Technische Universität Dresden

Plug and Produce für modulare verfahrenstechnische Anlagen

Dipl.-Ing.

Michael Obst

der Fakultät Elektrotechnik und Informationstechnik der Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender: Prof. Dr.-Ing. habil. Ercan Altinsoy
Gutachter: Prof. Dr.-Ing. habil. Leon Urbas
Gutachter: Prof. Dr.-Ing. Dr. h. c. Michael Weyrich

Tag der Einreichung: 09.01.2017
Tag der Verteidigung: 03.12.2018

Inhaltsverzeichnis

Abkürzungsverzeichnis

aHMI	*abstract Human-Machine Interface*
AML	*AutomationML*
AWL	*Anweisungsliste*
B&B	Bedienen und Beobachten
BDE	*Betriebsdatenerfassungs-System*
BEP	Break-Even-Point
CAEX	*Computer Aided Engineering Exchange*
CFC	*Continuous Function Chart*
CPPS	*Cyber Physical Production System*
CPS	*Cyber-physisches System (engl. cyber-physical system)*
DIN	*Deutschen Instituts für Normung*
DKE	*Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE*
DSM	*Design-Structure-Matrix*
DTM	*Device Type Manager*
EDD	*Electronic Device Description*
EDDL	*Electronic Device Description Language*
EN	*Europäische Norm*
ERP	*Enterprise-Resource-Planning*
FAT	*Factory Acceptance Test*
FBS	*Funktionsbausteinsprache*
FDI	*Field Device Integration*
FDT	*Field Device Tool*
GSD	*Generic Station Description*
GSDML	*General Station Description Markup Language*
HMI	*Human Machine Interface*
HTML	*Hypertext Markup Language*
IEC	*Internationale Elektrotechnische Kommission, engl. International Electrotechnical Commission*
ISA	*International Federation of the National Standardizing Associations*
ISO	*International Organization for Standardization*
IT	*Informationstechnik*
konti	kontinuierlich
KOP	*Kontaktplan*
LS	*Leitsystem*

mA	*Milliampere*
MES	*Manufacturing Execution System*
OPC	*OLE for Process Control, ab 11/2011 Open Platform Communications*
OPC UA	*OPC Unified Architecture*
PAAT	*Prozess-, Apparate- und Anlagentechnik*
PFE	*Prozessführungsebene*
PLT	*Prozessleittechnik*
PNML	*Petri Net Markup Language*
POE	*Programmorganisationseinheiten*
PPEAM	*Process Plant Engineering Activity Model*
R&I Fließschema	*Rohrleitungs- und Instrumentenfließschema*
RAMI	*Referenzarchitekturmodell*
SFC	*Sequential Function Chart*
SPS	*speicherprogrammierbare Steuerung*
ST	*Strukturierter Text Text*
UIML	User Interface Description Language
VDE	*Verband der Elektrotechnik Elektronik Informationstechnik e.V.*
VDI	*Verein Deutscher Ingenieure*
VDI/VDE-GMK	*VDI/VDE Gesellschaft Meß- und Regelungstechnik*
VDMA	*Verband Deutscher Maschinen- und Anlagenbau*
XAML	*Extensible Application Markup Language*
XIML	eXtensible Interface Markup Language
XML	*Extensible Markup Language*
XSD	*XML Schema*
ZVEI	*Zentralverband Elektrotechnik- und Elektronikindustrie e.V*

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Motivation

Die Absatzmärkte der Prozessindustrie, insbesondere im Bereich der Chemie, Pharmazie und Nahrungsmittelherstellung, werden stetig volatiler: durch die globale Verfügbarkeit von Alternativprodukten verkürzen sich Produktlebenszyklen, gleichzeitig unterliegen die nachgefragten Mengen starken regionalen und vor allem zeitlichen Schwankungen, welche zunehmend schwerer prognostizierbar sind. Um Produktinnovationen erfolgreich vermarkten zu können, benötigen diese bei einem zunehmend härteren Wettbewerb nach erfolgreicher Zulassung eine schnelle Serienreife des Prototyps. Sobald das Produkt vom Markt gut angenommen wurde, ist die Zeit bis zur bis zum Erreichen der geforderten Produktmenge in der geforderten Produktqualität wesentlich für die Wirtschaftlichkeit. Spätestens gegen Ende des Produktlebenszyklus sollte die Produktion nahe an den größten verbliebenen Absatzmärkten stattfinden, das heißt, entsprechend verlagert werden können. Die klassischen Produktionsverfahren der Prozessindustrie erfüllen diese Anforderungen nur unzureichend: Konti-Anlagen sind für eine bestimmte Produktionsmenge pro Zeiteinheit optimiert, die möglichst über Jahre nicht verändert werden sollte, die höhere Flexibilität konventioneller Batch-Anlagen geht mit unproduktiven Zeiten, zum Beispiel während des Umrüstens und damit Ineffizienz einher. Besonders vielversprechend erscheint in dieser Hinsicht die Modularisierung von verfahrenstechnischen Anlagen, bei der Anlagen durch die Kombination von Modulen flexibel aufgebaut werden. Einzelne Module realisieren jeweils standardisierte Produktionsschritte und können entsprechend des herzustellenden Produkts kombiniert werden. Änderungen des Produkts werden durch den Tausch von Modulen realisiert, die Produktionsmenge kann durch Hinzufügen gleichartiger Module erhöht werden (Urbas, Doherr, Krause & Obst, 2012).

1.2 Forschungsfragen

Die Integration eines Moduls in ein übergeordnetes klassisches Leitsystem ist mit den heute verfügbaren Informationsmodellen und Werkzeugen mit großen manuellen Aufwänden verbunden. Verschiedene Aspekte der Automatisierung wie Bedienbilder, Zustände von Ablaufketten oder Verriegelungen müssen für die Visualisierung und Führung des Moduls in einem übergeordneten Leitsystem manuell nachgebildet werden. Jedoch sind heutige Leitsysteme nicht dafür vorbereitet, das geforderte flexible Führen (NE 148) einer aus verschiedenen Modulen aufgebauten Anlage zu ermöglichen. Hierzu ist eine modulare Plug-and-Produce Methodik notwendig. Für diese Methodik wird eine durchgängige Informationsmodellierung, beginnend bei einem modularen funktionsorientierten integrierten Engineering benötigt. Diese Arbeit untersucht dabei folgende Fragestellungen:

- Welche Inhalte sind für die Einbindung eines Moduls in die übergeordnete Leitsystemstruktur notwendig?
- Welche Beschreibungsmittel sind geeignet, um die zuvor identifizierten Inhalte zu transportieren?

Um diese Punkte zu untersuchen, wird eine Architektur sowie ein strukturierter Prozess zur Integrationen eines Moduls erarbeitet.

1.3 Aufbau der Arbeit

Die vorliegende Dissertation gliedert sich neben einer Einleitung und der Schlussbetrachtung in insgesamt vier Teile.

In Kapitel 2 werden die notwendigen Grundlagen für das Verständnis der Arbeit vorgestellt. Neben der Einführung in Verfahrenstechnische Anlagen und deren Engineering wird ein Einblick in das allgemeine Prinzip des modularen Engineerings geben. Darauf folgend werden die spezifischen Eigenschaften modularer prozesstechnischer Anlagen vorgestellt, grundlegende Vorteile und Herausforderungen beschrieben, sowie auf den allgemeinen Life-Cycle eines prozesstechnischen Moduls und weitere wichtige Begriffe Bezug genommen.

Das folgende Kapitel 3 widmet sich dem aktuellen Stand der Technik zur Beschreibung eines prozesstechnischen Moduls. Basierend auf zuvor aufgestellten Integrationsaspekten werden, sowohl spezifisch für eine oder mehrere Aspekte, als auch allgemeine Beschreibungsmittel hinsichtlich ihrer Eignung untersucht. Das Kapitel schließt mit einer Zusammenfassung und einer entsprechenden Empfehlung zur Auswahl eines Beschreibungsmittels. Ausgehend von den vorgestellten Integrationsaspekten und der Analyse, wird ein entsprechender Handlungsbedarf beschrieben, aus welchem in dem sich anschließenden Kapitel 4, eine allgemeine Informationsmodellierung für ausgewählte Integrationsaspekte resultiert. Die hierfür notwendige Architektur und ein entsprechender Integrationsprozess eines Moduls werden zunächst erarbeitet. Das Kapitel widmet sich anschließend der Überführung der Modelle in konkrete Beschreibungsmittel und zeigt deren Vor- und Nachteile auf.

Basierend auf den Implementierungsvorschlägen des vierten Kapitels, werden in Kapitel 5 zwei Umsetzungen in die Praxis vorgestellt.

2 Grundlagen

In diesem Kapitel werden zunächst Grundlagen klassischer verfahrenstechnischer Anlagen und deren Engineering erläutert. Im weiteren Verlauf werden allgemeine Modulare Engineering Ansätze erläutert, sowie deren Anwendung auf verfahrenstechnische Anlagen beschrieben. Abschließend werden die Grundlagen zum Thema „Plug and Produce“ vorgestellt.

2.1 Verfahrenstechnische Anlagen und deren Engineering

Verfahrenstechnik wird durch (Christen, 2010, S. 4) als allgemeiner Prozess, in dem Stoffe und Materialien verarbeitet werden, definiert. Dieses Verfahren findet unter anderem bei der Herstellung von Fahrzeugen, Haushaltgeräten oder Verpackungen Anwendung. (Christen, 2010, S. 4) präzisiert diese Definition als chemische Verfahrenstechnik oder Chemie-Ingenieur-Technik für die Herstellung chemischer Erzeugnisse. Das eigentliche Ziel der chemischen Verfahrenstechnik beschreibt (Christen, 2010, S. 3) als die Stoffumwandlung. Die Stoffumwandlung kann erfolgen durch:

- Änderung der Zusammensetzung, beispielsweise durch Filtration einer Suspension, durch Destillation einer Lösung oder durch Trocknung eines Pulvers,
- Änderung der Form und Größe, beispielsweise durch Zerkleinerung von Erzgestein oder durch Kristallisation eines Salzes,
- Änderung der Stoffart, beispielsweise durch chemische oder biochemische Reaktionen.

Für die technische Realisierung der Verfahren in der Produktion kommen verschiedenste Prozessführungs- und Anlagentypen zum Einsatz. Nach (Hertwig & Martens, 2007, S. 7) können grundsätzlich die diskontinuierliche oder kontinuierliche Betriebsweise eines technischen Prozesses unterschieden werden.

- In der *diskontinuierlichen Betriebsweise* (Batch Prozess) erfolgt zu Beginn des Vorganges das Einfüllen der zu behandelnden Stoffmengen in den Apparat, um dort einer oder mehreren verfahrenstechnischen Operationen unterzogen werden zu können.
- Die *kontinuierliche Betriebsweise* (Fließbetrieb, Konti-Prozess) ist durch einen stetigen Zu- und Ablauf der Ein- und Austrittsströme gekennzeichnet. Im jeweiligen Apparat läuft in der Regel ein Hauptprozess ab, der aber vielfach von weiteren Operationen begleitet wird (beispielsweise chemische Reaktion mit Wärme- und Stoffübertragungsvorgängen).
- Als Mischform kommt in chemischen und biotechnischen Verfahren auch die *halbkontinuierliche Betriebsweise* (Teilfließbetrieb) zum Einsatz. In solchen Fällen werden in einem Reaktor ein oder mehrere Reaktionspartner diskontinuierlich vorgelegt und mindestens ein weiterer Stoff kontinuierlich dosiert.

Anlagentypen der chemischen Verfahrenstechnik können nach Art der Produktion (Fittler, 2015; Früh, 2015; Lier, 2013) oder nach der physikalischen Struktur (Fittler, 2015) unterschieden werden. Die Produktionsbedingte Klassifizierung unterscheidet Einproduktanlagen, Mehrzweck- und Mehrproduktanlagen (Lier, 2013).

- *Einproduktanlagen* (Monoanlagen) eignen sich zum Herstellen von einem Produkt oder einer festgelegten Gruppe von Produkten. Sie sind dafür ausgelegt, eine feste Prozessabfolge mit den gleichen Edukten auszuführen. Die Flexibilität solcher Anlagen ist gering, die Produktlaufzeit lang, der Automatisierungsgrad hoch und der Einsatzbereich eng begrenzt (Lier, 2013).
- *Mehrproduktanlagen* finden zur Herstellung unterschiedlicher Varianten eines Grundproduktes oder von Produkten mit unterschiedlichsten Spezifikationen ihre Anwendung. Die Durchsätze von Mehrproduktanlagen sind meist gering und deshalb eher für die Produktion von geringen Nachfragemengen vorgesehen. Die Produktlaufzeit solcher Anlagen ist kurz, der Automatisierungsgrad geringer und der Einsatzbereich breiter (Lier, 2013).

Mehrproduktanlagen werden vorrangig für Batch Prozesse eingesetzt (Fittler, 2015). Diese werden in Einstrang-Anlagen, Mehrstrang-Anlagen und Mehrstrang-Mehrwege-Anlagen differenziert (siehe Abbildung 1).

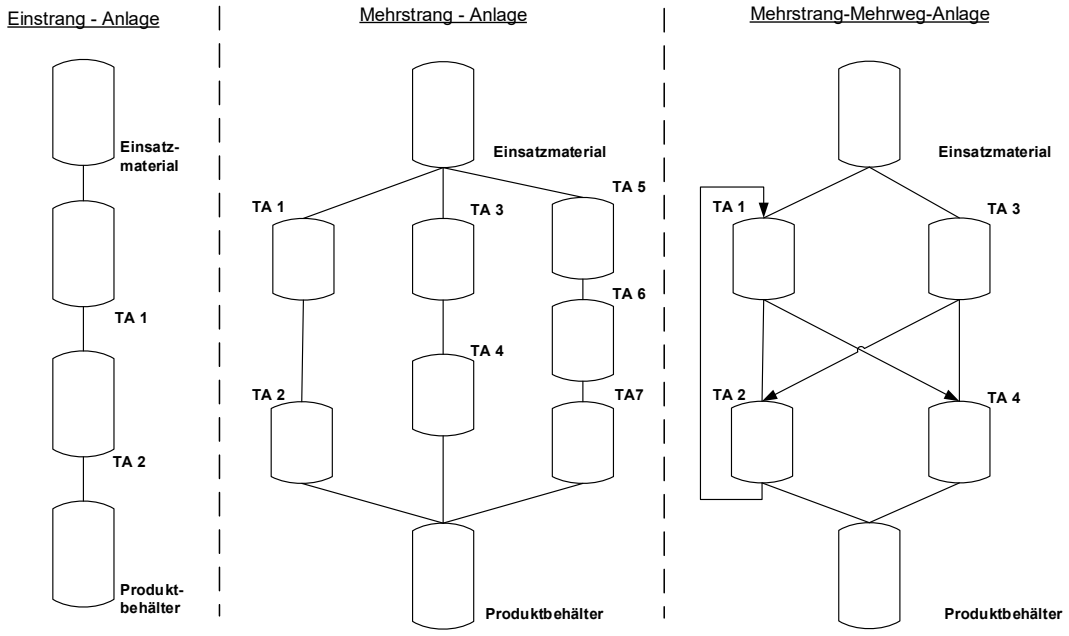


Abbildung 1:Physikalische Anlagenstrukturierung nach IEC 612512-1 (Fittler, 2015)

- Eine *Einstrang-Anlage* ist eine Gruppe von Teilanlagen, die eine Charge sequenziell und zumeist in fester Reihenfolge bearbeitet.
- *Mehrstrang-Anlagen* bestehen aus mehreren parallelen Einzelsträngen, zwischen denen kein Produkttransfer stattfindet.

- Bei einer *Mehrstrang-Mehrwege-Anlage* können die Wege zwischen den Strängen fest oder flexibel geschaltet werden.

Entsprechend (Lier, 2013, S. 62) sind Reaktionsapparate im kontinuierlichen Betrieb leichter automatisierbar und eignen sich aufgrund der hohen Investitionskosten, für die Produktion von Massenprodukten. Für die Produktion von kleinen Produktmengen oder verschiedenen, sich wechselnden Produkten finden daher diskontinuierliche Prozesse bei höhere Flexibilität ihre Anwendung. Nachteil sind hierbei die langen Zeiten für das Befüllen und Entleeren und gegebenenfalls. das Reinigen der Apparate.

Von der Idee bis zur produzierenden verfahrenstechnischen Anlage sind eine Vielzahl von Tätigkeiten notwendig. Diese werden als Engineering zusammengefasst. Nach (Alznauer, 1999) umfasst der Begriff Engineering „*all die ingenieurtechnischen Aufgaben und Tätigkeiten, die während Konzeption, Planung, Montage (Errichtung), Inbetriebnahme, Betrieb und Außerbetriebnahme (Demontage, Abriss) einer technischen Anlage ausgeführt werden*“. Der im Deutschen oft verwendete Begriff Anlagenprojektierung kann nicht als strenge Übersetzung für Engineering im vorgenannten Sinne herangezogen werden, da der Übergang zwischen Projektierung und Betrieb (Konfiguration, Parametrisierung) und im laufenden Betrieb (Instandhaltung, Optimierung) nicht berücksichtigt werden.

Ebenso beschreibt (Fay, 2009) das Engineering als Planung, Realisierung und Inbetriebnahme einer Anlage, sowie spätere Ingenieurstätigkeiten während des Betriebs wie die Überprüfung, Optimierung, Erweiterung und Modernisierung der Anlage. (Fay, 2009) charakterisiert das Engineering als eine phasenunabhängige Tätigkeit, die die systematische Anwendung von Kenntnissen über physikalische Gesetzmäßigkeiten umfasst. Auch (Brendenberger & Scherwietes, 2015) charakterisieren Engineering als alle Vorgänge und Schritte, die erforderlich sind, um von einer ersten Prozessidee zu einer fertigen Prozessanlage zu gelangen.

Weiter wird die Beschreibung des Engineerings im Bereich des Projektes Industrie 4.0 formuliert. Hier werden unter einem durchgängigem Engineering alle Phasen im gesamten Lebenszyklus einer Anlage, also auch Phasen, die die Außerbetriebnahme und Entsorgung der Anlage enthalten, verstanden (DIN / DKE, 2015).

Die allgemeine Planung einer verfahrenstechnischen Anlage ist durch die Phasen: Konzeptplanung, Basisplanung, Ausführungsplanung, Errichtung, Inbetriebnahme und Instandhaltung gekennzeichnet (Ulrich, 2009, S. 5).

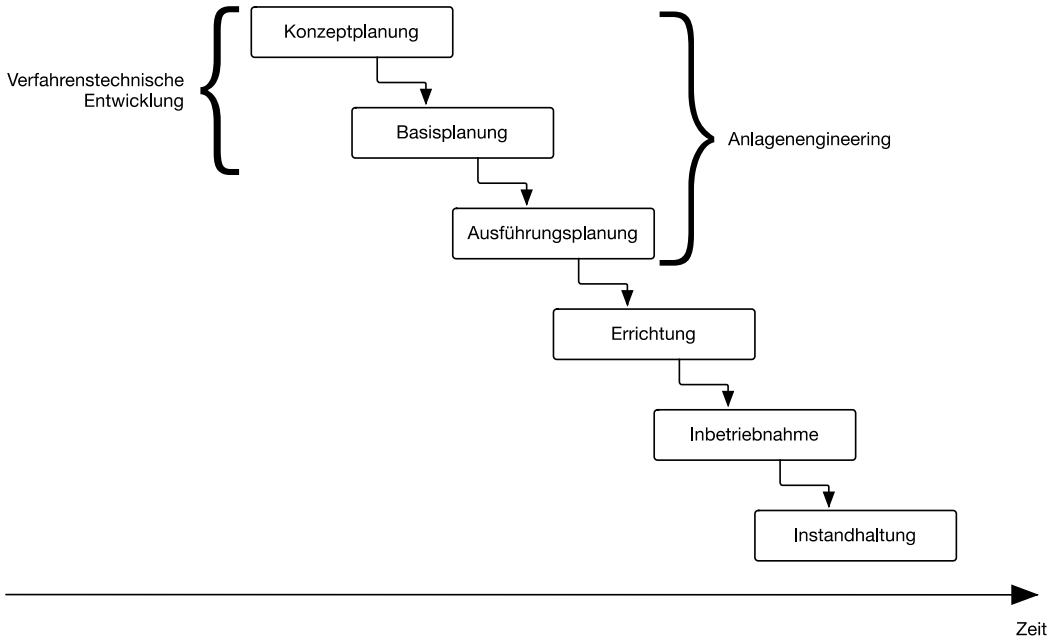


Abbildung 2: Lebenszyklus einer verfahrenstechnischen Anlage (Ulrich, 2009, S. 5)

Für die Abwicklung der PLT-Planung einer verfahrenstechnischen Anlage wurde durch die NAMUR mit dem Arbeitsblatt 35 (NA 35) ein Vorgehensmodell erarbeitet. Die Intention hinter dem Modell kann allgemein wie folgt formuliert werden:

"WAS ist WIE und WANN zu tun, um die Planung eines PLT-Projektes effizient und effektiv durchführen zu können." (NA 35)

Es findet damit eine Erläuterung und Zuordnung der Engineering-Tätigkeiten zu bestimmten Methoden statt, sowie eine Einteilung der zeitlichen Abfolge in Phasen. Dieses Vorgehensmodell soll die Kommunikation der prozessleittechnischen Planung mit dem Auftraggeber und der Verfahrensentwicklung verbessern und bei Beteiligung von mehreren Unternehmen die Koordination unterstützen. Es gliedert sich in die drei Hauptbereiche Projektierung, Qualitäts- und Projektmanagement. Diesen sind jeweils Tätigkeiten zugeordnet, die strukturiert mit untergeordneten Aktivitäten, bestimmten Methoden und Hilfsmitteln bearbeitet werden.

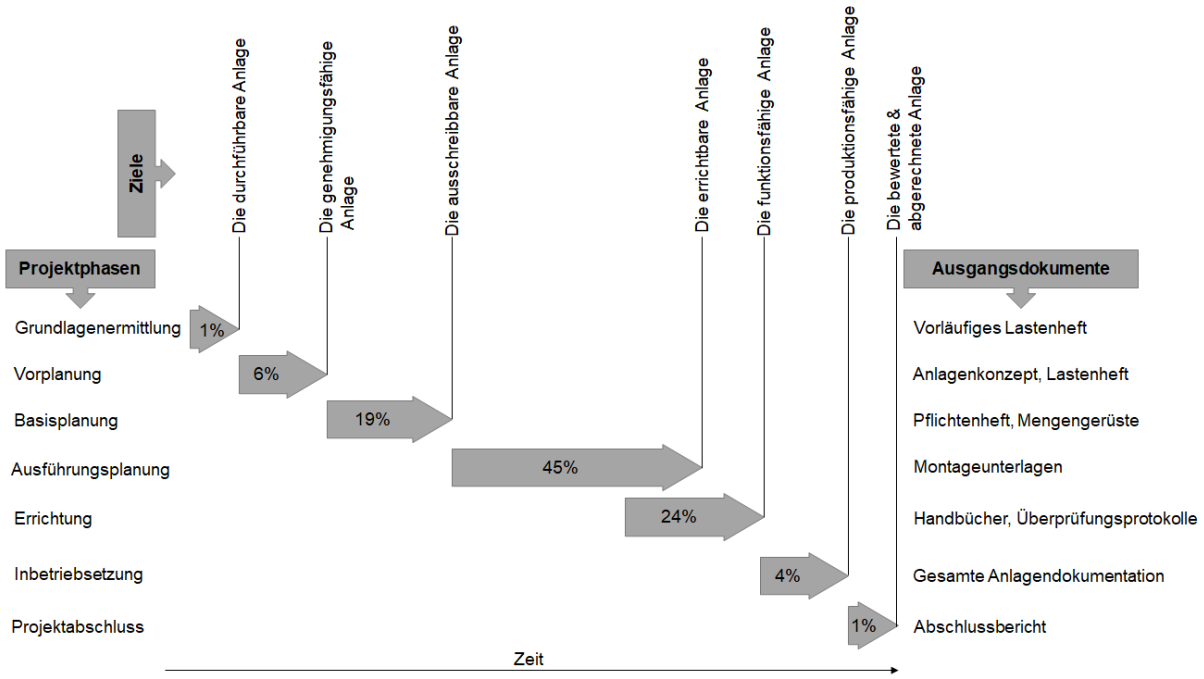


Abbildung 3: Phasenmodell eines PLT-Projektes nach NAMUR (NA 35, S. 11)

Die Planungsphase ist durch umfangreiche Aufwände und Investitionen geprägt. Üblicherweise beläuft sich die Dauer von der Produktentwicklung bis zur Inbetriebnahme einer Anlage auf circa fünf bis zehn Jahre (Bott & Schembecker, 2009), in deren Zeitraum kaum Einnahmen verbucht werden können. Durch Verkürzung dieser Phase könnte der Break-Even Point (BEP), der Zeitpunkt ab dem die erzielten Einnahmen die Kosten überschreiten, deutlich früher eintreten. Auch wenn durch den Einsatz computergestützter Planungswerkzeuge bereits ein enormer Beitrag zur Effizienzsteigerung und damit zur Verkürzung der Planungsphase beigetragen wurde, befindet sich der Cashflow, also der abgezinste Zahlungsstrom für eine typische Chemieanlage auf dem sogenannten „Slow track" (siehe Abbildung 4, orangene Kurve).

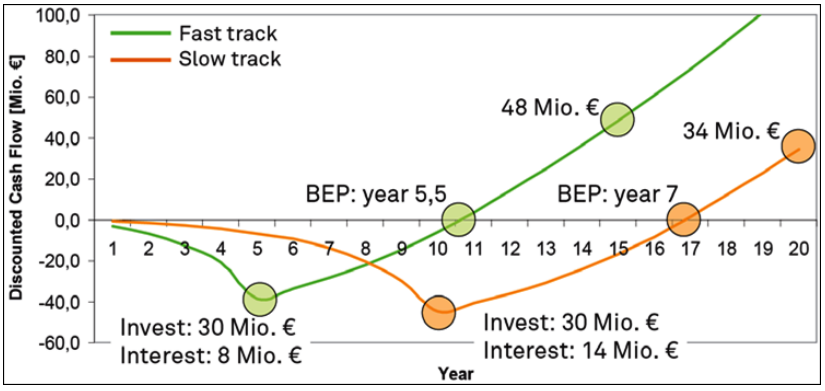


Abbildung 4: Cashflow für eine typische Chemieanlage (Bott & Schembecker, 2009)

Um eine qualitativ äquivalente Planung in einem kürzeren Zeitraum gewährleisten zu können, ist die Entwicklung neuer Methoden und Technologien erforderlich. Im Rahmen des 48. Tutzing- Symposiums wurden daher die folgenden PAAT Tutzing-Thesen (ProcessNet, 2009) aufgestellt:

1. Sei schnell, denn der Markt wartet nicht
2. Denke und plane in Modulen und Standardlösungen
3. Nutze wieder verwendbare Modelle für Prozesse, Informationen und Arbeitsabläufe
4. Kenne deinen Einfluss auf Wirtschaftlichkeit und Risiko des Projektes
5. Vermeide Perfektionismus, denn eine Punktlandung kostet Zeit und Geld
6. Vertraue deinem Kunden / Lieferanten
7. Bringe Kontinuität ins Projekt, von der Entwicklung bis zur Inbetriebnahme

Das Engineering einer automatisierten Produktionsanlage ist ein zeit- und kostenintensiver Prozess. Er ist zwar zeitlich und im Verhältnis zum übrigen Lebenszyklus der Anlage relativ kurz, verursacht allerdings einen erheblichen Teil der Gesamterstellungskosten einer Anlage (Fay, 2009). Die Arbeit von (Holm, 2016) zeigt, dass aufgrund des kürzeren Engineerings modulare Produktionsanlagen in der chemisch-pharmazeutischen Industrie grundsätzlich vorteilhaft sind. Durch die Anwendung geeigneter Mechanismen in der Automatisierungstechnik kann der projektspezifische Aufwand des Gesamtanlagen-Engineerings deutlich verringert werden. Durch die Nutzung von standardisierten Modulen ist außerdem eine Basis für ein umfangreiches Wiederverwendungskonzept gegeben. Dies ist insofern für die Planung und Entwicklung von Modulen wichtig, da dieser Vorgang bei mehrmaliger Veräußerung von baugleichen Entitäten mehr einer Produktentwicklung, als einem (Teil-) Anlagen-Engineering entsprechen wird.

2.2 Modulares Engineering allgemein

Ziel der Modularisierung ist es, eine Reduktion der Systemkomplexität durch Vereinfachung zu erreichen. „Modularisierung stellt eine gestalterische Strategie zur Reduktion von Unklarheit bei der Systemgestaltung dar“ (Göpfert, 1998). Unter Modularisierung wird die geeignete Gliederung eines Produktes verstanden, in dem die Abhängigkeiten zwischen Elementen (Modulen) verringert beziehungsweise die Schnittstellenvarianten reduziert werden. Ergebnis der Modularisierung sind vordefinierte Systembausteine. Eine schnelle und einfache Realisierung und Veränderung eines Systems wird erreicht, indem die Systembausteine zusammengesetzt, ausgetauscht oder angepasst werden. Damit eine Modularisierung Wettbewerbsvorteile bringt, muss sie sowohl technisch als auch organisatorisch vollzogen werden (Göpfert & Steinbrecher, 2000). Es werden sechs Typen der Modularität unterschieden (Pine, 1994; Schuh, 2005).

1. Modularität durch Gemeinsamkeit von Bestandteilen: Das gleiche Teil ist in unterschiedlichen Produkten einzusetzen.
2. Modularität durch Austausch von Bestandteilen: In Ergänzung zu Typ 1 wird ein Basisprodukt um unterschiedliche Anbauteile ergänzt.
3. Modularität durch passenden Zuschnitt: Die Größe oder Länge eines Produktes wird in gewissen Grenzen variiert, ohne damit Funktion und Schnittstellen zu ändern.

4. Misch-Modularität: Bei Produkten aus einem Gemische unterschiedlichen Substanzen wird dieses Verhältnis variiert, beispielsweise der Farbton.
5. Bus-Modularität: Es wird eine Grundstruktur verwendet, an diese verschiedene Bauteile über eine standardisierte Schnittstelle angekoppelt werden.
6. Teil-Modularität: Dieser Type baut auf dem gleichen Prinzip wie Modularität durch Gemeinsamkeit von Bestandteilen (Typ 1) auf, mit dem Unterschied, dass ein Maximale Variabilität gewährleitstet werden soll, in dem bestimmten Formen erst später definiert werden aber durch standardisierte Schnittstellen gekoppelt werden können.

Dabei bildet die Modularität durch Gemeinsamkeiten der Bestandteile die einfachste Form und erlaubt eine große Vielfalt ohne wesentliche Veränderungen, während die Teil-Modularität kundenindividuelle Produkte zulässt, bei denen die Struktur grundlegend verändert wird (Schuh, 2005).

Die Typen der Modularisierung lassen sich auf den modularen Anlagenbau wie folgt übertragen: In verschiedenen Anlagen immer wiederkehrende identische Module lassen sich als Typ 1 Modul klassifizieren. Module zur Anpassung eines Produktes an spezifische Anforderungen, beispielweise eines lokalen Marktes, ordnen sich hingegen in Typ 2 ein. Werden von einem Modul gleichen Typ's unterschiedliche Größen, die sich beispielsweise im Durchsatz unterscheiden, so werden diese dem Typ 3 zugeordnet. Eine mögliche Backbonestruktur zur Versorgung der Module, wie sie in Kapitel 0 vorgestellt wird wird hingegen als Typ 5 eingeordnet. An dieser Einordnung möglicher Modultypen im modularen Anlagenbau speziell in der Prozessindustrie lässt sich auch ein Kritikpunkt der Klassifizierung nach (Pine, 1994) erkennen. So lassen sich etliche Produkte mehreren Modultypen zuornden (Gonsior, 2008). Diese Einteilung finden sich ebenfalls in den Baukastensystemen der Konstruktionslehre wieder (Abbildung 5).

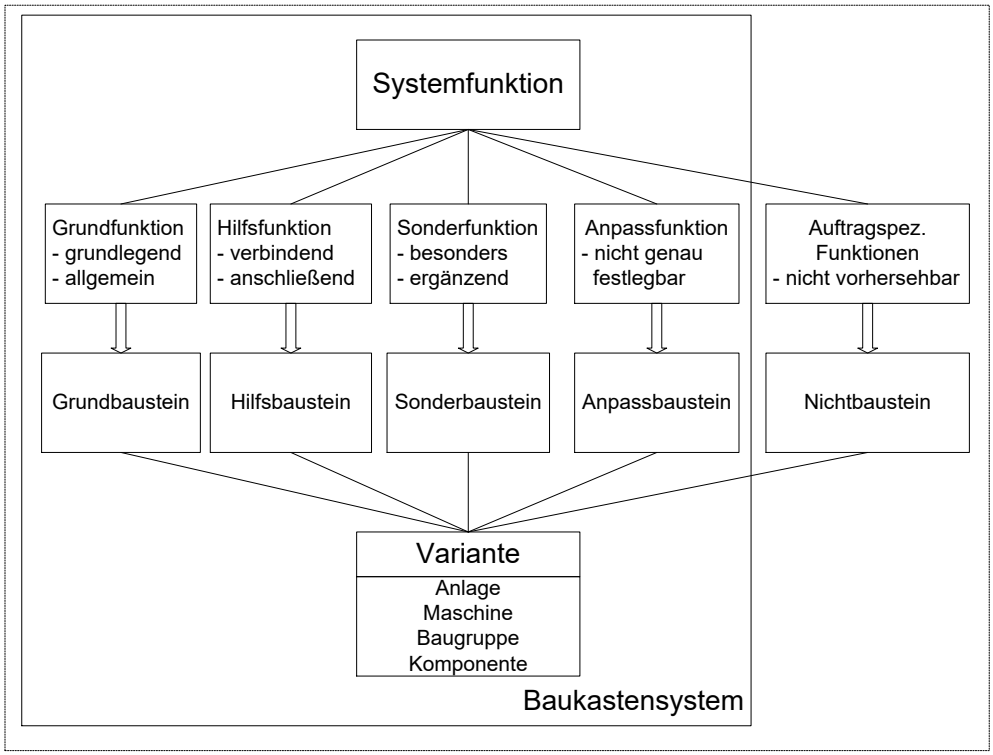


Abbildung 5: Baukastensystem der Konstruktionsmethodik (Pahl & Beitz, 1997)

Die Schwierigkeit bei der Definition von Modulen in der Prozessindustrie liegt in der Begründung der Grenzen zwischen den einzelnen Teilen. Es muss fundiert argumentiert werden, warum ein Planungsobjekt zu einem bestimmten Modul zugehörig ist. Dabei gilt es, die Kriterien und Richtlinien der Modulerstellung zu beachten. Im Bereich des Software Design haben (Baldwin & Clark, 2000) Kriterien zur Modulbewertung aufgestellt. Diese lauten: innere Festigkeit und Abgeschlossenheit, geringe Kopplung, Geheimnisprinzip, Handhabbarkeit und Wiederverwendbarkeit. Der Vorteil dieser modularen Betrachtungsweise besteht in der Abgeschlossenheit einzelner Module und der minimalen Rückkopplung auf andere Module. Für die verfahrenstechnische und automatisierungstechnische Auslegung bedeutet Modularisierung jedoch einen erhöhten Materialbedarf und einen größeren Umfang im Bereich der Instrumentierung. Dies liegt an der sich für viele Fälle ergebenden Überdimensionierung der Module sowie einem erhöhten Instrumentierungsbedarf zur Minimierung von ungewollten Wechselwirkungen zwischen den Modulen, zum Beispiel bedingt durch zusätzliche Verriegelungen. Dies ist unproblematisch, sofern dieser Mehraufwand durch hohe Stückzahlen und Serienfertigung deutlich kompensiert werden kann.

