eines Frames zu übertragende Datenmenge
- Die Anforderungen der Anwendung an die Datenübertragung (z. B. Zykluszeit und maximale Übertragungsdauer)

### 2.2.6 Ethernet/IP

Bei dem von der Open DeviceNet Vendor Association (ODVA) entwickelten Ethernet Industrial Protocol (EIP) handelt es sich um eine Adaption des für Automatisierungsanwendungen entworfenen Common Industrial Protocol (CIP) auf Ethernet. CIP ist als Applikationsprotokoll gemäß Open Systems Interconnection (OSI)-Modell prinzipiell unabhängig von den unterliegenden Übertragungsschichten. So gibt es beispielsweise mit DeviceNet eine auf den im Automobilbereich verbreiteten Controller Area Network (CAN)-Bus angepasste Variante von CIP. EIP als Ethernet-Portierung von CIP setzt für die Datenübertragung in seiner Grundvariante auf die TCP/IP-Protokollfamilie und ist damit gemäß Abbildung 2.5 ein RTE der Klasse 1.

#### Architektur

Wie auch in Profinet wird bei EIP zwischen Echtzeitkommunikation und nicht nicht-zeitkritischer Kommunikation unterschieden. Allerdings basiert in EIP jegliche Kommunikation –im Gegensatz zu Profinet auch die zeitkritische– auf den Protokollen der TCP/IP-Familie. Zur Unterscheidung werden in CIP zwei Arten von Nachrichten definiert:

**Explizite Nachrichten** werden für die nicht zeitkritische Kommunikation wie z. B. das Lesen oder Schreiben von Konfigurationsdaten verwendet und benutzen das Transmission Control Protocol (TCP) als Transportprotokoll. Der Begriff explizit beruht darauf, dass in diese Art von Nachrichten eine explizite Beschreibung der übertragenden Daten enthalten. Dahingegen werden **implizite Nachrichten** für zeitkritische, in der Regel zyklisch gesendete, Daten verwendet, sie nutzen das User Datagram Protocol (UDP) als Transportprotokoll. Die Bedeutung der Daten wird in impliziten Nachrichten nicht mit übertragen, womit sich eine verringerte Nutzdatenlast ergibt.

Nicht alle Geräte in EIP unterstützen alle Nachrichtenarten, analog zu Profinet gibt es auch hier verschiedene Geräteklassen: Der **Explicit Message Server** reagiert nur auf explizite Anfragen eines Clients. Wenn der Client ebenfalls nur diese Anfrage/Antwortorientierte Kommunikation unterstützt, handelt es sich um einen **Explicit Message Client**. Das Äquivalent zu dieser Klasse in Profinet stellen die IO-Supervisors da. Ein **IO-Adapter** umfasst neben der Funktionalität eines Explicit Messages Servers die weitere Fähigkeit, seine Eingangs- bzw. Ausgangsdaten nach einmaliger Aufforderung zyklisch in impliziten Nachrichten zu empfangen bzw. zu senden und entspricht damit einem IO-Gerät in Profinet. Die entsprechenden Aufforderungen erhält er von einem **IO-Scanner**,

welcher in der Lage sein muss, entsprechende Verbindungen für implizite Nachrichten zu den IO-Adaptern herzustellen. Der IO-Scanner entspricht daher der Rolle eines IO-Controllers in Profinet.

## Datenmodell

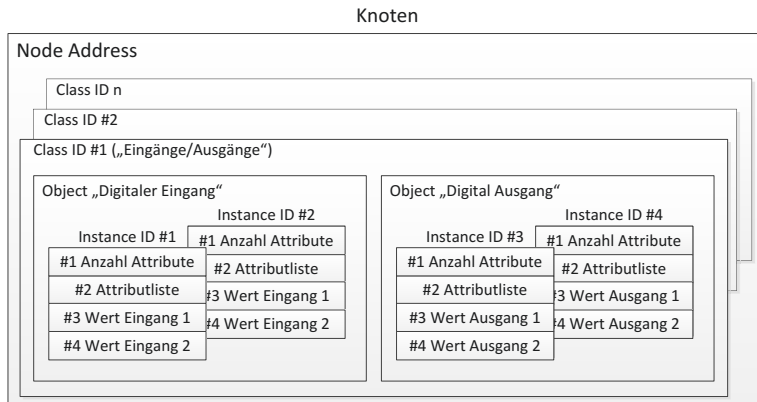
Während in Profinet ein signalorientiertes Datenmodell zur Adressierung der Ein- und Ausgabedaten verwendet wird, setzen CIP und damit EIP auf einen objektorientierten Datenzugriff. Dazu wird ein Endgerät (auch Knoten genannt) als eine Sammlung von abstrakten Objekten modelliert und über die folgenden Eigenschaften adressiert (siehe auch Beispiel in Abbildung 2.11):

- Durch die **Node Address** wird ein Knoten innerhalb eines Netzwerks eindeutig identifiziert. In EIP entspricht sie der IP-Adresse des entsprechenden Knotens.
- Eine Gruppe von Objekten, die dieselbe Systemkomponente beschreiben, wird in einer Klasse zusammengefasst. Eine Klasse wird über den **Class Identifier** (Class ID) adressiert.
- Die konkrete Repräsentationen eines Objekts wird als Instanz bezeichnet und anhand des **Instance Identifiers** (Instance ID) angesprochen.
- Klassen oder Objektinstanzen können über Attribute verfügen, welche über den **Attribute Identifier** (Attribute ID) adressiert werden.
- Der **Service Code** gibt an, welche Aktion die adressierte Objektinstanz oder das adressierte Objektattribut ausführen sollen, wenn eine an die jeweilige Adresse gerichtete explizite Nachricht eintrifft. Dies kann beispielsweise ein Lese- oder Schreibvorgang sein.

## Kommunikationsbeziehungen

CIP ist wie Profinet ein grundsätzlich verbindungsorientiertes Protokoll und unterscheidet zwei Arten von Verbindungen:

**I/O Connections** werden für den Datenaustausch gemäß des Producer-Consumer Modells eingesetzt. Bei dieser Verbindungsart sendet eine Daten produzierende Applikation implizite Nachrichten zyklisch an eine oder mehrere Daten konsumierende Applikationen. Die Realisierung einer Anwendung mit mehreren Konsumenten erfolgt



**Abbildung 2.11:** Beispiel für das in CIP genutzte Adressierungsmodell

in EIP über die Verwendung von Multicast-Gruppen. Der Nutzung einer I/O Connection erfordert immer einen vorherigen Verbindungsaufbau, in welchem beispielsweise das Sendeintervall der folgenden Daten festgelegt wird.

Dem Client-Server Kommunikationsmodell folgen die **Explicit Messaging Connections**. Diese Punkt-zu-Punkt Verbindungen werden für den anlassbezogenen Datenaustausch mittels expliziter Nachrichten eingesetzt. Ein vorheriger Verbindungsaufbau ist optional.

## Echtzeitverhalten

EIP nutzt in seiner Grundvariante UDP für den zyklischen Prozessdatenaustausch, so dass sich aus den in Unterabschnitt 2.2.3 genannten Gründen nur Anwendungen der Echtzeitklasse 1 realisieren lassen. Allerdings können durch die Erweiterung CIP Sync auch die Anforderungen der Echtzeitklasse 3 erfüllt werden. CIP Sync setzt dazu auf einen Zeitsynchronisierungsmechanismus gemäß dem Standard IEEE 1588, wodurch ein gemeinsames Zeitverständnis aller Knoten erreicht wird. Zusätzlich werden in CIP Sync Zeitstempel eingesetzt, welche die Erfassungszeitpunkte von Sensordaten und die Ausführungszeitpunkte von Aktordaten vorgeben.

Durch die Zeitstempelung der Prozessdaten kann auf eine genaue Kenntnis der Übertragungsverzögerungen verzichtet werden – es muss lediglich sichergestellt werden, dass Nachrichten ihren Empfänger rechtzeitig erreichen. Allerdings wird für die Unterscheidung von zeitkritischen und NRT-Daten bei EIP lediglich eine Priorisierung von Frames

eingesetzt, eine genaue netzwerkweite Kommunikationsplanung wie bei Profinet IRT findet bei EIP nicht statt. Daraus resultiert zwar ein geringerer Projektierungsaufwand für EIP-Systeme – die Einhaltung der Zeitbedingungen kann allerdings auf Netzwerkebene nicht garantiert werden. Insbesondere bei anspruchsvollen Anwendungen müssen daher andere Methoden für die Sicherstellung des korrekten Echtzeitverhaltens wie Simulationen oder schlichtweg eine „trial and error“-Vorgehensweise gewählt werden.

## **Inbetriebnahme**

Die Inbetriebnahme eines EIP-Netzes wird analog zu dem bereits in Unterabschnitt 2.2.4 für Profinet vorgestellten Verfahren durchgeführt. Dies umfasst ebenfalls die Schritte Projektierung (Definition der vorhandenen EIP-Geräte und der Beziehungen zwischen deren Kommunikationsobjekten) und Download der erzeugten Konfigurationsdatei in die SPS. Anschließend bauen die einzelnen Geräte die projektierten Kommunikationsbeziehungen auf.

### **2.2.7 Ethernet Powerlink**

Ethernet Powerlink (EPL) ist ein RTE-Standard, welcher heute von der Ethernet Powerlink Standardization Group (EPSG) weiterentwickelt wird. Wie auch bei EIP wird in EPL kein eigenständiges Applikationsprotokoll verwendet, sondern mit CANopen ein bereits etabliertes Protokoll adaptiert. EPL nutzt ebenso wie Profinet IRT ein zeitschlitzbasiertes Medienzugriffsverfahren, welches sich zusätzlich auch auf die NRT-Phase erstreckt. Wird der modifizierte Medienzugriff ausschließlich in Software realisiert, handelt es sich bei EPL um ein RTE der Klasse 2, ansonsten wird es der Klasse 3 zugeordnet.

## **Architektur**

EPL orientiert sich bei der Definition der Geräteklassen wie auch Profinet eng an der Automatisierungshierarchie: In jedem Netz gibt es eine zentrale Instanz, welche in EPL als Managing Node (MN) bezeichnet wird. Der MN steuert die gesamte Kommunikation zwischen ihm und den untergeordneten Controlled Nodes (CNs). Eine Besonderheit von EPL ist der bevorzugte Einsatz von Hubs als Verbindungselement anstatt der üblichen Switches. Die Gründe werden unter dem Punkt Echtzeitverhalten genannt.

Auch EPL setzt für die Unterscheidung von RT- und NRT-Daten zwei getrennte logische Kanäle ein. Auf TCP/IP basierende Applikationen können so parallel zum Echtzeit-Steuerungsprozess betrieben werden. Allerdings muss sichergestellt werden, dass jede

Ethernetschnittstelle in einem EPL-Netz das entsprechende Medienzugriffsverfahren unterstützt. Daher ist es in EPL nicht möglich, unmodifizierte Standardkomponenten einzusetzen.

Datenmodell

Die Adressierung der Geräte in EPL erfolgt über Node IDs, welche direkt am Gerät gesetzt werden müssen (beispielsweise über Schalter). Die Daten eines EPL-Knotens sind innerhalb eines Objektverzeichnisses organisiert (siehe Abbildung 2.12). Jedes Objekt wird durch einen eindeutigen Index adressiert und enthält neun festgelegte Parameter. Davon ist nur der Parameter, welcher den aktuellen Wert des Objekts repräsentiert, von außen zugänglich. Die übrigen beschreiben Objekteigenschaften wie Datentyp und Zugriffsrichtung. Zusätzlich kann jedes Objekt bis zu 255 Unterobjekte enthalten, welche wiederum durch ihren Subindex adressiert werden.

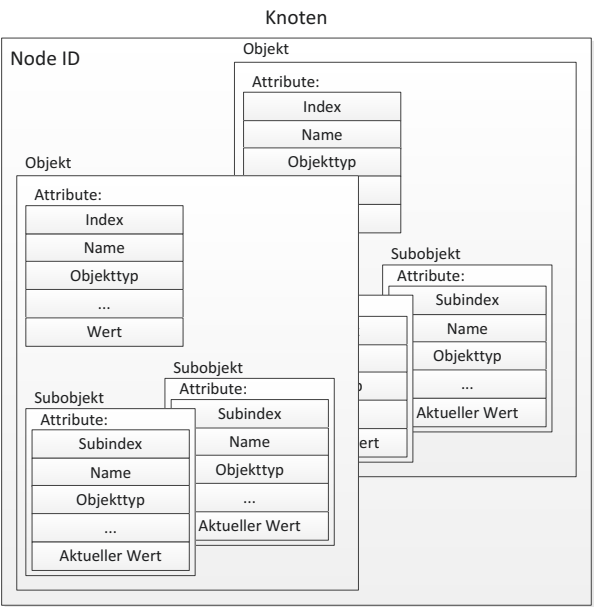


Abbildung 2.12: Adressierungsmodell in Ethernet Powerlink

## Kommunikationsbeziehungen

EPL unterscheidet zwei Formen der Kommunikation: Nicht-zeitkritische Daten werden in Form von Service Data Objects (SDOs) nach dem Client-Server Modell ausgetauscht. Dabei sendet der Client eine SDO read-Nachricht an den Server, welche den Index und ggf. Subindex des angefragten Objekts enthält. Der Server antwortet entsprechend mit dem Wert des Objekts.

Zyklische Daten werden in Echtzeit mittels Process Data Objects (PDOs) gemäß dem Producer-Consumer Modell zwischen dem sendenden und einem oder mehreren empfangenden Knoten ausgetauscht. Da in einem EPL-Netz vorzugsweise nur Hubs als Verbindungselemente eingesetzt werden und damit jeder Frame physikalisch immer an alle Netzknoten übermittelt wird, entsteht durch die Broadcast-Übertragung der PDO-Frames kein zusätzlicher Datenverkehr. In diesem Punkt unterscheidet sich EPL beispielsweise von Profinet, wo für jede Applikationsbeziehung zusätzliche Übertragungsressourcen benötigt werden.

## Echtzeitverhalten

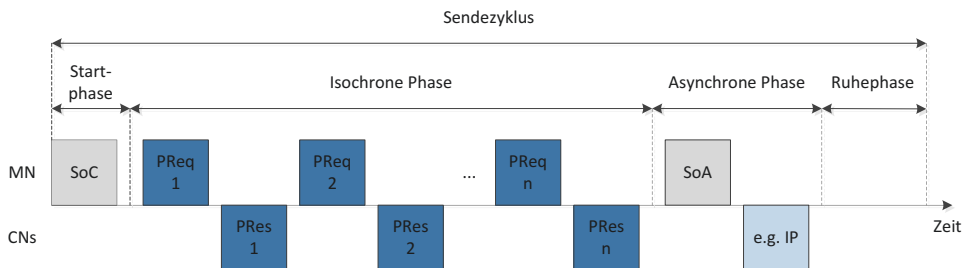
Wie Profinet IRT setzt auch EPL auf ein TDMA-basiertes Medienzugriffsverfahren. Während jedoch in Profinet IRT jeder Knoten selbstständig seinen Medienzugriff zu vorgegebenen Zeiten beginnt, wird in EPL ein Slot Communication Network Management (SCNM) genanntes Verfahren eingesetzt, bei dem jeder Knoten nur nach Aufforderung durch den MN mit dem Senden beginnen darf. Diese Vorgabe gilt für alle Verkehrsklassen –auch für nicht-zeitkritische Daten–, sodass es in einem EPL-Netz zu keinen Kollisionen kommen kann. Der Einsatz von Switches zur Kollisionsverhinderung ist in EPL daher obsolet. Stattdessen empfiehlt sich die Nutzung von Hubs, die eingehende Frames ohne weitere Analyse auf alle Ausgangsports weiterleiten und damit im Vergleich zu Switches niedrigere Verzögerungszeiten verursachen.