Als Erweiterung der reinen Wiederverwendung beschreiben (Weyrich & Klein, 2012) das Ziel der Modularisierung als die Identifizierung kleinster mechatronischer Einheiten, komplexere Module zu identifizieren. Komplexere Module sollen den Nutzen der Wiederverwendung erhöhen. Kern dieses Ansatzes ist eine Design-Structure-Matrix (DSM), in der die Zusammenhänge zwischen den Komponenten des Systems dargestellt werden. Die Zusammenfassung der Komponenten der Produktionsanlage zu Modulen wird durch eine Sortierung der Zusammenhänge ermöglicht.

2.3 Modulare prozesstechnische Anlagen

Klassische Produktionsverfahren der Prozessindustrie erfüllen die wachsenden Anforderungen der Märkte nach Flexibilität und Schnelligkeit bis zur Produkteinführung nur unzureichend: So sind Konti-Anlagen für eine bestimmte Produktionsmenge pro Zeiteinheit optimiert, die möglichst über Jahre nicht verändert werden sollte. Die höhere Flexibilität konventioneller Batch-Anlagen geht mit unproduktiven Zeiten, zum Beispiel während des Umrüstens und damit einer Ineffizienz einher.

Die gegenwärtige Situation in der Fertigungsindustrie zeigt Analogien: Die Chargenproduktion entspricht der Werkstattfertigung, bei der an einem Arbeitsplatz zahlreiche Produktionsschritte nacheinander durchgeführt und Produktionsmittel nur suboptimal genutzt werden. Die Konti-Anlage entspricht der Produktion am Fließband. Damit konnte ein großer Produktivitätsfortschritt erreicht werden, aber um den Preis, dass die Anlage für einen bestimmten Arbeitspunkt ausgelegt und daher unflexibel hinsichtlich wechselnder Produktionsmengen ist.

Wie auch in der Fertigungsindustrie wird in der Prozessindustrie nach Anlagenkonzepten gesucht, die die Vorteile beider Verfahren kombinieren. Besonders vielversprechend erscheint in dieser Hinsicht die Modularisierung von verfahrenstechnischen Anlagen, bei der Anlagen durch die Kombination von Modulen flexibel aufgebaut werden. Einzelne Module realisieren jeweils standardisierte Produktionsschritte und können entsprechend des herzustellenden Produkts kombiniert werden. Änderungen des Produkts werden durch den Tausch von Modulen realisiert, die Produktionsmenge kann durch das Hinzufügen gleichartiger Module erhöht werden (Obst, Drumm, Doherr, Bauer & Urbas, 2012; Urbas et al., 2012).

2.3.1 Vorteile und Herausforderungen der Modularisierung

Ein wesentlicher Vorteil eines modularen Anlagenkonzepts liegt in der Verkürzung der Zeit für Konzeption, Planung, Aufbau und Inbetriebnahme der Anlage und der flexiblen und späten Anpassbarkeit von Kostenstrukturen an sich verändernde Marktbedingungen (Lier, 2013; Urbas et al., 2012). Im Vergleich gegenüber einer konventionellen Anlage zeigen sich zwar zunächst höhere Investitionskosten, da die Produktion jedoch bei einer modularen Anlage deutlich früher beginnen kann, führt dies zu einem früheren Ertrag, durch den die Investition schneller amortisiert wird und die Gesamtkosten unter Umständen sogar geringer ausfallen lässt. Dieser Zeitgewinn, verbunden mit den weiteren Vorteilen einer modularen Anlage wie beispielsweise der Skalierbarkeit, ist nur zu erreichen, wenn:

- die Module in ihrer verfahrenstechnischen Funktion mindestens die geforderten Vorgaben der Anlage erfüllen,
- die Modulhersteller notwendige Anpassungen an sicherheits-, produkt-, und umweltspezifische Besonderheiten ihrer Kunden durch automatisierte Engineering-Workflows, Baureihenkonzepte und Variantenmanagement effizient bedienen können,

- die Kombination einer Anlage aus Modulen beziehungsweise die Integration eines Moduls in eine bestehende Anlage durch geeignete Beschreibungsmittel, Methoden und Werkzeuge wesentlich schneller erfolgen kann, als die typische Zeit für Planung und Bau einer konventionellen Anlage,
- und schließlich die Modulhersteller auf Grund der größeren Verantwortung für den verfahrenstechnischen Prozess ihre Module als hybride Leistungsbündel, also als eine Kombination von Sach- und Dienstleistung über den gesamten Lebenszyklus, anbieten (Meier & Uhlmann, 2012; Obst et al., 2013).

Neben der räumlichen, mechanischen und elektrotechnischen Integration ist dafür insbesondere eine Integration der Leittechnik erforderlich (Obst et al., 2013), die sowohl die vertikale Kommunikation zwischen Modul und übergeordneten Informationssystemen als auch die horizontale Kommunikation zwischen den Modulen ermöglicht (Fay et al., 2014).

Für den vertikalen Zugriff eines übergeordneten Leitsystems auf die Module definiert die Namur-Empfehlung 148 (NE 148) drei Varianten: Variante 1 bezieht sich auf Module, die keine differenzierten Funktionsschritte benötigen, sondern eine Funktionsauslösung, verbunden mit der zugehörigen Parametrierung. Komplexere Module mit einzelnen Funktionsschritten, die in Variante 2 und 3 berücksichtigt werden, benötigen für eine optimale Einbindung dieser, den Zugriff auf die einzelnen Funktionen innerhalb der Module.

Bei der Modularisierung verfahrenstechnischer Anlagen ist weiterhin das F^3-Factory Projekt (Buchholz, 2012) zu nennen. Vergleicht man dieses Projekt mit den Anforderungen der NE 148 ist festzustellen, dass der Schwerpunkt dieser Arbeiten auf der Modularisierung der der verfahrenstechnischen Geräte und Apparate sowie auf deren Planungsdaten lag. Die Realisierung der Automatisierungstechnik ist im gegenwärtigen Ausbau nicht modular aufgebaut. Zwar lassen sich die entworfenen Module auf verfahrenstechnischer Ebene in sogenannte PEA's und PEC's differenzieren, das eingesetzte Automatisierungssystem ist jedoch nach wie vor zentralistisch aufgebaut. Ein PEA (Process Equipment Assembly) stellt die kleinste modulare Einheit dar, beispielsweise Reaktoren oder Pumpen. Mehrere dieser PEA's werden in einem PEC (Process Equipment Container) entsprechend der geforderten Prozessfunktionalität zusammengeschaltet. Ein oder mehrere PEC werden anschließen entsprechend der gewünschten Skalierung der Produktionsmengen über einen standardisierten Anschluss an einen Backbone angekoppelt.

Weiterhin müssen technologische und organisatorische Voraussetzungen zur Umgestaltung der Engineering-Abläufe geschaffen werden, um den Engineering-Aufwand während der Integration eines Moduls in eine Anlage zu minimieren. Damit wird ein Großteil des Engineerings durch den Lieferanten des Moduls bereits vorweggenommen, um eine geeignete informationstechnische Repräsentanz des Moduls an den Schnittstellen für die Integration (offline beim Engineering, online im Betrieb) zu ermöglichen.

Die Integration in das Engineering bedingt eine digitale Beschreibung des Modules, basierend auf einer Standardmethodik. Diese kann und sollte sich an bestehende Technologien anlehnen, muss jedoch die Besonderheiten verfahrenstechnischer Module berücksichtigen. Die (NE 148) definiert die zu beschreibenden notwendigen Eigenschaften eines Modules wie folgt:

- verfahrenstechnische Funktionen
- automatisierungstechnische Verriegelungen
- Bedienbilder (HMI)
- Diagnosedaten
- Alarmierungsfunktionen
- Prozesswerte
- Asset Management Parameter
- Informationen zur Generierung eines Betriebsdatenerfassung-Systems (BDE)

Soll das Modul in ein System integriert werden, bedarf es einer digitalen Modellierung dieser Eigenschaften. Das Format der Modulbeschreibung sollte, wie auch die Kommunikationsprotokolle, offen verwendbar und keine proprietäre Lösung sein. Nur so lässt sich sicherstellen, dass die Informationen effizient im Integrations-Engineering verwendet werden können.

2.3.2 Begriffsdefinition

Im Kontext modularer Anlagen ist davon auszugehen, dass Module unterschiedlicher Hersteller und funktionaler Ausprägung zu einer Gesamtanlage zusammengefügt werden müssen. Die Gesamtanlage setzt dabei die Existenz einer Integrationsumgebung (Backbone) voraus (siehe Abbildung 6), die sowohl die Anforderungen der Verfahrenstechnik als auch die der Automatisierungstechnik abdeckt. Bei der Betrachtung von Modulen steht der Aspekt der geschlossenen funktionalen Einheit im Fokus. Das Modul wird daher nicht allein über die räumlichen Abmessungen definiert, sondern vor allem über die verfahrenstechnischen Funktionen.

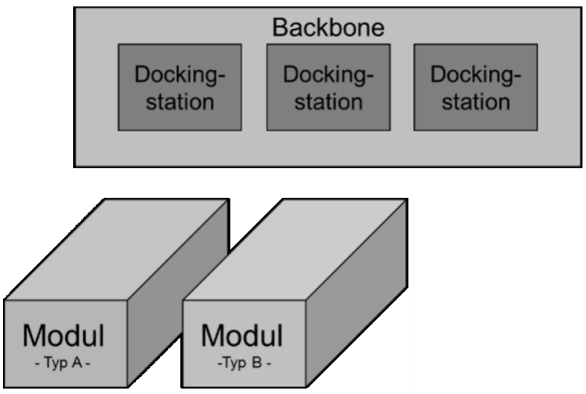


Abbildung 6: Integrationsumgebung für Module nach (Urbas et al., 2012)

In einer den betriebsspezifischen Anlagenanforderungen entsprechenden Infrastruktur können die einzelnen Module eingebracht, notwendige Versorgungsleitungen verschaltet und in die

Automatisierungssysteme integriert werden. Dies soll weitestgehend über standardisierte herstellerunabhängige Schnittstellen erfolgen. Je nach Umfang des Eingriffs, beziehungsweise der Eingriffsmöglichkeit in die einzelnen Module, sind unterschiedliche Ausprägungen denkbar. Dabei lassen sich drei unterschiedliche Modulvarianten ableiten.

- Modulvariante I: Autonome Module
- Modulvariante II: Integrierbare Module
- Modulvariante III: Modulare Module

Stößt ein Modul beziehungsweise eine Anlage an die jeweiligen Produktionsgrenzen kann eine Kapazitätserhöhung bevorzugt durch ein Numbering-up anstelle eines Scale-up geschehen. Dies reduziert den Aufwand einer Erweiterung. Im Folgenden werden die drei Varianten und die dazugehörige Infrastruktur (Backbone) beschrieben.

Infrastruktur (Backbone)

Jegliche Form von Anlage, die aus Modulen zusammengestellt wird, bedarf einer entsprechenden Infrastruktur. Diese Infrastruktur muss die Möglichkeit der Versorgung und Entsorgung von entsprechenden Medien, Energien und Daten bieten. Art und Umfang der angebotenen Medien, Energien und Daten kann von Backbone zu Backbone variieren, sollte aber einem festzulegenden Mindeststandard genügen. Die Integration in die Infrastruktur muss modulunabhängig durchgeführt werden können.

Diese für den Betrieb der Module notwendige Infrastruktur muss bereitgestellt werden. Dies kann sämtliche Aspekte der Planung, Realisierung und des Betriebs umfassen. Dabei ist es denkbar, dass sowohl anlagenspezifische als auch universelle Standardschnittstellen zum Einsatz kommen. Das angebotene Leistungsspektrum der Infrastruktur, beispielsweise Energieverbrauch oder Medienversorgung, muss spezifiziert werden. Über die Spezifikation hinausgehende Anforderungen müssen individuell gelöst werden, insbesondere durch die verursachenden Module und deren Hersteller

Modulvariante I: Autonome Module

Autonome Module sind in sich geschlossene Einheiten (Abbildung 7). Ein derartiges Modul kann nach Anschluss an die Ver- und Entsorgung unabhängig betrieben werden. Besteht die Anlage aus mehreren Modulen, funktionieren diese autark voneinander. Dies kann die Notwendigkeit entsprechender Pufferbehälter zwischen den Modulen und manuelle Produktionskoordination zur Folge haben.

Die für die Funktion und den Ablauf des geschlossenen Prozesses im Modul benötigte Automatisierung ist bereits vollständig im Modul integriert. Eingriffe und Beobachtungen sind lokal oder aus dem zentralen Leitstand möglich. Die autonomen Module verfügen über eine standardisierte Schnittstelle zum Erfassen von Betriebsdaten. Inbetriebnahme, Remote-Access sowie Anschluss an zentrale Betriebsdatenerfassungssysteme (BDE) sind ohne Konfiguration möglich. Eine übergeordnete

automatisierungstechnische Verbindung zwischen den Modulen ist nicht vorgesehen. Diese Variante ist weitestgehend mit einer nicht integrierten Package Unit vergleichbar.

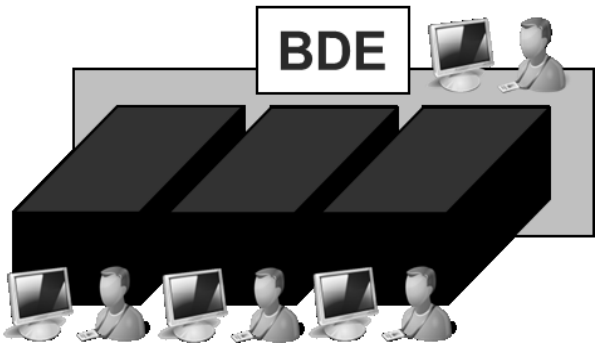


Abbildung 7: Grundstruktur für Variante I (Urbas et al., 2012)

Modulvariante II: Integrierbare Module

Als weiterentwickelte Variante können integrierbare Module angesehen werden (Abbildung 8). Diese sind hinsichtlich Funktion und Einsatzbereich im Wesentlichen fest definiert und unveränderlich. Die Integration in eine Gesamtanlage findet sowohl stofflich, energetisch als auch automatisierungstechnisch statt. Es besteht die Möglichkeit, modulübergreifend zu interagieren. Die Module besitzen zwar eigene Automatisierungskomponenten, diese können aber über ein übergeordnetes Leitsystem (LS) direkt beeinflusst werden.

Durch die Integration in ein übergeordnetes Leitsystem wird die modulübergeordnete und -übergreifende Steuerung ermöglicht. Die modulübergeordnete Perspektive betrachtet den Zusammenschluss einzelner Module als Gesamtanlage. Modulübergreifende Funktionen können somit realisiert werden. Für den gleichzeitigen automatisierten Betrieb mehrerer Module muss es möglich sein, die verschiedenen Automatisierungssysteme der Module anzupassen, ohne auf einer der verschiedenen Engineering-Ebenen eingreifen zu müssen. Die Sicherheit ist unabhängig vom Backbone zu gewährleisten.

Ziel muss es sein, die automatische Erkennung und Integration in ein Leitsystem beim Anschließen und bei der Integration neuer Module in eine bestehende Infrastruktur zu ermöglichen. Entsprechendes gilt auch für die Abkoppelung von Modulen. Eine eindeutige Identifizierbarkeit muss hierbei ebenso gewährleistet sein, wie eine eineindeutige Messstellenbezeichnung der einzelnen automatisierungstechnischen Komponenten

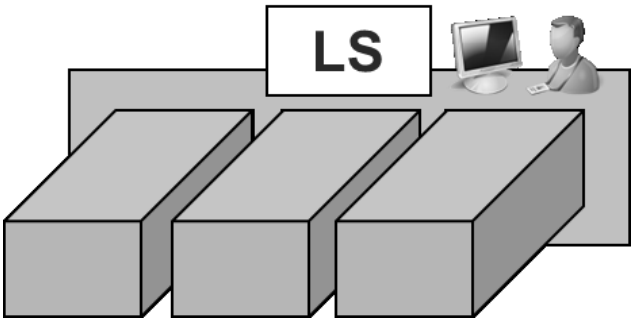


Abbildung 8: Grundstruktur für Variante II (Urbas et al., 2012)

Modulvariante III: Modulare Module

Der modulare Aufbau von Modulen stellt eine dritte Variante dar (Abbildung 9). Innerhalb der ersten Modulebene wird dabei wiederum eine modularisierte Struktur, eine zweite Modulebene, integriert. Dieser Aufbau verschafft dem Modul eine erhöhte Flexibilität, ohne die Gesamtstruktur der Anlage zu verändern. Das Modul kann dabei als eine Komposition aus einzelnen Modulen verstanden werden, wobei auch die gemischte Installation der Varianten II und III möglich und sinnvoll erscheint. Die Integration eines Moduls in das übergeordnete Leitsystem (des übergeordneten Moduls) entspricht grundsätzlich der zweiten Variante. Durch die skizzierte Kaskadierung können aus Modulen einzelner verfahrenstechnischer Funktionen verschiedene Teilanlagen abgebildet werden.

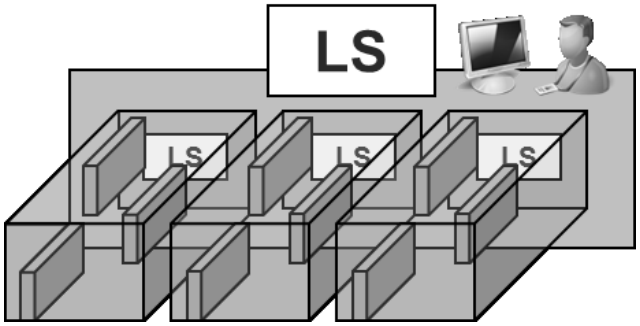


Abbildung 9: Grundstruktur für Variante III (Urbas et al., 2012)

2.3.3 Life-Cycle einer modularen verfahrenstechnischen Anlage

Das NAMUR Arbeitsblatt 35 (NA 35) bildet ein Vorgehensmodell für die Durchführung der leittechnischen Projektierung für verfahrenstechnische Anlagen ab. Darin werden Engineeringtätigkeiten erläutert, ihnen bestimmte Methoden zugeordnet und ihre zeitliche Abfolge in Phasen vorgenommen. Es gliedert sich in die drei Hauptbereiche Projektierung, Qualitäts- und Projektmanagement. Die Projektierung beinhaltet insgesamt sieben Phasen, die jeweils ein bestimmtes Ziel in Hinblick auf die Realisierung einer technischen Anlage verfolgen. Dabei liegt der Fokus auf der Planung der Errichtung einer technischen Anlage. Darüber hinaus definiert beispielsweise das “Process Plant Engineering Activity Model”, kurz PPEAM (PI-Step Consortium, 1994), sowohl die gesamte verfahrenstechnische Planung, als auch die Betriebsphase und die Demontage einer Anlage. Beides zusammengefasst und vereinfacht ergeben sich die in Abbildung 10 dargestellten vier Phasen: Planung,

Errichtung, Betrieb inkl. Umbauten sowie Demontage. Anhand dieser Phasen werden im Folgenden die Einflüsse des modularen Anlagendesigns diskutiert.

Abbildung 10: Phasen im Anlagen Life Cycle (Obst et al., 2013)

Planung

Die Planung einer verfahrenstechnischen Anlage ist geprägt durch die konstruktive Zusammenarbeit des Anlagenplaners, des Entwicklers und/oder späteren Betreibers der Anlage sowie den Lieferanten der Apparate, Mess- und Stellgeräte. Diese Zusammenarbeit im Entstehungsprozess beeinflusst den späteren Betrieb der Anlage.

Die Planung modularer Anlagen erfordert ein Umdenken bei der Auswahl des einzusetzenden Equipments sowie der Beziehung zwischen Planer und „Hersteller" der verfahrenstechnischen Anlage bzw. Module. Die Nutzung vorgefertigter Lösungen aus einem Katalog (siehe Abbildung 11 und Abbildung 12) birgt sowohl aus verfahrenstechnischer als auch automatisierungstechnischer Sicht erhebliche Einsparpotentiale. Dem entgegen steht die verringerte Individualität, d. h. Anlageneffizienz, der Produktionsanlage, d.h. insbesondere eine ungünstigere Optimierung bezüglich der späteren Anlageneffizienz. Innerhalb der Produktionsanlage erhöht sich die Vielfalt der eingesetzten Techniken. Im Folgenden werden die Anforderungen für diese Planungsphase aus Sicht der Automatisierungstechnik vorgestellt.

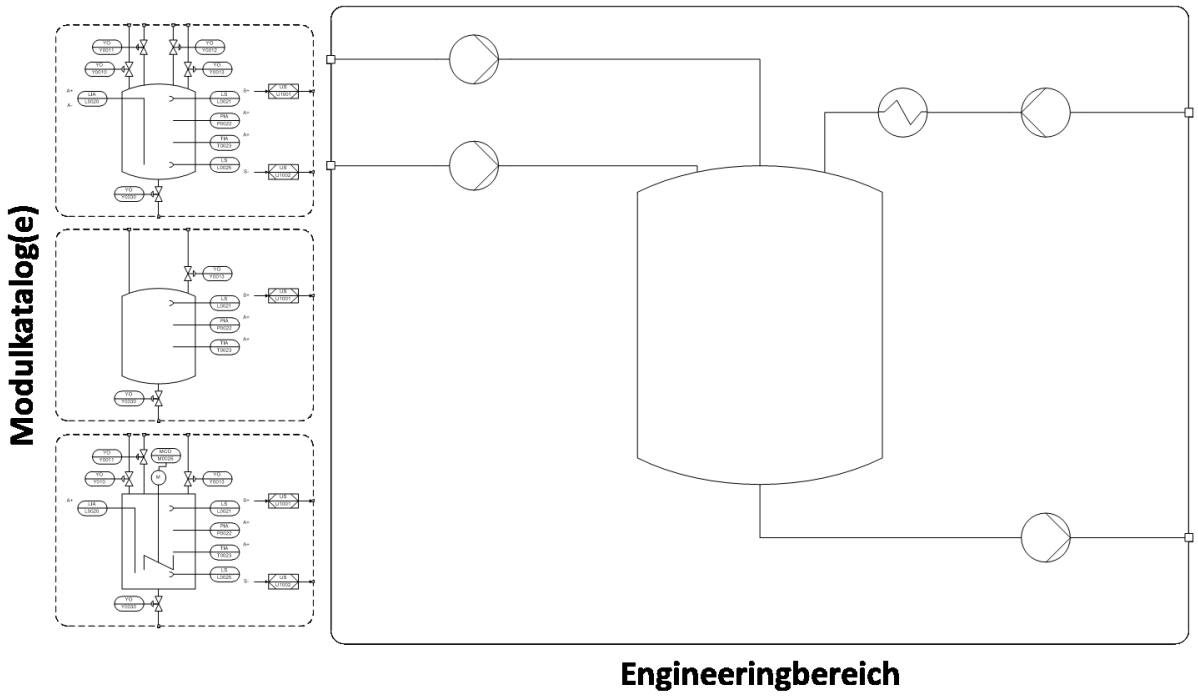


Abbildung 11: Modulauswahl anhand Modulkatalog (Teil 1) (Obst et al., 2013)

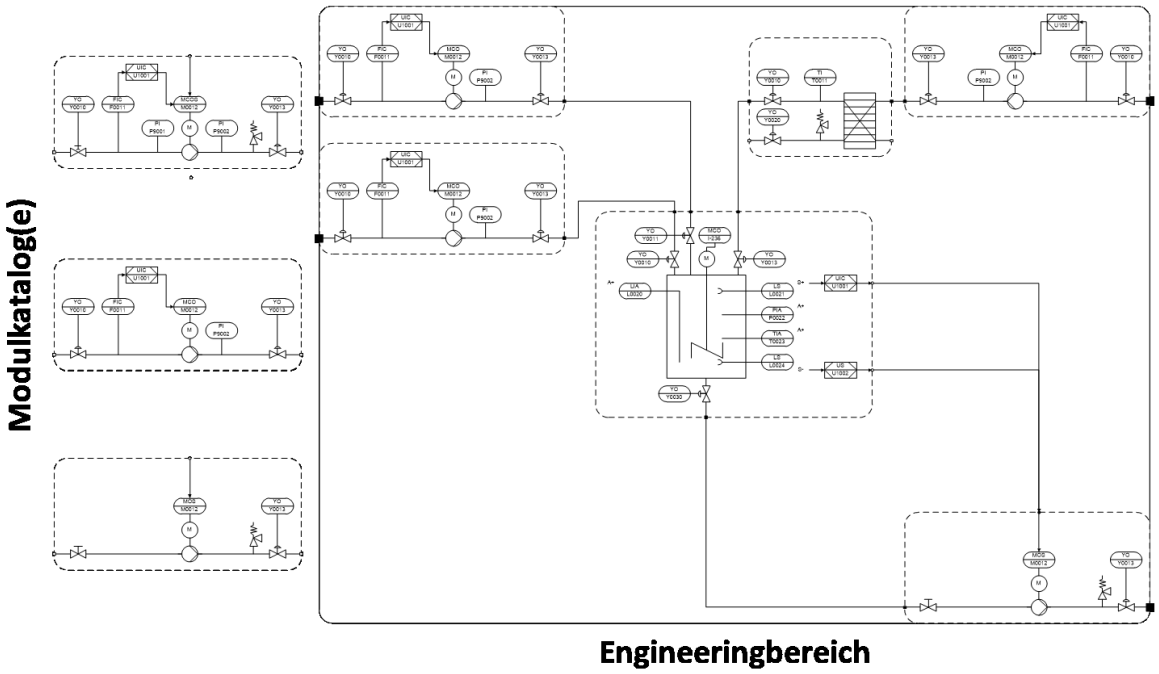


Abbildung 12: Modulauswahl anhand Modulkatalog (Teil 2) (Obst et al., 2013)

Im Planungsprozess einer modularen Anlage kann sich das Risiko eines Missverständnisses bei der Auswahl des Moduls aus dem Katalog des Lieferanten erhöhen (unzureichende Erfüllung der Anforderungen). Wie in (Urbas et al., 2012) beschrieben, erscheint es hilfreich eine gemeinsame „Sprache“ zwischen Lieferanten und Anlagenplaner zu finden, um Fehlinvestitionen zu vermeiden. Auch kann die Auswahl und Auditierung geeigneter Lieferanten hinsichtlich der Erfüllung firmenspezifischer Standards beispielsweise, hinsichtlich des Feldgerätetyps, sinnvoll sein.

Darüber hinaus kann sich eine erhöhte Abhängigkeit von den Lieferanten hinsichtlich der Kompetenz bezogen auf das Modul ergeben. So genügt im zukünftigen Planungsprozess die Kenntnis einiger relevanter Kenngrößen zur Modulauswahl, dies schafft aber keinen tieferen Einblick in den Aufbau des Moduls. Daher ist insbesondere der mögliche Wegfall eines Lieferanten, z. B. bei Insolvenz, bei der Lieferantenauswahl zu berücksichtigen und ggf. vertraglich abzusichern.

Selbst bei einer umfassenden Dokumentation hoher Güte wird der Planer mitunter nicht umhinkommen, den Lieferanten in die Maßnahmenbewertung der Risikoanalysen zur Produkt- und Anlagensicherheit einzubeziehen. Dies muss aus technischer Sicht kein Nachteil sein. In einigen Fällen werden Teile des Produktionsprozesses gegenüber den Lieferanten offengelegt werden müssen. Dies erfordert ein erhöhtes Maß an Vertrauen zwischen Lieferant, Betreiber und Planer.

Innerhalb eines Moduls ist der Modulhersteller für den Aufbau des Automatisierungssystems verantwortlich. Bei der Zusammenstellung verschiedener Module zu einer Anlage ist die Automatisierungshardware innerhalb der Module also bereits vorgegeben. Auch Einzelsteuerungen und diverse Verriegelungslogiken können auf Modulebene bereits implementiert sein. Dies verringert den

Planungs- und Implementierungsaufwand, setzt aber eine herstellerübergreifende Kompatibilität der Module voraus.

Generell kann sich die Rollenverteilung zwischen (Modul-)Lieferant, Planer und Betreiber der Anlage verändern. Konkret können in diesem Zuge, Teile des Detail-Engineerings an den Modullieferanten ausgelagert werden. Der Betreiber oder Planer der Anlage hingegen wird zunehmend die Aufgabe des Modul-Integrators übernehmen und verschiedene Module zu einem Gesamtsystem integrieren, was eine Verringerung des Planungsaufwandes bedeutet. Auswahl und Optimierung des Automatisierungssystems werden hierbei deutlich eingeschränkt, jedoch überwiegen die Vorteile bei dieser Lösung. Kritisch ist allerdings der Aspekt zu sehen, dass auch sicherheitsrelevante Funktionen vom Modulhersteller implementiert werden können und müssen, die sicherheitstechnische Verantwortung jedoch weiterhin in der Hand des Anlagen- bzw. Modulbetreibers verbleibt.

Errichtung

Die Errichtung und Montage eines Moduls liegt in der Verantwortung des Modulherstellers. Er kann die in seinem Katalog gelisteten Eigenschaften und Leistungsmerkmale des Moduls nach eigenem Ermessen realisieren und anbieten. Eine individuelle Konstruktion nach Vorgabe des Anlagenbetreibers kann zu einer verzögerten Modulbereitstellung und höheren Kosten führen. Nach Fertigstellung des Moduls wird die Erfüllung der Spezifikation innerhalb eines Factory Acceptance Test (FAT) überprüft. Soweit möglich, wird eine Leistungsfahrt des Moduls auf einem Teststand durchgeführt. Die Modultests liegen im Verantwortungsbereich des Modulherstellers und sind als Leistungsnachweis zu dokumentieren.

Dem FAT kommt bei der Errichtung einer modularen Anlage eine ganz besondere Bedeutung zu. Durch die zeitliche Straffung der Anlagenerrichtung müssen zahlreiche Aktivitäten in den FAT verlagert werden, um später keine Zeitverzögerungen zu erleiden. Entsprechende Testumgebungen müssen von dem Modulhersteller vorgesehen werden. Dies bietet auch die Möglichkeit, Schwachstellen und Fehler bereits in der Umgebung des Herstellers und nicht erst auf der Baustelle zu erkennen. Durch das Verlagern von Prüfungen in die Testumgebung des Modulherstellers, lassen sich neben Zeitvorteilen auch Kostenvorteile und eine Risikominimierung für das Gesamtprojekt erzielen.

Besonderes Augenmerk ist auf die Schulung der Anlagenbediener und des Instandhaltungspersonals zu legen, da bei der späteren Inbetriebnahme nur ein verkürzter Zeitrahmen zur Verfügung steht. Die Modulhersteller sind hier ebenfalls gefordert, entsprechende Test- und Schulungsmöglichkeiten vorzusehen. Spätestens zu diesem Zeitpunkt muss auch die Umsetzung vereinbarter Service Level abgeschlossen sein, so müssen zum Beispiel Ersatzteile zur Verfügung stehen.

Im Rahmen der Inbetriebnahme ist die Integration des, vorher durch den Anlagenbetreiber gewählten Moduls in die bestehende Infrastruktur eines Betriebsstandorts notwendig. Hierfür muss es am vorgesehenen Standort aufgestellt und angeschlossen werden. Der Anschluss an die übergeordnete

Infrastruktur erfolgt durch das Verbinden der vorher vereinbarten, idealerweise standardisierten Schnittstellen. Der Modullieferant und der Betreiber haben zu überprüfen, dass alle festgelegten (und im FAT getesteten) Schnittstellendefinitionen eingehalten wurden. Beim Anschluss werden auch die benötigten Informationen vom Modul an das übergeordnete Automatisierungssystem übertragen und umgekehrt. Dabei ist auf eine rückwirkungsfreie Einbindung in das Gesamtsystem zu achten. Eine automatisierte Überprüfung der Kompatibilität des Moduls zu der jeweiligen Andockstelle ist eine nützliche Zusatzfunktion.