Die Grundstruktur der Kommunikation in EPL besteht aus Zyklen konstanter Längen, welche wiederum in vier Phasen aufgeteilt werden (siehe Abbildung 2.13). In der Startphase sendet der MN ein Start of Cycle (SoC)-Frame zur Synchronisierung aller Knoten. In der folgenden isochronen Phase werden die zyklischen Daten ausgetauscht. Dazu fordert der MN jeden CN durch ein Poll Request (PReq) genanntes Paket auf, seine Daten zu senden. Gleichzeitig enthält das PReq-Paket die Ausgangsdaten des MNs in Richtung des jeweiligen CNs. Die Antwort des CNs, die Poll Response (PRes), beinhaltet wiederum die Eingangsdaten für den MN. Die PRes-Pakete werden als Broadcasts versendet und

können so auch von anderen CNs verarbeitet werden. Ein CN muss nicht in jedem Zyklus durch den MN abgefragt werden, dies kann auch in ganzzahligen Vielfachen des EPL-Zyklus erfolgen.

Die asynchrone Phase startet mit einem Start of Acyclic (SoA)-Frame durch den MN. In diesem Abschnitt darf nur der CN seine Daten senden, welcher im SoA-Frame vom MN hierzu aufgefordert wurde. Ihre Sendewünsche müssen die CNs in der isochronen Periode durch das Setzen eines Flags innerhalb ihrer PRes-Pakete dem MN mitteilen. Die asynchrone Phase kann zum Senden beliebiger Nutzdaten, beispielsweise für TCP/IP-Pakete, genutzt werden.

Durch die abschließende Ruhephase soll sichergestellt werden, dass keine Frames über die Grenze des Zyklus hinaus gesendet werden. In diesem Abschnitt darf kein Knoten einen Sendevorgang neu beginnen.



**Abbildung 2.13:** Phasen des Ethernet Powerlink Sendezyklus

## Inbetriebnahme

Die Inbetriebnahme eines EPL-Netzes ähnelt im Wesentlichen der Prozedur bei Profinet RT. Auch in EPL werden für die Netzwerkkonfiguration und die Prozessdatenzuordnung Gerätebeschreibungen, welche u. a. die Objekte und deren Attribute enthalten, benötigt. Diese werden hier als XML Device Description (XDD)-Dateien bezeichnet.

### 2.2.8 Ethercat

Ethernet for Control Automation Technology (Ethercat) ist ein von der EtherCAT Technology Group (ETG) entwickelter RTE-Standard. Er unterscheidet sich von den anderen in dieser Arbeit vorgestellten RTE-Varianten durch den Einsatz des Summenrahmenverfahrens, welches wesentlich von der bei Ethernet üblichen Datenübermittlung mittels

individueller Frames abweicht. Es handelt sich bei Ethercat daher um ein RTE der Klasse 3.

## Architektur

Ein Ethercat-System besteht aus einem Master und mehreren Slaves, welche logisch durch eine Ring-Topologie verbunden sind (siehe Abbildung 2.14). Die Vorgabe dieser Netzstruktur ist ein besonderes Ethercat-Merkmal, da in Ethernet-basierenden Netzen üblicherweise durch den Einsatz von Switches oder Hubs beliebige Topologien eingesetzt werden können – außer einer Ring-Topologie.<sup>1</sup>

Der Ethercat-Master bildet das offene Ringende eines Ethercat-Netzes. Alle nachfolgenden Slaves werden in Reihe hinter dem Master verkettet angeschlossen (sog. Daisy-Chain-Verkabelung). Der Datenaustausch geschieht in der Weise, dass im Upstream der Master einen Frame in den Ring sendet und jeder der nachfolgenden Slaves den Frame empfängt, verarbeitet und den ggf. modifizierten Frame an seinen Nachfolger weiterleitet. Der letzte Slave des Rings wird mit seinem physikalischen Vorgänger verbunden, so dass der Frame nun im Downstream wiederum jeden Slave durchläuft bis er schließlich wieder den Master erreicht. Aufgrund der Full-Duplex Eigenschaft von Ethernet müssen zwei Knoten nur mit einem Kabel verbunden werden.

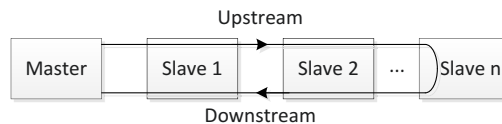


Abbildung 2.14: Ethercat Architektur

## Datenmodell

Im Ethercat-Standard selber ist kein spezielles Applikationsprotokoll spezifiziert. Stattdessen können alle Slaves eines Ethercat-Systems als ein verteilter Speicher angesehen werden, auf den der Master lesend oder schreibend zugreifen kann. Aus Sicht einer Steuerungsapplikation können so beispielsweise Variablen direkt einer Ethercat-Speicheradresse zugewiesen werden. Zur Adressierung des Speichers dient ein 32 Bit breites Feld im Ethercat Datagramm, welches auf zwei Wegen genutzt werden kann:

<sup>1</sup>Eine Ring-Verkabelung in Ethernet, bsp. aus Redundanz-Gründen, ist zwar möglich, dabei müssen jedoch durch geeignete Mechanismen die zur Ring-Bildung führenden Pfade deaktiviert werden.

Bei der **physikalischen Adressierung** werden 16 Bit des Adressfeldes dazu genutzt, einen einzelnen Slave entweder durch seine Position im Ethercat-Ring oder durch seine Stationsadresse anzusprechen. Durch die verbliebenen 16 Bit können  $2^{16}=64$  kB des jeweiligen Slave-Speichers direkt adressiert werden. Der Nachteil dieser Adressierungsart ist, dass pro Datagramm nur ein Slave –oder bei Broadcasts dieselbe Speicherstelle aller Slaves– angesprochen werden kann. Sie wird daher meist für Konfigurationsdaten verwendet.

Bei der **logische Adressierung** hingegen wird ein einzelner Slave nicht mehr individuell adressiert, stattdessen bilden alle Slaves einen gemeinsamen  $2^{32}=4$  GB großen Speicherbereich. Die bitweise Übersetzung der logischen in eine physikalische Adresse innerhalb der Slaves erfolgt mittels einer Fieldbus Memory Management Unit (FMMU). Die Übersetzungstabelle der FMMU enthält zu diesem Zweck Einträge, welche die bitgenaue logische Startadresse, die bitgenaue physikalische Startadresse im Slave-Speicher und die Größe des zu übersetzenden Speicherbereichs enthalten. Der Master wird bei dieser Adressierungsart in die Lage versetzt, in einem Ethercat Datagramm mit derselben logischen Adresse mehrere Slaves gleichzeitig anzusprechen. Die logische Adressierung findet ihre Anwendung meist im zyklischen Austausch von Prozessdaten.

Um Ethercat leichter in bestehende Anwendungen zu integrieren, wurden diverse Applikationsprotokolle auf Ethercat adaptiert. So gibt es die CANopen over EtherCAT (CoE)-Spezifikation, welche die Abbildung von CANopen-Objekten wie Service und Process Data Objects (siehe auch die Beschreibung von Ethernet Powerlink in Unterabschnitt 2.2.7) auf den Ethercat-Adressraum regelt.

## Kommunikationsbeziehungen

Unter Nutzung der im Abschnitt Datenmodell genannten Adressierungsmethoden ist der Master in der Lage, die Slaves per Uni-, Multi- oder Broadcasts zu erreichen – physikalisch wird jeder vom Master ausgehende Frame von allen Slaves empfangen. Soll nur eine Teilmenge der Slaves per Multicast angesprochen werden, so kann dies über die logische Adressierung und eine entsprechende Konfiguration der FMMUs realisiert werden.

Eine Slave-zu-Slave Kommunikation ist ebenfalls möglich. So können die FMMUs verschiedener Slaves dahingehend konfiguriert werden, dass die Ausgangsdatenadresse eines Slaves mit der Eingangsdatenadresse weiterer Slaves übereinstimmt. Ist der sendende Slave topologisch vor dem empfangenen angeordnet, so kann der Datenaustausch innerhalb eines Zyklus erfolgen, andernfalls werden zwei Zyklen benötigt.

## Echtzeitverhalten

Ethercat nutzt für die Datenübertragung das Summenrahmenverfahren, bei dem die Nutzdaten für mehrere Empfänger innerhalb eines einzigen Ethernet Frames angeordnet sind. Durch dieses Verfahren soll die Übertragungseffizienz gesteigert werden, da ansonsten Prozessdaten mit einer üblichen Länge von wenigen Bytes einen Ethernet Frame, dessen Nutzdatenfeld eine Mindestlänge von 46 Bytes haben muss, nicht vollständig ausfüllen. Die Zykluszeit  $t_{\text{zyklus}}$  beschreibt den Zeitraum zwischen dem Sende- und Empfangszeitpunkt eines Frames am Master (siehe Abbildung 2.15). Diese setzt sich maßgeblich aus den Übertragungsverzögerungen  $t_{\text{ÜV},n}$  für den Datentransport zwischen den einzelnen Knoten und den Bearbeitungsverzögerungen  $t_{\text{BV},n}$ , welche durch die Verarbeitung der Frames innerhalb der Ethercat-Stacks der einzelnen Slaves verursacht werden, zusammen.

Da ein Frame jeden Slave je einmal in Upstream- und Downstream-Richtung durchlaufen muss, können sich die Bearbeitungsverzögerungen bei einer großen Anzahl von Slaves negativ auf die Performance eines Ethercat-Systems auswirken. In diesem Falle können RTEs wie Profinet, welche eine Linientopologie nicht zwangsweise voraussetzen, bessere Zykluszeiten erreichen [Jasperneite u. a. 2007]. Ebenfalls kann ggf. Ethercat von einer Erhöhung der Bitübertragungsrate nicht in dem Maße profitieren wie andere RTEs, wobei [Jasperneite u. a. 2007] und [Prytz 2008] bei dem Vergleich der Zykluszeiten von Ethercat und Profinet IRT für Übertragungsraten von jeweils 100 MBit/s und 1 GBit/s zu unterschiedlichen Ergebnissen kommen.

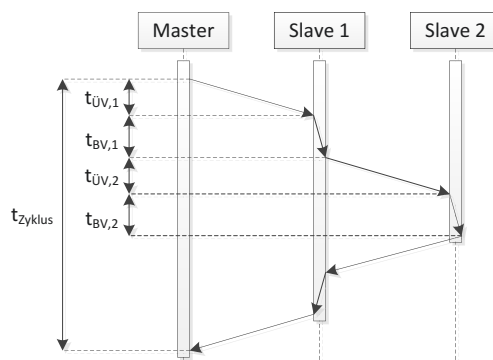


Abbildung 2.15: Weiterleitung von Frames in Ethercat

Um eine synchronisierte Ausführung von Applikationen zu gewährleisten, ist in

Ethercat mit dem Distributed Clocks Synchronization Protocol ein optionales Verfahren zur Uhrensynchronisation spezifiziert. Ähnlich wie bei CIP Sync in Ethernet/IP können so über Zeitstempel die Erfassungszeitpunkte von Sensordaten und die Ausführungszeitpunkte von Aktordaten synchronisiert werden.

### **Inbetriebnahme**

Auch in Ethercat wird im Wesentlichen die gleiche Inbetriebnahmeprozedur benötigt, wie sie unter Profinet RT (Unterabschnitt 2.2.4) beschrieben ist. Die Gerätebeschreibungsdateien bei Ethercat werden als Ethercat Slave Information (ESI)-Dateien bezeichnet.

## **2.2.9 Zusammenfassende Betrachtung der RTE-Standards**

In den Unterabschnitten 2.2.4 bis 2.2.8 wurden die verbreitetsten und in dieser Arbeit verwendeten RTE-Varianten vorgestellt. Eine Zusammenfassung der wesentlichen Eigenschaften findet sich in Tabelle 2.2.

Tabelle 2.2: Vergleich der verschiedenen industriellen Echtzeit-Ethernet Varianten

	Profinet RT	Profinet IRT	Ethernet/IP	Ethernet Powerlink	Ethercat
RTE-Klasse	2	3	1	2 <sup>1</sup> / 3 <sup>2</sup>	3
Echtzeit-Klasse	2	3	1 / 3 <sup>3</sup>	2 <sup>1</sup> / 3 <sup>2</sup>	3
Applikationsprotokoll	Profinet-spezifisch	Profinet-spezifisch	CIP	CANopen	z. B. CoE
Topologie	beliebig <sup>4</sup>	beliebig <sup>4</sup>	beliebig <sup>4</sup>	beliebig <sup>4</sup>	Ring
Zentraler Knoten erforderlich (Bezeichnung)	ja (IO-Controller)	ja (IO-Controller)	nein	ja (Managing Node)	ja (Master)
Bezeichnung dezentrale Knoten	IO-Gerät	IO-Gerät	entfällt	Controlled Node	Slave
Kommunikationsmodell	Producer/Consumer	Producer/Consumer	Producer/Consumer <sup>5</sup> Producer/Consumer <sup>7</sup>	Client/Server <sup>6</sup>	Client/Server
Adressierung	Signalorientiert	Signalorientiert	Objektorientiert	Objektorientiert	Verteilter Speicher
Echtzeit-Mechanismus	Priorisierung	Zeitschlitze	Priorisierung	Zyklische Abfrage	Summenrahmen
Uhrensynchronisation	nein	ja	optional <sup>3</sup>	nein	optional
Zeitstempelung von Prozessdaten	nein	nein	optional <sup>3</sup>	nein	optional

<sup>1</sup>Software-Implementierung <sup>2</sup>Hardware-Implementierung <sup>3</sup>Bei Verwendung von CIP Sync <sup>4</sup>Eine Ringbildung muss durch geeignete Maßnahmen verhindert werden <sup>5</sup>Gilt für I/O Connections <sup>6</sup>Gilt für Explicit Messaging Connections <sup>7</sup>Gilt für Process Data Objects <sup>8</sup>Gilt für Service Data Objects

### 2.2.10 Ausblick: IEEE 802.1 Time-Sensitive Networking

Die in den vorherigen Unterabschnitten beschriebenen RTE-Varianten spiegeln den Stand der Technik im Bereich der Ethernet-basierten Echtzeitkommunikation in der Automatisierungstechnik wieder. Die Entwicklung der verschiedenen Standards war der Tatsache geschuldet, dass Ethernet ein an sich nicht echtzeitfähiges Kommunikationsmedium darstellt – es jedoch aufgrund der hohen Leistungsfähigkeit und Verbreitung in der Informationstechnik ebenfalls in der Automatisierungstechnik eingesetzt werden sollte (siehe auch Abschnitt 2.2).

Die Erweiterung von Ethernet um Echtzeitfähigkeit resultierte jedoch zum einen darin, dass verschiedene Hersteller von Automatisierungskomponenten eigene RTE-Standards entwarfen, welche untereinander nicht kompatibel sind. Zum anderen führten die Echtzeit-Mechanismen ebenfalls zu Inkompatibilitäten zwischen den einzelnen RTEs und Standard-Ethernet –beispielsweise aufgrund unterschiedlicher Hardwareanforderungen und Medienzugriffsverfahren–, sodass sich eine vollständige Interoperabilität zwischen Netzwerken in der Informations- und Automatisierungstechnik nicht realisieren lässt.

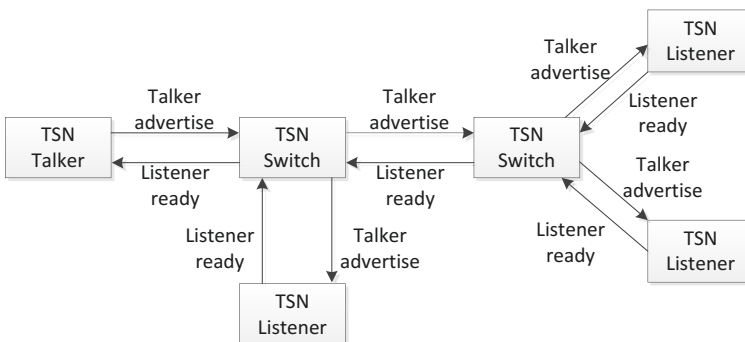
Das IEEE als Standardisierungsgremium von Ethernet hat in Form der IEEE 802.1 Time-Sensitive Networking (TSN) Task Group inzwischen eigene Aktivitäten zur Definition von Protokollen und Mechanismen aufgenommen, welche letztendlich dazu führen sollen, dass Ethernet ohne proprietäre Erweiterungen in allen Bereichen eingesetzt werden kann, in denen zeitkritische Kommunikation gefordert ist. Langfristig soll Ethernet dadurch in der Lage sein, domänenspezifische Echtzeitnetzwerke abzulösen. Dies trifft neben den RTEs in der Automatisierungstechnik insbesondere auch auf die Automobilbranche zu, welche beispielsweise mit FlexRay [Makowitz u. Temple 2006] einen eigenen Kommunikationsstandard für die zeitkritische Kommunikation innerhalb von Kraftfahrzeugen einsetzt. Die TSN Task Group ist Ende 2012 aus der Audio/Video Bridging (AVB) Task Group hervorgegangen. Letztere hat Protokolle für die dynamische Bandbreitenreservierung entworfen, durch deren Nutzung die simultane Übertragung mehrerer Datenströme –insbesondere solcher für Audio- und Videodaten– über ein Netzwerk ohne eine gegenseitige Beeinflussung ermöglicht wird. In der TSN Task Group werden die bestehenden AVB-Standards für die Realisierung deterministischer Kommunikation weiterentwickelt.

Ob und in welcher Form die TSN-Erweiterungen von Ethernet (im Folgenden nur als TSN bezeichnet) Einzug in die Automatisierungstechnik halten und gegebenenfalls

bestehende RTEs ersetzen, ist gegenwärtig noch nicht absehbar. Auch lassen sich noch keine Aussagen zum notwendigen Engineering-Aufwand für die Inbetriebnahme TSN-basierter Netze für Applikationen in der Automatisierungstechnik treffen. Diese Arbeit wird sich daher auf die Entwicklung von Konzepten für die automatische Inbetriebnahme der bereits etablierten RTEs beschränken – um jedoch einen Überblick über Ansätze zur Realisierung von Echtzeitkommunikation über Ethernet außerhalb der Automatisierungstechnik zu geben, werden in diesem Abschnitt die von TSN verwendeten Verfahren kurz vorgestellt.

Aus der Einordnung der TSN Task Group in die Normenreihe IEEE 802.1 lässt sich bereits erkennen, dass die Echtzeitmechanismen von TSN nicht auf Modifikationen der OSI-Ebenen 1 und 2 von Ethernet beruhen – diese sind beispielsweise in der Norm 802.3 für drahtgebundenes und 802.11 für drahtloses Ethernet spezifiziert. Die Norm 802.1 beschreibt stattdessen generelle Architekturen für lokale und Weitverkehrsnetze. Entsprechend sollen die TSN-Verfahren unabhängig vom verwendeten Kommunikationsmedium einsetzbar sein.

TSN ist kein alleinstehender Standard, sondern setzt sich aus einer Vielzahl verschiedener Protokolle zusammen. So werden beispielsweise in [IEEE 802.1AS 2011] Verfahren zur Uhrensynchronisation spezifiziert. Eines der grundlegenden Protokolle in TSN ist das Stream Reservation Protocol (SRP), welches in [IEEE 802.1Qat 2010] und der im Moment in der Entwurfsphase befindlichen Erweiterung [IEEE 802.1Qcc 2015] beschrieben wird. Es basiert auf der in Abbildung 2.16 gezeigten Unterteilung von Netzknoten in Talker, Switches (auch Bridges genannt) und Listener.



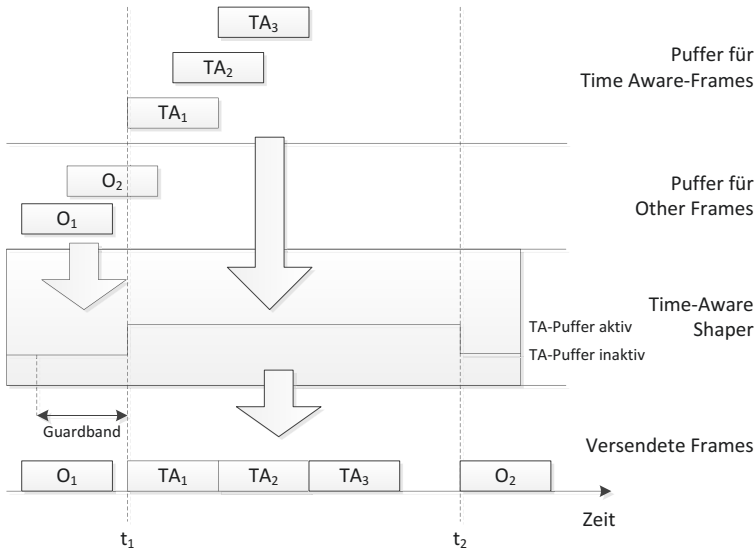
**Abbildung 2.16:** Reservierung von Übertragungskapazitäten in IEEE 802.1 Time Sensitive Networking

Ein Talker bietet dabei Informationen an, welche wiederum von einem Listener konsumiert werden. TSN geht von einem kontinuierlichen Datenstrom zwischen diesen beiden Arten von Knoten aus. Bevor ein Talker mit der Datenübertragung beginnt, bietet er seinen Datenstrom mittels einer Talker advertise-Nachricht im gesamten Netzwerk an. Diese Bekanntmachung enthält weitere Informationen über den Datenstrom wie z. B. die benötigte Bandbreite. Alle an den Daten interessierte Listener abonnieren den Strom mittels Listener ready-Nachrichten. Die Switche reservieren anschließend die für den Datenaustausch benötigten Kapazitäten. Sollte dies nicht möglich sein, beispielsweise bei nicht mehr ausreichend freier Bandbreite, lehnen die Switche den Aufbau der Verbindung zwischen Talker und Listener ab. Zukünftige Erweiterungen sollen das SRP dahingehend ergänzen, dass neben der Bandbreite insbesondere die für die Echtzeitkommunikation wichtigen Eigenschaften wie maximale Verzögerungszeiten und maximaler Jitter während der Datenstrom-Reservierung festgelegt werden können.

TSN setzt für Realisierung von Echtzeitkommunikation im Gegensatz zu den meisten RTEs nicht auf eine Trennung von zeitkritischer und nicht-zeitkritischer Kommunikation. Stattdessen sollen alle Arten von Datenströmen das Netzwerk parallel nutzen können. Die zentrale Rolle in diesem Konzept kommt den Switches zu – diese stellen die Komponenten des Netzwerks dar, an denen mehrere Datenströme zusammentreffen können. Ein Switch muss dann die unterschiedlichen Ströme vor seinen Weiterleitungsentscheidungen so gewichten, dass die angeforderten Echtzeiteigenschaften eingehalten werden können. Eine rein prioritätsgesteuerte Weiterleitung, wie sie schon heute eingesetzt wird [IEEE 802.1Q 2014], ist für diese Zwecke nicht mehr ausreichend – beispielsweise können höherprioräre Frames verzögert werden, sofern der Zielport des Switches bereits durch einen Sendevorgang eines Frames niedrigerer Priorität belegt ist.

Aus diesen Gründen befindet sich derzeit ein zeitgesteuerter Weiterleitungsmechanismus für Switche in der Standardisierung [IEEE 802.1Qbv 2015], welcher als Time-Aware Shaper bezeichnet wird. Dieser basiert ebenfalls wie die prioritätsgesteuerte Weiterleitung auf mehreren (bis zu 8) Ausgangspuffern pro Port, wobei die zu sendenden Frames gemäß ihres Priority Code Points in den entsprechenden Puffer eingeteilt werden (siehe Unterabschnitt 2.2.3). Die zusätzliche Funktionalität des Time-Aware Shapers liegt darin, dass jedem Ausgangspuffer ein Transmission-Gate vorgeschaltet ist, sodass der Shaper aktiv steuern kann, welcher Puffer zu welchem Zeitpunkt auf den Ausgangsport zugreifen kann. Bei Puffern für zeitkritische Frames wird mittels eines Guardband genannten Intervalls sichergestellt, dass der Ausgangsport zum vorgesehenen Sendezeitpunkt nicht durch Frames anderer Puffer blockiert wird. Ein Beispiel mit zwei Ausgangspuffern ist in

Abbildung 2.17 dargestellt.



**Abbildung 2.17:** Sendezeitregelung durch den Einsatz eines Time-Aware Shapers (nach [Steiner u. a. 2015])

In diesem Beispiel gibt es zwei Ausgangspuffer: jeweils einen für zeitkritische Frames (Time Aware-Frames) und einen für nicht-zeitkritische Frames (Other Frames). Der Zugriff der Puffer auf den Ausgangsport wird durch den Time-Aware Shaper geregelt, der zum Zeitpunkt  $t_1$  den TA-Puffer aktiviert. Der Frame O1 kann noch vor dem Frame TA1 gesendet werden, da seine Übermittlung noch innerhalb des Guardband-Intervalls fertiggestellt werden kann. Für den Frame O2 gilt dies nicht mehr, er kann daher erst nach der Deaktivierung des TA-Puffers zum Zeitpunkt  $t_2$  gesendet werden.

Die Zeitpunkte für die Aktivierung und Deaktivierung der einzelnen Puffer müssen in einem Kommunikationsfahrplan festgelegt werden. Die Berechnung dieses Fahrplans, sein Format und der Mechanismus für seine Verteilung an die einzelnen Switches sind noch offene Fragestellungen.

## 2.3 Engineering von automatisierungstechnischen Anlagen

Der Kern dieser Arbeit beschäftigt sich mit der Reduzierung des Inbetriebnahme-Aufwandes von Echtzeit-Kommunikationssystemen in automatisierungstechnischen Anlagen. Wie auch schon in Kapitel 1 erwähnt, stellt die Konfiguration der Kommunikation jedoch nur einen Teil des gesamten in der Automatisierungstechnik notwendigen Engineerings dar. Zur besseren Einordnung des von dieser Arbeit adressierten Teilaspekts soll dieser Abschnitt einen Überblick über den Engineering-Prozess von automatisierungstechnischen Anlagen geben.

Alle Teilschritte des Engineerings lassen sich in einem ersten Schritt in der Regel einem bestimmten Teil des Lebenszyklus einer automatisierungstechnischen Anlage zuordnen, welcher nach [Obst u. a. 2013] in die in Abbildung 2.18 dargestellten Phasen unterteilt werden kann.



**Abbildung 2.18:** Lebenszyklus einer automatisierungstechnischen Anlage [Obst u. a. 2013]

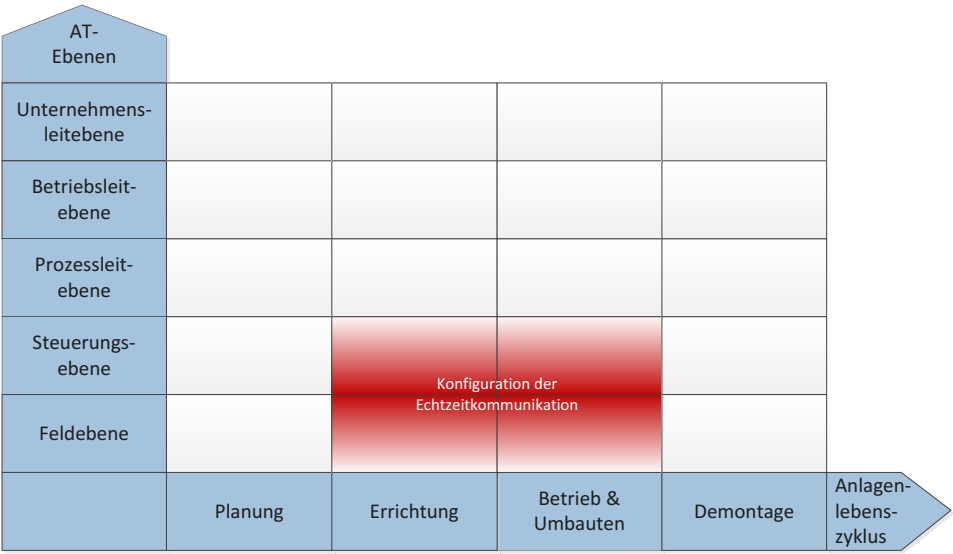
In der **Planung** wird zuerst die zu erfüllende Aufgabe der automatisierungstechnischen Anlage festgelegt und in Teilfunktionalitäten und Prozessschritte zerlegt. Anschließend werden die zur Umsetzung des Prozesses notwendigen konkreten technischen Hilfsmittel bestimmt. Beispielhafte Aufgaben sind hier die Auswahl von Automatisierungskomponenten aus den Katalogen der Hersteller sowie die mechanische, elektrische und hydraulische Planung der Anlage. Durch den Einsatz von Planungswerkzeugen (z. B. EPLAN [EPLAN Software 2016] zur Erstellung von Stromlaufplänen) entstehen in dieser Phase sogenannte Engineering-Artefakte, welche später auch zur automatischen Konfiguration des Echtzeitnetzwerkes genutzt werden können (siehe Abschnitt 5.2).

Während der **Errichtung** werden die Vorgaben aus der Planungsphase in die Praxis umgesetzt und die Anlage wird in den Betriebszustand versetzt. Zum einen findet die mechanische Montage der Anlage in dieser Phase statt, zum anderen werden die elektronischen Komponenten wie Steuerungen und Feldgeräte in Betrieb genommen.

Während der Phase des **Betriebes** werden Alarm- und Warnmeldungen vom Bedienerpersonal beobachtet und ggf. werden entsprechende Maßnahmen zur Behebung der Meldungsursache durchgeführt. Es können jedoch auch **Umbauten** der Anlage

stattfinden, beispielsweise zur Instandsetzung defekter Komponenten oder zur Prozessoptimierung. Am Ende des Lebenszyklus steht die **Demontage** der Anlage, wobei einzelne Komponenten in anderen Anlagen unter Umständen auch wiederverwendet werden können.