Im Rahmen der Inbetriebnahme werden auch bei einem modularen Anlagenkonzept verschiedene Tests erforderlich sein. Zum einen gibt es übergreifende Funktionen der Module, zum anderen muss die Funktion des Backbones im Zusammenspiel mit den Modulen getestet werden. Insgesamt steht für die Inbetriebnahme aber ein sehr viel kürzeres Zeitfenster und wenig Eingewöhnungszeit für das Betriebspersonal zur Verfügung. Ebenso liegt das technische Know-how beim Betreiber noch nicht vor und muss erst in der betrieblichen Praxis erlangt werden. Insbesondere bei einer zeitminimierten Inbetriebnahme ist daher eine Unterstützung vor Ort durch den Modulhersteller notwendig.

Zusammenfassend ergibt sich bei der Errichtung modularer Anlagen unter dem Aspekt der Zeitminimierung eine starke Verschiebung von Aktivitäten in den Zeitbereich des FAT hinein. Die Modulhersteller sind gefordert, stärkere Unterstützung bei der Inbetriebnahme und Schulungskonzepte für Bediener und Instandhaltung zu einem frühen Zeitpunkt anzubieten. Eine große Herausforderung besteht für die Hersteller von Automatisierungssystemen, um die Grundlage für das notwendige „Plug and Produce“ (Urbas et al., 2012) der Systemkomponenten zu schaffen und gleichzeitig zu verhindern, dass die Automatisierungstechnik modulare Anlagenkonzepte ausbremst.

Betrieb inklusive Umbauten

In einer klassisch errichteten Produktionsanlage ist die über Jahrzehnte gewonnene Betriebserfahrung in der Regel berücksichtigt worden. Beispiele hierfür sind die durchgängige Bedienung und einheitliche Geräte in den Anlagen. Hier wird es zwangsläufig zu Abstrichen kommen, da unterschiedliche Modulhersteller mit unterschiedlichen Erfahrungen aus den verschiedenen Anwendungsbereichen auch eine unterschiedliche Ausrüstung ihrer Module vornehmen werden.

Weiterhin sind die heutigen Geschäftsmodelle optimiert und ausgereift für klassische Anlagen. Für modulare Anlagen eröffnen sich weitere Möglichkeiten der Zusammenarbeit, wie z.B. durch Leasingmodelle für Module, Ersatzteilvorhaltung und Life Cycle Management durch externe Servicepartner. Dies bedeutet nicht, dass zwingend neue Geschäftsmodelle erforderlich werden, es ergeben sich aber je nach Verteilung des Know-hows und der Ressourcen zwischen Modullieferant und Betreiber neue Möglichkeiten, um bestehende Geschäftsmodelle zu ergänzen.

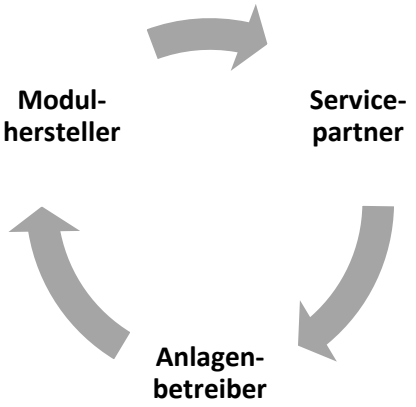


Abbildung 13: Neue Geschäftsmodelle zwischen Modulhersteller und Betreiber nach (Obst et al., 2013)

Als Beispiel sei hier das Life Cycle Management im Rahmen der Instandhaltung angeführt, das System-Updates oder auch den Gerätetausch am Ende der individuellen Nutzungsdauer umfasst.

Neue Möglichkeiten ergeben sich bei größeren Instandsetzungsmaßnahmen durch Modultausch anstelle eines Komponentenwechsels, hierbei ist eine Kompatibilität auf Modulebene ausreichend. Selbstverständlich muss die Nutzung und die Instandhaltung eines Moduls lückenlos dokumentiert werden, um beim Tausch eines Moduls eine „Checkhefthistorie“ mitgeben zu können.

Weiterhin muss der Modulhersteller für Schulungen über die Funktionalitäten der Module Personal zur Verfügung stellen. Dies gilt insbesondere für die Erstinbetriebnahme, aber auch bei Bedarf in der nachfolgenden Zeit. Die Koordination dieser Schulungen für die jeweiligen Module obliegt dem Betreiber der Gesamtanlage.

Wie in (Urbas et al., 2012) bereits ausgeführt, soll für die Gesamtanlage eine einheitliche Bedienung und Beobachtung ermöglicht werden. Durch Nutzung von Modulen unterschiedlicher Hersteller ist diese Einheitlichkeit aber oft nicht gewährleistet. Hier sind die PLT-Stellen mit dem jeweiligen Modulkennzeichnungssystem des Modulherstellers ausgestattet. Um dem Bedien- und Wartungspersonal Eingriffsmöglichkeiten zu geben, muss der Bezug zwischen örtlicher Kennzeichnung innerhalb des Moduls und der Kennzeichnung im übergeordneten Automatisierungssystem bekannt gemacht werden. Gemäß (Urbas et al., 2012) erfolgt diese Zuordnung im übergeordneten Automatisierungssystem und ist auch von diesem zu dokumentieren.

Demontage

Bei der Demontage ist grundsätzlich zwischen der eines Moduls und der des Backbones, als Bezeichner für die eigentliche Anlage, zu unterscheiden. Wird ein Modul, als Teil einer Anlage, außer Betrieb genommen, ist der Abkoppelvorgang unabhängig von der anschließenden Nutzung des Moduls. Somit ist dieses Vorgehen auch bei Abkoppelvorgängen im Rahmen von Umbauten der Anlage zu beachten.

Grundsätzlich ist während und nach der Demontage eines Moduls auf Rückwirkungsfreiheit zu achten. Weder das mechanische, noch das datentechnische Entfernen des Moduls aus der Anlage darf die

Funktionsfähigkeit der verbleibenden Anlage einschränken. Nach Abschluss des Demontagevorgangs eines Moduls muss die Anlage mit den verbliebenen Funktionalitäten wieder funktionstüchtig sein. Die Auswirkungen des Demontagevorgangs müssen sich auf die unmittelbar betroffenen Funktionen beschränken.

Durch die Anwendung von, vom heutigen konventionellen Anlagenbau abweichenden, Geschäftsmodellen ist es denkbar, dass ein Modul durch dessen Hersteller zurückgenommen, erneut eingelagert oder in andere modulare verfahrenstechnische Anlagen integriert wird. Dabei stellt sich die Frage nach dem Verbleib, der im Modul gespeicherten Informationen. Um Know-how-Schutz zu gewährleisten, ist die Prozesshistorie innerhalb des Moduls zu löschen. Ein generelles und vollständiges datentechnisches Rücksetzen des Moduls ist allerdings nicht zielführend, da für eine weitere und über den aktuellen Gebrauch hinausgehende Instandhaltungsplanung, alte Betriebszustände und Laufzeiten benötigt werden. Dies ist vergleichbar mit der Nutzung eines Gebrauchtwagens, hierbei werden weniger die Informationen über den bisherigen Ort, die Anzahl der Insassen und Fahrten des Wagens benötigt werden, vielmehr ist es wichtig, Informationen über den Kilometerstand, vorgenommene Reparaturen und einen lückenloses Serviceheft zu erhalten. Analog zur Nutzung eines Moduls müssen Informationen über die Nutzung eines Moduls, wie eingebrachte Stoffe, Oberflächenbeeinträchtigungen und Lastzustände dokumentiert werden.

Auch in Hinblick auf das übergeordnete Prozessleitsystem muss der datentechnische Abkoppelvorgang rückstandslos vonstattengehen. Nach der Entfernung eines Moduls dürfen keine „Altlasten“ im übergeordneten Backbone verbleiben. Alte Bedienbilder, dem abgekoppelten Modul entsprechende Funktionsbausteine oder Rezeptfunktionen, sollten in einem dafür vorgesehenen Archivbereich abgelegt werden. Der Vorteil wäre eine vereinfachte erneute Anbindung des Moduls an die Anlage. Hierbei lassen sich z. B. notwendige Firmware-Updates des Moduls vermeiden. Dieser Aspekt kann über ein angeschlossenes Manufacturing Execution System (MES) übernommen werden.

Neben der digitalen Reinigung des Moduls, muss dieses auch chemisch gereinigt werden, um einen weiteren Einsatz des Moduls zu gewährleisten. Dem eigentlichen Abkoppelvorgang muss somit ein Reinigungsvorgang vorgeschaltet werden. Die Funktionalität und die Belieferung mit Lösungsmitteln und weiteren Hilfsmitteln, zum Reinigen des Moduls, müssen vom übergeordneten Leitsystem bzw. von den dort angeschlossenen Modulen stammen. Kann oder soll die Reinigung eines Moduls nicht erfolgen, wird der ungereinigte Zustand in dem Modullogbuch hinterlegt.

Die Außerbetriebnahme des Backbones entspricht in Umfang und Qualität dem einer konventionellen verfahrenstechnischen Anlage. Um Produktionschargen rückverfolgbar zu gestalten, ist es unter Umständen notwendig, über die Existenz der Anlage hinaus, Prozesswerte und Zustände zu speichern. Auch hierfür könnte die Nutzung eines MES in Frage kommen.

2.4 Modularisierung im Zeitalter der Digitalisierung

Im Zuge der Digitalisierung der Wertschöpfungsnetzwerke muss sich auch eine modulare Anlagenarchitektur in dieses Spannungsfeld einordnen. Die wichtigsten Begriffe und Entwicklungen sollen im Folgenden zusammengefasst werden.

Cyber-physische Systeme (CPS) sind gekennzeichnet durch eine Verknüpfung von realen (physischen) Objekten und Prozessen mit informationsverarbeitenden (virtuellen) Objekten und Prozessen über offene, teilweise globale und jederzeit miteinander verbundene Informationsnetze (Plattform Industrie 4.0). CPS umfassen eingebettete Systeme, Produktions-, Logistik-, Engineering-, Koordinations- und Managementprozesse sowie Internetdienste, die mittels Sensoren unmittelbar physikalische Daten erfassen und mittels Aktoren auf physikalische Vorgänge einwirken, mittels digitaler Netze untereinander verbunden sind, weltweit verfügbare Daten und Dienste nutzen und über multimodale Mensch-Maschine-Schnittstellen verfügen. Cyber-physische Systeme sind offene soziotechnische Systeme und ermöglichen eine Reihe von neuartigen Funktionen, Diensten und Eigenschaften (acatech, 2013).

Um den Standardisierungsprozess im Zuge von Industrie 4.0 strukturiert stattfinden zu lassen, haben ZVEI, VDI/VDE-GMK, DKE sowie Partner der Verbändeplattform Industrie 4.0 Bitkom und VDMA gemeinsam ein Referenzarchitekturmodell für Industrie 4.0, auch RAMI 4.0 genannt, entwickelt (Hankel, 2015). RAMI 4.0 ist dabei im Prinzip eine 3D-Karte (Abbildung 14) zur Einordnung in den Gesamtprozess Industrie 4.0, das nach (Hankel, 2015) folgende Achsen enthält:

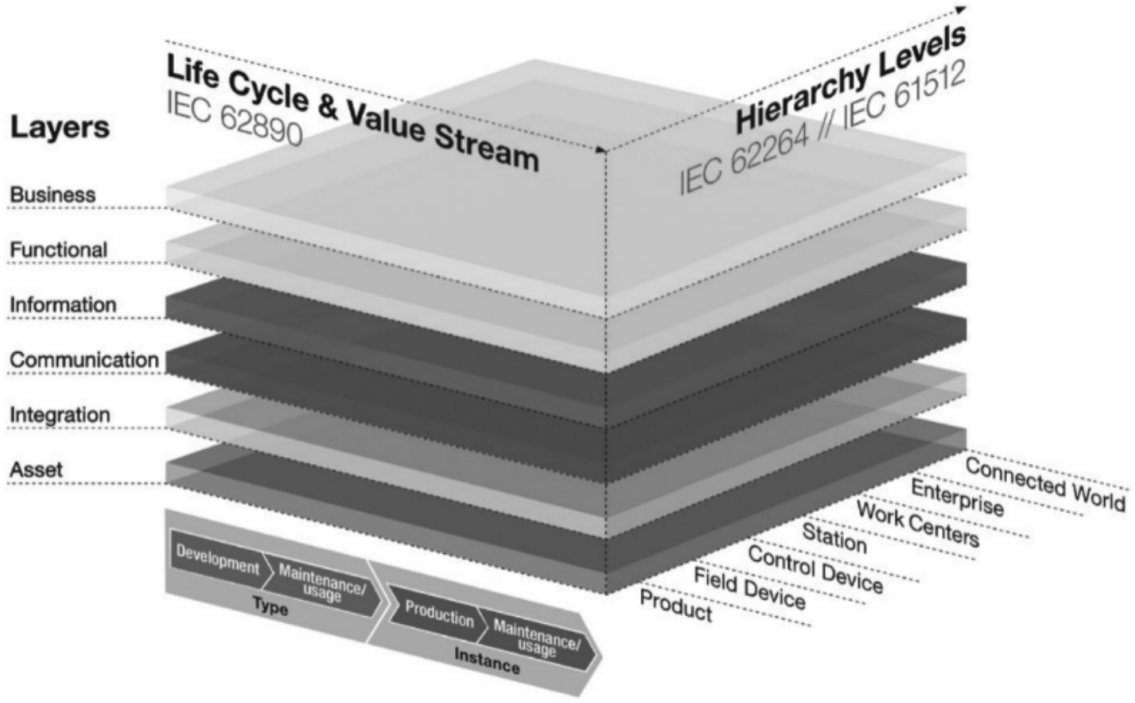


Abbildung 14: Referenzarchitekturmodell RAMI 4.0 (Hankel, 2015)

Die rechte Achse stellt die Hierarchiestufen, angelehnt an die IEC 62264 und ergänzt um „Product“ und „Connected World, dar. Die Achse Life Cycle & Value Stream bildet den Lebenszyklus ab, wobei der Übergang von Type zu Instance erfolgt, sobald Entwicklung und Prototypenfertigung abgeschlossen ist. Die Achse Layers beschreibt die Ebenen der IT-Repräsentanz, also das digitale Abbild bzw. den Abstraktionsgrad der Information.

In dieser Arbeit wird die Beschreibung des Funktionsumfanges zur Integration bereits funktionsfähiger Module erarbeitet, damit ist sie bezüglich der Hierarchiestufen zwischen *Control Device* (Steuerung, z.B. SPS) und *Station* (Prozessführungsebene) einzuordnen. Weiterhin werden die Möglichkeiten der Modularisierung innerhalb von Produktionsanlagen erörtert. Hinsichtlich des Lebenszyklus ist diese Arbeit mit der Produktfertigung im Bereich der *Instance* angesiedelt. Die wesentlichen Ebenen aus Sicht der Informationstechnik stellen *Functional*, *Information* und *Communication* dar. Mit *Communication* ist die Kommunikationsanbindung an das Modul gemeint, das betrifft beispielsweise die Definition von Datentypen und die Kommunikation mit der Modulsteuerung. Zudem sollen die Informationen des Moduls möglichst abstrakt dargestellt werden, ohne zu starke Abhängigkeiten von der darunterliegenden Implementierung aufzuweisen. So werden unter anderem keine direkten Sensorwerte aus dem Modul entnommen, sondern komplex verarbeitete Werte. Die Abstraktion der Information lässt sich bis auf Funktionssicht (Functional) fortsetzen, beispielsweise bei der Definition von Diensten.

Mit dem Modell der *Industrie-4.0-Komponente* wird die Vernetzung zwischen realen Objekten der Produktion mit virtuellen Objekten und Prozessen über definierte Eigenschaften genauer beschrieben. Jede Industrie 4.0 fähige Maschine muss Eigenschaften erfüllen, die über die notwendige Kommunikationsfähigkeit der realen Objekte bereitgestellt werden. Diese Daten und Funktionen müssen über den kompletten Lebenszyklus hinweg alle relevanten Daten in einem elektronischen abgesicherten Container sammeln, mit sich tragen und den am Wertschöpfungsprozess beteiligten Unternehmen zur Verfügung stellen. Dieser elektronische Container wird im Modell als *Verwaltungsschale* (siehe Abbildung 15) bezeichnet.

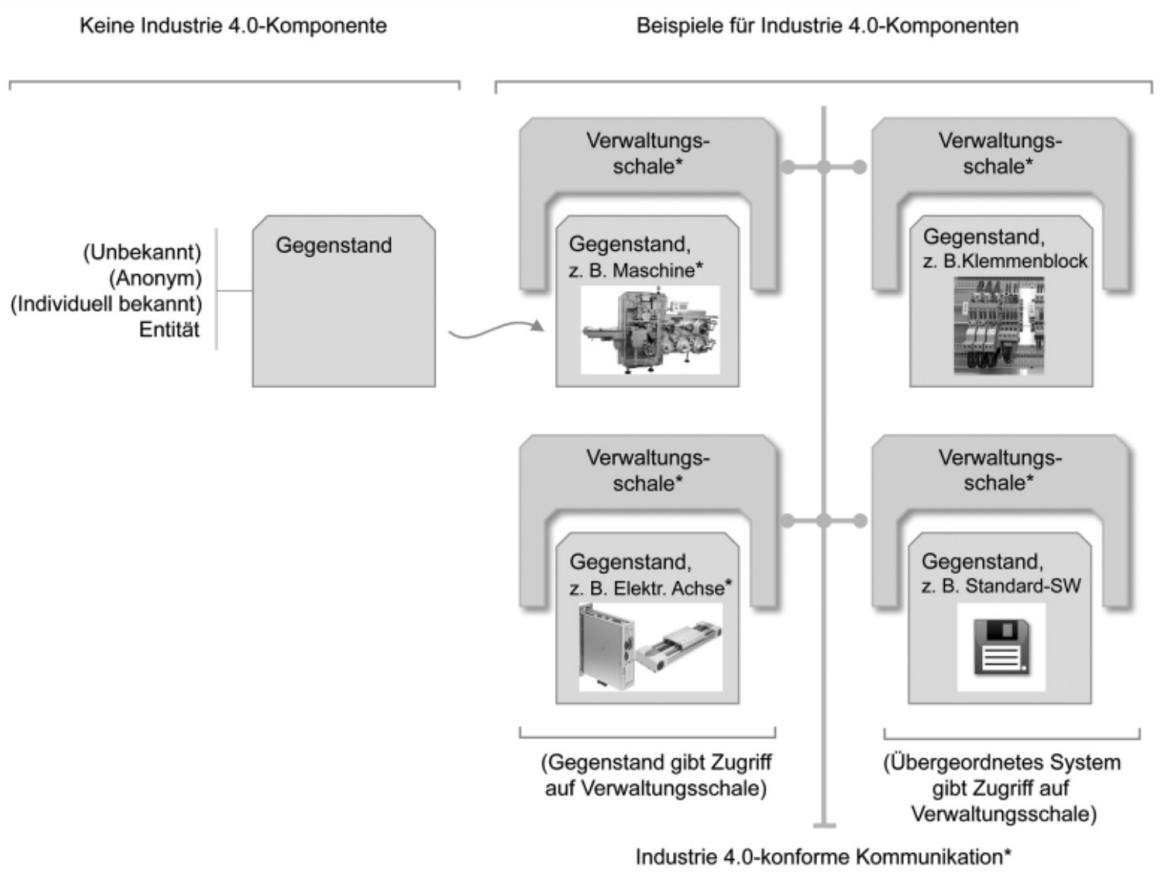


Abbildung 15: Verwaltungsschale einer Industrie 4.0 Komponente (Hoffmeister, 2015)

Werden diese Industrie 4.0 Komponenten oder CPS in Produktionsumfeld eingesetzt, wird von einer Smarten Fabrik oder auch einem Cyber Physical Production System (CPPS) gesprochen. Das Prdoukt stellt in diesem Fall ebenfalls ein CPS dar, wird mit Hilfe einer Verwaltungsschale beschrieben und kennt somit seine notwendigen Bearbeitugnschritte und den gewüschten Zielzustand. Um die Produkte in der geforderten Flexibilität zu fertigen, besteht die Notwendigkeit die Produktionsananlagen wandlungsfähig zu gestallten (acatech, 2013). Zwischen der Smart Factory beziehungsweise einem CPPS und einer modularen Anlage in der Prozessindustrie sind hier Parallelen zu erkennen. Innerhalb einer CPPS müssen die CPS entsprechenden technische Verarbeitunsmöglichkeiten anbieten, um die geforderten Bearbeitugnschritte durchzuführen. Werden diese Verarbeitunsmöglichkeiten auf verfahrenstechnische Grundoperationen überführt werden die CPS zu einem verfahrenstechnsichen Modul und die Gesamtanlagen zu einem CPPS. Eine modularen Anlage aus einzelnen dezentralen für sich automatisierten und vernetzten Modulen stellt einen wichtigen Schritt auf dem Weg zu Industrie 4.0 in der Prozessindustrie dar.

3 Stand der Technik und Forschung

Die Integration eines Moduls in eine übergeordnete Prozessführungsebene, beispielsweise einem Prozessleitsystem erfordert eine klare Beschreibung des Funktionsumfanges des Moduls. Entsprechend des BMW-Prinzips nach (Schnieder, 1999) werden im folgenden Kapitel mögliche Beschreibungsmittel, Methoden und Werkzeuge hinsichtlich ihrer Eignung untersucht. Die Aspekte, die während des Integrations-Engineerings von Modulen in ein übergeordnetes Prozessleitsystem benötigt werden, sind durch die (NE 148) im Abschnitt „Datenaustausch und Informationsschnittstellen“ beschrieben worden (siehe auch Abschnitt 2.3.1). Diese Daten sind für die wunschgemäße Ausführung einer Modulfunktion beim Abruf durch das Prozessleitsystem notwendig. Betrachtet man die genannten Daten und Informationen aus funktionaler Sicht, entsteht eine Klassifizierung, die auch unter der Bezeichnung der „Tauchnitz´schen Torte“ (Tauchnitz, 1996) bekannt geworden ist (siehe Abbildung 16).

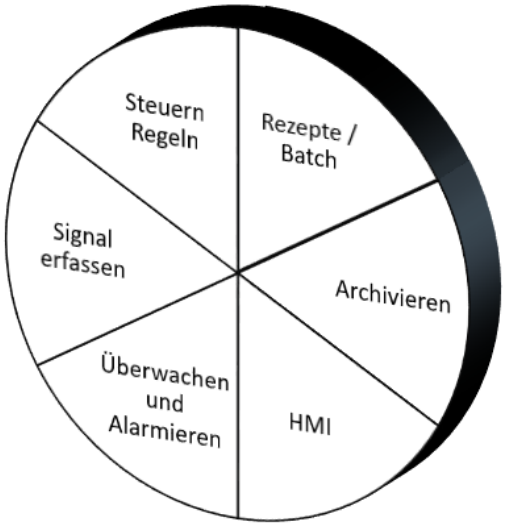


Abbildung 16: „PLS-Torte“ gemäß (Tauchnitz, 1996)

Das Automatisierungssystem eines Moduls kann entsprechend der (NE 148), sowohl die komplette Prozessführung des Moduls, als auch die normierte Bereitstellung des Prozessabbildes an das übergeordnete System übernehmen. Für den Zugriff auf das Modul durch die Prozessführungsebene, müssen diese Information geeignet beschrieben werden. Als Voraussetzung für die Beschäftigung mit der Informationsmodellierung ist es notwendig, die grundlegenden Konzepte und Eigenschaften von Modellen zu betrachten. Im Folgenden werden zunächst wichtige Begriffe präzisiert, sowie Beschreibungsmittel, Werkzeuge und Methoden spezifisch für die bereits aufgestellten Integrationsaspekte vorgestellt. Anschließend folgen Technologien, die weder einem Aspekt, noch mehreren Aspekten eindeutig zugeordnet werden können.

3.1 Begriffsdefinitionen

Für die weitere Analyse geeigneter Informationsmodelle zur Beschreibung der Integrationsaspekte eines Moduls werden zunächst wichtige Begriffe zusammengefasst.

Beschreibungsmittel dienen nach (Schnieder, 1999) der (formalen) Formulierung der Aufgaben und Lösung eines Problems. Dabei ist die Anlage, zum Beispiel einer Automatisierungseinrichtung, sowohl mit ihren physikalischen, als auch ihren informationellen Verhalten zu berücksichtigen. *Methoden* helfen bei der planmäßigen Lösung eines Problems durch entsprechende Hilfsmittel und Verfahren (Schnieder, 1999). Unterstützend kommen Werkzeuge zu Einsatz, die sich auf bestimmte Beschreibungsmittel oder Methoden konzentrieren. Für Entwurfsaufgaben haben sich beispielsweise Werkzeuge unter der Bezeichnung Computer Aided/Assited (System) Engineering (CAE) etabliert (Schnieder, 1999). Die detaillierte Betrachtung von Beschreibungsmitteln der Automatisierungstechnik macht die Untersuchung verschiedenster Modelle notwendig.

Das Internationale Elektrotechnische Wörterbuch (IEC 60050) definiert, dass ein *Modell* eine mathematische oder physikalische Darstellung eines Systems oder Prozesses ist, die das System oder den Prozess aufgrund bekannter Gesetzmäßigkeiten, einer Identifikation oder getroffener Annahmen genügend genau abbildet. Nach (Stachowiak, 1973, S. 131–137) ist es durch mindestens drei Merkmale gekennzeichnet:

- Abbildungsmerkmal: Modelle sind stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.
- Verkürzungsmerkmal: Modelle erfassen im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellschaffern und/oder Modellbenutzer relevant scheinen.
- Pragmatisches Merkmal: Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion:
    - für bestimmte – erkennende und/oder handelnde modellbenutzende - Subjekte (Für Wen?),
    - innerhalb bestimmter Zeitintervalle (Wann?),
    - unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen (Wozu?).

Nach (Kastens & Kleine Büning, 2005) ist der Begriff des *Modells* vom lateinischen Wort *modulus* für Maß, Maßstab abgeleitet. Er wird in vielen verschiedenen Zusammenhängen mit recht unterschiedlichen Bedeutungen verwendet. So kann er das Abbild eines vorhandenen Originals bezeichnen, beispielweise ein Schiffsmodell oder das Vorbild für ein herzustellendes Original, wie ein Gebäudemodell oder ein Vorbild für Maler oder Bildhauer. Das Modell kann konkret sein, wie das Schiffsmodell oder abstrakt, wie ein Modell zur Rentenberechnung. Auch das Modellierte kann konkret sein, wie das Schiff, oder abstrakt, wie die zahlenmäßige Entwicklung der Bevölkerung. Weiterhin führen (Kastens & Kleine

Büning, 2005) aus, dass die Modelle in der Informatik im Allgemeinen abstrakte Abbilder oder Vorbilder zu konkreten oder abstrakten Originalen darstellen. Diese Modelle sind absichtlich nicht originalgetreu; sie heben bestimmte Eigenschaften hervor und lassen andere weg. Der intendierte Verwendungszweck des Modells bestimmt, welche Eigenschaften modelliert werden und welches Kalkül zu deren Beschreibung besonders geeignet ist. Mit dem fertiggestellten Modell werden meist weitere Arbeiten durchgeführt, die der Zweckbestimmung entsprechen, beispielsweise:

- Operationen, die man am Original nicht durchführen kann;
- bestimmte Aspekte eines komplexen Gebildes untersuchen und verstehen;
- als Kommunikationsmittel zwischen Auftraggeber und Hersteller.

Bei solchen Auseinandersetzungen wird häufig das Modell so lange angepasst, bis eine Einigkeit über die beschriebenen Eigenschaften hergestellt ist. Dann fixiert das Modell dieses Ergebnis, z. B. als Anforderungen und Spezifikation in der Software-Erstellung.

Der letzte Aspekt ist von besonderer Bedeutung: Modelle können geprüft und angepasst werden, sie weisen eine korrekte und vollständige Erfassung der für den Zweck relevanten Eigenschaften nach. Dieser Vorgang wird auch als Validierung des Modells oder engl. model checking bezeichnet. So ist es möglich im Rahmen eines Finanzplanes zu überprüfen, ob alle Kosten erfasst, korrekt aufsummiert wurden und die Kostengrenze nicht überschritten wird.

(Lee, 1999) definiert, dass ein *Informationsmodell* durch, für den Bedeutungsinhalt von Daten und in einem ausgewählten Fachkontext beschriebene, notwendige Konzepte, Zusammenhänge, Einschränkungen, Regeln und Operationen repräsentiert werden. Diese Informationsmodelle werden entweder in Textform oder mittels einer formal definierten Syntax einer Informationsmodellierungssprache erstell. Ergänzt werden diese Eigenschaften durch den Zweck eines Informationsmodells.

3.2 Prozesswerte und Geräteintegration

Für die Integration einzelner Prozesswerte in eine Leitsystemstruktur können verschiedene Ansätze hinsichtlich ihrer Eignung für die Modulintegration untersucht werden. Für die Integration in die Kommunikationskonfiguration hat sich beispielsweise die Generic Station Description (GSD) etabliert (Diedrich, 2015). Mit Hilfe der GSD (Generic Station Description) werden Eigenschaften eines PROFIBUS/PROFINET Feldgerätes textuell beschrieben, wobei die hierarchische Struktur eines Feldgeräts in der GSD nur eindimensional – in Listenform – abgebildet ist (Diedrich, 2015). Mit der Einführung der GSDML (General Station Description Markup Language) wird die Eigenschaft XML basierter Dokumente, beliebige Hierarchiestufen bilden zu können, dazu genutzt, das hierarchische Gerätemodell möglichst unverändert abzubilden. Bei der Erstellung der XML Schemata wurde versucht, möglichst viel von der Semantik der GSD zu übernehmen, aber gleichzeitig dort wo es sinnvoll und notwendig erschien, die Abbildung neu zu gestalten. Für modular aufgebaute Feldgeräte bietet die

GSDML weiterhin die Möglichkeit, Modul und Submodulbeziehung herzustellen. Übertragen auf ein Modul bedeutet dies, dass ein Modul als ein Feldgerät implementiert werden müsste.

Für die Geräteintegration und Parametrierung geben (Mahnke et al., 2011) einen Überblick über verschiedene Standards zur Geräteintegration. Die EDD (Electronic Device Description) beschreibt ein Gerät und enthält dabei auch Konstrukte, wie eine Benutzerschnittstelle beispielsweise über verschiedene Menüs organisiert ist. Bei FDT (Field Device Tool) werden neben ausprogrammierten Benutzerschnittstellen auch Strukturen definiert, mit denen die Information über die Parameter eines Gerätes erfasst werden.

Für die Parametrierung der Feldgerätefunktionen hat sich die in der IEC 61804-3 normierte Electronic Device Description Language (EDDL) etabliert (Diedrich, 2015). Diese erlaubt eine formale Beschreibung des nach außen sichtbaren Informationshaushalts eines Feldgeräts (beispielsweise Parameter), die Definition des Kommunikationsverhalten, sowie die Definition von Navigationsstrukturen und von Bediendialogen. Ergänzt wird dies durch die Definition von Methoden zur Beeinflussung des Interaktionsverhaltens (Wizards, Eingabevalidierung) und zum Ablauf komplexer Vorgänge wie beispielsweise eine Kalibrierung. Die EDD wird von einem Werkzeug wie zum Beispiel einem Simatic PDM oder isEDDview eingelesen, interpretiert und dargestellt, ähnlich wie eine HTML-Datei von einem Browser.

Das EDDL Cooperation Team und die FDT Group entwickeln seit 2007 eine einheitliche Lösung zur Field Device Integration (FDI). Diese soll eine vom Anwender genutzte und vom Systemumfeld unabhängige Geräteintegration ermöglichen. Dabei sollen die Vorteile der heute eingesetzten Technologien EDDL und FDT/DTM vereint werden. FDI befindet sich derzeit in der Spezifikation. Das FDI-Basiskonzept definiert die Komponenten FDI Package, FDI Server und FDI Client (siehe Abbildung 17).

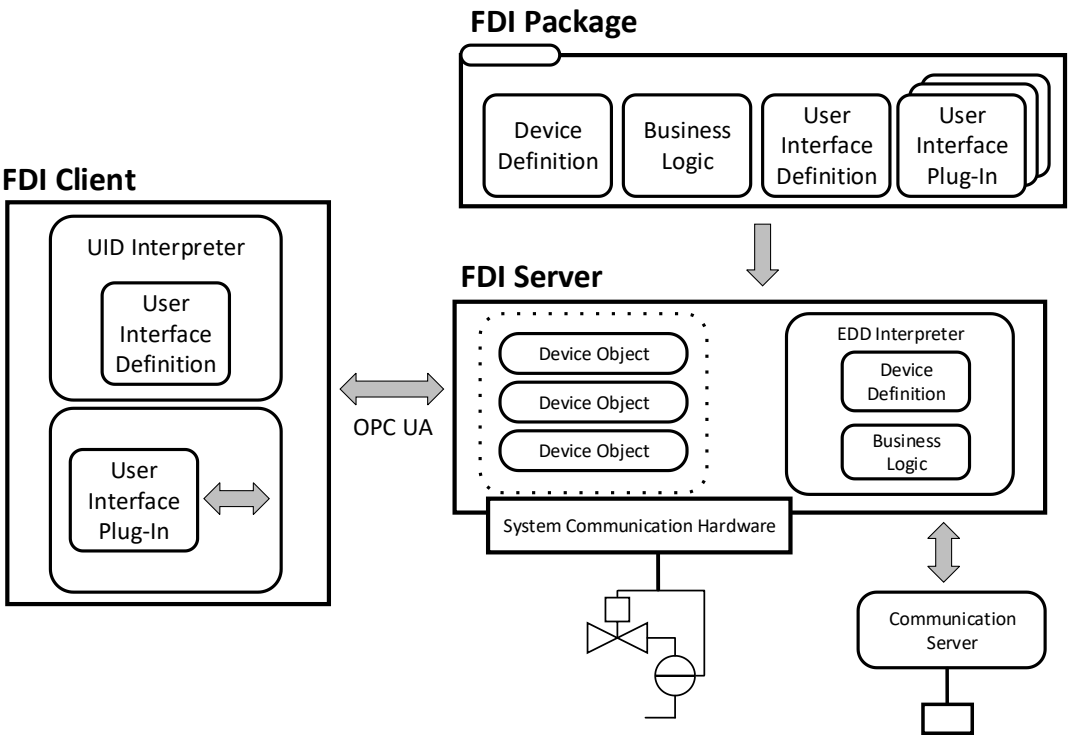


Abbildung 17: FDI-Basiskonzept (Kumpfmüller & Lange, 2010)

FDI Packages werden vom Gerätehersteller geliefert und enthalten alle Informationen, die für eine Geräteintegration notwendig sind (siehe Abbildung 18). Die Device Definition umfasst Verwaltungsinformationen sowie das Gerätemodell. Die Konsistenzsicherung dieses Gerätemodells sowie die Kommunikationslogik zum Gerät erfolgt über die Business Logic. Die Beschreibung der Präsentation der Geräteparameter und Gerätefunktionen ist Aufgabe der User Interface Description. Diese Bestandteile des FDI Packages werden mit EDDL nach IEC 61804-3 beschrieben. Zusätzlich können Oberflächenanteile auch als programmierte Komponenten (User Interface Plug-ins) integriert werden, deren Basiskonzept dem FDT/DTM-Konzept nach IEC 62453 entspricht.