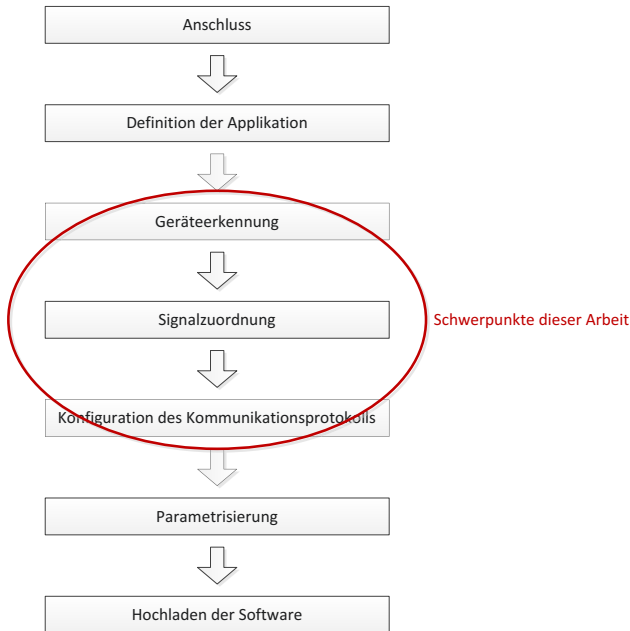
Zur Klassifizierung der einzelnen Engineering-Schritte ist eine Unterteilung in Lebenszyklusphasen jedoch nicht ausreichend. Es müssen weiterhin auch die verschiedenen Ebenen der Automatisierungstechnik gemäß der Automatisierungspyramide in Abbildung 2.2 berücksichtigt werden, da je nach Ebene unterschiedliche Engineering-Aufgaben zu leisten sind. Daher wird zur Einordnung dieser Arbeit das in [Hodek 2013] gezeigte zweidimensionale Schema genutzt, in dem alle zu leistenden Engineering-Aufwände für automatisierungstechnische Anlagen bestimmten Feldern zugeordnet werden können. Die Konfiguration der Echtzeitkommunikation als Kernthema dieser Arbeit ist an der Schnittstelle zwischen Steuerungs- und Feldebene angesiedelt und betrifft die Errichtungsphase bei der Erstinbetriebnahme einer Anlage sowie die Betriebs- und Umbauphase bei notwendigen Rekonfigurationen. Die entsprechenden Felder sind in Abbildung 2.19 rot markiert.



**Abbildung 2.19:** Einordnung dieser Arbeit in das Klassifizierungsschema für Engineering-Aufwände nach [Hodek 2013]

Innerhalb der identifizierten Felder müssen im Einzelnen die in Abbildung 2.20 aufge-

fürten und im Folgenden beschriebenen Schritte zur Inbetriebnahme von Feldgeräten durchgeführt werden [Hammerstingl u. Reinhart 2015]. Die Reihenfolge der Ausführung der einzelnen Schritte kann dabei variieren.



**Abbildung 2.20:** Schritte der Inbetriebnahme auf Feld- und Steuerungsebene

Zu Beginn muss der physikalische **Anschluss** des Feldgeräts zum Kommunikationssystem hergestellt werden. Bei RTEs wird dabei meist ein Ethernet-kompatibler RJ45-Stecker verwendet.

Während der **Definition der Applikation** wird die für die Steuerung des zu automatisierenden Prozesses benötigte Applikationslogik erstellt. In der Regel stellen die Hersteller von Steuerungskomponenten integrierte Entwicklungsumgebungen für die Programmierung ihrer Steuerungen zur Verfügung, wobei als Programmiersprachen meist die in der Norm [IEC 61131-3] beschriebenen Sprachen unterstützt werden.

Anschließend müssen die Feldgeräte dem zentralen Kommunikationsknoten (meistens der SPS) bekannt gemacht werden und ihnen müssen Adressen zugeteilt werden. Manche Hersteller von Konfigurationswerkzeugen unterstützen den Anwender dabei mit einer automatischen **Geräteerkennung**, sofern dies bei dem verwendeten Kommunikationssystem möglich ist.

Damit die Kommunikationsobjekte der Steuerungslogik, beispielsweise externe Variablen, den korrekten Ein- und Ausgängen der Feldgeräte zugeordnet werden können, müssen die Objekte durch eine **Signalzuordnung** den entsprechenden physikalischen Adressen der Sensorik/Aktorik zugewiesen werden.

Während der **Konfiguration des Kommunikationsprotokolls** müssen für alle Prozessdaten, die zwischen Steuerung und Feldgeräten ausgetauscht werden sollen, bestimmte Parameter wie Datentyp, Datenlänge und zeitliche Eigenschaften (z. B. Zykluszeit) festgelegt werden. Zur Unterstützung des Anwenders können Gerätebeschreibungsdateien (siehe Unterabschnitt 2.3.2) in das Konfigurationswerkzeug geladen werden, welche die entsprechenden Informationen für das zugehörige Feldgerät bereits enthalten. Die Automatisierung dieses Schrittes stellt die Kernmotivation dieser Arbeit dar.

Während der **Parametrisierung** werden applikationsspezifische Betriebsparameter der Feldgeräte eingestellt, wie z. B. die Kalibrierungsdaten von Sensoren. Dieser Schritt wird entweder mittels herstellerspezifischer Tools durchgeführt oder über einheitliche Schnittstellen und Werkzeuge wie EDDL, FDT oder FDI (siehe Unterabschnitt 2.3.1).

Nach Durchführung der vorherigen Schritte erfolgt das **Hochladen der Software**, welche die Steuerungslogik als auch die Einstellungen des Kommunikationssystems enthält, in die SPS. Dieser Schritt erfolgt in der Regel unter Nutzung der Entwicklungsumgebung des SPS-Herstellers über ein proprietäres Protokoll.

Die vorgenannten Schritte zur Inbetriebnahme müssen weitestgehend manuell ausgeführt werden und müssen ggf. während der Betriebsphase erneut durchlaufen werden, sollte es zu Umbauten der Anlage kommen. Der dadurch hervorgerufene Konfigurationsaufwand steht, wie in Kapitel 1 erwähnt, im Widerspruch zur Forderung nach einfach rekonfigurierbaren automatisierungstechnischen Systemen. Der in Kapitel 3 vorgestellte Ansatz zur Autokonfiguration soll den Konfigurationsaufwand möglichst minimieren, wobei er nur Lösungen für die Schritte Geräteerkennung, Signalzuordnung sowie Konfiguration des Kommunikationsprotokolls anbietet. Für die anderen Schritte wurden in der Automatisierungstechnik bereits verschiedene Hilfsmittel entwickelt, durch deren Nutzung eine Anlage schneller und damit kostengünstiger in den betriebsbereiten Zustand versetzt werden kann. Der folgende Unterabschnitt gibt einen Überblick über diese Technologien.

### 2.3.1 Integrationstechnologien für Feldgeräte

Wie in Abschnitt 2.3 erläutert, sind zahlreiche manuelle Konfigurationsschritte für die Inbetriebnahme eines Automatisierungssystems notwendig. Ein wesentlicher Bestandteil dieses Prozesses ist die Integration von Feldgeräten in die Steuerungs- und Prozessleitebene. Die folgenden Unterabschnitte stellen Technologien vor, die das Inbetriebnahmepersonal bei seiner Arbeit unterstützen und die Integration erleichtern. Anschließend wird geprüft, inwiefern die genannten Konzepte sich auch für die RTE-Autokonfiguration nutzen lassen.

### 2.3.2 Kommunikationsspezifische Gerätebeschreibungsdateien

Für die Integration von Feldgeräten in die Steuerungsebene verwenden nahezu alle industriellen Echtzeit-Kommunikationssysteme Gerätebeschreibungsdateien. Diese elektronischen Datenblätter enthalten zahlreiche Informationen über das jeweilige Feldgerät, welche für Inbetriebnahme der Echtzeit-Kommunikation und für die Verknüpfung zwischen Steuerungslogik und Feldgerät notwendig sind. Beispielhafte Inhalte einer Gerätebeschreibung sind Informationen zur Geräteidentifikation, erforderliche Parametrierungsdaten, Ein- und Ausgabewerte incl. Datentypen und geräteinterner Adressen sowie Informationen über das zeitliche Verhalten des Gerätes wie die minimal unterstützte Zykluszeit.

Das Format der Gerätebeschreibungsdatei hängt vom verwendeten Kommunikationssystem ab. So verwendet Profinet General Station Description (GSD)-Dateien, welche in der Generic Station Description Markup Language (GSDML) definiert werden. In Ethernet/IP werden die Gerätebeschreibungsdateien als Electronic Data Sheet (EDS) bezeichnet, in Ethernet Powerlink und Ethercat wird das von CANopen genutzte XML Device Description (XDD)-Format eingesetzt. Allen Varianten ist gemein, dass die jeweiligen Formate die Extensible Markup Language (XML) als Basis zur hierarchischen Informationsgliederung nutzen.

### 2.3.3 Electronic Device Description Language

Die EDDL wird genutzt, um Gerätebeschreibungsdateien im Electronic Device Description (EDD)-Format zu spezifizieren. Der Einsatzzweck einer EDD-Datei liegt im Gegensatz zur Klasse der kommunikationsspezifischen Beschreibungsdateien nicht in der Konfiguration der Kommunikation, sondern in der Parametrierung, Wartung und Diagnose von Feldgeräten. Durch die EDD-Dateien wird eine einheitliche Schnittstelle zu den

Feldgeräten bereitgestellt, um diese zentral durch ein einheitliches Software-Tool über das vorhandene Kommunikationsnetzwerk konfigurieren zu können. EDD adressiert damit den in Abbildung 2.20 genannten Schritt „Parametrierung der Feldgeräte“.

Eine EDD-Datei enthält Beschreibungen aller Parameter des jeweiligen Feldgerätes. Im Gegensatz zu GSD-Dateien, in denen nur die Kommunikationseigenschaften angegeben sind, enthalten EDDs weitere Informationen wie Bereichsgrenzen, physikalische Einheit und Standardwert eines Parameters sowie grafische Oberflächen zur Visualisierung eines Feldgerätes und seiner Eigenschaften. Neben den Parametern können auch Funktionen innerhalb einer EDD definiert werden, mit deren Hilfe einfache Vorgänge wie z. B. die Kalibrierung eines Gerätes angesteuert werden können. EDD-Dateien sind nicht als Ersatz für die kommunikationsspezifischen Gerätebeschreibungen gedacht, welche für die Integration der Feldgeräte in die Steuerungsebene und die Kommunikationskonfiguration weiterhin benötigt werden.

### **2.3.4 Field Device Tool**

Die FDT-Technologie soll –ähnlich dem EDD-Konzept– einheitliche Schnittstellen für die Parametrierung von Feldgeräten und darüber hinaus für deren Einbindung in Prozessleitsysteme bereitstellen. Im Gegensatz zu EDD stellt FDT jedoch keine Beschreibungssprache dar – es handelt sich stattdessen um eine Software-Schnittstelle, mit der gerätespezifische Applikationen, die sogenannten Device Type Manager (DTM), auf standardisierte Weise in Leitsysteme oder Parametrierungstools integriert werden können. Die einzelnen DTMs werden auf dem FDT-Host, bei dem es sich üblicherweise um einen PC handelt, in die FDT-Rahmenanwendung geladen, welche sich als Laufzeitumgebung für DTMs auffassen lässt.

Ein DTM enthält Benutzeroberflächen für den Zugriff auf Feldgeräteparameter sowie für die Konfiguration und Bedienung der Geräte, wobei auch grafische Oberflächen mit komplexen hinterlegten Algorithmen realisiert werden können. Die Basistechnologien von FDT basieren auf den Microsoft-Diensten Component Object Model (COM) zur Kommunikation zwischen FDT-Rahmenanwendung und den DTMs sowie auf ActiveX für die Gestaltung der Benutzeroberflächen und der Programmlogik. FDT lässt sich daher nur in Zusammenhang mit dem Betriebssystem Windows von Microsoft verwenden.

### 2.3.5 OPC Unified Architecture

Die OPC Unified Architecture (OPC UA) ist eine plattformunabhängige Technologie für die Modellierung und den Austausch von Informationen. Sie wurde entwickelt, um Interoperabilität zwischen heterogenen Systemkomponenten über verschiedene Arten von Kommunikationssystemen im Bereich der Automatisierungstechnik zu schaffen. OPC UA ist eine Weiterentwicklung der Open Platform Communications (OPC)-Technologie, welche einen weitverbreiteten Standard für die Kommunikation zwischen Komponenten der Steuerungs- und Prozessleitebene darstellt.

Im Bereich der Modellierung stellt OPC UA objektorientierte Methoden für die Strukturierung von Informationen zur Verfügung ohne selbst konkrete Informationsmodelle festzulegen. Aufgrund dessen lässt sich für OPC UA kein bestimmtes Einsatzfeld wie z. B. bei FDT angeben. Es obliegt stattdessen den Nutzern von OPC UA, das Anwendungsgebiet und damit einhergehende Informationsmodelle zu definieren. Dementsprechend werden im Folgenden nur die generischen Eigenschaften von OPC UA beschrieben, eine konkrete Anwendung findet sich in der in Unterabschnitt 2.3.6 erläuterten FDI-Technologie.

Abbildung 2.21 zeigt die wesentlichen Elemente einer typischen OPC UA Client/Server-Architektur und deren Zusammenhänge. Eine OPC UA-Applikation greift sowohl server- als auch clientseitig auf eine einheitliche Programmierschnittstelle (englisch: Application Programming Interface (API)) zu, welche unabhängig vom verwendeten Kommunikationssystem ist. Die Umsetzung der Applikationsdaten auf konkrete Übertragungsprotokolle übernimmt der Kommunikationsstack, wobei der OPC UA-Standard derzeit Anbindungen an die Protokolle Hypertext Transfer Protocol (HTTP), SOAP und an ein OPC UA-internes Binärprotokoll definiert. Letzteres erzeugt den geringsten Mehraufwand bei der Datenübertragung und wird daher genutzt, wenn Daten hochperformant übertragen werden müssen.

Die Server-Applikation wird in OPC UA durch Objektstrukturen und deren Instanzen repräsentiert, eine Objektinstanz kann beispielsweise einen konkreten Sensor mit seinen Parametern und ggf. auch seinen Funktionen darstellen. Innerhalb des OPC UA-Adressraums werden alle Objekte einschließlich deren Instanzen durch verknüpfte Knoten abgebildet. Auf die entstehende Knotenhierarchie kann ein Client durch verschiedene Methoden zugreifen.

Weiterhin kann der Client den Server durch eine Subskription anweisen, bestimmte Knoten zu überwachen. Der Client erhält anschließend bei Änderungen des überwach-

ten Elements Mitteilungen vom Server.

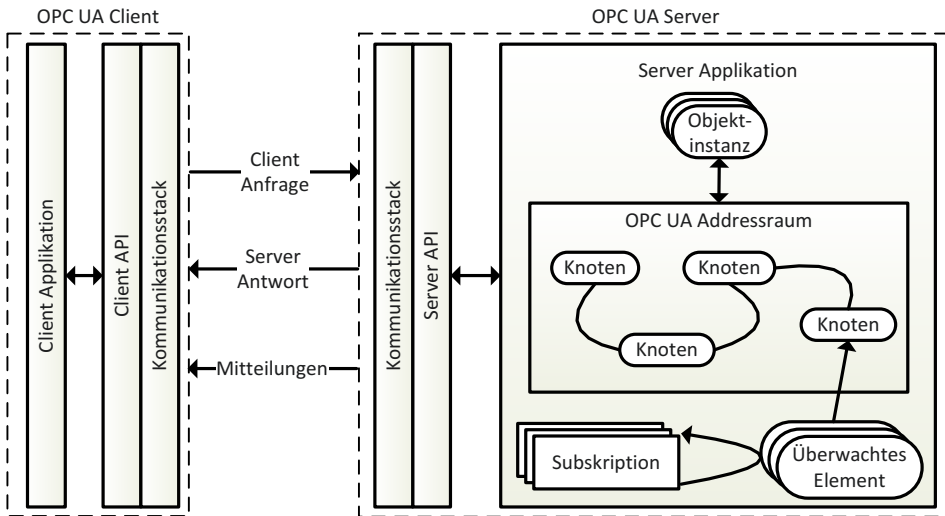
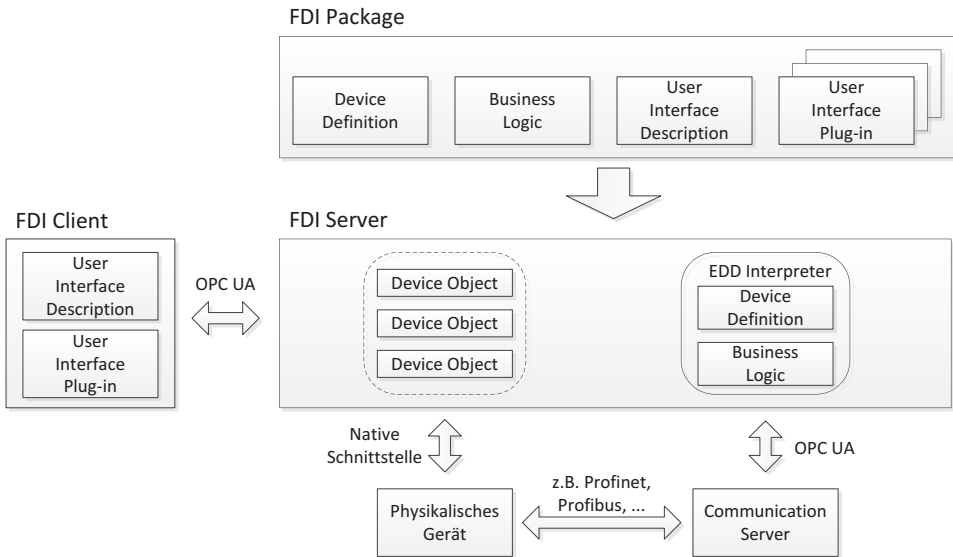


Abbildung 2.21: Kernelemente der OPC Unified Architecture [Dürkop u. a. 2013]

### 2.3.6 Field Device Integration

Das FDI-Konzept soll zukünftig die derzeit am Markt vorherrschenden Technologien zur einheitlichen Feldgeräteparametrierung, EDDL und FDT, in einem gemeinsamen Standard vereinen. FDI kombiniert dabei die Vorteile von EDDL wie Plattformunabhängigkeit und Einfachheit sowie von FDT wie die Möglichkeit zur Realisierung komplexer Funktionalitäten. Einen Überblick über die FDI-Architektur gibt Abbildung 2.22.