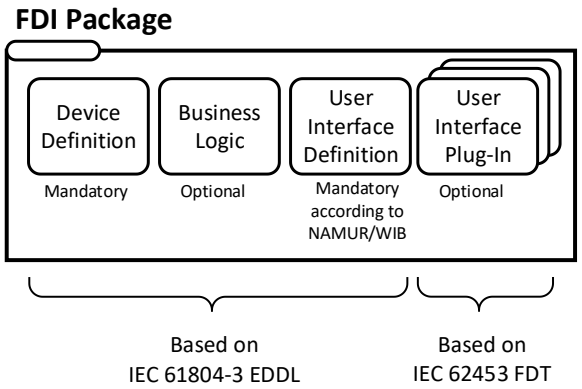


Abbildung 18: FDI Package (Kumpfmüller & Lange, 2010)

In (Theurich, Stoss, Wollschlaeger & Urbas, 2012) wird an zwei Beispielen überprüft, ob und in welchem Umfang das FDI Konzept zu einer Beschreibung von Sinneinheiten geeignet ist, die aus

mehreren intelligenten Feldgeräten zusammengesetzt sind. Die Ergebnisse dieser Untersuchungen machen deutlich, dass die Beschreibung einzelner Aspekte einer aus vielen Feldgeräten komponierten Einheit mit FDI möglich ist. Verschiedene Restriktionen von FDI erfordern jedoch komplexe Kommunikationswege, beispielsweise über den optionalen OPC-UA-Dienst eines FDI Servers.

3.3 Automatisierungstechnische Verriegelungen & Abläufe

Für die Implementierung von Verriegelungen und Abläufen werden in der Prozessautomation Funktionsbausteine bzw. CFC's sowie Ablaufketten oder SFC's eingesetzt. Für den herstellerunabhängigen Austausch von Funktionsbausteinen wurde von der PLCopen Organisation das Austauschformat PLCopen XML entworfen. Bei der Entwicklung dieses Datenformates wurde nicht nur der Datenaustausch zwischen verschiedenen Programmierumgebungen berücksichtig, vielmehr ist das Datenformat PLCopen XML ebenso als Datenaustauschmedium für Netzwerkkonfigurations-, Debugging-, Simulations- oder auch Dokumentationswerkzeuge entwickelt worden (PLCopen, 2009). PLCopen XML basiert auf der Extended Markup Language (XML), für den Datenaustausch ergibt sich daraus der Vorteil, dass ein Dokument mittels einer Grammatik (z.B. einer Dokumenttypdefinition oder eines XML-Schemas) validiert werden kann. Mit Hilfe des PLCopen XML Formates können die in der IEC 61131-3 definierten Beschreibungssprachen Sequential Function Chart (SFC), Strukturierter Text Text (ST), Anweisungsliste (AWL), Funktionsbausteinsprache (FBS) und Kontaktplan (KOP) abgebildet werden. Diese können sowohl auf der Ebene eines IEC 61131-3 Projektes als auch auf Ebene der Programmorganisationseinheiten (POE) ausgetauscht werden. Somit bietet dieses Format ein Konstrukt, die Abbildung des gesamten Programms eines Moduls, als auch Teile von diesem, die in Funktionsbausteinen oder Funktionen beschrieben sind, zu realisieren.

Mit Hilfe von Schrittketten bzw. SFCs werden Abläufe von Anlagenkomponenten beschrieben. Sie stellen eine Kette von Steuerungsschritten dar, welche durch Weiterschaltbedingungen (Transitionen) miteinander verbunden sind. Mit dem Format PLCopen XML lassen sich ebenfalls Abläufe in Form von SFCs abbilden und austauschen. Zu den fünf in der IEC 61131 genormten Fachsprachen fehlt jedoch eine Sprache, mit der sequentielle und nebenläufige Algorithmen grafisch entworfen und beschrieben werden können. Die Signal-interpretierten Petri-Netze bieten hier eine Ergänzung (Frey, Minas & John, 2002).

Für den Austausch solcher Verhaltensbeschreibungen, basierend auf Petri-Netzen, wurde die Petri Net Markup Language (PNML) entwickelt. PNML ist ein XML-basiertes Austauschformat, das Petri-Netze als einen mit Petri Net Type Definitions markierten Graphen abstrahiert. Die Petri Net Type Definitionen spezifizieren die Syntax einer Kantenmarkierung, wie es zum Beispiel mit XML Schemata für XML Dateien möglich ist. Weiterhin wurde die PNML um einen modularen Ansatz erweitert (Urbas et al., 2012), mit welchem Teile eines Petri-Netzes gekapselt und wiederverwendet werden können. PNML wurde im Februar 2011 in der ISO/IEC 15909 standardisiert.

3.4 Prozessführung

In (Schaudel, 2009) wird der Begriff *Prozessführung* als ein Ansatz für die Integration der Prozessführungsstrategien der Automatisierungstechnik mit den verfahrenstechnischen und chemischen Prozessen in einer Anlage beschrieben. Die Prozessführung dient zur optimalen Ressourcennutzung, zum An- und Abfahren einer Anlage, zum sicheren Beherrschen dieser und zur Herstellung von Produkten mit gewünschter Qualität zu niedrigen Kosten. (Birk, 1999) definiert die „Prozessführung als die Gestaltung und Beherrschung des Verhaltens eines Prozesses, durch zielgerichtete technische Maßnahmen (z. B. Verfahrenstechnik, Automatisierungstechnik und anderen technischen Disziplinen) sowie durch die Tätigkeit der Anlagenfahrer“. Sowohl für Batch- als auch für kontinuierliche Prozesse existieren entsprechende Prozessführungsmodelle, diese werden im Folgenden näher erläutert.

3.4.1 Batch Prozesse

Batch Prozesse sind durch die Verarbeitung einer abgeschlossenen Stoffmenge, einem Batch, charakterisiert. Dabei wird eine definierte Menge an Edukten durch eine geordnete Abfolge von Verarbeitungsschritten zu einem oder mehreren Produkt(en) verarbeitet. Die Verarbeitungsschritte sind spezifischem Equipment zugeordnet (Brandl, 2007). Viele Prozesse aus den Bereichen Pharmazie, Spezial- und Feinchemie, sowie der Lebensmittelproduktion werden in Batch Prozessen geführt.

Für ein einheitliche Struktur der Batch Prozessen definiert die (DIN EN 61512-1) Terminologie und Modelle für chemische Anlagen und beschreibt ein Prozessführungsmodell. Das Modell setzt sich aus der Kombination des physischen Modells, des prozeduralen Modells und des Prozessmodells zusammen (Abbildung 7).

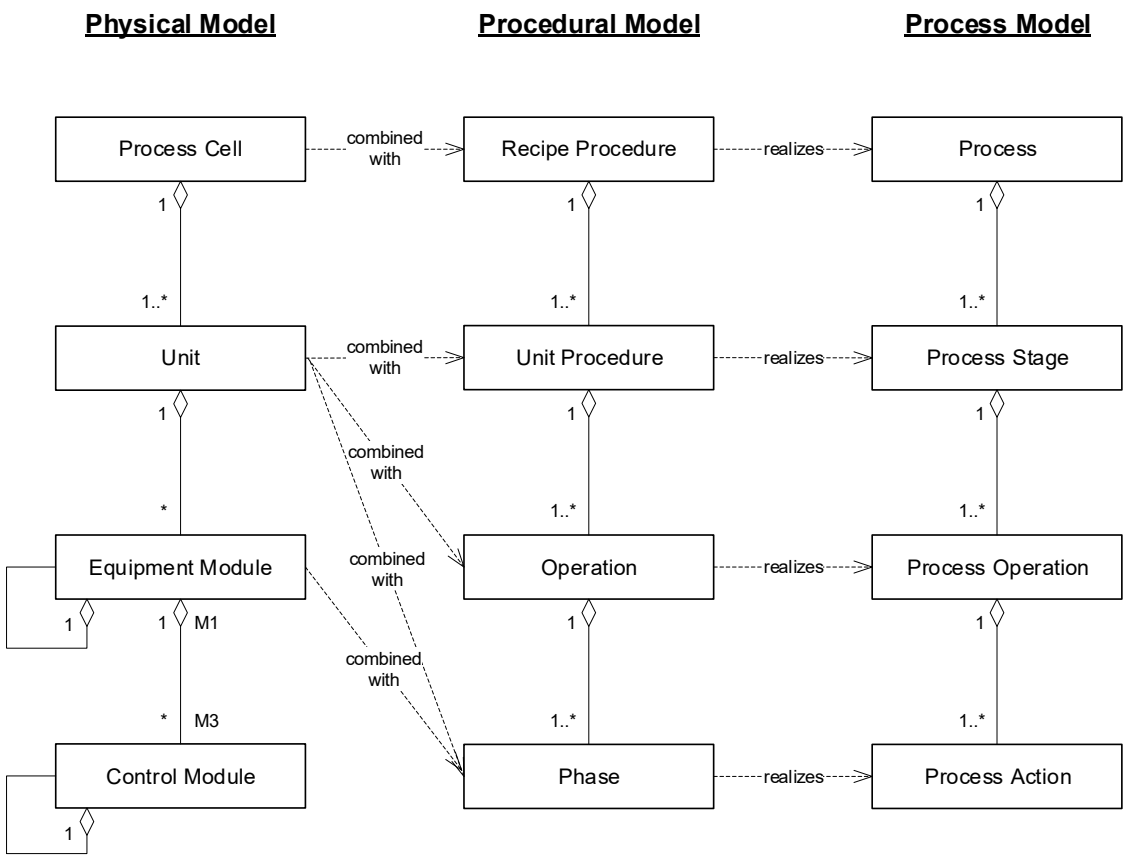


Abbildung 19: Hierarchisches Modell eines Batch Prozesses mit Elementbeziehungen in UML-Notation, nach (Mahnke et al., 2011)

Die Beschreibung von physikalisch-chemisch zielorientierten Änderungen von Stoffen, als eine Hierarchie aus Prozessabschnitten, Prozessoperationen und Prozessschritten erfolgt im Prozessmodell (Process Model). Ein Prozess wird aus einem oder mehreren Prozessabschnitten (Process Stage) gebildet. Diese können seriell oder parallel ablaufen und sind unabhängig voneinander. Jeder Abschnitt besteht wiederum aus Prozessoperationen (Process Operation), die eine Kombination von gekoppelten Aktivitäten darstellen, um ein Zwischenziel zu erreichen. Der Prozessschritt (Prozess Action) ist die unterste Ebene im Modell und beinhaltet einfache und automatisierte Aktionen, die wiederverwendbar sind (Urbas, 2012). Das Prozedur Modell (Procedural Model), auch Vorgehensmodell genannt, beschreibt die Verfahren, Vorgänge und Phasen zur Realisierung des Prozesses. Die festgelegten Ebenen können in allen Rezepttypen (Verfahrensrezept, Werksrezept, Grundrezept, Steuerrezept) oder deren steuerungstechnischen Modulen (Prozedursteuerungen) enthalten sein. Die Prozedur oder Rezeptprozedur (Procedure / Recipe Procedure) ist die allgemeine Strategie für das Produzieren einer Charge. Die Hierarchie besteht neben der Prozedur als oberster Ebene, aus Teilprozeduren (Unit Procedure), Operationen (Operation) und Funktionen (Phase). Um eine Strategie für die Ausführung eines zusammenhängenden Prozesses in einer Teilanlage zu realisieren, werden Teilprozeduren benötigt. Diese werden durch aufeinanderfolgende Operationen und den benötigten Schemata für die Auslösung, Organisation und Steuerung dieser Operationen abgebildet. Eine Operation wird auch als

Prozedurelement bezeichnet, das eine unabhängige Verarbeitungsaktion bestimmt. Beispiele für Operationen sind:

- Reinigung: Leeren und säubern eines Reaktors
- Füllung: Füllen des Reaktors mit destilliertem Wasser und Lösungsmittel
- Reaktion: mit Edukten beschicken und heizen

Die unterste Stufe eines Prozedurelements im Modell der Prozedursteuerung ist die Funktion (Phase). Beispiele für prozessorientierte Funktionen können Lenkung, Dosieren oder Heizen sein.

Das physikalische Modell (Physical Model) ist ein hierarchisches Modell der Dekomposition einer Batch Anlage und betrachtet dessen physikalische Elemente. Auf der obersten Ebene sind Anlage (Process Cell) und Teilanlage (Unit) zwei obligatorische Elemente. Eine Anlage bildet eine logische Gruppierung von Einrichtungen, die die zur Herstellung einer oder mehrerer Chargen benötigten Einrichtungen beinhaltet. Durch eine Gruppierung von zusammengehörenden Einzelsteuereinheiten (Control Modul) und / oder technischen Einrichtungen (Equipment Module), in denen größere Verarbeitungsaktivitäten ausgeführt werden können, entstehen Teilanlagen. Beispiele für Units sind Reaktion, Kristallisation oder Lösen. Technische Einrichtungen sind funktionale Zusammenfassungen von Einrichtungen, die kleinere Verarbeitungsaktivitäten ausführen können (z. B. Dosieren und Wägen). Jede technische Einrichtung enthält Einzelsteuereinheiten, die als einzelne Entität betrieben werden. Eine Einzelsteuereinheit realisiert die Basisautomatisierung und stellt eine Sammlung von Sensoren, Aktoren und anderen Einzelsteuereinheiten dar (Urbas, 2012).

3.4.2 Kontinuierliche Prozesse

In (Brandl, 2007) werden kontinuierliche Prozesse beschrieben, als die Verarbeitung von Materialien in einem kontinuierlichen Fluss. Diese Materialien sind durch ein Gewicht oder ein Volumen charakterisiert und besitzen keine diskreten Entitäten derselben. Diese Materialen werden durch spezialisiertes Equipment verarbeitet, das in einem stabilen Zustand operiert und einen bestimmten Teil des Produktionsprozesses übernimmt. Da der Prozess in einem stabilen Zustand verfährt, ist er nicht abhängig von der Bearbeitungszeit. Start, Überführung und Abfahren sind üblicherweise keine Prozessphasen in denen die gewünschte Güte erreicht wird. Die dabei entstehenden Produkte müssen meist separat behandelt werden. Dennoch sind diese Phasen sehr wichtig für den sicheren und effizienten Betreib der Anlage.

Die (ISA 106) beschreibt entsprechende Modelle für kontinuierliche Prozesse. Das Prozessführungsmodell besteht aus dem physischen Modell (Physical Model), dem Modell für Prozessanforderungen (Procedure Requirements Model) und dem Prozessimplementierungsmodell (Procedure Implementation Model).

Das physische Modell stellt die physischen Komponenten, die eine kontinuierliche Produktion ermöglichen, dar. Das Modell der Prozessanforderungen zeigt die verschiedenen Ebenen des physischen

Modells, in welchen Prozessanforderungen notwendig sein könnten. Die Umsetzung der Prozessanforderungen aus den einzelnen Ebenen des physischen Modells wird im Modell der Prozessimplementierung beschrieben.

3.4.3 Hybride Prozesse

Viele grundsätzlich kontinuierlich betreibbare Apparate wie Desorber oder Filter haben nur eingeschränkte Standzeiten und müssen regelmäßig gereinigt oder regeneriert werden. Während der Regenerierungsphase steht der Apparat nicht zur Verfügung. Um solche Verfahrenseinheiten in eine kontinuierliche Anlage zu integrieren, werden die Komponenten redundant ausgeführt und im Wechselbetrieb eingesetzt. Dies kann als Phase bezeichnet werden.

3.5 Bedienbilder (HMI)

Die Mensch-Maschine Schnittstelle zur Anzeige der Bedienbilder wird in der (VDI/VDE 3699-2, S. 3) definiert, als ein Sammelbegriff für alle Komponenten und Bedienelemente wie Anzeiger nebst wahrnehmbarer Aktionen und Reaktionen einer technischen Einrichtung, die der Kommunikation mit ihren Benutzern dienen. Die Aufgabe der Mensch-Maschine Schnittstelle ist es, den Bediener bei seiner Aufgabe zu unterstützen. Dazu zählt im besonderen Maße die Prozessführung, die als der bestimmungsgemäße wirtschaftliche und umweltverträgliche Betrieb einer verfahrenstechnischen Anlage definiert ist (VDI/VDE 3699-2, S. 9). Das Fließbild ist eine schematische Darstellung von Komponenten samt deren Verbindung durch Fließlinien zur Wiedergabe der Beziehungen in einer verfahrenstechnischen Anlage und der Leittechnik (VDI/VDE 3699-1, S. 14). Dabei wird zwischen dem verfahrenstechnischen und dem leittechnischen Fließbild unterschieden. Das verfahrenstechnische Fließbild bildet die Struktur der Anlage sowie der Verfahrensfluss ab. Das leittechnische Fließbild stellt die Funktionen, Funktionsbaugruppen und deren Verknüpfung dar. Um einer gute Bedienbarkeit zwischen den einzelnen Fließbildern realisieren zu können, ist eine Hierarchisierung der Bilder notwendig.

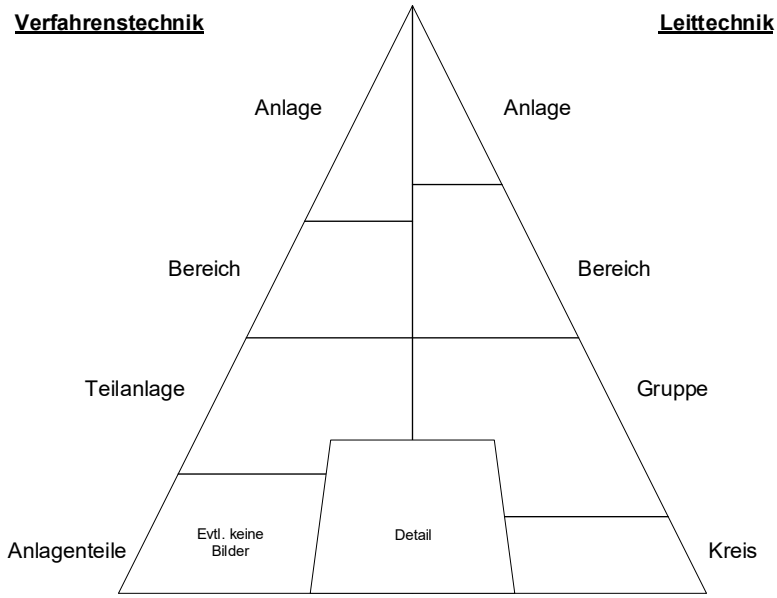


Abbildung 20: Hierarchieebenen von Fließbildern entsprechend (VDI/VDE 3699-3, S. 15)

Eine bewährte Einteilung entsprechend der (VDI/VDE 3699-3) sieht wie folgt aus:

- Das *Anlagenbild* ist eine vereinfachte Darstellung aller Bereiche der Anlage. Jeder Bereich ist direkt anwählbar und zeigt den aktuellen Zustand an. Es werden die verfahrenstechnischen Fließwege, sowie die Ein- und Ausgangsstoffe angezeigt, die Grobstruktur der Leittechnik wird visualisiert. Dabei können Rechtecke oder an die Anlage angelehnte Strukturen zum Einsatz kommen.
- Mit dem *Bereichsbild* werden verfahrensrelevante Apparate, Aggregate, Rohrleitungen, Energieströme und ggf. leittechnische Signale der Bereiche vereinfacht mit Symbolen veranschaulicht.
- Auf dem *Teilanlagen-* oder *Gruppenbild* werden funktionale Zusammenhänge (PLT-Stellen, Rohrleitungen, einfache Regelungs- und Steuerungsstrukturen) dargestellt.
- Das *Detailbild* visualisiert detaillierte Zusammenhänge einzelner Apparate und Aggregate mit Wirklinien. Signalflüsse lassen sich vom Sensor bis zum Aktor verfolgen. Es dient der Parametrierung, Inbetriebnahme und Aufklärung von Fehlern.

Für eine formale Beschreibung von Visualisierung existieren eine Reihe von Beschreibungsmitteln. Die Visualisierungssprache XVCML ist eine technologieunabhängigen Beschreibungsform, um webbasierte Visualisierungssysteme entwickeln zu können. Dabei wurde eine, auf die Aufgabenstellung angepasste XML-basierte Beschreibungssprache entwickelt, die alle Anforderungen von modernen Prozessvisualisierungen erfüllen soll. Ziel ist es, eine für den Entwickler ausreichende Bibliothek von vorgefertigten Visualisierungselementen zur Verfügung zu stellen, die mittels XML-basierter Programmierung konfiguriert und zu einer Visualisierung zusammengefügt werden können (Braune & Hennig, 2007).

Die Extensible Application Markup Language (XAML) ist eine Auszeichnungssprache, die auf XML basiert und für die Beschreibung von Oberflächen-Hierarchien von Microsoft entwickelt wurde. Sie definiert dabei eine Vielzahl von Eigenschaften, zum Beispiel grafische Elemente (Widgets), Benutzeroberflächen, Verhaltensweisen, Animationen, Transformationen oder die Darstellung von Farbverläufen. Die Logik hinter den grafischen Elementen wird in Form von .Net-Code eingefügt, der eine Trennung von Darstellung und Funktionalität ermöglicht. Die Darstellung wird über XML-Stylesheets definiert, XAML-Dateien selber sind hierarchisch strukturiert. Eigenschaften und Einstellungen zum Beispiel einer Schaltfläche, werden wie bei XML bzw. HTML im Tag als Attribute aufgeführt (Kühnel, 2008).

Das in der User Interface Description Language (UIML) hinterlegte Modell besteht aus drei Abschnitten. Der erste Teil definiert das Interface, in welchem Struktur, Inhalte, Darstellung und das Interaktionsverhalten abgehandelt werden. „Peers“ übernehmen das Mapping auf eine bestimmte Technologie und beschreiben die Applikationslogik. Den dritten Abschnitt bilden die Templates. Darin können Objekte mit festgelegten Eigenschaften definiert werden (Souchon & Vanderdonckt, 2003).

Bei der eXtensible Interface Markup Language (XIML) erfolgt die Beschreibung der Informationen mittels so genannten Ontologien. Das Modell lässt sich in Komponenten, Relationen und Eigenschaften zerlegen. In den Komponenten wird das Vorgehensmodell beschrieben (z.B. Task Model, Dialog Model). Relationen werden genutzt, um Elemente innerhalb einer Komponente oder Komponenten untereinander zu verbinden. Über die Eigenschaften (Attribute) werden den Elementen Werte zugewiesen (Puerta & Eisenstein, 2002).

Für die automatische Generierung von Bedienbildern werden in der Fachliteratur verschiedene Lösungsansätze beschrieben. Diese nutzen HTML5 (Hickson et al., 2014), SVG (Schmitz, 2010) oder proprietäre HMI-Formate (Calvary et al., 2003; Hickson et al., 2014; Reuther, 2003; Urbas, Hennig, Hager, Doherr & Braune, 2011). Diese Formate benötigen entweder umfangreiche MDA-Rahmenwerke (Calvary et al., 2003; Hickson et al., 2014; Reuther, 2003; Urbas et al., 2011), sind für die Prozessautomation nur bedingt geeignet (Calvary et al., 2003; Hickson et al., 2014; Reuther, 2003) oder lassen sich nur mit speziellen Ergänzungen mit den für ein semantisches Layout notwendigen Attributen erweitern (Hickson et al., 2014; Schmitz, 2010). Mit den Ergebnissen in (Obst et al., 2012) wurde ein durchgängiges automatisches HMI-Engineering aus CAE Planungsdaten ermöglicht. Mit den darin entwickelten formalen Ansätzen ist ein systemunabhängiger Austausch von Bedienbildern möglich. Die formale Beschreibung der Bedienbilder erfolgt mit Hilfe der Beschreibungssprache aHMI (abstract Human-Machine Interface), die das Metamodell CAEX benutzt. Mit Hilfe des aHMI kann der Aufbau eines Bedienbildes selber, die Anordnung der Bedienbilder einer Hierarchie und eine Verknüpfung der Bedienbilder untereinander abgebildet werden. Die Beschreibung der einzelnen Bedienbild-Elemente für die Darstellung beispielsweise einer Pumpe wird hierbei nicht ermöglicht. Diese werden in aHMI abstrakt über das CAEX Rollenkonzept definiert.

Das XML-basierte Beschreibungsmittel GraphML (Brandes, Eiglsperger, Herman, Himsolt & Marshall, 2002; GraphML Working Group, 2006) ist ein Format zur Beschreibung der Struktur eines Graphen sowie einem Erweiterungsmechanismus für applikationsspezifische Daten. GraphML unterstützt gerichtete, ungerichtete und gemischte Graphen, Hypergraph sowie hierarchische Graphen. Es bietet Möglichkeiten zur Beschreibung der graphischen Repräsentation des Graphen, Referenzen auf externe Daten, sowie anwendungsspezifische Attribute. Damit bietet GraphML eine einfache Möglichkeit, strukturelle Zusammenhänge wie sie auf einem Bedienbild vorzufinden sind, zu beschreiben.

3.6 Allgemeine Modelle der Prozessautomatisierung

Neben aspektbezogenen Beschreibungsmitteln finden eine Reihe von allgemeinen Beschreibungsmitteln und Modellen in der Prozessautomation Anwendung. Für die Abbildung mehrerer Aspekte mit einem Beschreibungsmittel, sollen im Folgenden allgemeine Beschreibungsmittel untersucht werden.

3.6.1 AutomationML (DIN EN 62714)

AutomationML (AML) ist ein nach (DIN EN 62714-2) genormtes Datenaustauschformat für Engineeringdaten in sowohl der Fertigungs- als auch Prozessindustrie und wurde entwickelt, um eine lizenzgebührenfreie, herstellerunabhängige Datenbasis für Entwicklungswerkzeuge bereitzustellen. Die im Folgenden aufgeführte inhaltliche und funktionale Beschreibung basiert in wesentlichen Teilen auf (Drath, 2010). Derzeit ist in den meisten Fällen kein durchgehendes Engineering in Planungsprozessen möglich, wie es bei der Nutzung eines oder mehrerer Werkzeuge mit einer gemeinsamen Datenbasis wie beispielsweise in der Engineering Software COMOS[1] realisierbar ist. Häufig wird dies zumeist durch den hohen Detailgrad kleinerer Spezialwerkzeuge bedingt, die eine größere Anzahl von Einstellungen und Parametern aufweisen und auf Hardwarekomponenten bestimmter Hersteller abgestimmt sind (Herstellerabhängigkeit). Durch diesen hohen Spezialisierungsgrad sind fast alle Datenformate untereinander inkompatibel. Daten, die von einem Planungswerkzeug zum nächsten verwendet werden sollen, müssen durch den Anwender manuell übertragen werden, was in einem höheren Zeit- und Kostenaufwand sowie eine erhöhte Fehleranfälligkeit resultiert. Das Ziel von AutomationML ist es, diese „Papierschnittstelle“, das heißt die Extrahierung relevanter Daten aus dem vorherigen Planungsschritt für den nächsten auf Papier, sowie die manuelle Eintragung in das nächste Werkzeug, aus dem Planungsprozess zu entfernen. Die Stärken von AutomationML liegen hierbei in der Verwendung bereits etablierter Datenformate, einer Separierung beziehungsweise Kapselung von Daten, in der nicht alle Daten in einer Datei abgebildet sind, sowie einer geschickten Referenzierung der Daten, um Redundanzen zu vermeiden. Weiterhin sind viele Modellierungsansätze optional und Kopplungen mitunter relativ lose, um möglichst viele Modellierungsvarianten abbilden zu können und

[1] http://w3.siemens.com/mcms/plant-engineering-software/de/Seiten/Default.aspx (letzter Zugriff 21.5.2016)

eventuelle Einschränkungen bei der Modellierung in den Aufgabenbereich der Werkzeuge zu verlagern. Die strukturelle Integrität kann zwar mittels einer XSD-Datei verifiziert werden, die Verantwortung für die inhaltliche Konsistenz wird allerdings an die verwendeten Werkzeuge übertragen, da eine solche Prüfung in AutomationML nicht vorgesehen ist.

Aufbau und Struktur

AutomationML bedient sich mehrerer etablierter Datenformate, um unter anderem eine schnellere Marktverbreitung und Annahme durch die Hersteller zu gewährleisten. Abbildung 21 zeigt hierbei die verwendeten Grundformate. Als Dachformat wird CAEX eingesetzt, um alle Teile und Aspekte der Anlage modellieren zu können. Sofern es sinnvoll ist, ist eine Abbildung von Objekten durch Unterformate möglich. Für die Modellierung von geometrischen und kinematischen Informationen wird COLLADA verwendet, während Steuerungen und Verhalten im PLCopen Format dargestellt werden.

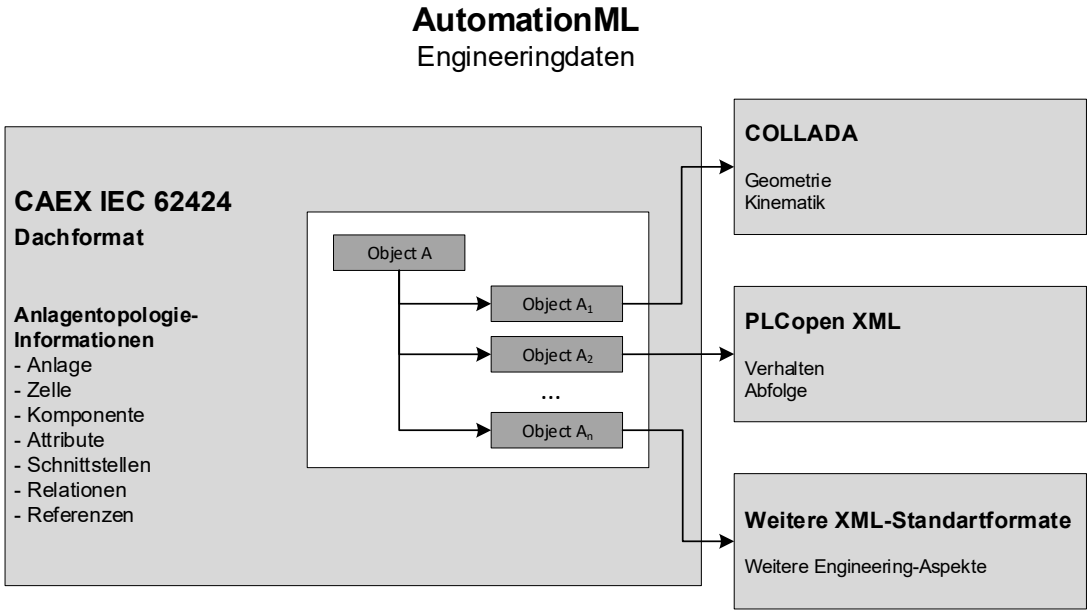


Abbildung 21: AutomationML: Aufbau und Struktur (Drath, 2010)

Ein wichtiger Unterschied zu anderen objektorientierten Sprachen besteht in der fehlenden Strukturvorschrift für Objektinstanzen von Klassen in AutomationML. Sie dient als Kopiervorlage, wobei Instanzen anschließend nicht nur in ihren Parametern, sondern auch strukturell verändert werden können. In letzterem Fall weist lediglich noch eine Referenz auf die sehr lose Verbindung zwischen Instanz und Klasse hin. Um einen möglichst großen Kreis an Werkzeugen anzusprechen und wenig Einschränkungen zu bieten, belässt AutomationML die Entscheidung den darauf aufbauenden Werkzeugen, diese Verbindung enger zu gestalten. Als Ergebnis dieser Modellierung ist es möglich, dass Objektinstanzen die gesamte Information ihrer Klasse enthalten. Methoden oder Logik werden mit CAEX nicht dargestellt, sondern lediglich modellierte Objekte und Strukturen und deren Beziehungen zueinander. Grundlegend besteht AutomationML aus folgenden Bestandteilen:

- RoleClassLib: Bibliothek mit Rollenklassen,
- SystemUnitClassLib : Bibliothek mit ausspezifizierten Klassen,
- InterfaceClassLib: Bibliothek mit Schnittstellenklassen.

Im Folgenden werden für das Verständnis wichtige Elemente, diskutiert. Sämtliche der im Folgenden beschriebenen Elemente, außer Attribute, verfügen in AutomationML außerdem über:

- Header Information (Versionierung, Copyright und Beschreibung),
- Revisions Informationen zur Versionsverwaltung,
- einer ID zur indeutige Objektidentifizierung,
- ein nicht notwendigerweise eindeutiger Name, der im Fall von Klassen zur Referenzierung genutzt wird,
- Ref-Pfade (z.B. RefBaseClassPath, RefRoleClassPath). Diese sind nicht bei Bibliotheken vorhanden und verweisen auf Elternklassen bzw. Klassen im Fall einer Objektinstanz. Der Pfad stellt nur eine willkürlich definierbare Hierarchie dar. Somit kann ausschließlich die direkte Elternklasse bestimmen werden, für die Bestimmung weiterer Abhängigkeiten muss demnach rekursiv vorgegangen werden.

Abbildung 22 und Abbildung 23 verdeutlichen die Zusammenhänge zwischen den Elementen, auf welche in den nächsten Unterabschnitten näher eingegangen wird. Hierbei stehen die Abkürzungen RC für RoleClass, SUC für SystemUnitClass und IE für InternalElement.