Das FDI Device Package enthält alle für die Parametrierung eines Gerätes notwendigen Informationen. Es enthält eine Device Definition mit Verwaltungs- und Geräteinformationen sowie eine Business Logic, welche die Kommunikationsschnittstelle beschreibt und zur Konsistenzsicherung dient. So kann die Business Logic beispielsweise eine Methode beinhalten, die bei einem Wechsel der physikalischen Einheit eines Parameters (z. B. von Grad Fahrenheit zu Grad Celsius) den Zahlenwert entsprechend anpasst. Weiterhin enthält das Device Package eine User Interface Definition für die Beschreibung der Geräteparameter und -funktionen sowie User Interface Plug-Ins für frei programmierbare grafische Benutzeroberflächen. Letztere Komponente wird entsprechend dem aus FDT bekannten DTM-Prinzip realisiert, die drei anderen Komponenten werden in der EDDL beschrieben.



**Abbildung 2.22:** Architektur des Field Device Integration-Konzepts (nach [Rachut u. a. 2013])

Die FDI Device Packages werden in einen FDI-Server geladen, welcher für jedes Gerät ein Device Object anlegt und die ihm zur Verfügung gestellten einzelnen Gerätebeschreibungen zu einem einheitlichen Informationsmodell aggregiert. Die Struktur dieses Modells wird im FDI-Standard [IEC 62769-5 2015] festgelegt, die Implementierung erfolgt mittels OPC UA. Die im Device Package enthaltenen EDDL-Beschreibungen der Device Definition und der Business Logic werden vom FDI-Server mittels eines Interpreters direkt ausgeführt, während die User Interface-Komponenten nur verwaltet werden, da diese letztendlich auf den FDI-Clients ausgeführt werden müssen. Die Clients, beispielsweise Parametrierungswerkzeuge, greifen auf die im FDI-Server hinterlegten Informationen ebenfalls über OPC UA zu.

Nimmt der Client Änderungen an Geräteparametern vor oder ruft Methoden eines Gerätes auf, muss der FDI-Server diese Informationen an das entsprechende Gerät weiterleiten. Dies erfolgt entweder über eine native Schnittstelle zwischen FDI-Server und Gerät oder über einen FDI Communication Server. Letzterer wird eingesetzt, wenn die zu parametrierenden Geräte nicht direkt mit dem FDI-Server verbunden sind, sondern beispielsweise über einen Feldbus. Der Communication Server fungiert in diesem Fall als ein Gateway zwischen dem eingesetzten industriellen Kommunikationssystem und dem FDI-Server. Die Kommunikation zwischen FDI-Server und Communication Server

erfolgt ebenfalls über OPC UA, für die Anbindung an das unterliegende Kommunikationsprotokoll definiert FDI verschiedene Profile, unter anderem für Profinet.

### **2.3.7 Bewertung**

Wie in Kapitel 3 dargestellt wird, ist der Mechanismus zur automatischen RTE-Konfiguration auf externe Informationsquellen angewiesen, aus denen er die zur Inbetriebnahme des Netzwerkes erforderlichen Parameter extrahiert. Die in den vorhergehenden Unterabschnitten vorgestellten Integrationstechnologien können als entsprechende Quellen genutzt werden. Insbesondere auf die kommunikationsspezifischen Gerätebeschreibungsdateien muss während der Autokonfiguration zurückgegriffen werden, da diese für den Betrieb des Echtzeitnetzes unabdingbare Informationen enthalten.

Die weiteren Beschreibungstechniken wie EDDL, FDT und FDI dienen in erster Linie der vereinfachten Feldgeräteparametrierung und -überwachung. Diese Schritte stellen zwar ebenfalls wichtige Teilaufgaben im Lebenszyklus einer automatisierungstechnischen Anlage dar – sie stehen jedoch nicht im Fokus dieser Arbeit, welche sich primär, wie in den Abbildungen 2.19 und 2.20 gezeigt, mit der automatischen Konfiguration des Echtzeitkommunikationssystems zwischen Feld- und Steuerungsebene beschäftigt.

Jedoch benötigt die Autokonfiguration nicht nur kommunikationsspezifische Informationen über die einzelnen Feldgeräte – so müssen für die automatische Prozessdatenzuordnung (siehe Abschnitt 5.2) auch Informationen über den Inhalt und die Bedeutung der Feldgeräteparameter bekannt sein. Zu diesem Zweck kann auf die mittels EDDL in der Device Definition des entsprechenden FDI Packages formulierten Parameterbeschreibungen zurückgegriffen werden. Zum Abruf der Beschreibungen bietet sich ebenfalls die von FDI genutzte RTE-unabhängige OPC UA-Technologie an. Die Integration dieser Lösungen in das Konzept zur automatischen Konfiguration wird in Kapitel 5 näher beschrieben.

## **2.4 Plug & Play-Konzepte in der Informationstechnik**

Das in dieser Arbeit entwickelte Konzept zur automatischen Konfiguration von Echtzeitkommunikationsnetzwerken soll dazu beitragen, dass Feldgeräte an ein Netzwerk angeschlossen werden können und ihre Funktionalitäten einer Steuerungsapplikation möglichst ohne manuelle Eingriffe zur Verfügung stehen. Das Konzept einer konfigurationslosen Integration von sowohl Hardware- als auch Software-Komponenten in über-

geordnete Applikationen ist in der Informationstechnik weit verbreitet und wird dort mit dem Begriff „Plug and Play“ –in diesem Abschnitt mit PnP abgekürzt– bezeichnet. Durch das Wort „Play“ wird der besondere Bezug auf Endanwender verdeutlicht, denen es beispielsweise ermöglicht werden soll, neue Komponenten an einen Heimcomputer anzuschließen und diese sofort nutzen zu können.

Ein frühes Beispiel für PnP ist die Installation von Erweiterungskarten bei PCs der x86-Architektur. Bei dem ursprünglich verwendeten Industry Standard Architecture (ISA)-Bus war es erforderlich, dass der Benutzer die benötigten Systemressourcen der Erweiterungskarte mittels Steckbrücken oder kleinen Schaltern zuteilen musste. Dieser fehlerträchtige Vorgang wurde beim ISA-Nachfolger, dem Peripheral Component Interconnect (PCI)-Bus, durch eine automatische Ressourcenzuweisung ersetzt.

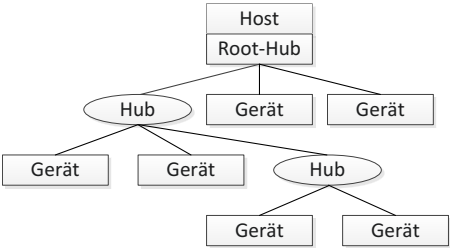
Die folgenden Unterabschnitte geben einen Überblick über zwei aktuelle und verbreitete Standards in der Informationstechnik, bei denen die Unterstützung von PnP eine der Kernfunktionalitäten darstellt. In Unterabschnitt 2.4.3 wird geprüft, inwiefern die von beiden Technologien genutzten Konzepte zum konfigurationslosen Betrieb auf diese Arbeit übertragen werden können.

### **2.4.1 Universal Serial Bus**

Der Universal Serial Bus (USB) ist eine Schnittstelle zum einheitlichen Anschluss von Peripheriegeräten an Computer. Vor der Einführung von USB wurden im PC-Bereich verschiedene Standards zum Geräteanschluss wie die serielle und parallele Schnittstelle sowie PS/2 genutzt. Wie auch beim ISA-Bus mussten den dort angeschlossenen Geräten bestimmte Systemressourcen zugeteilt werden, weiterhin konnte pro Anschluss in der Regel nur ein Gerät angeschlossen werden. Bei nicht ausreichenden Schnittstellen mussten Erweiterungskarten installiert werden, wobei unterschiedliche Gerätearten oftmals unterschiedliche Stecker erforderten. Der Anschluss eines neuen Gerätes war mit dem Neustart des kompletten Systems verbunden.

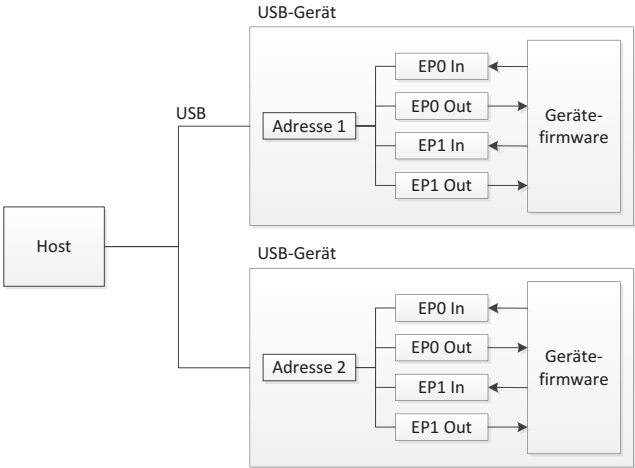
Die Einführung von USB hat die genannten Nachteile beseitigt und hat weiterhin durch kontinuierliche Weiterentwicklung immer höhere Datenübertragungsraten ermöglicht, sodass USB die meisten anderen Schnittstellen für Peripheriegeräte heute ersetzt hat. USB folgt einer Master/Slave-Architektur, in der bis zu 127 Geräte, die Slaves, in einer mehrstufigen Sterntopologie an einen zentralen Host, dem Master, angeschlossen werden können. Die Mehrstufigkeit wird durch Hubs erreicht, welche jeweils den Anschluss mehrerer Geräte erlauben, wobei maximal sechs Hubs kaskadiert werden

können. Ein Beispiel für eine mögliche USB-Topologie wird in Abbildung 2.23 gezeigt.



**Abbildung 2.23:** Beispielhafte Topologie eines USB-Systems

Wie in Abbildung 2.24 gezeigt wird, enthält jedes USB-Gerät einen oder mehrere Endpunkte, zu denen der Host jeweils eigenständige Kommunikationsverbindungen aufbauen kann. Ein Endpunkt kann dabei uni- oder bidirektionale Kommunikation unterstützen sowie in einem von vier Übertragungsmodi betrieben werden: Der isochrone Transfer wird für Verbindungen mit garantierter Datenrate verwendet, der Interrupt-Transfer für die zyklische Abfrage von Geräten, der Bulk-Transfer für große und nicht zeitkritische Datenmengen sowie der Control-Transfer für den Austausch von Konfigurationsdaten. Die genaue Konfiguration der Endpunkte wird vom jeweiligen USB-Gerät festgelegt.



**Abbildung 2.24:** USB-Kommunikationsstruktur

Die PnP-Fähigkeit von USB beruht hardwareseitig auf der automatischen Konfigurati-

on der businternen Kommunikation sowie softwareseitig auf einheitlichen funktionalen Schnittstellen für bestimmte Geräteklassen. Die Grundzüge beider PnP-Mechanismen werden im Folgenden dargestellt. Die Beschreibungen beziehen sich auf den USB-Standard 2.0.

### **Konfiguration der Kommunikation**

Die Konfiguration der Kommunikation, welche in USB Enumeration genannt wird, ist für den Endanwender vollkommen transparent und besteht u. a. aus den folgenden Schritten:

- **Geräteerkennung:** Physikalisch verwendet USB ausschließlich Punkt-zu-Punkt Verbindungen zwischen Hub und Gerät. Wird ein neues Gerät an einen Hub angeschlossen, so erkennt der Hub den Einsteckvorgang über die veränderten elektrischen Eigenschaften am entsprechenden Port.
- **Aushandeln der Datenrate:** USB unterstützt mehrere Geschwindigkeitsklassen. Nachdem der Hub den Anschluss eines neuen Gerätes erkannt hat, handeln Hub und Gerät über eine sogenannte Chirp-Sequenz die höchstmögliche Geschwindigkeit aus, die von beiden Partnern unterstützt wird.

Nachdem die physikalische Verbindung zwischen Hub und Gerät aufgebaut wurde, informiert der Hub den Host über den Anschluss eines neuen Gerätes. Der Host übernimmt anschließend den weiteren Konfigurationsvorgang:

- **Adresszuweisung:** Im uninitialisierten Zustand ist das neue Gerät über die Standard-Adresse 0 erreichbar. Der Host nutzt diese Adresse, um mit dem Gerät eine Kommunikationsverbindung aufzubauen und diesem anschließend eine neue eindeutige Adresse zuzuweisen.
- **Abruf der Gerätekonfiguration:** Der Host ruft grundlegende Informationen über das Gerät ab. Diese Daten enthalten u. a. eine Hersteller-ID, eine Produkt-ID, eine Seriennummer sowie die Anzahl der Endpunkte einschließlich deren Konfigurationen. Die Informationen sind in festgelegten Datenstrukturen, den Deskriptoren, enthalten. Der Zugriff auf die Deskriptoren erfolgt immer über den Endpunkt 0, den somit jedes USB-Gerät unterstützen muss.
- **Kommunikationsplanung:** Die zur Verfügung stehende Bandbreite wird in Zeitschlitze unterteilt, welche vom Host den einzelnen Endpunkten gemäß ihren in

den Deskriptoren enthaltenen Anforderungen zugeteilt werden. Isochrone und Interrupt-Transfers dürfen dabei in der Summe maximal 80 % der zur Verfügung stehenden Bandbreite reservieren. Überschreitet die Anforderung eines neu angeschlossenen Gerätes diese Grenze, schlägt der Verbindungsaufbau fehl. Für Control-Transfers werden statisch 20 % der Bandbreite reserviert. Bei Bulk-Transfers erfolgt keine Bandbreitenreservierung, diese werden immer dann ausgeführt, wenn im jeweiligen Kommunikationszyklus alle anderen Transfers abgeschlossen sind.

- **Abschluss der Konfiguration:** Nachdem die Kommunikation zwischen Host und Gerät aufgebaut ist, meldet der Host den Anschluss eines neuen Gerätes an das Betriebssystem. Ab diesem Zeitpunkt kann eine Applikation unter Nutzung eines generischen USB-Treibers Daten mit den Endpunkten des Gerätes austauschen.

### **Einheitliche funktionale Schnittstelle**

In USB wird die komplette Buskonfiguration, wie soeben beschrieben, autark von internen Mechanismen gesteuert, sodass Applikationen nach dem Hinzufügen eines Gerätes automatisch auf dessen Endpunkte zugreifen können. Um die Kommunikation mit der Hardware zu erleichtern, werden zusätzlich Gerätetreiber verwendet, die den Zugriff auf das Gerät abstrahieren, sodass die Anwendung keine binären Rohdaten mit den Geräte-Endpunkten austauschen muss. Stattdessen bietet der Treiber für die Nutzung der Gerätefunktionalitäten entsprechende Funktionsaufrufe an und setzt diese intern in die Steuersignale und -daten um.

Allerdings müssen Applikationen bei der Verwendung von Gerätetreibern normalerweise an den gerätespezifischen Treiber angepasst werden, da verschiedene Treiber die gleiche Gerätefunktionalität unterschiedlich implementieren können. Im Ergebnis könnte eine Applikation nur mit den Geräten derjenigen Hersteller zusammenarbeiten, für deren Treiber sie explizit entworfen wurde.

Aus diesen Gründen nutzt USB das Prinzip der Geräteklassen. Eine Geräteklasse fasst dabei die Ansteuerung verschiedener Geräte ähnlicher Funktionalität zusammen, indem in ihr einheitliche Attribute und Funktionen, über die alle Geräte einer Klasse gesteuert werden können, definiert werden. Durch die Verwendung eines einzigen Klassentreibers wird eine Applikation damit in die Lage versetzt, unterschiedliche Geräte verschiedener Hersteller zu nutzen. Die Zugehörigkeit zu einer bestimmten Geräteklasse teilt ein USB-Gerät dem Host während der Enumeration durch entsprechende Deskriptoren mit.

Durch die Verwendung von Klassentreibern wird es dem Endbenutzer ermöglicht, Geräte an einen PC anzuschließen und diese ohne weitere manuelle Schritte zu benutzen. Die Voraussetzung dafür sind zum einen die Unterstützung der jeweiligen Geräteklasse durch das Betriebssystem und zum anderen die Existenz einer entsprechenden Klasse für das Gerät. Letzterer Punkt ist weniger eine technologische als eine organisatorische Problemstellung. So ist die Definition und Ausgestaltung von Geräteklassen maßgeblich davon abhängig, ob der Standardisierungsprozess von den Geräteherstellern unterstützt wird und ob sie ihre Produkte den Vorgaben der Klassendefinition entsprechend konzipieren. Im Falle von USB haben sich Hard- und Softwarehersteller in Form des USB Implementers Forum zusammengeschlossen, welches eine Vielzahl von Geräteklassen (z. B. die Human Interface Device Class, welche u. a. Definitionen für Tastaturen und Mäuse enthält) geschaffen hat. Die breite Herstellerunterstützung hat maßgeblich zum Erfolg von USB beigetragen.

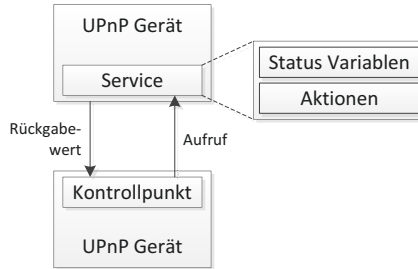
Allerdings müssen USB-Geräte nicht zwangsläufig kompatibel zu einer bestimmten Klasse sein, sodass es drei Möglichkeiten für den Entwurf von USB-Geräten gibt:

- Das USB-Gerät entspricht vollständig einer Klassendefinition. Dementsprechend steht seine Funktionalität uneingeschränkt einer Applikation zur Verfügung (vorausgesetzt, dass diese die Klasse ebenfalls unterstützt).
- Das USB-Gerät implementiert eine Klassendefinition, bietet darüber hinaus jedoch noch zusätzliche Funktionalitäten an. In diesem Fall können Applikationen, welche auf der entsprechenden Geräteklasse beruhen, nur eine Grundfunktionalität des Gerätes nutzen. Für alle weiteren Funktionen muss die Applikation gerätespezifisch angepasst werden.
- Das USB-Gerät lässt sich keiner Geräteklasse zuordnen. Hier können die Gerätefunktionalitäten nur von Applikationen genutzt werden, die speziell auf das Gerät ausgelegt sind.

## 2.4.2 Universal Plug and Play

Universal Plug and Play (UPnP) bezeichnet einen auf Protokollen aus dem Internet-Umfeld basierenden Standard für die konfigurationslose Zusammenarbeit von verteilten IT-Komponenten, welche über ein IP-basiertes Netzwerk miteinander verbunden sind. In UPnP stellen Endgeräte ihre Funktionalitäten in Form von Diensten zur Verfügung (siehe Abbildung 2.25), welche von Kontrollpunkten aus aufgerufen werden können. Ein

Dienst beinhaltet dabei ausführbare Aktionen und abrufbare Statusvariablen. In einem Gerät können ebenfalls mehrere Dienste und Endpunkte sowie eine Kombination aus beiden realisiert werden. UPnP ist damit eine in der IT eingesetzte Ausprägung des SOA-Paradigmas (siehe Unterabschnitt 2.5.1).



**Abbildung 2.25:** Grundlegende Architektur eines UPnP-Systems

UPnP legt fest, wie durch Einsatz bereits existierender Protokolle Dienste definiert, aufgefunden und kontrolliert werden können:

- **Adressierung:** Jedes UPnP-Gerät enthält einen Client für das Dynamic Host Configuration Protocol (DHCP), welches für die dynamische Zuweisung von IP-Adressen eingesetzt wird. Befindet sich kein entsprechender DHCP-Server im Netzwerk, weist sich jedes Gerät im entsprechenden Netz selbstständig eine IP-Adresse gemäß dem Internet-Standard Request for Comments (RFC) 3927 [RFC 3927 2005] zu.
- **Erkundung:** Geräte geben ihre Dienste im Netzwerk per Multicast bekannt und nutzen dazu das Simple Service Discovery Protocol (SSDP). Kontrollpunkte können weiterhin ebenfalls über SSDP nach bestimmten Diensten suchen.
- **Beschreibung:** Nachdem ein Endpunkt einen passenden Dienst gefunden hat, ruft er dessen Beschreibung mittels HTTP ab. UPnP definiert Vorgaben für das Format der Dienstbeschreibung, welche u. a. eine Liste der aufrufbaren Aktionen, Parameter für jede Aktion sowie eine Liste von Statusvariablen enthält.
- **Kontrolle:** Ein Endpunkt kann mittels HTTP die ihm nach Abruf der Beschreibungen bekannten Aktionen eines Dienstes aufrufen. Die Ergebnisse des Aufrufs spiegeln sich in einer Änderung der Statusvariablen des Dienstes wieder.

- Benachrichtigung: Wenn sich der Zustand eines Dienstes ändert, verschickt dieser Ereignisbenachrichtigungen. Um diese zu empfangen, muss ein Kontrollpunkt sich vorher beim Dienst registrieren und die gewünschten Benachrichtigungen abonnieren.
- Darstellung: UPnP-Geräte können optional in ihrer Gerätebeschreibung eine Adresse anbieten, über die ein Endanwender nähere Informationen über das Gerät abrufen kann. Dabei kann es sich beispielsweise um die URL des Web-Interfaces des Gerätes handeln.

UPnP gibt über die oben genannten Methoden den Rahmen vor, wie verschiedene Geräte über ein Netzwerk miteinander interagieren können. Zusätzlich definiert UPnP –vergleichbar mit den Geräteklassen in USB– konkrete Gerätetypen und deren Dienst-Schnittstellen.

### 2.4.3 Bewertung

Im Folgenden soll geprüft werden, welche der vorgestellten Plug-and-Play Konzepte von USB und UPnP sich für die automatische Konfiguration von Echtzeit-Ethernet eignen. Die Ergebnisse dieser Analyse werden in Kapitel 5 bei der Erstellung des Autokonfigurations-Konzepts aufgegriffen.

#### USB

- Geräteerkennung: Ein ähnliches Verfahren zur elektrischen Erkennung von Netzknoten und zur Aushandlung der Geschwindigkeitsklasse ist unter dem Begriff Autonegotiation bereits im Ethernet-Standard definiert. Eine weitere Eigenschaft von USB ist es, dass der Host über den Anschluss bzw. das Entfernen eines Gerätes automatisch informiert wird. Einen vergleichbaren Mechanismus gibt es bei Ethernet nicht, insbesondere existiert kein mit der Rolle des Hosts vergleichbarer zentraler Knoten, welcher informiert werden könnte.

*Fazit: nicht übertragbar*

- Adresszuweisung: Auf Ethernet-Ebene besteht keine Notwendigkeit für eine automatische Zuweisung von Adressen, da jeder Ethernet-Knoten bereits eine vom Hersteller voreingestellte weltweit einmalige Hardwareadresse enthält. Für höhere Protokollschichten kann jedoch eine Adressvergabe ähnlich dem in USB eingesetzten Verfahren geeignet sein. So müssen beispielsweise in Ethernet Powerlink

alle Geräte manuell mit einer einmaligen Node ID versehen werden. In Analogie zu USB könnte stattdessen jeder Powerlink-Knoten mit einer einheitlichen vordefinierten Node ID das Netzwerk betreten, woraufhin der zentrale Managing Node den Knoten ihre eindeutigen IDs automatisch zuweist. Als Voraussetzung für ein solches Verfahren müsste die starre Prozessdatenzuordnung zwischen Steuerungsapplikation und Feldgerät über die Node ID aufgehoben werden.

*Fazit: bedingt übertragbar*

- Abruf der Gerätekonfiguration: In Profinet muss für modulare Geräte der Ausbaugrad während des Engineerings angegeben werden. Würde der Autokonfigurations-Mechanismus die Gerätekonfiguration dahingegen selbstständig auslesen, könnte dieser manueller Schritt der Inbetriebnahme entfallen.

*Fazit: übertragbar*

- Kommunikationsplanung: USB basiert wie die meisten RTEs auf einer zentral durchgeführten TDMA-basierten Ressourcenzuteilung. Jedoch nutzt USB eine andere Informationsquelle als Planungsgrundlage als dies in RTEs der Fall ist. So werden in USB die Ressourcenanforderungen von den Endgeräten spezifiziert – bei RTEs werden die zeitlichen Parameter für den Prozessdatenaustausch manuell vom Benutzer vorgenommen.

Eine Übertragung des USB-Prinzips auf die Automatisierungstechnik ist nur bedingt möglich: Prinzipiell könnten auch die Ressourcenanforderungen der Feldgeräte aus ihren Gerätebeschreibungsdateien ausgelesen werden, da dort alle zyklisch zu übertragenden Daten in Form von Ausgangs- und Eingangsparametern beschrieben werden. Zusätzlich enthalten die Dateien für jedes Gerät die minimal unterstützte Zykluszeit. Auf Basis dieser Informationen könnte ein „worst case“-Kommunikationsplan berechnet werden, welcher für jegliche Parameter aller Feldgeräte eine Übertragung in der minimalen Zykluszeit vorsieht. Auf diese Weise kann sichergestellt werden, dass alle Prozessdaten rechtzeitig zwischen Steuerung und Feldgerät ausgetauscht werden.

Eine wesentliche Einschränkung dieses Ansatzes stellt jedoch die erzeugte Netzlast dar. Da der Datenaustausch unter Umständen in wesentlich kürzeren Zeitabständen erfolgt als dies für die korrekte Funktionalität der Applikation notwendig wäre, kann die maximale Leistungsfähigkeit des Echtzeit-Netzes deutlich früher erreicht werden als bei der konventionellen manuellen Kommunikationsplanung.

*Fazit: bedingt übertragbar*

- Einheitliche funktionale Schnittstelle: Dies ist die aus Sicht eines Applikationsentwicklers wichtigste PnP-Fähigkeit von USB. Sie ermöglicht es, Anwendungen ohne Wissen über die konkret eingesetzten Peripheriekomponenten zu entwickeln, solange ein geeigneter Klassentreiber zur Verfügung steht. Weiterhin muss der Endbenutzer keine manuellen Einstellungen vornehmen, um die Kommunikation zwischen Applikation und Peripherie zu ermöglichen. Ein ähnliches Verfahren könnte in der Automatisierungstechnik die manuelle Prozessdatenzuordnung zwischen Steuerung und Feldgeräten ersetzen. Anstatt über Variablen oder ähnliche Kommunikationsobjekte, die mit den physikalischen Adressen der Ein- und Ausgänge verbunden werden müssen, würde eine Steuerungsapplikation nur noch mit den entsprechenden Klassentreibern kommunizieren. Die Weiterleitung der Prozessdaten an die Feldgeräte würde von den Treibern übernommen werden.

Dieses Prinzip lässt sich jedoch nur bedingt auf die Automatisierungstechnik übertragen: Zur Schaffung entsprechender Klassendefinitionen für Feldgeräte müsste seitens der Hersteller die dafür notwendige breite Unterstützung gewährt werden. Im Bereich von USB ist dies gelungen – dort wird jedoch mit Endverbrauchern eine andere Zielgruppe adressiert: Wäre im Konsumentenbereich die Inbetriebnahme von PCs mitsamt ihrer Peripherie nur mit Expertenwissen möglich, würde dies viele Verbraucher vor Investitionen abschrecken und der Markt für IT-Technik wäre wesentlich kleiner. Demgegenüber muss bei automatisierungstechnischen Komponenten eine entsprechende Nachfrage nicht erst geschaffen werden, diese ergibt sich durch die Notwendigkeit zur Produktions- und Prozessautomatisierung. Würden die Hersteller hier ihre Produkte standardisieren und damit leichter austauschbar machen, könnte dies zu einer starken Wettbewerbszunahme führen, welche nicht im Interesse der etablierten Hersteller sein dürfte. Im Gegenteil findet eher eine Individualisierung durch herstellerspezifische Funktionalitäten statt.

Weiterhin gibt es aus technischer Sicht Hindernisse bei einer direkten Übertragung des Klassentreiber-Prinzips auf die Automatisierungstechnik: Bei PCs erfüllen die meisten Peripheriegeräte eindeutige Funktionalitäten (z. B. Maus, Tastatur, Monitor) – in der Automatisierung gibt es jedoch auch Feldgeräte wie digitale Eingabe/Ausgabe-Komponenten, welche erst durch ihre Integration in den Prozess einer bestimmten Funktionalität zugeordnet werden können.

Allerdings gibt es auch in der Automatisierungstechnik in der Form von Profilen Ansätze zur einheitlichen Beschreibung von Feldgerätefunktionalitäten. So definiert

die Normenreihe IEC 61800-7 [IEC 61800-7 2007] beispielsweise eine universelle Schnittstelle für die Antriebstechnik und deren Umsetzung auf verschiedene Kommunikationssysteme und Applikationsprotokolle wie CIP (CIP Motion), CANopen (CiA 402) oder Profinet (PROFIdrive). Eine Abdeckung der meisten auf dem Markt befindlichen Feldgeräte durch entsprechende Profile ist in der Automatisierungstechnik jedoch –auch aufgrund der Vielzahl unterschiedlicher Geräte– jedoch nicht in Sicht.

*Fazit: bedingt übertragbar*

## UPnP

- Adressierung: Eine automatische Adressvergabe ist bei der RTE-Autokonfiguration prinzipiell möglich. Allerdings ist zu berücksichtigen, dass während der herkömmlichen Inbetriebnahme eines RTEs die Prozessdaten der Steuerung fest mit den Adressen der Feldgeräte verknüpft werden. Werden die Adressen erst während des Hochlaufens des Netzwerkes vergeben, so muss eine Lösung für eine dynamische Prozessdatenzuordnung gefunden werden.