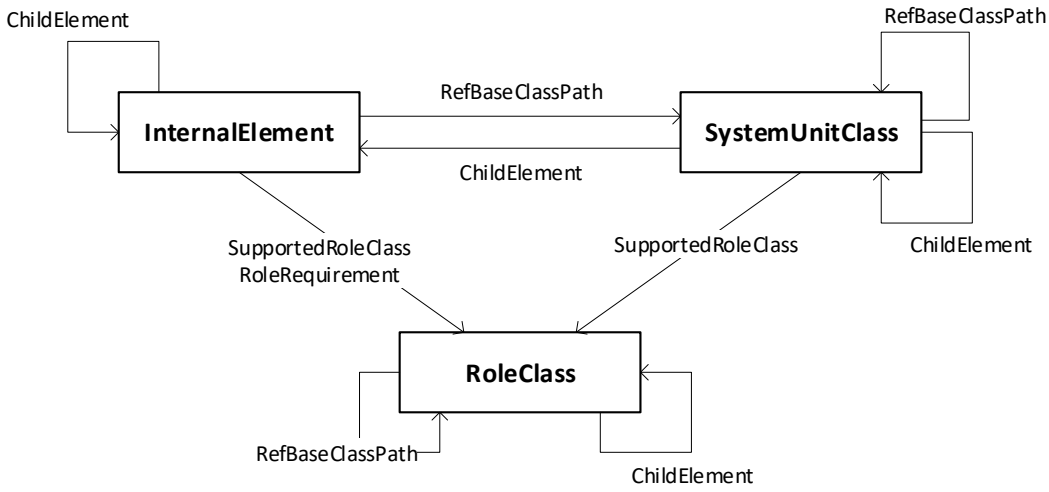


Abbildung 22: CAEX Struktur Teil 1 (Henßen & Schleipen, 2014)

Attribute (Atribut)

Ein Attribut ist ein Datenwert, der von jeder in den folgenden beschriebenen Strukturen in unbegrenzter Anzahl beinhaltet werden kann. Ein Attribut kann Daten aus verschiedensten Datentypen enthalten oder einen komplexen Typ besitzen und wiederum aus anderen Attributen aufgebaut sein. Attribute sind die einzigen Elemente in AutomationML, die eine Wertinformation beinhalten. Alle anderen Strukturen

dienen nur zur Modellierung von Strukturbeziehungen untereinander und zur Interpretation der Attributinformation. Jedes Attribut erhält weiterhin einen eindeutigen Namen, eine Beschreibung, einen Standardwert und eine Einheit. Außer dem Namen und dem Datentyp ist jedes Feld optional, es kann demzufolge auch keinem Datenwert zugewiesen werden. Weiterhin können Attribute im Gegensatz zu Objektinstanzen keine Klassenstruktur besitzen, zwar können Attributen neben den bereits genannten Basiselementen wie Name und Datentyp andere Daten in Form von enthaltenen Sub-Attributen mitgegeben werden, allerdings kann deren Mitgabe über die Definition eines Attributtyps nicht erzwungen werden.

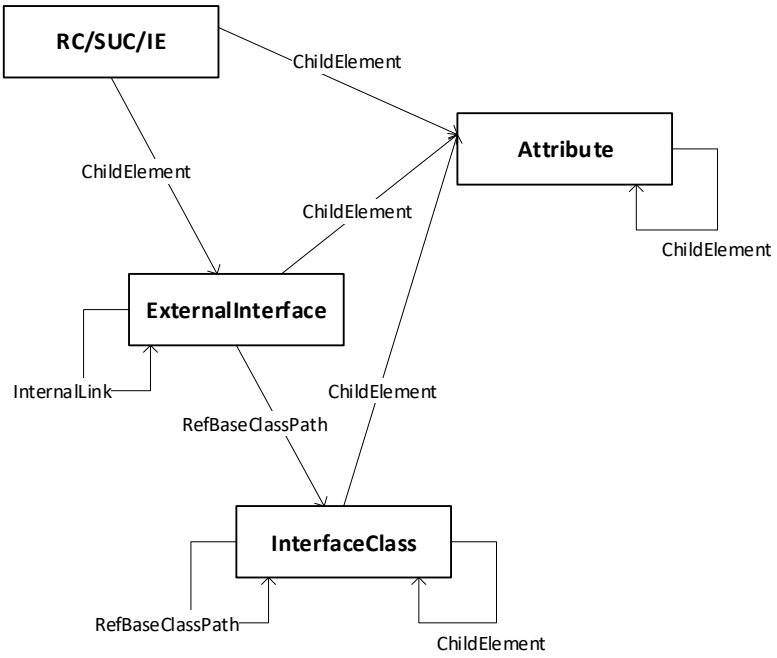


Abbildung 23: CAEX Struktur Teil 2 (Henßen & Schleipen, 2014)

Objektinstanz (InternalElement)

Ein InternalElement ist eine Objektinstanz und beschreibt damit ein spezifisches Objekt eines Objekttyps (Klasse). Objektinstanzen können sowohl in der SystemUnitClassLib als auch der InstanceHierarchy genutzt werden. Sie finden in beiden Bereichen gleiche Anwendung, jedoch kann in der InstanceHierarchy eine zusätzliche Zuweisung zu genau einer Rollenklasse ermöglicht werden. Weiterhin ist eine spezifische Zuordnung von Objektinstanzen zu einer jeweiligen SystemUnitClass möglich. Die Instanz bildet praktisch eine Kopie der Klasse, wobei die komplette Struktur und deren Inhalt theoretisch verändert werden können, sodass nur noch die Referenz zur Klasse auf die Verbindung hinweist. Die Einschränkung möglicher Strukturänderungen wird dabei in den Aufgabenbereich der AutomationML unterstützenden Werkzeuge verschoben. Objektinstanzen unterstützen keine Vererbung, dies geschieht auf Typ- bzw. Klassenebene.

Rollenklassen (RoleClass)

Rollenklassen in CAEX beziehungsweise AutomationML dienen der Darstellung abstrakter Objekte. Dies kann, wie der Name bereits suggeriert, die Rolle sein, die das Objekt in der Anlage oder dem Prozess einnimmt, so kann ein Motor als Rollenklasse definiert werden. Eine Rollenklasse besitzt nur eine strukturelle Definition, mit welcher eine Festlegung über das Vorhandensein von Eigenschaften eines Motors wie Drehzahl und die Leistung ermöglicht wird. Der genaue Wert der erbrachten Eigenschaft ist jedoch abhängig vom speziell gewählten Motor, der mit dem Klassentyp der SystemUnitClass abgebildet wird.

Im ursprünglichen CAEX ist es nur möglich, einem Objekt eine Rolle zuzuweisen. Es existiert eine Beziehung mit dem Namen „SupportedRoleClass“, die die Angabe mehrerer Rollenklassen in SystemUnitClassLib und InstanceHierarchy erlaubt. Diese wären aber Oder-verknüpft, das heißt das Objekt müsste eine der Rollen ausüben. Die erläuterte Zuweisung zu einer Rolle in der InstanceHierarchy hätte hierbei die Auswahl getroffen. Für AutomationML war dies ein Hindernis, sodass man hier eine andere Interpretation, nämlich eine Und-Verknüpfung verwendete. Ein Objekt kann nun mehrere Rollen gleichzeitig ausüben, was Mehrfachvererbungen ermöglicht.

Die Vererbung zwischen Rollenklassen wird unterstützt. Eine erbende Rollenklasse erbt alle Attribute der Elternklasse und kann diese überschreiben, ein Entfernen im Sinne einer Strukturänderung ist, im Gegensatz zu einer Klasse-Instanz-Relation, nicht möglich.

Eine Aggregation von Rollenklassen wird für Modellierungszwecke nicht unterstützt, denn sie dient vorrangig der hierarchischen Ordnung von Klassen. Die Darstellung einer Rollenklasse, die mehrere Objekte in Form von Rollenklassen enthält, ist nicht möglich, da die Rollenklasse nicht andere Rollenklassen, sondern deren Instanzen enthält. Instanzen sind der SystemUnitClassLib und der InstanceHierarchy vorbehalten. Jedoch können neben Attributen auch Schnittstellen in Rollenklassen enthalten sein.

Ausspezifizierte Klassen (SystemUnitClass)

Ausspezifizierte Klassen (SystemUnitClass) werden in AutomationML zur Darstellung real existierender Objekte oder Prozesse verwendet. Bei der Anlagenplanung könnte die gesamte Anlage aus Instanzen von, in dieser Bibliothek spezifizierten, Objekten zusammengebaut werden. Während in einem früheren Planungsschritt mit Rollenklassen eine abstrakte Funktion angegeben wurde, kann in einem späteren Schritt eine Zuweisung zu einem realen Objekt erfolgen. Zu diesem Zweck ist eine SystemUnitClass in der Lage, mit der SupportedRoleClass-Beziehung Zuweisungen auch in Form von Mehrfachvererbungen zu Rollen vorzunehmen. Eine Schwachstelle an dieser Zuweisung stellt dabei die optionale Übernahme von Attributen von der Rollenklasse dar. Es kann nicht sichergestellt werden, dass eine auf einer Role Class basierende SystemUnitClass, alle Attribute dieser besitzt.

Vererbung ist analog zu den Rollenklassen auch hier möglich. Eine erbende Klasse übernimmt alle Attribute sowie die ausgeübten Rollen der Elternklasse, eine strukturelle Änderung der Elternelemente ist nicht möglich, allenfalls ein Überschreiben des Inhaltes.

Aggregation ist im Gegensatz zu den Rollenklassen möglich und für die Modellierung einsetzbar. Zu diesem Zweck wird das AutomationML-Element InternalElement genutzt. In der SystemUnitClassLib ist es daher möglich, im Gegensatz zur RoleClassLib, Instanzen mit Zuweisungen zu Rollen oder Klassen zu verwenden. Dies ist dadurch gegeben, dass eine SystemUnitClass als Kopiervorlage für eine Instanz angesehen werden kann und somit strukturell dieselben Eigenschaften besitzt. Eine Aggregation von Klassen, um eine bestimmte hierarchische Ordnung ohne Einfluss auf die Modellierung ähnlich wie bei Rollenklassen zu gewährleisten, ist hier ebenfalls möglich.

Schnittstellenklassen (InterfaceClass)

Eine Schnittstellenklasse ist eine Vorlage für eine Schnittstelleninstanz. Eine Schnittstelle dient der Kopplung von Objekten und kann in allen drei anderen Bereichen, das heißt in der InstanceHierarchy, SystemUnit- ClassLib und RoleClassLib verwendet werden. Die Kopplung selbst findet zwischen zwei Schnittstellen in Instanzen über einen sogenannten InternalLink statt. Die Attribute und die Klasse der Schnittstelle definieren dabei die semantische Bedeutung, die Art der Kopplung. AutomationML gibt bereit zahlreiche Schnittstellen vor, die entweder direkt oder als Basisklassen für andere Schnittstellen verwendet werden können. Beispiele wären eine Port-Schnittstelle oder spezielle Schnittstellen zur Anbindung bzw. Referenzierung externer Dateien außerhalb von CAEX. Eine Aggregation ist bei Schnittstellen nicht verfügbar, dafür unterstützen sie die Vererbung analog zu Rollenklassen und ausspezifizierten Klassen. Auch hier werden Attribute der Elternklasse übernommen und können inhaltlich verändert, aber nicht entfernt werden. Schnittstellen können allen Klassen und Instanzen zugewiesen werden. Eine Kopplung kann jedoch nur zwischen instanziierten Objekten erfolgen und ist daher auf die Instanzhierarchie beschränkt.

Instanzhierarchie (InstanceHierarchy)

Die Instanzhierarchie dient der Modellierung der tatsächlichen Anlage beziehungsweise des realen Prozesses. Während die anderen drei Bereiche den Modellierungswerkzeugen die Bibliotheken zur Verfügung stellen, steckt die eigentliche Information in diesem Abschnitt. Je nach Phase des Planungsprozesses können die Elemente abstrakt definiert, also in Form einer Rollenklassenzuweisung oder direkt als System UnitClass spezifiziert sein.

Es handelt sich jedoch stets um Instanzen, deren Eigenschaften auch für die Instanzhierarchie gelten. Eine Vererbung ist somit nicht möglich, dafür jedoch die Aggregation. Da Instanzen sämtliche Information ihrer möglichen Elternklassen beinhalten, ist es möglich, ausschließlich die Instanzhierarchie zum finalen Zeitpunkt zu betrachten, Dies ist dann interessant, wenn eine auf AutomationML basierende Strukturabbildung unveränderlich sein soll.

Nutzung externer XML-Formate

Es existieren spezifische Problemstellungen, für die eine Abbildung in CAEX zwar möglich, aber sehr aufwendig ist. Sofern bereits etablierte Formate zur Behandlung existieren, werden diese von AutomationML bevorzugt verwendet. Bereits integriert sind hierbei PLCopen und COLLADA. Ersteres dient hauptsächlich zur Darstellung von Programmlogik einer SPS (basierend auf der IEC 61131-3) und ist in der Lage, die fünf gängigen SPS-Programmiersprachen Anweisungsliste, Ablaufsprache, Strukturierter Text, Kontaktplan und Funktionsbausteinsprache abzubilden. Der Name setzt sich aus PLC (Programmable Logic Controller) und open (für frei verfügbar; herstellerunabhängig) zusammen. COLLADA (Collaborative Design Activity) ist für den Datenaustausch von Geometrie und Kinematik entworfen worden. Zusätzlich bietet AutomationML die Möglichkeit, weitere externe Formate einzubinden. Zu diesem Zweck existiert eine AutomationML Schnittstelle ExternalDataConnector, von dem die beiden Schnittstellen COLLADAInterface und PLCopenXMLInterface bereits erben und verfügbar sind. Zum Einbinden weiterer Formate kann ExternalDataConnector erweitert werden und dient als Referenz auf die Datei in dem entsprechenden Format.

3.6.2 NE 150 - Namur Datencontainer

Im Rahmen der NE 150 wurde ein agiler Standardisierungsprozess gewählt. Durch eine schrittweise Standardisierung wurde zunächst nur die PLT-Stelle betrachtet und in den Datenaustausch eingebracht. Weitere Artefakte können in den nächsten Schritten aufgenommen werden, wodurch proprietäre Datenmodelle durch Standards ersetzt werden. Der Prozess zur Erstellung der NE 150 wurde in (Schüller et al., 2015) zusammengefasst. Die im Folgenden aufgeführten Inhalte zu NE 150 wurden, soweit nicht anders gekennzeichnet aus (Schüller et al., 2015) entnommen.

Die PLT-Stelle, als Schnittstelle zwischen der Prozesssteuerung und dem eigentlichen Prozess, wird von mehreren Gewerken bearbeitet. Sie ist während des Engineerings und des Betriebs einer Anlage von zentraler Bedeutung. Ein durchgängiges Engineering wird jedoch durch die verschiedenen Werkzeuge der einzelnen Gewerke erschwert. Daher ist ein Datenaustausch zwischen den Werkzeugen notwendig, wozu Export- und Import-Schnittstellen erforderlich sind. Aufgrund des parallelen Arbeitens ist ein bidirektionaler Austausch notwendig. Für einzelnen Aspekte existieren bereits ein Reihen von Beschreibungsmitteln. Eine Auswahl der Beschreibungsmittel und deren Zuordnung ist der Abbildung 24 zu entnehmen.

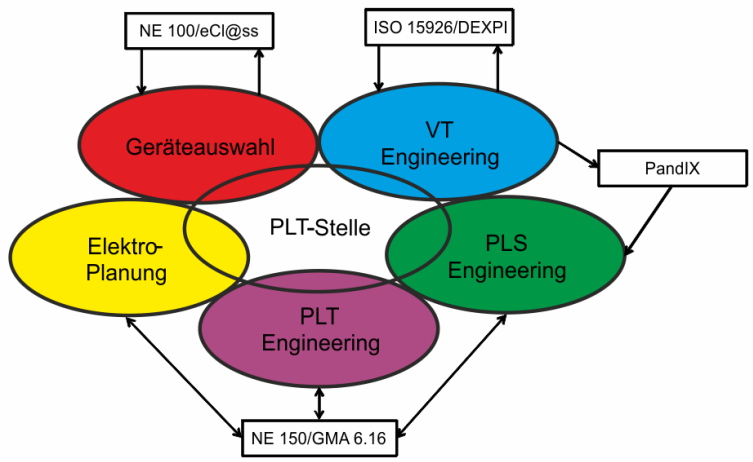


Abbildung 24: Einordnung der PLT-Stelle zwischen den Gewerken und deren Beschreibungsmittel an den Schnittstellen (Schüller et al., 2015)

Zentrales Element der NE 150 ist das Klassenmodell der PLT Stelle, welches in Abbildung 25 dargestellt ist.

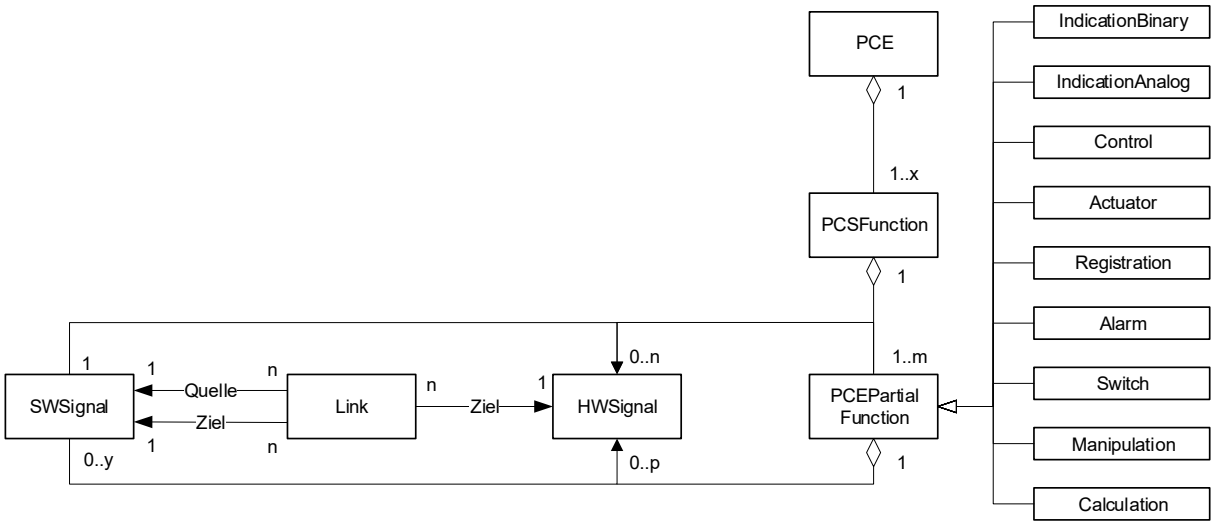


Abbildung 25: NE150 Klassenmodell der PLT-Stelle nach (Schüller et al., 2015)

Eine PLT-Stelle umfasst häufig genau eine PLT-Funktion, sie kann aber auch einen Loop mit mehreren PLT-Funktionen umfassen. Die PLT-Funktion (IEC 62424: PCE-Kategorie) wird üblicherweise durch den Erstbuchstaben der Typical-Kennzeichnung bestimmt („F“ für Durchfluss oder „P“ für Druck). Die Auflistung der möglichen Erstbuchstaben und ihrer Bedeutung ist in der IEC 62424 enthalten. Jede PLT-Stelle besteht aus mindestens einer PLT-Teilfunktion (IEC 62424: PCE-Verarbeitungsfunktion). Die Teilfunktionen sind üblicherweise durch die Folgebuchstaben der Typical-Kennzeichnung beschrieben. Das Datenmodell beschreibt diese mit Hilfe je einer eigenen Klasse und unterstützt im ersten Schritt die Verarbeitungsfunktionen Anzeige von Binärsignalen (IndicationBinary), Analoganzeige (IndicationAnalog), Regelung (Control), Aktor (Actuator), Aufzeichnung (Registration), Alarm (Alarm), Schaltfunktion (Switch), Manipulierung (Manipulation) und Berechnung (Calculation). Diese Klassenbibliothek ist erweiterbar. Weiterhin besitzt jede PLT-Funktion genau ein Software-Signal, dass das leittechnisch interpretierte Signal modelliert (den Füllstand 0 bis 1 m). Auf dieses „Stamm-Software-

Signal“ beziehen sich alle PLT-Teilfunktionen. So werden beispielsweise Schaltfunktionen beim Über- oder Unterschreiten eines definierten Werts dieses Software-Signals ausgelöst. Im Regelfall ist diesem Software-Signal genau ein Hardware-Signal (z.B. 4 bis 20 mA) zugeordnet, wodurch die Umsetzung von 4 bis 20 mA zu 0 bis 1 m definiert wird. Es sind jedoch auch PLT-Funktionen wie Redundanzen oder Soft-Sensoren mit mehreren oder keinen Hardwaresignalen denkbar. Einige Teilfunktionen besitzen ihrerseits Software- und Hardware-Signale, beispielsweise für Schaltungen oder Reglerausgänge.

3.6.3 OPC UA (IEC 62541)[2]

OPC UA ist ein plattformunabhängiger Standard, der in der Industrie eingesetzt wird, um den Datenaustausch sowie die Regelung und Steuerung von Feldgeräten über verschiedene Netzwerkstrukturen zu ermöglichen. Die nachfolgende Beschreibung basiert auf der IEC 62541 (IEC 62541-1). OPC UA stellt die folgenden Grundlagen bereit:

- Informations- und Datenmodell für Struktur, Verhalten und Semantik
- Nachrichtenmodell für Interaktion zwischen Applikationen
- Kommunikationsmodell für Datentransport
- Konformitätsmodell zur Gewährleistung der Interoperabilität zwischen Systemen

Weiterhin werden Dienste, Alarm- und Meldesysteme, Zugriff auf historische Daten, sowie eine zuverlässige und sichere Kommunikation unterstützt beziehungsweise bereitgestellt. Zur Anwendung kommt hierbei eine Server-Client-Struktur, in der Feldgeräte oder Prozessleitsysteme als Client fungieren und der Server eine Art Datenvermittler darstellt. Die Darstellung von Informationen wie von Feldgeräten, findet im OPC UA Server über einen oder mehrere Knoten statt (siehe Abbildung 26).

[2] Die in Kapitel 3.6.2 dargestellten Ausführungen zum Stand der Technik des Beschreibungsmittels AutomationML, wurden erstmals, in der vom Autor betreuten studentischen Abschlussarbeit (Wassilew (2015)) zusammengestellt.

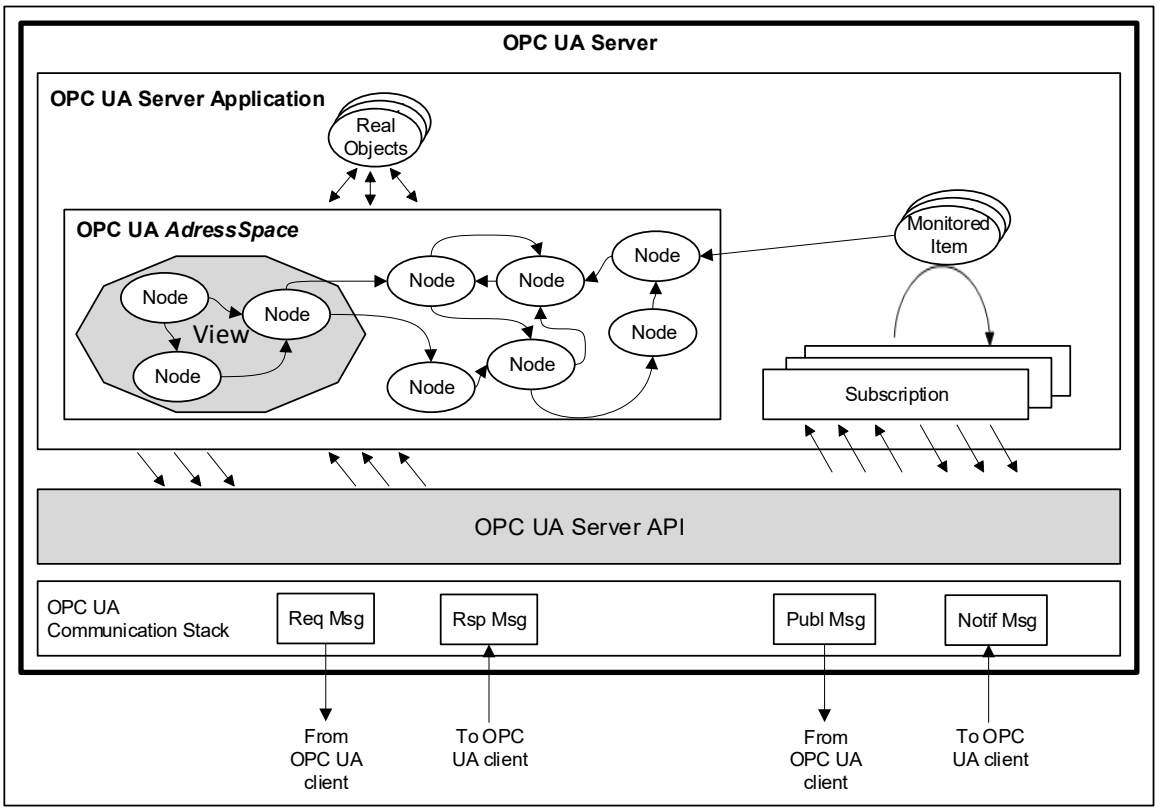


Abbildung 26: OPC UA Architektur nach (IEC 62541-1)

Zu jedem beliebigen Knoten kann ein *MonitoredItem* erstellt werden, eine Referenz auf den Knoten und ein beobachtbares Objekt. Jeder Client kann nun über eine *Subscription* zu einem Beobachter werden und wird über jede Zustandsänderung des Knotens informiert. Ein direktes Anfordern des Knoteninhaltes ist daher nicht erforderlich. Weiterhin können je nach Einstellung des Knotens und Berechtigungen des Nutzers auch Änderungen geschrieben werden. Eine abstraktere objektorientierte Modellierung ist ebenfalls möglich.

Knotenmodell

Zur Abbildung von Informationen verwendet OPC UA ein Knotenmodell. Einer oder mehr Knoten (node) können zusammengefasst ein reales Objekt repräsentieren. Knoten selbst enthalten Attribute und Referenzen auf andere Knoten. Dies wird in Abbildung 27 verdeutlicht. Ein Attribut ist dabei als ein Datenwert zu verstehen, d.h. beispielsweise eine Zeichenkette, ein boolescher Wert oder eine Zahl. Es kann über den Zugriff auf einen Knoten direkt ausgelesen oder, je nach Zugriffsrecht, auch beschrieben werden. Eine Referenz definiert Verbindungen zwischen Knoten, wobei der Referenztyp die semantische Bedeutung der Verbindung festlegt. Knoten werden durch Knotenklassen spezifiziert, die bestimmen, um was für eine Art Knoten es sich handelt.

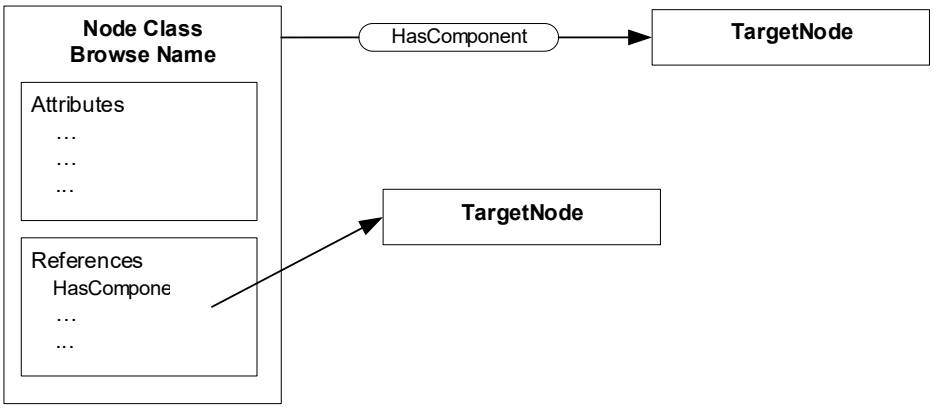


Abbildung 27: OPC UA Knoten im Adressraum nach (IEC 62541-1)

Objektmodell

Das Objektmodell ermöglicht eine abstraktere Abbildung von Knoten in Form von Objekten, Variablen und Methoden. Jeder Bestandteil der objektorientierten Modellierung wird dabei von einem Knoten mit einer bestimmten Knotenklasse ausgeübt, der den Modellbestandteil definiert. Abbildung 28 zeigt die Abhängigkeiten zu dem Basisknoten.

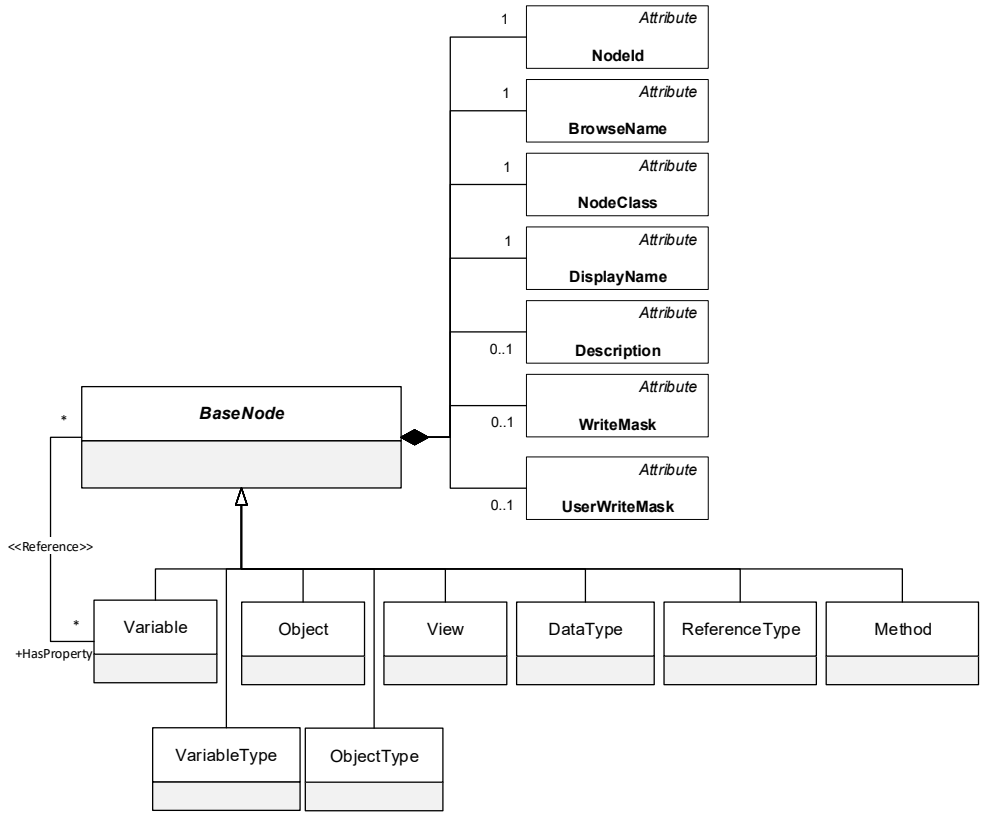


Abbildung 28: OPC UA Basisknoten nach (IEC 62541-1)

Im Folgenden werden die wichtigen Elemente, die gleichzeitig Knoten sind, genauer beschrieben.

Variablen und Variablentypen

Variablen können auch in Eigenschaften und Datenvariablen unterteilt werden. Eigenschaften sind serverdefinierte feste Werte, wie beispielsweise ein minimal oder maximal erlaubter Wert und können von außerhalb nicht geändert werden. Um Rekursionen zu vermeiden, sind Eigenschaften von Eigenschaften nicht erlaubt. Datenvariablen enthalten direkt zugreifbare Werte, die letztlich die Information abbilden. So kann ein Temperatursollwert über eine Datenvariable abgebildet werden. Abbildung 29 zeigt einen Überblick über den Aufbau von Variablen. Eine Variable verfügt unabhängig vom Typ über einige wichtige Attribute.

- *Value*: Das vielleicht wichtigste Attribut beinhaltet den derzeitigen Wert der Variablen).
- *ValueRank*: Dieser Wert gibt an, ob es sich um ein Array oder einen Skalarwert handelt bzw. wie die Dimension des Arrays ist.
- *ArrayDimensions*: Entgegen der intuitiven Annahme gibt dieserWert nicht die Dimension des Arrays an (das ist die Aufgabe von ValueRank), sondern legt für jede Dimension die Größe fest, beispielsweise (4,5) für eine 4x5 Matrix.
- *AccessLevel*: Die benutzerunabhängige Zugriffsstufe
- *UserAccessLevel*: Die benutzerabhängige Zugriffsstufe
- *MinimumSamplingInterval*: Die Aktualisierungsrate der Datenwerte in Millisekunden. Sie gibt an, wie regelmäßig der OPC UA Server Daten aktualisieren soll.
- *Historizing*: Gibt an, ob die Datenwerte archiviert werden sollen.

Der Variablentyp verfügt analog zur Variable über einen Value-, ValueRank und ArrayDimensions-Wert. Diese legen den Default-Wert für die Variable fest und verhalten sich ansonsten analog zur Variablen. Ein weiteres wichtiges Attribut ist der Datentyp. Er stellt als NodeId eine Referenz auf den Knoten mit dem entsprechenden Datentyp dar.

Objekte und Objekttypen

Objekte dienen zur Modellierung realer physischer Strukturen oder Prozesse. Wie in Abbildung 29 dargestellt, kann ein Objekt Variablen und Methoden enthalten und ist einem Objekttyp zugeordnet. Des Weiteren verfügt ein Objekt über das EventNotifier-Attribut, das festlegt, ob der entsprechende Objekt-Knoten sich bei einem Event anmelden kann bzw. Zugriff auf historische Events hat.

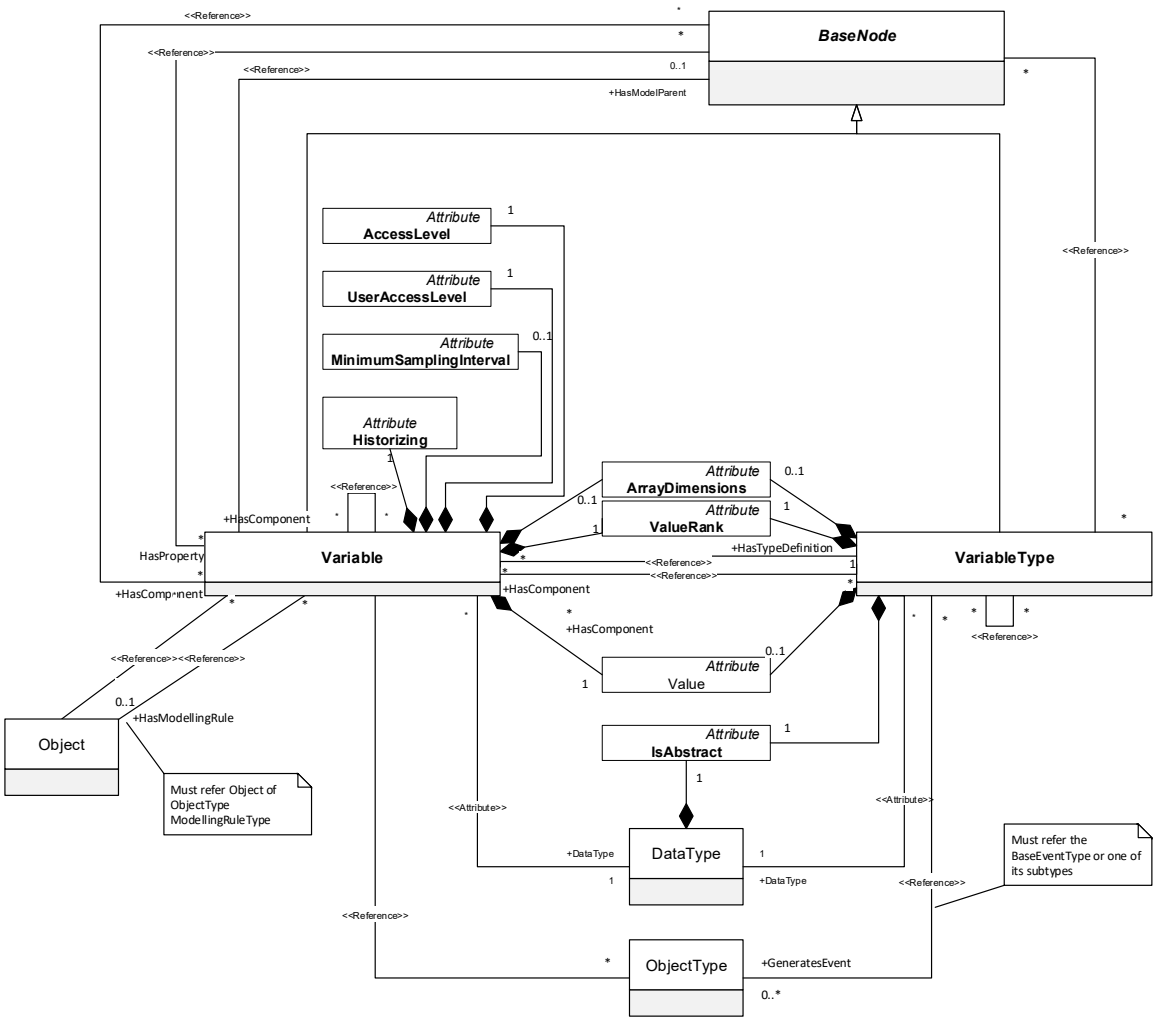


Abbildung 29 : OPC UA Variablen und Variablentypen nach (IEC 62541-1)

Sichten auf das Modell (View)

Eine Sicht (View) ist ein Knotentyp, der nicht direkt zur Modellierung genutzt wird, sondern einem OPC Client erlaubt aus einer Untermenge von Knoten auszuwählen. Somit dient nicht mehr der gesamte Adressraum als zu durchsuchender Raum, sondern nur noch die spezielle Sicht. Dies ist bereits in Abbildung 27 zu sehen. Die Abhängigkeiten von der Knotenklasse View werden in Abbildung 31 dargestellt. Außer dem EventNotifier-Attribut, das analog zu dem gleichnamigen Attribut eines Objektes auch hier dieselbe Funktion übernimmt, existiert noch das ContainsNoLoops-Attribut. Dieses legt über einen Binärwert fest, ob es in der Sicht bei dem Nachverfolgen von Referenzen zu einem geschlossenen Kreis kommen kann, dass heißt beim Folgen eines Pfades von Referenzen von einem Knoten aus, wieder eine Referenz auf diesen Knoten erhält.

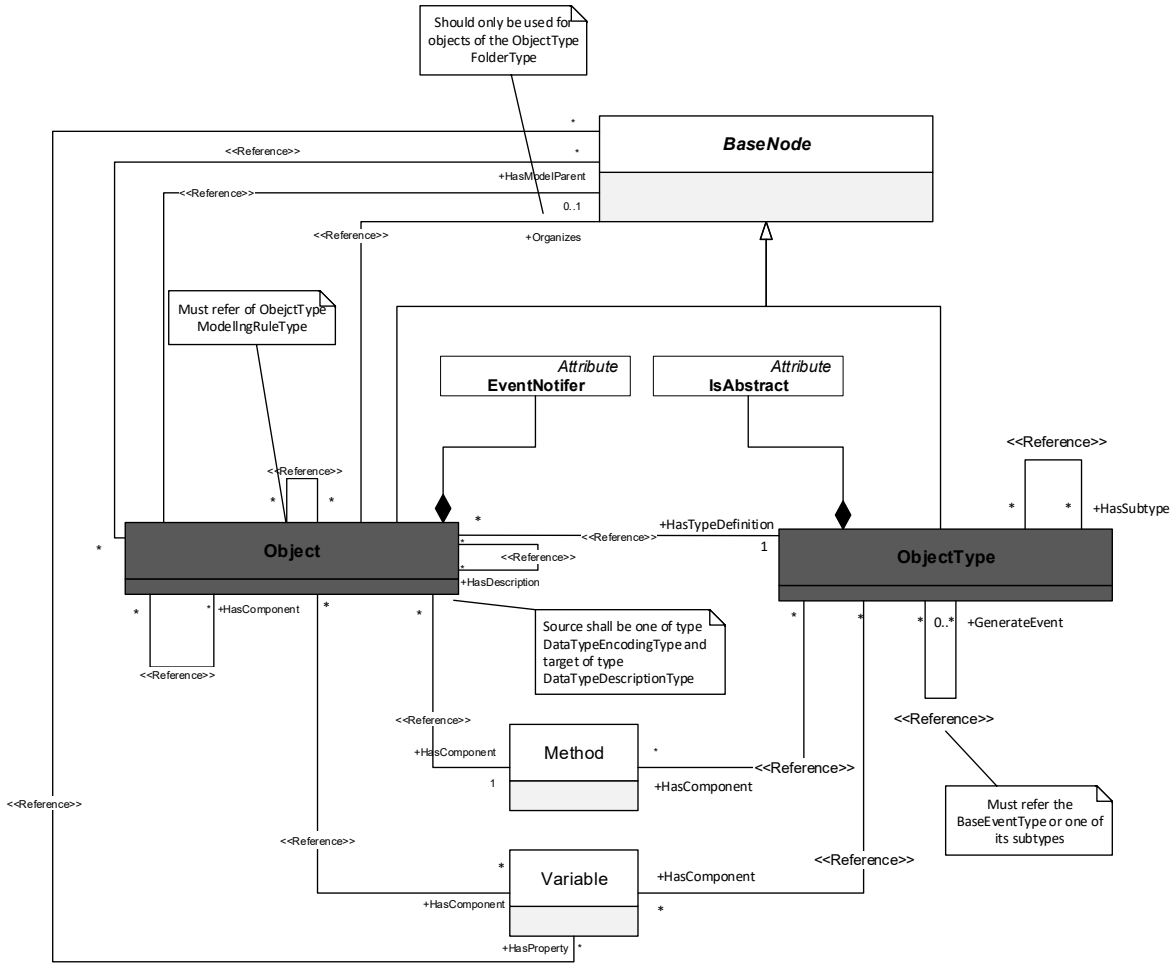


Abbildung 30: OPC UA Objekte und Objekttypen nach (IEC 62541-1)

Referenztypen

Referenzen definieren die Beziehung zwischen verschiedenen Knotenklassen wie Objekten, Variablen und Methoden. Die semantische Bedeutung wird dabei über den Referenztyp festgelegt. OPC UA definiert dabei bereits einige Referenztypen (dargestellt in Abbildung 32), ermöglicht aber auch ein nachträgliches Ergänzen zusätzlicher Referenztypen. Die wichtigsten Referenztypen im Hinblick auf die Modellierung sind:

- HasComponent: Realisiert Aggregation (Teil-von-Beziehungen),
- HasProperty: Verweis auf Eigenschaften des Knotens,
- HasTypeDefinition: Zuweisung von Instanzen von Variablen oder Objekten zu Typen bzw. Klassen,
- HasSubType: Realisiert Vererbungsbeziehung und kennzeichnet das erbendeElement,
- Organizes: Dient zum hierarchischen Ordnen von Elementen (vergleichbar mit einem Ordnerpfad). Ein Element verweist mit Organizes auf ein Subelement der Hierarchie.

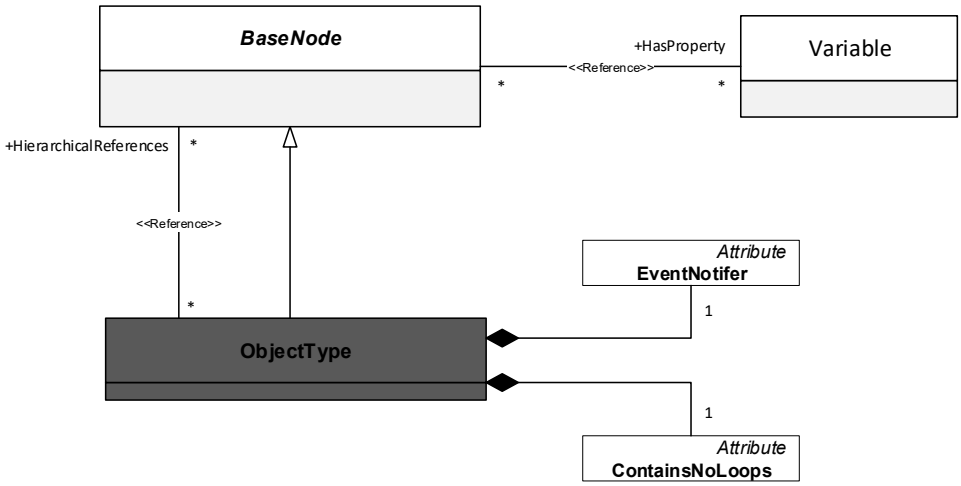


Abbildung 31: OPC UA Sicht auf das Modell nach (IEC 62541-1)

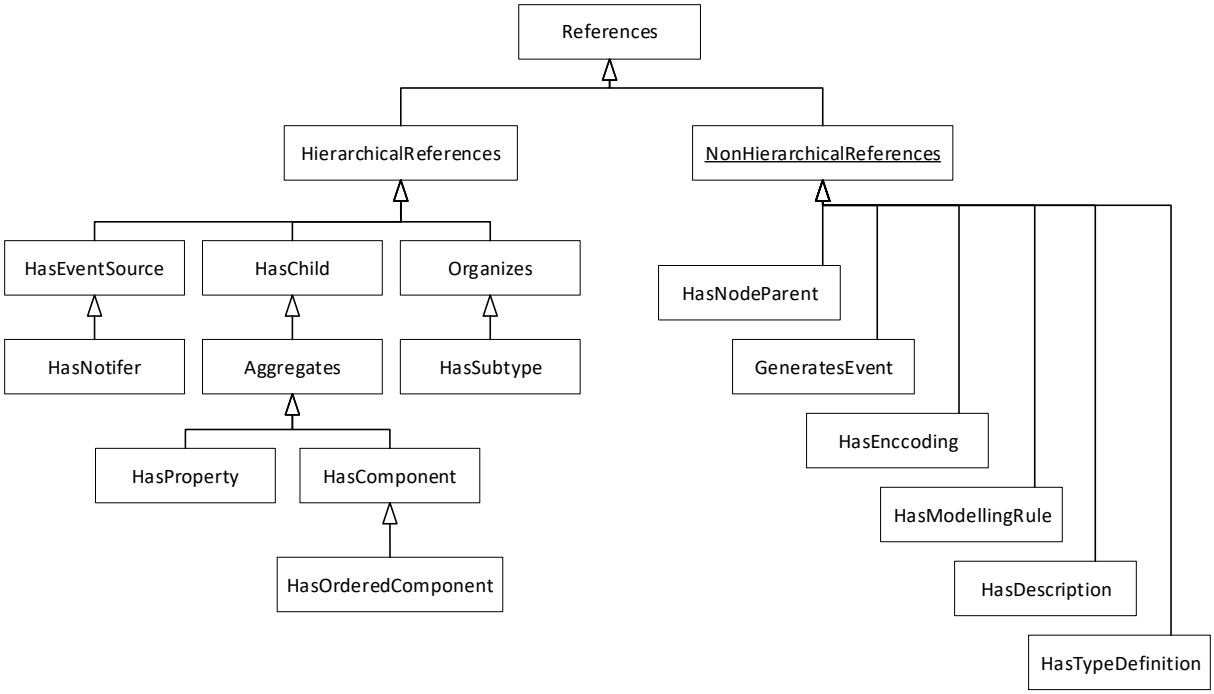


Abbildung 32: OPC UA Referenztypen mit Hierarchie nach (IEC 62541-1)

Methoden

Methoden sind einfache Funktionen, die als zustandsloses Element einem Objekt bzw. Objekttyp zugeordnet werden können. Sie können über die Mitgabe von Input-Daten Output-Daten errechnen. Jede Methode wird im Kontext der aktuellen Sitzung ausgeführt. Wird diese unterbrochen, so endet auch die Methodenausführung ergebnislos. Jede Methode besitzt noch je ein Attribut, um die nutzerunabhängige (Executable) und nutzerspezifische (UserExecutable) Ausführbarkeit der Methode festzulegen. Abbildung 33 stellt diese Zusammenhänge noch einmal dar.

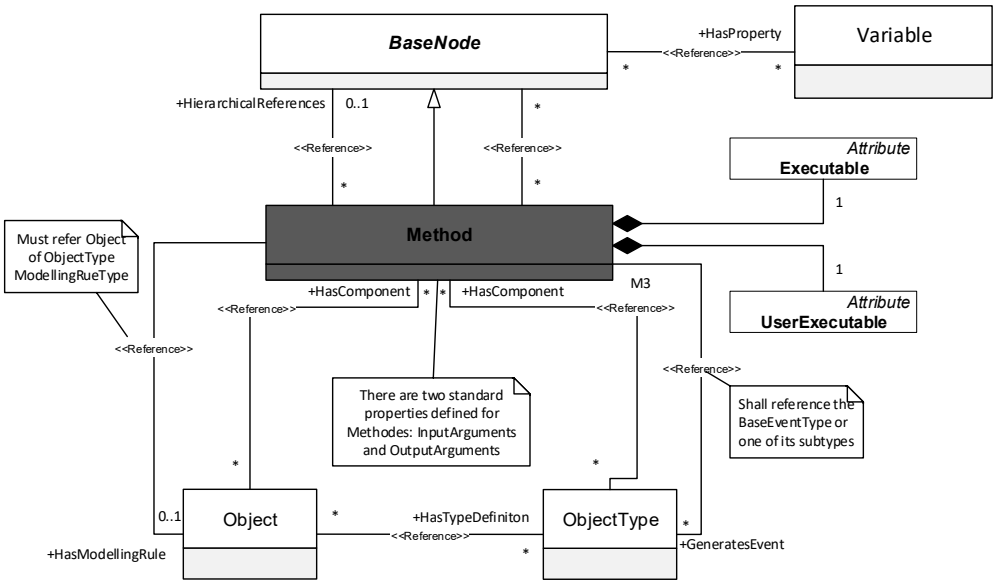


Abbildung 33: OPC UA Method nach (IEC 62541-1)

3.6.4 Merkmalleisten zur Beschreibung von Instrumenten (eClass/NE 100)

Der Produktdatenaustausch zwischen Gewerken und entsprechenden Systemen aus dem Bereich der Prozessleittechnik (Elektro- und MSR-Technik) kann nur dann reibungslos stattfinden, wenn die auszutauschenden Informationen eindeutig definiert und ihre Verwendung geklärt ist. Eine Merkmalsleiste leistet hierzu einen wesentlichen Beitrag.

Bisher wurden die Anforderungen an PLT-Geräte und Systeme durch den Kunden (Anwender) uneinheitlich beschrieben und bei Lieferanten (Herstellern) angefragt. Diese wiederum beschreiben die Geräte anhand ihrer eigenen Dokumentation in verschiedenen Strukturen und Systemen (Papier, Datenbank, CD, elektronischer Katalog). Ähnlich werden auch beim Planungs- bzw. Entwicklungsprozess Geräteinformationen häufig wiederholt in unterschiedliche IT-Systeme eingegeben.

Ein Prozess, der allen Beteiligten die Möglichkeit eröffnet, die im Planungs- und Beschaffungsvorgang benötigten und vorhandenen Informationen nur einmal aufzunehmmen und zur weitern Bearbeitung bereitzustellen, ermöglicht die Arbeit auf das Wesentliche zu konzentrieren. Voraussetzung dazu ist der standardisierter Infomationsaustausch zwischen allen Partner, sowie eine einheitliche Beschreibung der auzutauschenden Objekte.

Die eClass als branchenübergreifender Produktdatenstandard, sowie die Prolist und die NAMUR-Empfehlung NE 100 also branchenspezifischer Merkmalsleiste die 2013 in die eClass aufgenommen wurden (George, 2013), stellen einen solchen Standard zur Verfügung und versetzt somit Lieferanten sowie Kunden von PLT-Geräten und Systemen in die Lage, Abläufe innerhalb ihrer Unternehmen und zwischen den Unternehmen zu optimieren. Engineering-Firmen werden ihrer Position entsprechend als Kunden oder Lieferanten betrachtet (NE 100).

Die auszutauschenden Informationen der zu betrachteten Geräte werden mit Hilfe von Merkmalen beschrieben. Diese Merkmale sind in Merkmalleisten zusammengestellt, die jeweils einen Gerätetyp festlegen. Die eClass enthält sowohl allgemeine Merkmale, die in einer Anfrage oder einem Angebot genutzt werden können, als auch detaillierte Merkmale zur Integration eines PLT-Gerätes, z. B. in CAE-Systeme der Planung, Systeme der Instandhaltung, sowie in Prozessleitsysteme oder ERP-Systeme.

Im Einzelnen stellt sich der Nutzen von standardisierten Merkmalleisten für PLT-Geräte wie folgt dar:

Nutzen beim Kunden:

- Optimierung der Datenintegration, von der Beschaffung über die Planung, Inbetriebnahme bis hin zur Instandhaltung
- Standardisierter und strukturierter Datenaustausch zum Hersteller, z. B. über XML
- Keine Pflege für eigene Formulare und Datensysteme
- Bessere Vergleichbarkeit der Gerätedaten bei Angeboten
- Angebotene Strukturen in ERP-Systemen, Katalog-Systemen, CAE-Systemen nutzbar
- Stammdatenimport direkt vom Gerätelieferant (z. B. für CAE-Systeme und Instandhaltung)

Nutzen beim Lieferanten:

- Optimierung des Prozesses Anfrage/Angebot
- Standardisierter und strukturierter Datenaustausch zum Kunden, z. B. über XML
- Standardisierung der Spezifikationen für Anfrage- und Angebotsprozesse
- Integrierbarkeit der automatisierten Geräteauswahl
- Optimierung des Prozesses "Bereitstellungtechnischer Daten", Datenfluss im internen System
- Optimierung bei der Generierung von Katalogen

Merkmale sind bestimmte Eigenschaften, die zur Beschreibung beispielsweise von Geräten dienen. Zu diesen Eigenschaften zählen u. a. Anforderungen und Rahmenbedingungen, die durch die Umgebung vorgegeben oder zu beachten sind. Dies sind des Weiteren alle technischen Details des Gerätes.

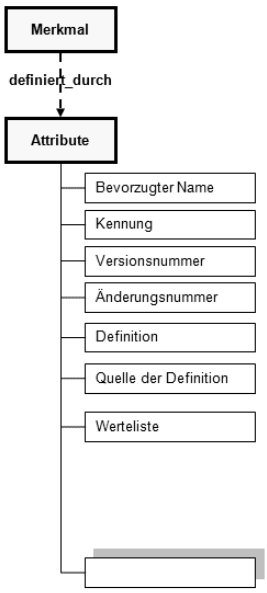


Abbildung 34: Allgemeine Beschreibung eins Merkmales nach (NE 100)

3.6.5 Weihenstephaner Standards

Die Weihenstephaner Standards, ursprünglich für die Lebensmittel- und Verpackungsindustrie entwickelt, definieren einerseits eine Schnittstelle für die Kommunikation verschiedener Maschinen- und Prozesssteuerungen mit einem übergeordneten MES, als auch (branchenspezifisch) die für die Datenerfassung notwendigen Informationen. Durch eine automatisierbare Parametrierung über eine Gerätebeschreibungsdatei wird die Kommunikation nach dem „Plug and Play“ Prinzip ermöglicht. Der Datenaustausch setzt sich aus verschiedenen vereinheitlichten Statuswörtern mit Befehlen (read, write, ...), ergänzt um Datenpunkte (Chargen, Zustände, ...), zusammen und basiert auf dem Ethernet-TCP/IP Protokoll. Das Ziel der Weihenstephaner Standards ist die Schaffung einer standardisierten Betriebsdatenerfassung. Zur Etablierung einer übergeorneten Prozessführungskomponente oder von Mechanismen, die eine Maschine in ein Gesamtsystem integrieren sollen, muss dieser Ansatz erweitert werden. Er eignet sich somit nicht als alleiniger Ansatz für die Umsetzung eines Plug-and- Produce. (Voigt & Kather, 2005)

3.7 Zusammenfassung und Handlungsbedarf

Die Aufstellung der untersuchten Beschreibungsmittel zeigt die Vielzahl von verfügbaren und zum Teil etablierten Beschreibungsmitteln in den verschiedensten Aspekten der Integration eines Moduls in eine übergeordnete Prozessführungsebene. In Tabelle 1 werden die untersuchten Technologien zunächst entsprechend als *Beschreibungsmittel*, *Methode* oder *Werkzeug* eingeordnet und weiterhin dem Umfeld ihrer Verbreitung zugeordnet.

Tabelle 1: Zusammenfassung der betrachteten Beschreibungsformate und Modelle

	BESCHREIBUNGSMITTEL	METHODE	WERKZEUG	STANDARD	INDUSTRIE	ACADEMIC	VERRIEGELUNGEN & ABLÄUFE	HMI	DIAGNOSEDATEN	ALARMIERUNGSFUNKTIONEN	PROZESSWERTE	BETRIEBSDATENERFASSUNG
PLCOPENXML	+			+			+					
PNML	+					+	+					
GSDML	+			+							+	
EDDL	+			+							+	
WEIHENSTEPHAN	+	+	+	+								+
AUTOMATIONML/CAEX	+		+	+			+	+	+	+	+	+
OPC UA	+	+		+			+		+	+	+	+
FDI		+	+	+					+		+	
IEC 61512/ISA 88		+		+			+					
ISA 106		+		+			+					
IEC 61131	+			+			+				+	
VDI 3699		+		+				+				
NE 107		+		+					+			
XVCML	+				+			+				
XAML	+				+			+				
UIML	+					+		+				
XIML	+					+		+				
AUTOHMI	+		+			+		+				
GRAPHML	+			+				+				

Alle als Standard oder Empfehlung (z.B. IEC, ISA, DIN, NAMUR) veröffentlichten Technologien werden in der Spalte *Standard* zusammengefasst. Technologien welche im industriellen Umfeld etabliert sind, werden der Kategorie *Industrie* zugeordnet. Technologien welche sich zurzeit in Entwicklung befinden oder durch entsprechende Forschungseinrichtungen veröffentlicht wurden, werden der Kategorie *Forschung* zugeordnet. Auf der rechte Seite der Tabelle werden die untersuchten Technologien hinsichtlich ihres Integrationsaspektes klassifiziert. Aus der Aufstellung geht hervor, dass für die Beschreibung der Integrationsaspekte eine Vielzahl von Beschreibungsmitteln zur Verfügung stehen. Spezifische Formate wie EDDL oder PLCopenXML sind sehr gut anerkannt für ihren jeweiligen Integrationsaspekt, großer Vorteil hierbei sind die existierenden Schnittstellen in den Prozessleitsystemen. Jedoch bedarf es für jeden Integrationsaspekt ein anderes Beschreibungsmittel sowie einen geeigneten Container. Allgemeine Formate wie beispielweise AutomationML oder OPC UA decken einen großen Bereich der Integrationsaspekte ab, sind jedoch in vielen Aspekten sehr allgemein gehalten, so dass entsprechende Schnittstellen noch auf die spezifische Modellierung beispielsweise eines Bedienbildes angepasst werden müssen.

Die zukünftige Weiterentwicklung der verwendeten Komponenten bei der Integration eines Moduls, wie die eingesetzte Software und Hardware in Prozessleitsystem und im Modul, machen die Verwendung eines flexibel erweiterbaren Formates entweder als Container mit mehreren spezifischen Formaten oder eines einheitlichen erweiterbaren Formates notwendig. Diese Erweiterbarkeit muss auch auf Ebene der Integrationsaspekte möglich sein, so dass in Zukunft noch weitere Aspekte berücksichtigt werden können.

4 Plug and Produce für modulare Anlagen

4.1 Integrationsarchitektur

Eine zentrale Herausforderung der Modularisierung ist das Engineering, das eine möglichst kostengünstige und schnelle Planung, Aufbau beziehungsweise Umbau modularer verfahrenstechnischer Produktionsanlagen realisieren soll. Bei bereits etablierten Prozessleitsystemen lässt sich eine geringe Flexibilität im Betrieb modularer Anlagen und im verteilten Engineering feststellen. (Urbas, 2014).

Grundgedanke einer modularen Anlagenarchitektur ist die Minimierung des Aufwands während der Integration eines Moduls in die Anlage. Dies kann durch die Verlagerung geeigneter Aufwände in das Engineering des Modullieferanten erreicht werden. Das Engineering der modularen Gesamtanlage zerlegt sich in zwei im Allgemeinen voneinander getrennte Engineering-Prozessen, mit der jeweiligen Verantwortung des Modulherstellers und des Anlagenbetreibers. Es ist des Weiteren anzunehmen, dass der Integrationsaufwand während des Gesamtanlagen-Engineerings mit zunehmender Intelligenz des Moduls abnehmen wird, da die ausschließlich das Modul betreffenden Aspekte schon im entsprechenden Softwarecode der Modulsteuerung umgesetzt sind und somit nicht in das übergeordnete Prozessleitsystem integriert werden müssen. Intelligenz wird hier, in Anlehnung an (Stern, 1935), definiert, als die Fähigkeit eines Moduls seinen inneren Zustand zu erkennen und in Grenzen zu beherrschen und damit ihre Integrität zu schützen, sowie die Fähigkeit, ihren Zustand nach außen zu kommunizieren, soweit gewünscht und erforderlich für einen Betrieb im Verbund mit anderen. Denkbar sind darüber hinaus Ansätze, bei denen Module über eine eigene Beschreibung verfügen und so die Integration unterstützen können (Kainz et al., 2013). Der Schwerpunkt liegt hierbei weniger in der Übertragung der Daten als in der geeigneten Modellierung der Information. Ein Modul ist somit ähnlich der bereits formulierten Modulvariante 2 bzw. 3 der (NE 148) mit dezentraler Intelligenz ausgestattet.

In Analogie zur physikalischen Modularisierung (Hady & Wozny, 2012) entspricht die funktionsgerichtete Modularisierung einem dienstbasierten Ansatz. Ein Modul stellt hierfür seine verfahrenstechnische Funktion als Dienst einem übergeordneten Prozessleitsystem zur Verfügung. Folgende Grundfunktionalitäten muss ein Modul des Prozessleitsystems unterstützen:

- Bedienen und Beobachten (B&B)
- Melden und Protokollieren
- Steuern und Überwachen

Zur Erfüllung dieser Funktionalität müssen während des Integrations-Engineering folgende Aspekte und Aufgaben realisiert werden:

- Netzwerk-Engineering - mit dem Ziel der Abbildung des physikalischen Kommunikationssystems und Ermöglichen der Parametrierung
- Realisierung von Teilen der Koordinierungs- und Prozedursteuerung zum zeitgerechten Abrufen und Überwachen (Orchestrierung) der Moduldienste
- HMI-Engineering zur Realisierung der Bedien- und Beobachten-Funktionalität

Wie bereits erläutert, sind die Engineering-Prozesse von Modulhersteller und Anwender voneinander entkoppelt (Abbildung 35).

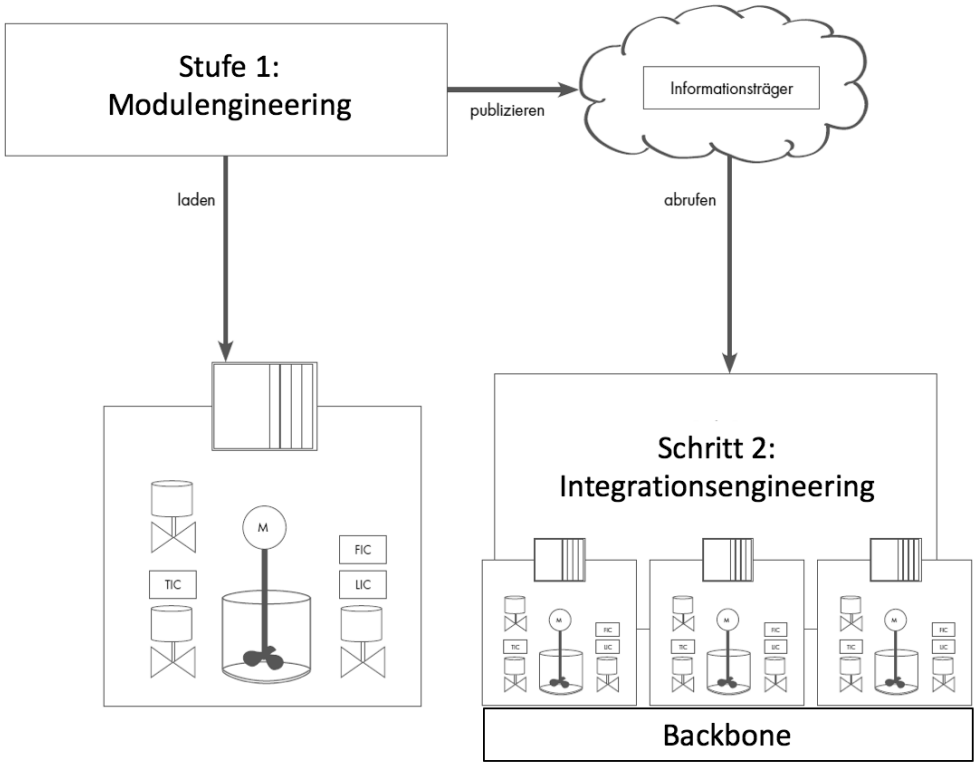


Abbildung 35: Engineering Prozess einer modularen Anlage mit dezentraler Intelligenz (Obst et al., 2015b)

Unabhängig vom Engineering der Gesamtanlage führt der Hersteller des Moduls das Modul-Engineering durch. Dies umfasst unter anderem das Erstellen und Laden des lauffähigen Softwarecodes der Modulsteuerung. Zur Übertragung der zur Integration notwendigen Ergebnisse des Modul-Engineerings werden diese in einem Informationsträger gespeichert und dieser veröffentlicht. Die benötigte Information wird so während des Integrations-Engineerings abrufbar und verwendbar.

Dem Bedarf nach einer Bedien- und Beobachtbarkeit des über mehrere Module verteilten Prozesses wird mit der Nutzung eines SCADA-Systems (als Teil eines PLS oder als Stand-alone-System) entsprochen. Die zentrale Herausforderung hierbei ist die Realisierung des nach (NE 148) formulierten einheitlichen Look-and-Feel einer modularen Anlage.

Das Bedienbild eines Moduls wird durch den Modulhersteller zur Verfügung gestellt. Die Kenntnis der in industriellen Anlagenprojekten meist projektspezifisch verwendeten vereinheitlichten Bedienbildbibliothek des übergeordneten Systems hat er zu diesem Zeitpunkt jedoch noch nicht, die Generierung dieser im Leitsystem der Gesamtanlage kann erst während des Gesamtanlagen-Engineerings erfolgen. Zur Umsetzung der modulspezifischen Bedienbilder in solche mit projekteinheitlichen Bedienbildelementen müssen die Bedienbilder in einer darstellungsunabhängigen Beschreibungsform vorliegen. Die Layout- und Rolleninformationen sind im Anschluss durch einen Algorithmus zugänglich, der die projektabhängigen Bedienbildelemente in gewünschter Darstellung und Lage auf das Bedienbild setzt und mit den Daten der Module verknüpft (Urbas & Doherr, 2011).

Für die datentechnische Verknüpfung eines Bedienbildelementes mit den Variablen des Modul-SPS-Codes muss ebenfalls ein zuvor abgestimmter Informationsraum vorliegen. Dazu arbeitet der Namur AK 1.12 an einer Bibliothek mit dem Ziel, die darzustellenden Informationen hinter einem Bedienbildelement durch eine harmonisierte eindeutige Merkmalskennung zu annotierten, um in allen beteiligten Automatisierungssystemen den gleichen Inhalt darzustellen.