*Fazit: bedingt übertragbar*

- Erkundung, Beschreibung, Kontrolle, Benachrichtigung: Bei UPnP werden die vorhandenen Geräte und ihre Dienste dynamisch zur Laufzeit erkundet. Anschließend werden zwischen Kontrollpunkt und Dienst Details ausgetauscht, welche zum Aufrufen des Dienstes erforderlich sind. Dieses Prinzip ließe sich auch auf die Automatisierungstechnik zum Ersatz bzw. zum Vereinfachen der statischen Prozessdatenzuordnung übertragen, indem man die Steuerungskomponente als Kontrollpunkt auffasst, welcher Dienste der Feldgeräte (z. B. Einlesen eines bestimmten Prozesswerts) aufruft.

Ähnlich wie bei der notwendigen Definition von Geräteklassen in USB benötigt allerdings auch dieses Konzept einen industrieweiten Standardisierungsprozess für die einheitliche Beschreibung von Diensten, um herstellerunabhängig Interaktionen zwischen der Steuerung und den von den Feldgeräten angebotenen Diensten zu gewährleisten. Weiterhin müssen die Echtzeitanforderungen der Automatisierung berücksichtigt werden, welche von den TCP/IP-basierten UPnP-Protokollen nicht erfüllt werden können. Diese Einschränkungen stellen jedoch die generelle Eignung des dienstbasierten Ansatzes nicht in Frage – tatsächlich ist er eine wesentliche Grundlage für viele Forschungsarbeiten zum Thema PnP in

der Automation (siehe auch Unterabschnitt 2.5.1).

*Fazit: übertragbar*

- Darstellung: Ähnlich wie bei UPnP könnten auch Feldgeräte die URL ihres Web-Interfaces im Netzwerk propagieren. Dieses könnte von Technikern genutzt werden, um das Feldgerät zu konfigurieren oder Diagnoseinformation abzurufen. Entsprechende Web-Server sind heute bereits auf einer Vielzahl von Feldgeräten enthalten – neu wäre die automatische Bekanntgabe der URL, sodass der Techniker beispielsweise nicht erst die IP-Adresse des Gerätes herausfinden muss.

*Fazit: übertragbar*

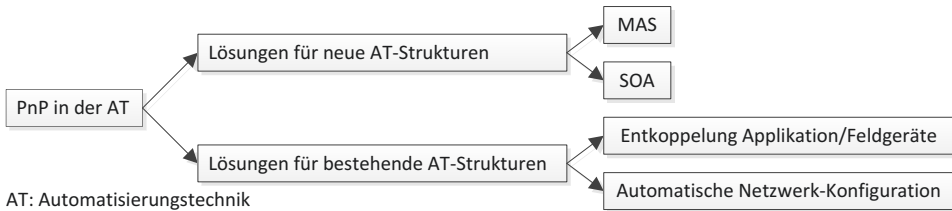
## 2.5 Plug & Play-Konzepte in der Automatisierungstechnik

Im Gegensatz zur Informationstechnik, in der sich PnP-Konzepte seit langem etabliert haben, sind entsprechende Konzepte in der Automatisierungstechnik bisher weitestgehend nur auf Forschungsebene betrachtet worden. Das generelle Ziel aller Arbeiten im Kontext PnP ist es dabei, manuelle Konfigurationsschritte zu reduzieren, um so im Idealfall eine automatisierungstechnische Anlage zu erhalten, die sich zur Inbetriebnahme oder nach Änderungen vollständig autonom (re-)konfigurieren kann.

Die verschiedenen Ansätze zur Realisierung von PnP, welche in diesem Abschnitt behandelt werden, lassen sich entsprechend Abbildung 2.26 gliedern. Die erste Differenzierungsebene orientiert sich an [Kothmayr u. a. 2015], welche zwei wesentliche Gruppen identifiziert haben: Die erste umfasst Vorschläge für völlig neue Automatisierungsarchitekturen – die Abkehr von der bekannten Pyramidenstruktur liegt in der Erkenntnis begründet, dass die üblichen Systeme zur Steuerung von Produktionsprozessen nicht die notwendige Modularisierung und Rekonfigurierbarkeit unterstützen [Rooker u. a. 2009]. Die zweite beinhaltet Ansätze, welche innerhalb der bekannten Strukturen der Automatisierungstechnik jeweils bestimmte Teilaspekte adressieren.

Letztere Gruppe wird in Unterabschnitt 2.5.2 näher untersucht, wobei nur Arbeiten betrachtet werden, welche sich entweder mit der Entkoppelung von Steuerungsapplikation und Feldgeräten oder mit der automatischen Netzwerk-Konfiguration beschäftigen.

Im Folgenden wird die Gruppe „Lösungen für neue Automatisierungstechnik-Strukturen“ betrachtet. Hier setzen alle Vertreter in ihrer Grundstruktur auf verteilte Systeme zur Realisierung von Automatisierungsaufgaben [Keddis u. a. 2013]. Die verbreitetsten



**Abbildung 2.26:** Gliederung der untersuchten PnP-Ansätze

Konzepte sind dabei Agentensysteme (Multi-Agent Systems (MASs)) sowie die in Kapitel 1 bereits erwähnten SOAs.

Bei einem MAS handelt es sich um ein System, welches aus mehreren autonomen Entitäten –den Agenten– besteht. Ein Agent ist in der Lage, ein vorgegebenes Ziel zu erreichen. Wenn dies aufgrund fehlenden Wissens oder fehlender Fähigkeiten nicht möglich sein sollte, interagiert er wiederum mit anderen Agenten zum Zwecke einer gemeinsamen Zielerreichung [Leitão 2009]. Aus der Gesamtheit aller Agenten ergibt sich ein MAS, welches in der Lage ist, durch das Verteilen von Funktionalitäten auf die einzelnen modularen, intelligenten und kombinierbaren Agenten Probleme autonom zu lösen [Leitão u. a. 2016]. Beispiele für die Anwendung von MASs in industriellen Anwendungen finden sich in [Nagorny u. a. 2012; Lepuschitz u. a. 2013; Yan u. Vyatkin 2013; Leitão u. a. 2013, 2016].

Das Grundgerüst einer SOA besteht aus unabhängigen, jedoch interoperablen, Diensten. Jeder Dienst legt dabei nur seine äußere Schnittstelle gegenüber anderen Diensten fest, die Implementierung ist von außen nicht sichtbar. Ein weiteres SOA-Grundprinzip ist die lose Kopplung: Alle Dienste können im Prinzip unabhängig voneinander operieren, ihre Interaktionen untereinander sind nicht zustandsbehaftet, asynchron und nicht kontextbezogen [Jammes u. a. 2005]. Die lose Kopplung der einzelnen Dienste führt zu einem hohen Grad an Modularität und Wiederverwendbarkeit. Beispiele von SOA-basierten industriellen Anwendungen finden sich in [Jammes u. Smit 2005; Cucinotta u. a. 2009; Jammes 2011; Delsing u. a. 2012; Karnouskos u. a. 2012; García Valls u. a. 2013; Loskyll 2013; Kyusakov u. a. 2013; Puttonen u. a. 2013; Girbea u. a. 2014; Ramis u. a. 2014; Colombo u. a. 2014; Dai u. a. 2015; Pfrommer u. a. 2015; Kothmayr u. a. 2015; Legat u. Vogel-Heuser 2015; Giret u. a. 2016; Kothmayr u. a. 2016].

Sowohl den MAS- als auch den SOA-basierten Ansätzen ist gemein, dass der Aspekt der Echtzeit-Kommunikation oft vernachlässigt wird. Zur Verdeutlichung wurden die

genannten Beispiele für industrielle MAS und SOA-Anwendungen daraufhin überprüft, inwiefern sie die Integration von echtzeitfähiger Kommunikation berücksichtigen. Wie in Tabelle 2.3 gezeigt wird, können die untersuchten Veröffentlichungen in sechs Klassen eingeteilt werden.

**Tabelle 2.3:** Klassifizierung beispielhafter industrieller MAS- und SOA-Anwendungen in Bezug auf ihre Echtzeit-Unterstützung

Klasse	Beschreibung und zugehörige Veröffentlichungen
I)	<i>Einsatz eines dedizierten Echtzeit-Kanals</i> Jammes u. Smit [2005], Cucinotta u. a. [2009], García Valls u. a. [2013], Colombo u. a. [2014], Kothmayr u. a. [2015]
II)	<i>Echtzeitkommunikation verlagert auf eine untere Ebene</i> Delsing u. a. [2012], Nagorny u. a. [2012], Leitão u. a. [2013], Leitão u. a. [2016]
III)	<i>Einsatz von IP-Protokollen für Echtzeit-Kommunikation</i> Cucinotta u. a. [2009], Jammes [2011], Kothmayr u. a. [2016]
IV)	<i>Echtzeit-Kommunikation explizit ausgeschlossen</i> Loskyll [2013], Kyusakov u. a. [2013]
V)	<i>Keine Vorschläge für die Integration der Echtzeit-Kommunikation</i> Karnouskos u. a. [2012]
VI)	<i>Echtzeit-Kommunikation nicht berücksichtigt</i> Lepuschitz u. a. [2013], Yan u. Vyatkin [2013], Puttonen u. a. [2013], Girbea u. a. [2014], Ramis u. a. [2014], Dai u. a. [2015], Pfrommer u. a. [2015], Legat u. Vogel-Heuser [2015], Giret u. a. [2016]

In der Mehrzahl der Ansätze wird der Aspekt der Echtzeit-Kommunikation nicht erwähnt und daher auch nicht weiter betrachtet (Klasse VI). In einer Veröffentlichung wird auf die Notwendigkeit der Echtzeit-Kommunikation hingewiesen, ohne jedoch weiter auf deren Integration einzugehen (Klasse V). Zwei weitere Veröffentlichungen schließen die Anwendbarkeit ihrer Lösungen auf Echtzeit-Systeme explizit aus (Klasse IV). Die Ansätze der Klasse III verwenden IP-basierte Protokolle für die Echtzeit-Kommunikation, welche sich jedoch nicht für alle Echtzeit-Anwendungen in der Automatisierungstechnik eignen (siehe Unterabschnitt 2.2.3). Die Veröffentlichungen der Klasse II setzen ihre vorgestellten MAS- bzw. SOA-Lösungen nur für nicht zeitkritische Kommunikation ein.

Komponenten mit Echtzeit-Anforderungen werden als abgeschlossene Subsysteme eingebunden, innerhalb derer herkömmliche Echtzeit-Kommunikationssysteme eingesetzt werden. Bei den Ansätzen der Klasse I werden dedizierte Echtzeit-Kanäle darüber hinaus auch für die Kommunikation zwischen Agenten bzw. Diensten eingesetzt – konkrete Vorschläge für die Realisierung dieser Kanäle finden sich jedoch in keiner der untersuchten Veröffentlichungen.

Zusammengefasst lassen sich den Ansätzen der Klassen 1 und 2 Möglichkeiten entnehmen, auf welche Weise Echtzeit-Kommunikationssysteme in MAS- und SOA-basierte Lösungen integriert werden könnten. Allerdings geht keine der Veröffentlichungen auf die Notwendigkeit der manuellen Konfiguration dieser Systeme ein, sodass eine vollständige PnP-Fähigkeit nicht gegeben ist. Das in dieser Arbeit erarbeitete Konzept für die automatische RTE-Konfiguration könnte perspektivisch insbesondere für die PnP-kompatible Realisierung des Echtzeitkanals in den MAS/SOA-Ansätzen der Klasse 1 verwendet werden. Da das Autokonfigurations-Konzept teilweise auf SOA-Techniken beruht, werden diese in Unterabschnitt 2.5.1 näher vorgestellt.

### **2.5.1 Serviceorientierte Architekturen**

Serviceorientierte Architekturen (SOAs) stammen ursprünglich aus dem IT-Bereich, wo sie zur Realisierung verteilter Geschäftsprozesse eingesetzt werden. Das europäische SIRENA-Projekt [Jammes u. Smit 2005] war eines der ersten Ansätze zur Portierung des SOA-Konzepts auf die Domäne der Automatisierungstechnik. In SIRENA wurde dazu das SOA-Konzept bis hinunter auf die Feldebene übertragen: Jedes Feldgerät bietet demnach seine Funktionalität in Form von Webservices an. Die entsprechenden Schnittstellen und die Interaktionen zwischen den Webservices werden mittels des Standards Device Profile for Web Services (DPWS) implementiert. Webservices sind dabei eine Technologie aus dem Internet-Umfeld, mittels derer verteilte Web-Anwendungen realisiert werden können. Die Schnittstelle eines Webservices wird in der XML-basierten Web Service Description Language (WSDL) beschrieben. Zur Kommunikation zwischen den Webservices wird das Protokoll SOAP in Verbindung mit HTTP genutzt.

Um aus einzelnen Diensten einen Geschäfts- oder Produktionsprozess zu generieren, müssen diese miteinander kombiniert werden und beispielsweise Ausführungsreihenfolgen festgelegt werden – dieser Vorgang wird als Orchestrierung bezeichnet. Eine gebräuchliche Methode zum Orchestrieren von Webservices ist die Business Process Execution Language (BPEL). Auf Basis der BPEL wurden im SIRENA-Nachfolgerprojekt

SOCRADES Werkzeuge entwickelt, welche den Orchestrierungsvorgang unterstützen, beispielsweise durch grafische Benutzeroberflächen.

Die in SIRENA und SOCRADES entwickelten Methoden können das Engineering von automatisierungstechnischen Systemen erleichtern, da das Inbetriebnahmepersonal nur noch für die korrekte Orchestrierung der Dienste verantwortlich ist, mit Details des Datenaustausches oder der Netzwerkkonfiguration muss es sich nicht mehr befassen. Für ein vollständiges PnP ist es allerdings darüber hinaus erforderlich, auch den Orchestrierungsprozess zu automatisieren.

Ein entsprechender Ansatz findet sich in [Loskyl 2013]. Das dort entwickelte Verfahren zur dynamischen Orchestrierung basiert auf der semantischen Beschreibung von Diensten, welche über die reine syntaktische Schnittstellenbeschreibung wie bei DPWS hinausgeht. Dabei werden alle semantisch annotierten Webservices, welche die Funktionalitäten der in einer automatisierungstechnischen Anlage vorhandenen Komponenten repräsentieren, in einem zentralen Webservice-Verzeichnis gespeichert. Jedes zu fertigende Produkt wird weiterhin durch eine abstrakte Prozessbeschreibung beschrieben, welche die zur Fertigung notwendigen Produktionsschritte enthält. Ein automatischer Fähigkeitsabgleich vergleicht anschließend die Prozessbeschreibung mit dem zentralen Dienstverzeichnis und bildet jeden abstrakten Prozessschritt auf eine konkrete Anlagenfunktionalität ab.

Ähnliche Ansätze werden auch in [Engel u. a. 2016], [Pfrommer u. a. 2015] und [Jasperneite u. a. 2015] vorgestellt. Engel u. a. setzen für die semantische Beschreibung von Produktionsmodulen allerdings nicht Webservices, sondern die Beschreibungssprache MathML ein. Sie präsentieren weiterhin einen konkreten Algorithmus zur Durchführung des Fähigkeitsabgleiches. Jasperneite u. a. erwähnen mit der fehlenden Echtzeitfähigkeit eine wesentliche Schwäche der bisherigen SOA-basierten PnP-Ansätze erwähnt. So werden für die Kommunikation zwischen den Diensten in der Regel IP-basierte Protokolle wie DPWS (so auch bei Loskyl und in SIRENA/SOCRADES) oder auch OPC UA (Pfrommer u. a.) eingesetzt. Wie im Abschnitt 2.2 beschrieben wurde, erfüllt IP in Verbindung mit Ethernet jedoch in der prozessnahen Kommunikation nicht die Echtzeitanforderungen der Klassen 2 und 3.