Alle Informationen die im Modul-Engineering erarbeitet und während des Gesamtanlagen-Engineerings benötigt werden, sind in einem instanzspezifischen Informationsträger hinterlegt.

4.2 Informationsmodellierung für die Modul-Integration

Bei modularen Anlagenarchitekturen muss die prozesstechnische Funktion der Gesamtanlage auf mehrere Module aufgeteilt werden. Wesentliche Anteile des Verhaltens der Anlage, also der Verfahrenstechnik, der Prozessführung und des Zeitverhaltens, sind somit direkt durch die Module und deren Aufbau bestimmt. Die Aspekte (Prozessgrafiken, Verriegelungs-, Steuerungs- und Regelungsstrukturen), die während des Integrations-Engineerings von Modulen in ein übergeordnetes Prozessleitsystem benötigt werden, sind in der NE 148 (NE 148) im Abschnitt „Datenaustausch und Informationsschnittstellen“ definiert (siehe Kapitel 3).

Für viele Anwendungsfälle ist nur ein Teil der dort genannten Funktionen Auswerten, Archivieren, Bedienen, Anzeigen, Überwachen und Alarmieren, Erfassen, Steuern, Regeln, Rezepte ausführen und Rezepte erstellen notwendig: Für eine zustandsbasierte Prozessführung, bei der die verfahrenstechnischen Funktionen eines Moduls in einem Dienst gekapselt wird und somit einer Service-Orientierten Architektur (SOA) entspricht, ist die Beschreibung der Dienste und deren Kommunikationsvariablen (SFC nach IEC 61131, Batch nach IEC 61512/ISA S88, Konti nach ISA 106) erforderlich. Zur Realisierung der Bedien- und Beobachtbarkeit muss die abstrakte Beschreibung der Mensch-Maschine-Schnittstelle (Mimics, Trends, Key-Performance-Indikatoren) in das HMI-System eingelesen und ausgewertet werden. Sind Aspekte erforderlich, die über die Realisierung des Produktionsprozesses hinausgehen, müssen Alarmmanagement, Archivierung sowie Diagnose und Instandhaltung betrachtet werden. Auch sind weitere funktionale Teilmodelle denkbar. Dies beinhaltet

beispielsweise Simulationsmodelle für die virtuelle Wasserfahrt, Stücklisten der verbauten Geräte und Apparate bis hin zu Hard- und Softwareaspekten integrierter Fehlerausbreitungsgraphen für eine modulübergreifende Fehleranalyse, siehe (Morozov & Janschek, 2011).

In den folgenden Abschnitten werden die einzelnen Aspekte der Modulintegration näher beleuchtet. Dabei werden zunächst die zu integrierenden Inhalte untersucht, im darauffolgenden Abschnitt folgen exemplarische Modellierungen der einzelnen Aspekte.

4.2.1 Integration automatisierungstechnischer Verriegelungen & Abläufe

Ablaufsteuerungen ermöglichen die Verarbeitung sequentieller oder paralleler Funktionen eines zeit- oder ereignisdiskreten Prozesses. Sie werden zum Koordinieren von kontinuierlichen Funktionen eingesetzt und steuern komplexe Prozesse. Diese bestehen im Allgemeinen aus einer oder mehreren Schrittfolgen. Die technische Umsetzung wird mithilfe der Sequential Function Charts (SFC) realisiert. Eine Schrittfolge besteht immer aus einem Start- und einem Endschritt. Dazwischen befinden sich Schrittfolgen, bestehend aus Aktionsschritten und Kanten.

Zur formalen Beschreibung von Ablaufsteuerungen werden neben der Ablaufsprache nach IEC 61131 auch Zustandsdiagramme oder Petri-Netze verwendet. Ein Vorteil der Zustandsdiagramme ist deren einfache Möglichkeit zur Fehlerdiagnose, sowie die einfache Transformation in bereits existierende Programmiersprachen. Jedoch funktioniert die Beschreibung mit Hilfe der Zustandsdiagramme ausschließlich für einfache oder alternativverzweigte Sequenzen, nicht für parallele, da sich das Diagramm immer in genau einem Zustand befinden muss.

Betrachtet man nun die Integration der Ablaufsteuerung eines Moduls in ein übergeordnetes Leitsystem, werden folgende Informationen während des Betriebes eines Moduls benötigt:

- Strukturinformationen zur Visualisierung der Ablaufkette
- Anzahl der Plätze, Transitionen,
- einheitliches Bezeichnungsschema,
- Strukturinformationen zur Verzweigungen und Zusammenführung, Art der Verzweigung
- Informationen über Verklemmungen
- Informationen über aktive Plätze und Transitionen

Die Struktur einer Ablaufsteuerung lässt sich herstellerneutral mithilfe eines bipartiten Graphens darstellen (siehe Abbildung 36). Hierbei spielt die aktuell gewählte Bezeichnung nur eine nebensächliche Rolle, da sich die Struktur des Graphen eindeutig mit Verknüpfungsmatrizen beschreiben lässt. Daraus entstehen zwei Matrizen, die sich allein auf der Grundlage ihrer binären Werte 1:1 zum abgebildeten Graphen wiederherstellen lassen. Eine Matrix beschreibt die Beziehung jeder Transition zu jedem Platz, eine zweite die Beziehung vom Platz zur Transition. Sobald eine direkte Verbindung vorhanden ist, wird eine 1 in der Matrix hinterlegt, andernfalls erscheint eine 0. Eine weitere

Informationsquelle innerhalb dieser Matrizen stellt die Aufeinanderfolge von mehreren Plätzen nach einer Transition dar, beziehungsweise der Folge ein und derselben Transition nach zwei unterschiedlichen Plätzen (blau markierter Bereich). Daraus lässt sich für die Parallelverzweigung folgende Regel ableiten: Folgen aus einer Transition zwei Plätze, öffnet sich die Parallelverzweigung, folgt aus zwei Plätzen dieselbe Transition, schließt sich die Verzweigung. Aufgrund dieser Regel lässt sich ohne Übermittlung zusätzlicher Informationen der Graph aus einfachen Matrizen wiederherstellen. Ein wesentlicher Vorteil dieser Darstellung gegenüber anderen, wie beispielsweise der Adjazenzmatrix ist die ausschließliche binäre Kodierung.

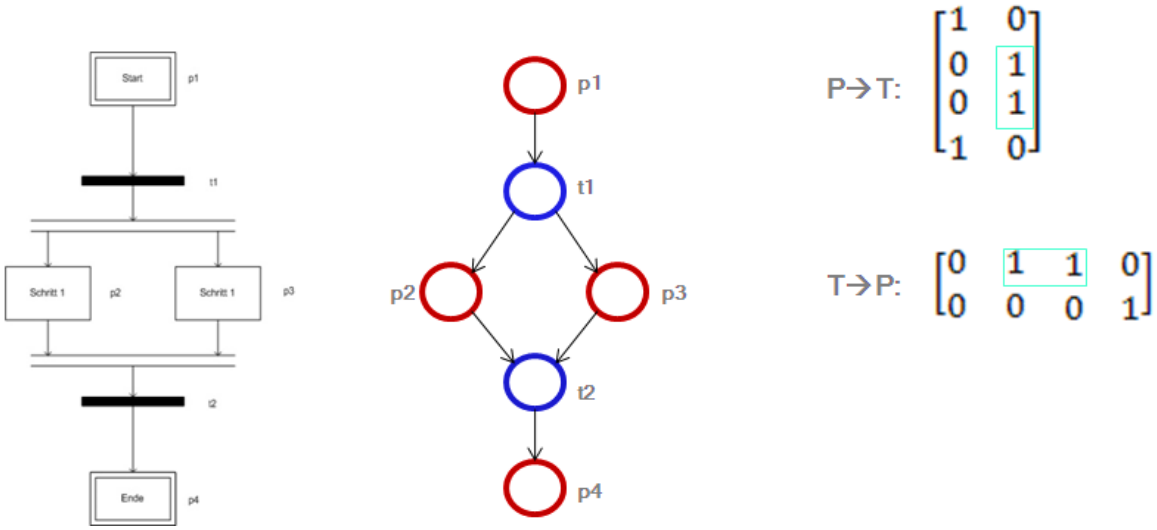


Abbildung 36: Darstellung einer Ablaufkette durch einen Graphen und Matrix (Obst et al., 2014b)

Die Beschreibung der aktivierten Schritte kann über zwei binären Vektoren mit Lese- und Schreibrechten übermittelt werden. Je ein Vektor beschreibt dabei die aktiven Plätze und einer die Transitionen. Mit Hilfe eines "Manual Override" kann der Operateur mit den entsprechenden Berechtigungen, bei Verklemmungen aktiv eingreifen und die Transition manuell weiterschalten.

Für den Betreiber eines Moduls ist es notwendig, während des Betriebes der Anlage die Verriegelungen zu beobachten und gegebenenfalls zu bedienen. Zur Sicherung seines Know Hows wird der Modul Hersteller jedoch nicht in allen Fällen einen vollständigen Zugriff auf die genaue Implementierung zulassen wollen. Ein häufig eingesetztes Beschreibungsmittel der Verriegelungen ist die prozessleittechnische Ursachen-Wirkungs-Matrix. Diese Matrix verbindet binäre Größen, die eine Aussage über den Zustand einer technischen Einrichtung oder des Prozesses erlauben – die Ursache - mit binären Größen, die einen geeigneten Stelleingriff in den Prozess zur Folge haben – der Wirkung. Eine Ursache kann mehrere Wirkungen zur Folge haben, ein und dieselbe Wirkung kann von verschiedenen Ursachen ausgelöst werden. Einige Hersteller von Prozessleitsystemen bieten für ihre Systeme Safety-Matrix-Werkzeuge an, beispielsweise (Yokogawa Europe, 2015) oder (Siemens AG), mit denen diese Methode nahtlos in das prozessleittechnische Engineering integriert werden kann. Eine solche Matrix kann mit Inhalten unterschiedlichen Datentyps, Länge und Zugriffsrechten in einem

Datenpaket übertragen werden. Doch was genau ist eine Safety-Matrix und wie unterscheidet sich diese von einer Matrix zur Beschreibung von Struktur der Ablaufsteuerung? Auch diese Matrix hat eine eindeutige Bezeichnung und besteht aus einer bestimmten Anzahl an Spalten und Zeilen. Diese Informationen müssen wie auch schon bei der Ablaufsteuerung separat übergeben werden. Den einzigen Unterschied stellen die Inhalte dar. Während bei der Ablaufsteuerung nur binäre Inhalte übergeben werden, muss an dieser Stelle jeder Inhalt einzeln innerhalb eines Strings weitergeleitet werden.

4.2.2 Zustandsorientierte Prozessführung einer modularen Anlage

In den beiden vorangegangenen Abschnitten wurde die Integration der in einem Modul realisierten Ablaufketten und Verriegelungen betrachtet. In Zukunft ist davon auszugehen, dass die prozesstechnische Funktion bei modularen Anlagenarchitekturen in den jeweiligen Modulen gekapselt ist. Die gewünschte Wandlungsfähigkeit dieser Produktionsanlagen bedingt auch die Kommunikation in geeigneter Weise zu kapseln. Die Prozessführung dieser Module sollte aus diesem Grund über eine abgesetzte Dienste-Schnittstelle erfolgen. Diese folgt den Grundsätzen, die durch die Oasis-Referenz-Architektur Foundation (Brown et all, 2012) formuliert wurden. Die Dienste werden dabei über zuvor definierte Zustände und Zustandsübergänge gesteuert und überwacht (Beispiel siehe Abbildung 37). Die Kommunikation bedingt zwischen PLS und Modul Kommunikationsvariablen, die die Aufforderung zur Erreichung eines der Zustände, oder die das Ergebnis über das Erreichen eines Zustandes abbilden. Die Kapselung der prozesstechnischen Funktion in einem Dienst muss allerdings anpassbar sein, um die erforderlichen produkt- und prozessspezifischen Gegebenheiten zu ermöglichen. Aus diesem Grund sind Parameter zur Anpassung des Dienstes vorzusehen. Mit ihnen können Sollwerte vorgegeben und aktuelle Istwerte abgefragt werden. Die Parameter müssen dazu sowohl Grenzwerte, also die Spanne zwischen Minimum und Maximum, in der ein gefahrloser und zerstörungsfreier Betrieb möglich ist, beinhalten, als auch deren Einheiten. Dazu sind ebenfalls Kommunikationsvariablen vorzusehen.

Durch Kombination mehrerer Module in einer Anlage entsteht die Notwendigkeit, die Dienste der angeschlossenen Module in eine für die Produktion des gewünschten Produktes erforderliche geordnete Folge zu bringen. So muss zum Beispiel bei einem kontinuierlich betriebenen Reaktionsprozess das Anfahren des Reaktors mit dem Vorliegen der Ausgangsprodukte abgestimmt werden. Da diese zusätzliche Orchestrierungsfunktion erst durch Kombination verschiedener Module notwendig wird, muss dies von einer noch während des Integrations-Engineering zugänglichen Automatisierungsinstanz, beispielsweise dem übergeordneten Leitsystem, übernommen werden. Um die Dienste modulübergreifend orchestrieren zu können, ist die Kenntnis der aktuellen Zustände, wie *startend*, *laufend* oder *gestoppt*, der Module notwendig. Diese Information wird durch die dezentrale Intelligenz eines jeden Moduls ermittelt und über eine Kommunikationsschnittstelle zugänglich gemacht. Die Definition der Zustände muss hersteller- und modulunabhängig und somit einheitlich über alle Module hinweg geschehen. In Abbildung 37 sind exemplarisch mögliche Zustände sowie deren Zustandsübergangsmodell in Anlehnung an (DIN EN 61512-1) aufgeführt.

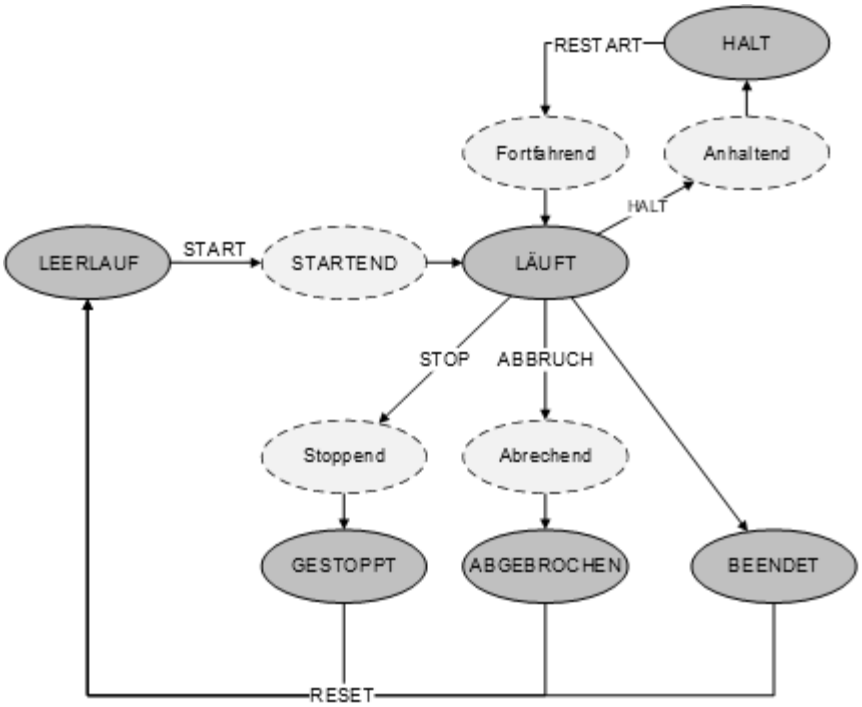


Abbildung 37: Zustände und Zustandsübergangsmodell exemplarisch nach (DIN EN 61512-1)

Die Orchestrierung der Dienste entspricht einer Prozedursteuerung gemäß (DIN EN 61512-1). Diese bestimmt, dass "Aktionen in einer geordneten Folge stattfinden, damit eine prozessorientierte Aufgabe durchgeführt wird." (DIN EN 61512-1). Zur Realisierung der Orchestrierungsfunktionalitäten bringen heutige Batch-Werkzeuge mit der Umsetzung der DIN EN 61512 Definitionen grundsätzlich alle erforderlichen Voraussetzungen mit, unabhängig davon, ob der zu steuernde Produktionsprozess kontinuierlicher oder diskreter Natur ist (Hawkins, Brandl & Boyes, 2010).

4.2.3 Bedienen und Beobachten

Die VDI-Richtlinie 3699 (VDI/VDE 3699-1, S. 9) beschreibt die Prozessführung als „eine Aufgabe des Operators mit dem Ziel, den bestimmungsgemäßen Betrieb einer verfahrenstechnischen Anlage wirtschaftlich und umweltverträglich durchzuführen. Ergänzend dient dem sicheren Betrieb das Schutzsystem, das der Operator nicht beeinflussen kann.“ Die nutzerfreundliche Gestaltung der Mensch-Maschine-Schnittstellen der Prozessleitsysteme (PLS) stellt somit einen wichtigen Faktor für den wirtschaftlichen Betrieb von Anlagen dar. Durch die Heterogenität der Engineering-Werkzeuge bei der Erstellung der modulspezifischen Bedienbilder besteht das Ziel eines Informationsträgers, diese Bedienbilder in einer neutralen Beschreibungsform abzubilden. Weiterhin muss den spezifischen Darstellungsmöglichkeiten der Bedien- und Beobachtungsysteme sowie projektspezifischen Anforderungen an die Darstellung einzelner Elemente eines Bedienbildes Rechnung getragen werden. Ebenfalls muss bei der Integration von Modulen unterschiedlicher Hersteller eine einheitliche Darstellung der Bedienelemente über die Modulgrenzen hinweg gewährleistet sein (siehe Abbildung 38). Daher setzt das Konzept auf eine rollenbasierte Beschreibung der einzelnen Bedienelemente (siehe Abbildung 39).

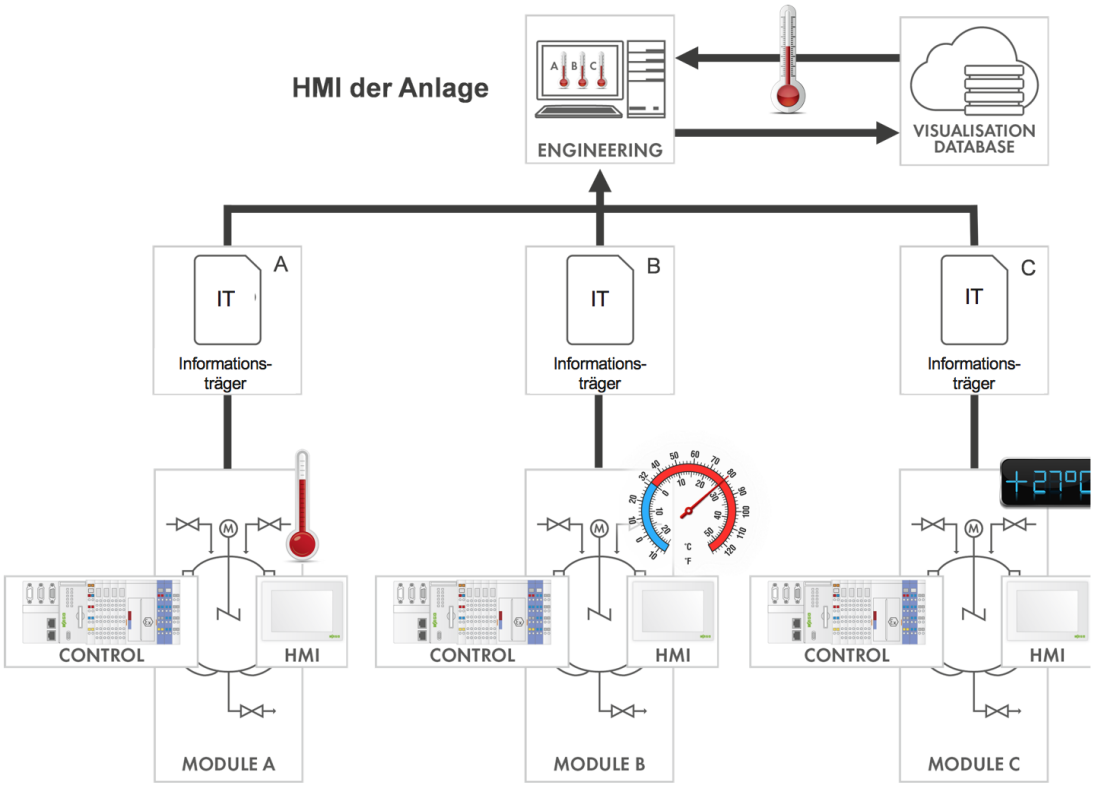
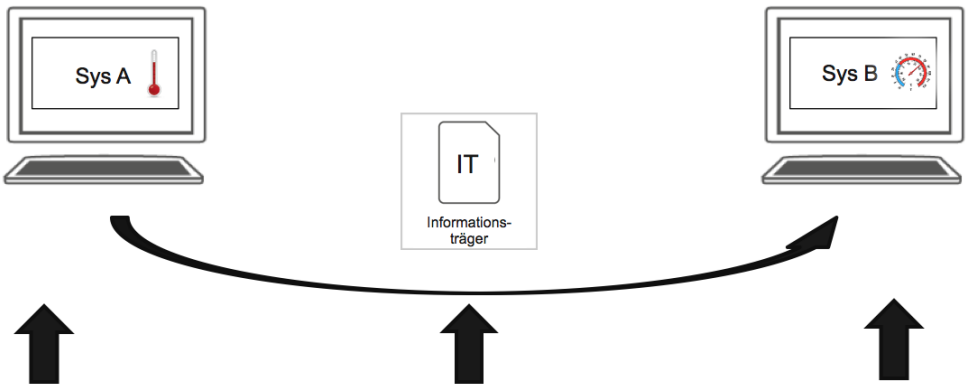


Abbildung 38: Einheitliches Bedienen und Beobachten unabhängig vom Quellsystem des Modulherstellers (Obst et al., 2015b)



Systemübergreifende HMI-Bibliothek

Rolle	Beschreibung	Parameter	Referenz (DIN IEC 60050-351)
NA_Temp01	Temperatur-Messstelle	Name	351-41-01
		Istwert	351-41-02
		Sollwert	351-41-03
		Messbereich	351-48-11
NA_Pump01	Pumpen-Faceplate	Name	351-41-01
		…	…

Abbildung 39: Rollenbasiertes Bibliothekskonzept zur HMI Integration (Obst et al., 2015b)

Ziel und Voraussetzungen

Sowohl auf Seiten der Bedienbilder-Erstellung beim Modulhersteller (Sys A) als auch beim Bedien- und Beobachtungsystem des Modul-Betreibers (Sys B) müssen Bibliotheken zum Einsatz kommen, die um eine semantische Bedeutung der einzelnen Bedienelemente ergänzt wurden (siehe Abbildung 39).

Dazu muss in einem Harmonisierungsprozess ein minimales Set an Elementen identifiziert werden, die zum Bedienen mehrerer Module notwendig sind. Dabei bedarf es einzig der Abstimmung, welche Elemente (Ventil, Antrieb, Messstelle, …) mit welchen Informationen (Sollwert, Istwert, …) es zu übertragen gilt. Keineswegs müssen die graphischen Darstellungen des Bedienelementes oder das Interaktionsverhalten harmonisiert werden. Dies ist und bleibt Differenzierungsmerkmal der integrierenden Systeme.

Abstrakte Beschreibung von Bedienbildern

Bedienbilder der Prozessindustrie bilden implizite und explizite Strukturinformationen des geführten Prozesses und der Automation ab. Die einzelnen visuellen Elemente, die Bedienbild-Elemente, sind sichtbar über Rohrleitungen und Wirklinien miteinander verknüpft, die aus der technologischen Kompositionshierarchie ableitbaren ist-Teil-von-Relationen der Elemente werden üblicherweise indirekt durch grafische Mittel des Layouts wie örtliche Nähe und regelmäßige Anordnung dargestellt. Die direkten Strukturmittel sind häufig so stark ausgeprägt, dass die Autoren von (Hoernicke, Christiansen & Fay, 2014) einen Ansatz vorstellen, wie die HMI in Brownfieldprojekten, in denen keine anderweitigen Informationsquellen zur Verfügung stehen, ausgewertet werden können, um die Anlagentopologie zu rekonstruieren. Die konkrete Ausprägung des Layouts durch 2-dimensionale x,y-Koordinaten ist jedoch sehr stark abhängig von der Implementierungsplattform. Um die geforderte Technologieunabhängigkeit zu wahren, sollen die Beziehungen zwischen den Elementen eines Bedienbilds in dem Informationsträger mathematisch-abstrakt als Graph, bestehend aus Knoten und Kanten, beschrieben werden. Die x,y-Koordinaten der Bedienbild-Elemente im Modul-Engineeringsystem werden als Attribute an den Knoten mitgeführt und können vom Integrations-Engineeringsystem als Layouthinweise verwendet werden. Im einfachsten Fall, wenn die Bildschirmgröße und Bedienbild-Elementgröße von Quell- und Zielsystem weitgehend übereinstimmen, können diese Layouthinweise direkt übernommen werden.

Die Beschreibung selbst muss in der Lage sein, mehrere Bilder darzustellen. Jedem dieser Bilder sollen als Parametersatz die Maße für Höhe und Breite, der beim Engineering verwendeten Pixel mitgegeben werden. Es soll aber bereits der Fall berücksichtichtig werden, dass diese Daten aufgrund ausgefeilter Routing-Algorithmen nicht mehr benötigt werden. Ausgehend von den in der (VDI/VDE 3699-3, S. 15) erwähnten Ebenen sollen drei Bedienbildebenen vorgesehen werden. Das oberste Level siedelt sich im Produktionsstrang an und soll kein komplettes Bedienbild, sondern nur ein Teilbild mit den wichtigsten Daten liefern. Weitere Anforderungen werden für dieses Bedienbild nicht definiert. Die nächstniedrigere

Ebene umfasst das Modul selbst und soll alle darin ablaufenden Prozesse in einer Übersicht darstellen. Üblicherweise handelt es sich hierbei um ein an das R&I Fließbild angelehntes Bedienbild, soll aber nicht darauf beschränkt sein. In der nächst-niedrigeren Ebene ist es vorgesehen, Teilbilder des Moduls für einzelne Equipment-Module oder andere Teil-Funktionen einzugruppieren, ebenso können sich auch Übersichtsbilder zu den Diensten einordnen. Um die Bilder unterscheiden zu können, ist Ihnen ein eindeutiger Name zu geben, der auch die Ebene der Darstellung beinhaltet. Weiterhin sollten Verknüpfungen zwischen den Bildern möglich sein und Verbindungen zwischen Bildobjekten in Bildern darstellbar sein. Sind Positionsdaten vorhanden, müssen diese angegeben werden.

Bei Bildobjekten wird davon ausgegangen, dass ihre zugehörigen Symbole stellvertretend durch Rechtecke beschrieben werden können. Die Objektkoordinaten müssen an der oberen linken Ecke eines solchen Rechtecks abgelesen werden und beinhalten die Platzierung innerhalb der Ebene (weiter vorn oder hinten). Es ist bereits hier zu beachten, dass diese Rechtecke nur als Platzierungsempfehlung dienen, vom Zielsystem aber nicht immer so umgesetzt werden können. Eine Orientierung für Bildobjekte muss nicht, kann aber mitgegeben werden. Sie soll sich hauptsächlich aus den Anschlüssen und Verbindungen ergeben.

Verbindungen zwischen Bildobjekten müssen an sogenannten Ports festgemacht werden. Diese Ports werden pro Bildobjekt definiert und stellen somit ihre Anschlüsse dar. Eine Positionierung der Ports innerhalb eines Bildobjektes muss durch absolute als auch relativer Koordinaten ermöglicht werden. Um eine Orientierung für Bildobjekte lesen zu können, müssen die Ports zusätzliche Informationen über ihre Verwendung als Ein- und/oder Ausgang besitzen. Weiterhin soll es möglich sein, Verbindungen den Typ des zu übertragenden Mediums zuzuweisen und die Art des Anschlusses festzulegen. So besitzt beispielsweise ein Druckluftventil nicht nur einen Ein- und Ausgang für das zu steuernde Medium, sondern auch noch mindestens einen Eingang für Druckluft. Es ist darauf hinzuweisen, dass nicht alle verfügbaren Anschlüsse auch zwingend dargestellt werden müssen.

Verbindungen sollen ebenfalls in der Lage sein, Routing-Informationen mitzuliefern. Diese Informationen werden als simple zusätzliche Eckpunkte angegeben. Sie sollen dabei in exakter Reihenfolge von Start- nach Endpunkt angegeben werden, diese beiden Punkte selbst aber nicht enthalten. Verbindungen sollen sowohl in gerichteter als auch in ungerichteter Form möglich sein.

Kommen in den Verbindungen Zweigstellen vor, so werden diese als extra Bildelemente mit einer entsprechenden Anzahl an Ein- und Ausgängen modelliert werden, um die grafische Überlagerung von Verbindungen zu vermeiden.

Sämtliche Kommunikationsvariablen die im Bedienbild benötigt werden, müssen als Referenz auf die im Informationsträger festgelegten Kommunikationsvariablen hinterlegt werden. Dadurch wird Datenredundanz vermieden und gleichzeitig eine Unabhängigkeit des HMI-Teil vom übrigen Informationsträger realisiert. Die Zuordnung der Rollen der Bildobjekte soll dabei über Verweise auf

ein ebenfalls vom Informationsträger unabhängiges Merkmalsystem wie beispielsweise eCl@ss erfolgen. Ob diese aus den bereits im Informationsträger vorhandenen Rollenklassen bzw. System Units ausgelesen werden oder ob das HMI-Engineering-Werkzeug sie selbst erzeugt, wird hierbei dem Tool-Ersteller überlassen. Die Verwendung der Kommunikationsvariablen in den Bildobjekten muss ebenfalls durch den Verweis auf ein Merkmalsystem eindeutig sein. Demzufolge müssen auch die Variablen selbst eine solche Referenz besitzen.

Weiterhin ist eine Rechteverwaltung notwendig. Jedes Bild, jedes Bildelement beziehungsweise jede Variable und jeder Parameter muss eine Angabe darüber besitzen, wer bedienen und wer nur beobachten darf. Dafür ist eine Abbildung der Nutzerrollen und der entsprechenden Rechte auf das Meldesystem des Bedien- und Beobachtungssystem der Prozessführungsebene vorzusehen. Es obliegt der Verantwortung des Modulherstellers, diese zuvor zu harmonisierten Nutzerrollen sinnvoll zu implementieren.

4.3 Integrationsprozess

Für die Einordnung der einzelnen Integrationsaspekte in einen Integrationsprozess gibt Abbildung 40 einen Überblick über die notwendigen Integrationsschritte eines Moduls in die Prozessführungsebene.

In einem ersten Engineering-Schritt werden die einzelnen Funktionen einer Anlage aus Modulen zusammengesetzt. Hier wird auf Ebene einer Verfahrenstechnischen Funktion, beispielsweise einer Reaktion, die Auswahl getroffen. In diesem Beispiel wird jedes Modul vereinfacht über die Anzahl, den Typ der Ein- und Ausgänge und die parallel nutzbaren Einheiten eines Typs beschrieben. Die *Modulklasse Tank* (2/2/2) in Abbildung 40 ist exemplarisch durch zwei Eingänge, die gleichzeitige Nutzung zweier unabhängiger Tank-Einheiten in dem Modul, sowie zweier Ausgänge charakterisiert. In diesem Schritt erfolgt eine Spezifizierung der Module auf Klassen-Ebene, wobei jede Klasse über einen definierten Satz generischer Eigenschaften und Funktionen definiert ist.

Im zweiten Schritt, dem modularen Automationsengineering, wird das Gesamtverhalten der Anlage konfiguriert. Dieser Schritt umfasst die dafür notwendigen Teilschritte, beginnend bei der Konfiguration (2.1). Hier wird zum Beispiel die Kommunikationsschnittstelle des Moduls mit dem übergeordneten PLS konfiguriert. Die weiteren Teilschritte (2.2) – (2.4) konfigurieren die modulübergreifende Logik, die Rezepte und die Bedienbilder. Aufgrund der Beschreibung der Gesamtfunktion der Anlage mit Hilfe der Modulklassen und der entsprechenden generischen Eigenschaften, können diese Konfigurationen ohne die Kenntnis der konkreten Modulinstanzen erfolgen. Dies ermöglicht zum einen ein sehr frühes Engineering des Automatisierungssystems der Gesamtanlage und später einen einfachen Austausch des Moduls gegen ein anderes, das die Fähigkeiten dieser Modulklasse unterstützt.