Loskyl hat dies ebenfalls erkannt und schließt die Anwendbarkeit seines Ansatzes für Anwendungen mit Echtzeitanforderungen explizit aus. Im SIRENA/SOCRADES-Umfeld wurde hingegen versucht, dem Echtzeit-Problem durch Verbesserungen an den Übertragungsprotokollen zu begegnen. So wird in [Colombo u. a. 2014] im Rahmen des SOCRADES-Nachfolgers IMC-AESOP vorgeschlagen, das in DPWS für den Datenaus-

tausch verwendete SOAP-Protokoll durch Efficient XML Interchange (EXI), welches eine binäres Übertragungsformat für XML darstellt, zu ersetzen. Es konnten zwar tatsächlich Effizienzsteigerungen festgestellt werden, jedoch setzt auch EXI letztendlich auf IP auf.

Andere Ansätze, welche die Echtzeitfähigkeit von SOAs verbessern sollen, finden sich beispielsweise im RI-MACS Projekt [Cucinotta u. a. 2009]. Dort wird eine separate API für zeitkritische Prozesse vorgeschlagen, welche parallel zur standardmäßigen DPWS-basierten API genutzt werden kann. Durch die getrennte API können, in Analogie zu RTEs der Klassen 2 und 3, die komplexen Protokollstapel von DPWS und TCP/IP umgangen werden. Eine ähnliche Lösung wurde im iLAND Projekt [García Valls u. a. 2013] in der Form einer Middleware für die Steuerung der Kommunikation zwischen den Diensten entwickelt. Die Middleware enthält neben dem Stack für nicht zeitkritische Kommunikation einen Platzhalter für die Echtzeitkommunikation. Beiden Forschungsprojekten ist gemein, dass sie keine näheren Angaben darüber machen, wie Echtzeitnetzwerke in ihre Ansätze integriert werden könnten.

## 2.5.2 Plug-and-Play für Echtzeit-Kommunikationssysteme

Dieser Unterabschnitt gibt einen Überblick über Arbeiten, welche sich mit der Reduzierung des Engineeringaufwandes für die Integration von Feldgeräten in die Steuerungsebene befassen.

Den meisten untersuchten Arbeiten ist gemein, dass sie für die Realisierung von PnP auf eine Entkopplung von Steuerungs- und Feldebene setzen, womit sie sich von der gegenwärtigen Praxis absetzen: In heutigen Steuerungsprogrammen wird häufig geräte- und herstellerspezifischer Programmcode zur Ansteuerung der Feldgeräte genutzt [Hodek 2013]. Demgegenüber steht das PnP-Prinzip in der Informationstechnik, welches einen reinen funktionalitätsorientierten Zugriff von Applikationen auf Peripheriegeräte unter Nutzung einer einheitlichen Programmierschnittstelle vorsieht (siehe Unterabschnitt 2.4.3). Die konkrete Ansteuerung der Peripherie wird durch gerätespezifische Treiber gekapselt.

Um dieses Prinzip auf die Automatisierungstechnik zu übertragen, wurde am Aachener Lehrstuhl für Prozessleittechnik (ACPLT) im Rahmen des Projekts Universaler Feldbuskanal (UniFeBu) [Epple 2009] eine einheitliche Schnittstelle für den Zugriff von PC-Anwendungen auf Feldbusse entwickelt. Zur Trennung von Applikation und Hardware wurden drei Ebenen eingeführt: Eine Diensteschicht stellt busunabhängige Funktionen zum Setzen und Lesen von Parametern und Variablen sowie zur Buskonfiguration zur

Verfügung, eine Feldbusschicht übersetzt die Dienstaufrufe in busspezifische Funktionen und eine Hardwareabstraktionsschicht stellt einen herstellerunabhängigen Zugang zur verwendeten Schnittstellenkarte bereit. Die Buskonfiguration (u. a. die vorhandenen Geräte) muss manuell aus der Anwendung heraus über die Methoden der Diensteschicht erfolgen, teilweise müssen auch die Konfigurationswerkzeuge der Hersteller der Schnittstellenkarte verwendet werden. Das UniFeBu-Prinzip erlaubt es damit, Applikationen unabhängig vom verwendeten Feldbus zu entwickeln – die Konfiguration des Busses muss jedoch manuell erfolgen.

Zur Trennung von Anwendung und Kommunikation wurde im PAPAS-Projekt [Plank u. a. 2006] ein Kommunikationsmodell entwickelt, welches sich an USB orientiert. Jedes Feldgerät wird als eine Ansammlung von Endpunkten aufgefasst, wobei die genaue Anzahl und die Funktionalität der Endpunkte durch geräteklassenspezifische Profile vorgegeben wird. Um der auf dem Master laufenden Anwendung einen einheitlichen Zugriff auf die Endpunkte zu ermöglichen, werden wie auch bei UniFeBu Abstraktionsschichten für den Buszugriff genutzt. Aus Kommunikationssicht besteht Plug-and-Play bei PAPAS in der automatischen Erkennung von Feldgeräten und dem anschließenden Laden eines entsprechenden Gerätetreibers. Für die automatische Konfiguration des Echtzeitbusses ist im PAPAS-Modell zwar ein eigenständiger Funktionsblock vorgesehen, dessen Funktionalität im Rahmen des Projektes jedoch nicht ausgefüllt wurde.

Der Schwerpunkt in Hodeks Arbeit [Hodek 2013] liegt ebenfalls in der Entwicklung eines einheitlichen Modells für den Zugriff einer Applikation auf Feldgerätefunktionalitäten: Seine Lösung für die automatische Integration von Feldgeräten in die Steuerungsebene führt eine API als Abstraktionsebene ein, durch deren Nutzung herstellerunabhängiger Steuerungscode geschrieben werden kann. Die Abbildung der API-Aufrufe auf geräteindividuelle Befehle erfolgt durch entsprechende Gerätetreiber. Im Gegensatz zu beiden letztgenannten Arbeiten adressiert Hodek jedoch auch die Selbstkonfiguration des Kommunikationssystem, wozu er eine Analyse bestehender Kommunikationsstandards durchführt. Im Ergebnis kommt er zu dem Schluss, dass sich industrielle Echtzeitkommunikationssysteme aufgrund des manuellen Konfigurationsaufwandes nicht für die automatische Feldgeräteintegration eignen. Er entwickelt daher ein eigenes UPnP-ähnliches Kommunikationskonzept für die Adressierung, Erkundung, Beschreibung und Nutzung von Feldgeräten, welches auf bestehende Protokolle aus dem TCP/IP-Umfeld aufsetzt. Ein echtzeitfähiger Prozessdatenaustausch ist mittels Hodeks Ansatz somit nicht möglich.

Ebenfalls mit der Konzeption einer funktional einheitlichen Applikationsschnittstelle,

jedoch speziell für Echtzeit-Ethernet, befasst sich die Arbeit von Lechler [Lechler 2011]. Dazu wurden verschiedene RTE miteinander verglichen und es wurden einheitliche Geräte- und Parametermodelle entwickelt, die einen Zugriff auf die Feldgeräte unabhängig vom verwendeten Netzwerk ermöglichen. Auf die Konfiguration der RTEs wird jedoch nicht eingegangen und es bleibt offen, inwiefern dieser Schritt innerhalb seines Ansatzes manuell oder automatisch durchgeführt werden kann.

Im Projekt CAPRI [Imtiaz u. Jasperneite 2013a] wurden verschiedene PnP-Verfahren für RTEs realisiert. Für die Entkopplung von Applikation und Kommunikationsschicht wird eine einheitliche Modellierung von Feldgeräten mittels eines OPC UA-Informationsmodells vorgeschlagen, welches unabhängig vom verwendeten RTE ist. OPC UA wird hier nur für die Beschreibung der Feldgeräte und ihrer Parameter eingesetzt, der echtzeitfähige Prozessdatenaustausch selber soll entweder weiterhin mittels RTEs oder der Nutzung der AVB-Erweiterung von Ethernet erfolgen. Wie das Informationsmodell auf ein bestimmtes Kommunikationssystem abgebildet werden kann, wird allerdings offengelassen. Weiterhin wurde im CAPRI-Projekt ein Mechanismus für die automatische Vergabe von MAC-Adressen an Geräte abhängig von ihrer Position in der Netzwerktopologie entwickelt.

Im Gegensatz zu den letztgenannten Arbeiten, welche sich mit der Vereinheitlichung des Applikationszugriffs auf Feldgeräte beschäftigen, finden sich zu der Thematik der automatischen Konfiguration von Echtzeitnetzwerken weitaus weniger Lösungsvorschläge in der Literatur. Um eine vollautomatische Integration von Feldgeräten in die Steuerungsebene zu ermöglichen, müssen jedoch beide Aspekte berücksichtigt werden.

In der Arbeit von Krug [Krug 2013] wird eine Methode für die automatische Konfiguration von Robotersystemen vorgestellt, wobei der Schwerpunkt auf der Entwicklung einer allgemeinen Architektur für die automatische Konfiguration der Echtzeitkommunikation zwischen Robotersteuerung und Peripheriegeräten liegt. Es werden fünf Schritte (Physikalische Verbindung, Geräteerkennung, Basiskonfiguration, Informationsgewinnung, Konfiguration) identifiziert, welche die Grundlage für den automatischen Konfigurationsprozess bilden. Die Durchführung dieser Schritte obliegt einem Konfigurationsmanager, der mittels TCP/IP-basierter Protokolle mit den Peripheriegeräten kommuniziert, um die für Konfiguration des Echtzeitnetzwerkes notwendigen Informationen zu erhalten. Allerdings verbleibt Krugs Arbeit auf einem abstrakten Level ohne auf konkrete Echtzeitnetzwerke einzugehen.

Das abstrakte 5-Schritte-Model von Krug bildet die Grundlage für konkrete Ansätze zur automatische RTE-Konfiguration: In [Reinhart u. a. 2010] wird es für die Autokon-

figuration des RTEs Ethernet Powerlink verwendet. In [Dürkop u. a. 2012b] wird ein entsprechend strukturierter automatischer Konfigurationsablauf für Profinet vorgestellt, wobei für die Geräteerkennung SOA-Funktionalitäten genutzt werden.

Ein sehr ähnliches Konzept wird in den Arbeiten von Krüning [Krüning u. Eppele 2013; Krüning 2015], welche eine Erweiterung des bereits erwähnten UniFeBu-Projektes darstellen, für die Autokonfiguration von Profinet eingesetzt. Jedes Feldgerät enthält dabei eine Selbstbeschreibung mit den für die Konfiguration des Profinet-Netzes erforderlichen Geräteeigenschaften. Der Abruf der Beschreibungen erfolgt unter Nutzung des NRT-Kanals von Profinet. Dies entspricht konzeptionell dem in der vorliegenden Arbeit vorgeschlagenen Verfahren für die Geräteerkennung in Profinet (siehe Abschnitt 5.4). In Krünings Arbeiten bleibt allerdings offen, wie aus den erlangten Gerätebeschreibungen eine konkrete Profinet-Parametrierung erzeugt werden kann. Er verweist hierzu auf das UniFeBu-Projekt, in dessen Dokumentation [Eppele 2009] allerdings zu dieser Thematik ebenfalls keine Informationen verfügbar sind.

Basierend auf dem Ansatz von [Dürkop u. a. 2012b] wird in [Regulin u. a. 2015] ein Verfahren für die automatische Konfiguration von Ethercat vorgestellt. Zur Geräteerkennung werden dort Ethercat-eigene Methoden verwendet, welche jedoch Erweiterungen am standardmäßigen Verhalten eines Ethercat-Masters bedingen. Ein Plug-and-Play Mechanismus für Profibus wird in [Pöschmann u. Krogel 2000] gezeigt: Die Autoconfiguration Management Framework genannte Komponente beschränkt sich jedoch auf die Identifizierung der vorhandenen Feldgeräte und auf eine automatische Adressvergabe.

Die genannten Ansätze beschränken sich überwiegend auf einzelne Netzwerktechnologien, sodass sie nicht auf RTE-übergreifende Möglichkeiten der automatischen Konfiguration eingehen. Eine maßgebliche Einschränkung ist weiterhin, dass der Einfluss der Applikationsebene auf die Konfiguration nicht weiter berücksichtigt wird. Regulin u. a. erkennen zwar eine entsprechende Notwendigkeit, liefern jedoch keine konkreten Lösungsansätze.

## 2.6 Zusammenfassende Bewertung

Sowohl die Inbetriebnahme als auch die Rekonfiguration automatisierungstechnischer Systeme erfordern heute hohe manuelle Konfigurationsaufwände, wobei sich die notwendigen Schritte je nach betrachteter Automatisierungsebene unterscheiden. So gibt es beispielsweise für die Integration von Feldgeräten in die Prozessleitebene bereits etablierte unterstützende Technologien wie EDDL und FDT sowie zukünftig eventuell

auch FDI.

Für das notwendige Engineering für die Feldgeräteintegration auf Steuerungsebene sind solche Hilfswerkzeuge dahingegen weniger verbreitet, hier müssen im Wesentlichen zwei Punkt manuell vorgenommen werden: Zum einen muss die Steuerungsapplikation an die verwendeten Feldgeräte angepasst und mit diesen verknüpft werden, zum anderen muss das Echtzeitkommunikationssystem parametrierbar werden. Aus Analogiebetrachtungen zur Informationstechnik können Rückschlüsse gezogen werden, wie der notwendige Konfigurationsaufwand für diese beiden Schritte minimiert werden kann. Ein entsprechendes Konzept in der IT ist unter dem Begriff Plug-and-Play bekannt, worunter „die automatische Erkennung, Konfiguration und Einbindung von Hardwarekomponenten ohne manuellen Eingriff“ [Hodek 2013] verstanden wird. Wie in der Bewertung von USB (siehe Unterabschnitt 2.4.1) gezeigt wurde, basiert seine PnP-Fähigkeit im Wesentlichen auf einer einheitlichen Programmierschnittstelle für Anwendungen, auf der Standardisierung von Gerätefunktionalitäten in Form von Geräteklassen sowie auf der für die Anwendung transparenten Konfiguration der Kommunikation.

Bei der Übertragung des USB-Konzepts auf die Automatisierungstechnik steht man vor der Schwierigkeit einer sehr heterogenen Kommunikationslandschaft. Während USB von vornherein als Ersatz für die bis zu seiner Einführung verbreiteten Schnittstellen gedacht war, müssen Konzepte für die Automatisierungstechnik eine Integration der etablierten Kommunikationsstandards berücksichtigen. Erst langfristig sich könnte mit der Einführung von TSN (siehe Unterabschnitt 2.2.10) die Möglichkeit einer einheitlichen Kommunikationsinfrastruktur ergeben.

Weiterhin benötigen die in der Automatisierung eingesetzten Echtzeitnetze –im Gegensatz zu den in der IT verbreiteten Standards– eine umfangreiche Parametrierung vor der Inbetriebnahme. Dieser Gesichtspunkt wird von nahezu allen Forschungsansätzen zu PnP in der Automatisierungstechnik nicht weiter adressiert. Stattdessen wird das Thema Echtzeitkommunikation in der Regel ausgeklammert oder eine Integration von normalerweise für den Echtzeitdatenaustausch genutzten Technologien wie RTEs wird nicht berücksichtigt (siehe Abschnitt 2.5).

Automatische Konfiguration von Echtzeit-Ethernet

Dürkop, L.

2017, X, 249 S. 82 Abb., 30 Abb. in Farbe., Softcover

ISBN: 978-3-662-54124-1