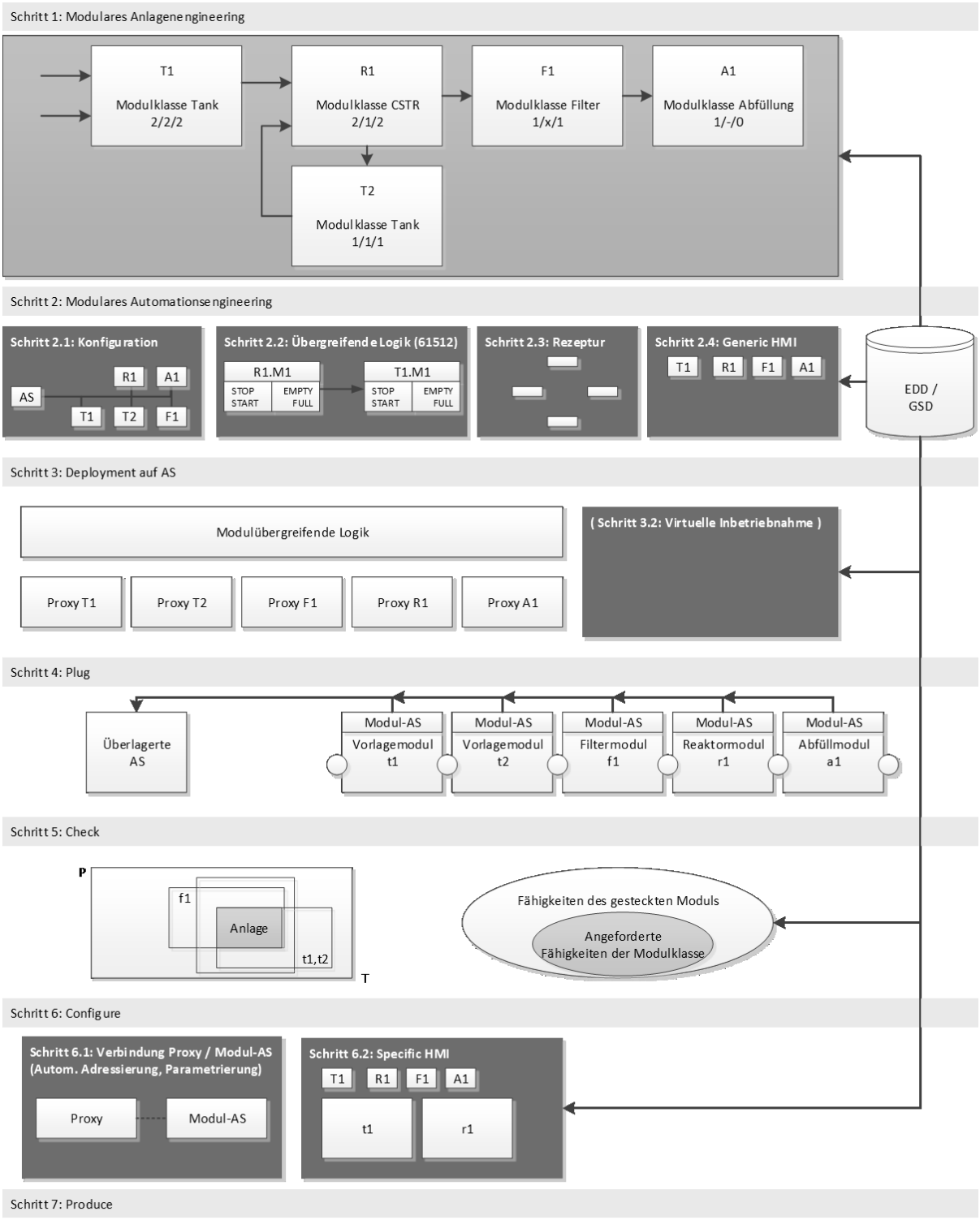


Abbildung 40: Integrationsprozess (Obst, Hahn & Urbas, 2014b)

Dieses Vorgehen kann um Profile, ähnlich wie bei der Integration von Feldgeräten über Feldbussen, ergänzt werden, um die Rückwärtskompatibilität zu älteren Modulklassenbeschreibungen zu gewährleisten. Im dritten Integrationsschritt wird die Konfiguration auf das Automatisierungssystem der Anlage geschrieben. Erst in einem vierten Schritt der Integration werden die konkreten Module, die Modulinstanzen, an das überlagerte Automatisierungssystem angeschlossen (Plug). Nachfolgendend werden die Randbedingungen (zum Beispiel Druckstufen, Grenzwerte), sowie der Funktionsumfang der

angesteckten konkreten Module gegeneinander überprüft (Check). Bevor die Module in Betrieb genommen werden können, erfolgt ein Konfigurationsschritt, in dem die spezifischen Module automatisch ihre Adressen zugewiesen bekommen oder entsprechend parametriert werden (Configure).

4.4 Überführung der Integrationsaspekte in einen Informationsträger

Ausgangspunkt der Definition der Anforderungen an einen Informationsträger ist ein Geschäftsmodell, in dem Modulhersteller, wie in der NE 148 beschrieben, Module als eine feste und gegebenenfalls parametrierbare Einheit aus verfahrenstechnischer Funktion und Automatisierungstechnik anbieten. Die wesentlichen Engineering-Schritte an der Schnittstelle von Verfahrenstechnik und Automatisierungstechnik finden beim Modulhersteller statt, ein Modulanwender soll ein solches Modul mit wenigen Schritten und ohne detaillierte Kenntnis des internen Aufbaus in eine modulare Anlage integrieren können. Daraus ergeben sich folgende Anforderungen an einen Informationsträger:

- *Automatische Generierbarkeit*: Ein Informationsträger muss direkt und ohne manuelle Anreicherung aus den Engineeringdaten des Moduls generierbar sein. Dies ist eine wesentliche Voraussetzung für eine fehlerfreie und nachvollziehbare Parametrierung, Versionierung und Variantenverwaltung von Modulen. Bei der Modellierung ist daher darauf zu achten, dass keine manuellen Anreicherungen bei der Generierung des Informationsträges notwendig sind.
- *Unabhängigkeit von Beschreibungsaspekten*: Eine weitgehende Unabhängigkeit der einzelnen Beschreibungsaspekte eines Moduls, sowohl methodisch als auch semantisch, stellt den geordneten Austausch und die unabhängige Entwicklung einzelner funktionaler Aspekte sicher („Tortenstücke“). Dies unterstützt zum einen eine iterative, anforderungsgetriebene Entwicklung bestehender Aspekte des Informationsträgers und gewährleistet zum anderen eine freie Erweiterbarkeit um neue Aspekte.
- *Trennung abstrakter und technischer Modelle*: Die Informationsmodelle sollten so weit möglich, technologieunabhängig spezifiziert werden können. Technologieabhängige Anteile eines Modells sind getrennt zu modellieren. Beispielsweise muss eine I/O-Variable unabhängig von den Einschränkungen der Kommunikationstechnologie identifiziert und benannt werden können. Die Abbildung auf die kommunikationsmittelspezifische Adressierung – beispielsweise Coil/Index bei Modbus, Slot/Index bei Profibus oder server/tagname bei OPC - sind in einem technologieabhängigen Anteil zu kapseln.
- *Automatische Prüfbarkeit und Lesbarkeit*: Das integrierende System muss den Informationsträger einlesen und die notwendigen Objekte im eigenen Datenhaushalt automatisch generieren können. Das Format des Informationsträgers muss daher einfach auf syntaktische Richtigkeit und Vollständigkeit geprüft werden können. Auch beim Prüfen und Einlesen sind Datentypen, Strukturen und Kennzeichnungssysteme, die einen manuellen Eingriff erfordern, kontraproduktiv. Eine gegebenenfalls notwendige Auflösung von Vieldeutigkeit (beispielsweise bei der Konfliktresolution in einem multikriteriellen

Optimierungsschritt wie der Layoutanpassung von HMI an andere Formfaktoren) muss wiederholbar parametrierbar sein.

Die konkreten Daten eines Informationsträgers beschreiben eine einzige Modulinstanz. Über eine Serien- und Versionsnummer wird die eindeutige Identifikation gewährleistet. Ein Informationsträger bildet immer den gegebenenfalls auch vorkonfigurierten Zustand des Moduls ab, der durch das Automatisierungsprojekt des Moduls vom Modulhersteller vorgegeben wird. Spätere Änderungen beispielsweise der Kommunikationsparameter (IP Adresse) müssen in dem zu integrierenden Automatisierungssystem über eine Parametrierung eingepflegt werden. Bei einem Abkoppeln des Moduls muss die Eindeutigkeit zwischen Informationsträger und Modul gewährleistet werden. Hier ist es vorstellbar, durch ein Abkoppeln das Modul in den Auslieferungszustand zurück zu versetzen oder eine entsprechende Änderung im Informationsträger zu generieren, die beispielsweise die veränderte Lebenszyklusinformation beinhaltet.

Bei der konkreten Modellierung eines Moduls als Informationsträger, können eine Vielzahl von Beschreibungsmittel zum Einsatz kommen. In Kapitel 3 wurden diese, aufgeschlüsselt nach ihren konkreten Einsatzgebieten, vorgestellt und bewertet. Bei der Auswahl ein Beschreibungsmittel müssen mehrere Faktoren berücksichtigt werden. Neben den oben genannten allgemeinen Anforderungen an den Informationsträger, die mit Hilfe des Beschreibungsmittels umgesetzt werden müssen, sollten des Weiteren die Einsatzumgebungen des Informationsträgers betrachtet werden. Für die Anwendung des Informationsträgers in der Praxis muss dieser in zahlreiche Werkzeuge integriert werden und eine hohe Nutzerakzeptanz erreichen. Im Folgenden werden exemplarisch drei Ansätze vorgestellt.

4.4.1 EDD Ansatz

Heutige Leitsysteme weisen Integrationspfade zur Integration einzelner Feldgeräte auf. Daher wurde in (Obst, Hahn & Urbas, 2014a) ein Ansatz unter Nutzung der Feldgeräte Beschreibungsmittel GSD und EDDL gewählt. Im Folgenden wird die Umsetzung der einzelnen Integrationsaspekte erläutert.

Variablen und Parameter

Bei der Integration von Feldgeräten in ein Prozessleitsystem bildet die Grundlage jeder Geräteparametrierung die Beschreibung der Variablen und Parameter. Bei der Beschreibung der Variablen und Parameter eines Moduls besteht die Zielstellung, auf möglichst einfache und wieder verwendbare Weise das vollständige Modul zu beschreiben. Diese Beschreibung soll auf ihre Kernkomponenten abstrahiert und eine einheitliche Bezeichnung eingeführt werden, so dass eine Vorlage für die Beschreibungen eines Moduls entsteht. Dabei wird die Beschreibung zwischen drei Ebenen untergliedert:

EBENE 1 - Deklaration aller verwendeten Variablen:

Die erste Ebene der Modellierung ist eine Abbildung aller zur Kommunikation zwischen Modul und Prozessleitsystem verwendeten Variablen. Grundsätzlich wird bei der Geräteparametrierung zwischen

statischen und dynamischen Parametern unterschieden. Darüber hinaus können dynamische Parameter in zyklische und azyklische unterteilt werden. Bei der Modulkommunikation ist die Unterscheidung ebenfalls notwendig. Sobald ein Parameter oder eine Variable dynamische Eigenschaften aufweist, wird diese Eigenschaft im *CLASS* Attribute festgehalten. Eine weitere Unterscheidung sowohl der statischen als auch der dynamischen Parameter findet im Attribute *HANDLING* statt, dieses definiert die Zugriffsrechte für den Nutzer (Lesen, Schreiben).

```
VARIABLE identifier                // eindeutige schematische Bezeichnung
{
       CLASS class-specifier;      // beschreibt Verwendungsweise der Variablen
       TYPE type-specifier;
       HANDLING specifier;
}
```

Abbildung 41: Eigenschaften einer EDD Variablen (Obst, Hahn & Urbas, 2014b)

EBENE 2 – Sinneinheiten

Abhängig von den Eigenschaften und deren späteren Verwendung werden sämtliche Parameter und Variablen in Ebene 2 entsprechend gruppiert. Eine solche Gruppierung von Variablen kann in EDDL wie folgt dargestellt werden:

```
COLLECTION OF VARIABLE identifier
{
       LABEL string-specifier;
       MEMBERS
       {
              member-specifier;     // Variablen gleichen HANDLINGs,
              member-specifier;     // dynamische/statische Eigenschaften,
              member-specifier;     // gleiche Verwendung
              …
       }
}
```

Abbildung 42: Modul Parameter als Teil einer COLLECTION (Obst, Hahn & Urbas, 2014b)

EBENE 3 – Kommunikation

Innerhalb der Ebene 3 werden Kommunikationsregeln festgelegt. EDD nutzt für die Kommunikation automatisch Kanal MS02 von Profibus, also eine azyklische Kommunikation zwischen Prozessleitsystem und dem Modul. Profibus ist ein modulbasiertes Kommunikationsprotokoll, welches die einzelnen Module über SLOT und die genaue Platzierung innerhalb des Moduls über INDEX festlegt. Beide Angaben müssen innerhalb der EDD hinterlegt sein. Darüber hinaus ist es nicht möglich, Inhalte gleichzeitig zu lesen und zu schreiben, was zur Folge hat, dass COLLECTIONS, dessen Inhalte gelesen und geschrieben werden können, innerhalb der Kommunikation in zwei getrennten Paketen behandelt werden müssen. Solche Kommunikationspakete werden folgendermaßen beschrieben:

```
COMMAND identifier
{
	INDEX		1…n;
	SLOT		1…254;
	OPERATION	READ | WRITE;
	TRANSACTION
	{
		REQUEST
		{
			data-items-specifier;
			data-items-specifier;
		}
		REPLY
		{
			data-items-specifier;
			data-items-specifier;
		}
}
}
```

Abbildung 43: Definition eines Kommunikationspaketes (Obst, Hahn & Urbas, 2014b)

Die zyklische Kommunikation über MS0 wird innerhalb der GSD Datei definiert. Dabei wird beschrieben, welche Adressen des Moduls in einem zyklischen Datenaustausch mit dem Prozessleitsystem unterstützt werden. Die Zykluszeit wird von den meisten Prozessleitsystemen automatisch generiert.

Beschreibung einer Ablaufkette mit EDDL

Die von dem Modul mitgelieferte Darstellung in Matrizenform (siehe Abschnitt 4.2.1) kann problemlos innerhalb der EDD beschrieben werden. Dazu werden die zwei Matrizen serialisiert, so dass zwei binäre Vektoren übertragen werden müssen. Darüber hinaus können die aktiven Plätze und Transitionen ebenfalls über zwei Vektoren oder auch einzeln übertragen werden (siehe Abbildung 44 und Abbildung 45).

```
VARIABLE MOD_SFC_P2T
{
	LABEL		"Struktur der AS: P zu T";
	CLASS		LOCAL;
	TYPE		BITSTRING
	DEFAULT_VALUE	"01100001";
	HANDLING		READ;
}
```

Abbildung 44: Beschreibung der Verbindungen der Plätze zu Transitionen (Obst, Hahn & Urbas, 2014b)

```
VARIABLE MOD_SFC_T2P
{
	LABEL			"Struktur der AS: T zu P";
	CLASS			LOCAL;
	TYPE			BITSTRING
	DEFAULT_VALUE	"10010110";
	HANDLING		READ;
}
```

Abbildung 45: Beschreibung der Verbindungen der Transitionen zu Plätzen (Obst, Hahn & Urbas, 2014b)

Zusätzlich müssen für die Deserialisierung im Prozessleitsystem, Informationen zur Anzahl an Sparten und Zeilen der Matrix übertragen werden. Dies bedeutet vier zusätzliche Variablen, die jedoch innerhalb derselben Kommunikationsgruppe wie die serialisierten Matrizen übermittelt werden können.

Beschreibung eines Bedienfließbildes mit EDDL

Im vorhergehenden Abschnitt wurde gezeigt, wie die als Graph abstrahierte Struktur eines SFC in einer EDD abgebildet werden kann. Um die Informationen eines Bedienfließbildes ebenfalls in einer EDD abzubilden zu können, werden die Unterschiede zur Ablaufkette zunächst dargestellt:

- Für die Bildelemente eines SFCs müssen Parallel- und Alternativverzweigungen aus der Topologie inferiert werden, während die in (Obst et al., 2012) betrachtete Struktur der Bedienfließbilder alle Bildelemente vollständig enthält.
- Alle Verbindungen eines SFCs werden gleich dargestellt. Die Verbindungen in Bedienfließbildern kodieren hingegen die Art der Verbindung (Stoff, Energie, Information), beispielsweise durch Farben und Linienstärke.
- Alle Visualisierungsobjekte eines SFCs werden einheitlich durch zwei Bildsymbole (Stelle oder Transition) dargestellt, während in prozessorientierten Bedienfließbildern Symbolbibliotheken zum Einsatz kommen.
- Der dynamische Informationshaushalt einer SFC-Darstellung kann durch eine boolesche Variable für den Zustand eines jeden Schrittes (Aktiv/Inaktiv), eine boolesche Variable für den Schaltzustand pro Transition (Frei/Gesperrt) und einer booleschen Variablen für die manuelle Überbrückung der Transition für die meisten Zwecke ausreichend gut dargestellt werden. Die Bildsymbole der Bedienfließbilder sind jedoch komplexe Bibliothekselemente, die typspezifische Informationsstrukturen erwarten.

Für die formale Beschreibung eines Bedienfließbildes in EDD bedeuten diese Unterschiede, dass eine reine Abbildung der Strukturinformationen nicht ausreichen. Um beispielsweise die Bildsymbole der Bedienfließbilder möglichst in einem einheitlichen Look & Feel bereitzustellen, müssen die Instanzen der Symbole durch das übergeordnete Leitsystem, durch eine separat zu konfigurierende Symbolbibliothek bereitgestellt werden. In dem generisch auszuliefernden Bedienfließbild des Moduls wird nachfolgend über eine Klassen- oder Typ-Bezeichnung die Referenz hergestellt (siehe Abbildung 47). Das formalisierte HMI orientiert sich an der in Abbildung 46 dargestellten Visualisierung.

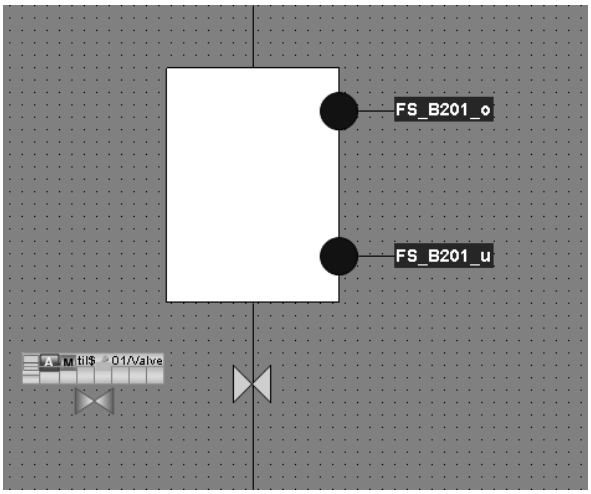


Abbildung 46: Beispiel HMI der Formalisierung in Abbildung 47 (Obst, Hahn & Urbas, 2014b)

```
COLLECTION OF VARIABLE MOD_HMI_BBD1_Tank1
{
	LABEL  "Alle Informationen eines Tanks";
	MEMBERS
	{
		Type, PU_HMI_BBD1_Tank1_Type;
		XCoord, PU_HMI_BBD1_Tank1_XCoord;
		YCoord, PU_HMI_BBD1_Tank1_YCoord;
		Connections, PU_HMI_BBD1_Tank1_Connections;
	}
}

VARIABLE MOD_HMI_BBD1_Tank1_Type
{
	LABEL		"Klasse des Bedienbild-Elementes Tank";
	CLASS		LOCAL;
	TYPE		ASCII;
	DEFAULT_VALUE	"TANK";
}
VARIABLE MOD_HMI_BBD1_Tank1_XCoord
{
	LABEL  "X-Koordinate Tank1";
	CLASS  LOCAL;
	TYPE   DOUBLE;
	DEFAULT_VALUE  394.0909;
}
```

Abbildung 47: Beschreibung eines statischen HMI-Elements in EDD (Obst, Hahn & Urbas, 2014b)

Jedes Bedienfließbild wird über eine COLLECTION beschrieben, in der alle darzustellenden Objekte zusammengefasst werden. Ein Objekt wird über den Typ, Koordinaten auf dem Bedienfließbild, weiteren objektspezifischen Eigenschaften sowie den Verbindungen (Variable: MOD_HMI_BBD1_Tank1_Connections) zu anderen Objekten beschrieben.

Für die Beschreibung der dynamisierten Bildelemente, also Bildelementen welche mit Prozessdaten oder Zuständen des Moduls verknüpft sind, muss innerhalb einer entsprechenden Struktur auf die im Variablenhaushalt des Moduls definierten Informationen referenziert werden (siehe Abbildung 48).

```
VARIABLE MOD_HMI_BBD1_Valve1_Fbk
{
       LABEL         "Verweise auf Signal der Rückmeldung";
       CLASS         LOCAL;
       TYPE          ASCII;
       DEFAULT_VALUE "PU_VAR_Valve1_Fbk_Open";
}
```

Abbildung 48: Beschreibung eines statischen HMI-Elements in EDD

Die Darstellung einer Rohrleitung oder auch einer Signallinie kann nicht alleine über eine X-Y Koordinate beschrieben werden. Hier muss der Verlauf der einzelnen Punkte der Linie mit beschrieben werden, zum Beispiel in Form einer einfachen Liste.

Mit den Algorithmen der autoHMI-Transformationskette (Doherr, Urbas, Drumm & Franze, 2011) wird das in der EDD gemäß Abbildung 47 formal beschriebene HMI des Moduls eingelesen und mit der Symbolbibliothek des Zielleitsystems instanziiert. Je nach Offenheit des Leitsystems erfolgt dies durch externe oder in das Leitsystem integrierte Werkzeuge.

Zusammenfassung & Bewertung

Die Beschreibung eines Moduls mit Hilfe eines EDDL basierten Informationsträgers unter Einhaltung normativer Vorgaben ist möglich. Wichtige Integrationsaspekte wie zum Beispiel die Abbildung einzelner Kommunikationsvariablen, einer SFC-basierten Verhaltensbeschreibung, sowie der systemneutralen Beschreibung einzelner Bedienbilder konnten umgesetzt werden. Jedoch musste für die eindeutige Referenzierung der einzelnen Objekte eine Semantik in die Bezeichner der Variablen integriert werden, was von der EDDL-Maintenance Group nicht befürwortet wird und beispielweise durch die Erweiterung mit einem merkmalsbasierten Ansatz, wie er in (John et al., 2013) vorgeschlagen wird, gelöst werden kann.

4.4.2 DIMA Ansatz

Der im Folgenden diskutierte und vorgestellte Informationsträger Module Type Package (MTP) wurde im Rahmen das Forschungsprojektes DIMA mit der Firma Wago, der Helmut-Schmidt-Universität in Hamburg und der Technsiche Universität Dresden entwickelt (Holm et al., 2014; Obst et al., 2015a). Dieser Informationsträger beschränkt sich auf zwei wesentliche Aspekte der Integration, der zustandsgeführten Prozessführung und des HMIs einer aus selbstständig automatisierten, mit dezentraler Intelligenz ausgestatteten Modulen aufgebauten prozesstechnischen Anlage. Dies entspricht der Typ-II-Moduldefinition der (NE 148).

Strukturierung

Zur Unterstützung der Versionierbarkeit und Verteilung eines MTPs ist dieser als Modellcontainer mit einer hierarchischen Ordnerstruktur angelegt, in denen beliebige Archivformate angewandt werden können. In dem in (Holm et al., 2014) vorgestellten Demonstrator kommt hierbei ein ZIP-Archiv zum Einsatz. Dies hat den entscheidenden Vorteil, dass zum einen für viele Sprachen und Systeme open-

source-Implementierungen von Funktionssammlungen verfügbar und zum anderen Funktionen zum Umgang mit ZIP-Archiven in nahezu allen gängigen Betriebssystemen vorhanden sind. Im Folgenden werden die Inhalte dieses Modellcontainers detaillierter beschrieben.

Manifest

An der MTP-Wurzel ist eine Manifest-Datei angelegt. Diese XML-Datei enthält die für den MTP relevante Meta-Daten, welche Auskunft über die Version der zugrundeliegenden MTP-Spezifikation und deren eindeutige Seriennummer geben und die in diesem enthaltene funktionale Aspekte beschreibt. Die Indizierung und Suche eines prinzipiell geeigneten MTP wird durch die Auswertung der Manifest-Daten beschleunigt. Die in der Manifest-Datei abgelegte Information zu den Aspekten beinhalten sowohl den Typ und den Pfad zur entsprechenden Datei, sowie die Version und ein Flag, das den Aktivitätsstatus des enthaltenen Aspektes anzeigt. Dies erlaubt beispielsweise bei komplexen Produktfamilien, zu den Modulvarianten Beschreibungsvarianten abzulegen und die relevanten Teilmodelle zu- oder abzuschalten (aktiv bzw. inaktiv). Weiterhin enthält das Manifest vordefinierte Einträge für häufig genutzte Dateiarten, beispielsweise Bilder (Images), Zertifikate (Certificates), Hilfedateien (Helpfiles) und Handbücher (Manuals). Eine weitere wichtige Informationsklasse sind Angaben zur Kommunikation mit der dezentralen Recheneinheit des Moduls (SPS oder Embedded Controller).

Integrationsaspekte

Jeder eigenständige Integrationsaspekt wird in einer eigenen Datei abgelegt, auf die die Manifest-Datei verweist. Durch diese strikte Trennung der Einzelmodelle ist eine einfache Erweiterung gegeben und das Hinzufügen neuer Integrationsaspekte möglich. Abbildung 49 stellt die hierarchische Strukturierung des MTP (Modul.mtp) dar. Im Projekt DIMA (Holm et al., 2014) wurden die zwei Integrationsaspekte untersucht, das Bedienen und Beobachten (hmi.xml) und die zustandsbasierte Prozessführung (control.xml), diese werden im Folgenden detailliert beschrieben. Die weitere Aufteilung der einzelnen XML-Dateien in einen technologieabhängigen und technologieunabhängigen Teil erfolgt durch die Strukturierungsmöglichkeiten der XML.

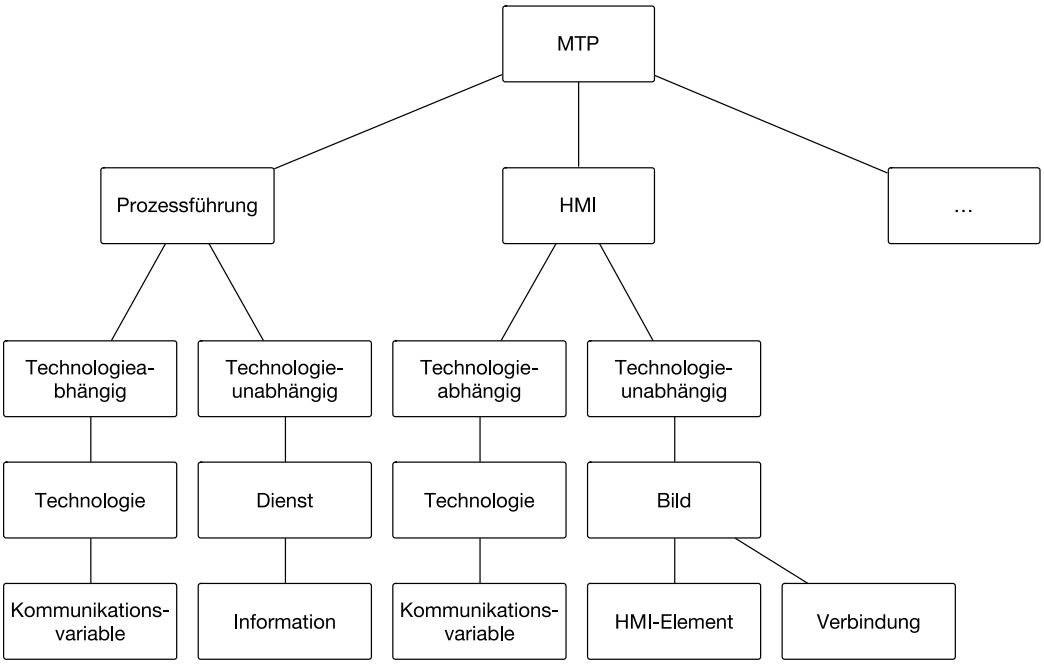


Abbildung 49: Inhaltliche Strukturierung des MTP

Technologieunabhängige und –spezifische Teilmodelle

Information, also die Verbindung von Struktur (Syntax) mit der Bedeutung der Strukturentitäten (Semantik), kann unabhängig von der Abbildung dieser Information durch Sprach- und Strukturmittel einer Kommunikationstechnologie, wie z.B. OPC-DA dargestellt werden. Auch innerhalb der Dateien für die jeweiligen Beschreibungsaspekte wird zwischen der Beschreibung der Variablen und Funktion des Moduls und deren Abbildung auf Kommunikationsvariablen einer Kommunikationstechnologie getrennt. Durch diese Trennung können vom Modulhersteller mehrere Kommunikationskanäle vorgesehen werden. Weiterhin ist es möglich, die inhaltliche Beschreibung des Moduls getrennt von möglichen Kommunikationstechnologien zu entwickeln.

Sowohl für die Beschreibung der Prozessführung als auch für die Beschreibung der Funktionen des Bedienens und Beobachtens wurde der gleiche Ansatz zur Beschreibung der Kommunikation gewählt. Im weiteren Verlauf werden die implementierten Anteile der technologieunabhängigen Beschreibung für die Aspekte Prozessführung und Bedienen und Beobachten näher erläutert.

Technologieabhängige Beschreibung der Kommunikation

Die Beschreibung der realen Kommunikation zwischen dem Prozessleitsystem und dem Modul muss technologieabhängig erfolgen.

Für jede Kommunikationstechnologie die das Modul unterstützt, wird ein neues Element pconnection eingefügt. Dieser Teil beinhaltet die Definition der Kommunikationsparameter sowie die Beschreibung der einzelnen Variablen pvalue spezifisch für die Kommunikationstechnologie. Die logische Verbindung zwischen der technologieunabhängigen und technologiebezogenen Variablen wird über eine im MTP eindeutige ID pvid hergestellt.

Abbildung 50 stellt eine entsprechende Variable unter Nutzung der OPC-DA-Technologie dar.

```
<pconnection type="OPCDA" server="localhost">
	<!--pvalue defines a variable for Communication between hmi and module-->
	<pvalue pvid="c5232ef6-6cd6-4387-b7e1-fec9f5a5020c"
		name="PLC_PRG.Panel_visu.L001Value" description="">
		<!-- In this case (OPC DA) this is the OPC TAG-->
		<adress>_750_8204_PFC200_2ETH_RS_CAN.Application.PLC_PRG.Panel_visu.
		L001Value</adress>
		<accesstyp>READWRITE</accesstyp>
		<datatype>REAL</datatype>
	</pvalue>
```

Abbildung 50: Beschreibung der technologieabhängigen Kommunikationsvariablen am Beispiel von OPC DA

Zustandsbasierte Prozessführung

Der technologieunabhängige Anteil der Beschreibungsdatei für die zustandsbasierte Prozessführung enthält Informationen zu den Diensten und den zugeordneten Kommunikationsvariablen für die Zustandsanforderung und -erreichung sowie die Variablen, mit denen die Parametrierung erfolgen kann. Dem Element des Moduls (modul) wird ein weiteres Element mit einem Attribut zur Anzahl der Dienste eingefügt (services). Das Element services enthält die Dienste (service) mit den entsprechenden Variablen. Jede Variable wird durch ein Element ServiceInterfaceInfo repräsentiert. Um Parametervariablen und Variablen zur Zustandskommunikation unterscheiden zu können, wird ein entsprechender Eintrag im Attribut ref genutzt. Hier sind Eintragungen mit Verweis auf z.B. eCl@ss-Elemente (eCl@ss, 2014) oder auch Ontologien wie Quandt (Hodgson et al., 2014) o.ä. denkbar. Abbildung 51 stellt ein den technologieunabhängigen Anteil des Moduls Plant_Module_1 dar. Es verfügt über zwei Dienste mit jeweils zwei Variablen zur Zustandskommunikation (ref-Eintrag nicht abgebildet). Der Dienst Tempering verfügt des Weiteren über eine Parametervariable (ref-Eintrag nicht abgebildet).

```
<modul id="" name="Plant_Module_1" ref="">
	<services count="2">
		<service id="" name="Cleaning" ref="" >
			<ServiceInterfaceInfos>
				<ServiceInterfaceInfo name="Cleaner_Regel_D"
					pvid="f60b023a-f610-4ae2-ac02-57586769d15d" ref=""/>
				<ServiceInterfaceInfo name="Cleaner_Regel_I" D"
					pvid="f60b023a-f610-4ae2-ac02-57586769d15d" ref=""/>
				<ServiceInterfaceInfo name="Cleaner_Zeit"
					pvid="f60b023a-f610-4ae2-ac02-57586769d15d" ref=""/>
				<ServiceInterfaceInfo name="CleanerVorlauftemp"
					pvid="f60b023a-f610-4ae2-ac02-57586769d15d" ref=""/>
			</ServiceInterfaceInfos>
		</service>
		<service id="" name="Tempering" ref="" >
			<ServiceInterfaceInfos>
				<ServiceInterfaceInfo name="SollTemp"
					pvid="f60b023a-f610-4ae2-ac02-57586769d15d" ref=""/>
				<ServiceInterfaceInfo name="CurrentTemp"
```

```
                                pvid="f60b023a-f610-4ae2-ac02-57586769d15d" ref=""/>
                    </ServiceInterfaceInfos>
            </service>
        </services>
            </modul>
```

Abbildung 51: XML-Darstellung des technologieunabhängigen Anteils für die Prozessführung des Moduls

Abbildung des HMI Aspektes mit GraphML

Aufgrund der guten Einbindung in Open-Source-Editoren für die Validierung der Beschreibung und der guten Unterstützung durch graphentheoretische Funktions- und Algorithmensammlungen fiel im DIMA-Projekt die Wahl auf das XML-basierte Beschreibungsmittel GraphML (Brandes et al., 2002; GraphML Working Group, 2006). Weitere Lösungsansätze zur Automatisierung der Bedienbildgenerierung nutzen die im Stand der Technik (Kapitel 3.5) aufgeführten Formate. Diese Formate benötigen entweder umfangreiche MDA-Rahmenwerke (Calvary et al., 2003; Reuther, 2003; Urbas et al., 2011), sind für die Prozessautomation nur bedingt geeignet (Calvary et al., 2003; Hickson et al., 2014; Reuther, 2003) oder lassen sich nur mit speziellen Ergänzungen mit den für ein semantisches Layout notwendigen Attributen erweitern (Hickson et al., 2014; Schmitz, 2010). Jedes Bedienbild eines Moduls wird als einzelner Graph mit dem in der GraphML-Spezifikation definierten Element vom Typ *graph* beschrieben (siehe Abbildung 52). Die *graph*-Elemente der Bedienbilder sind Kindelemente des Sammelelements *pictures,* das sich direkt unterhalb des Wurzelelements *HMI* befindet. Parallel zu *pictures* liegt das Element *pconnections*, das die technologieabhängigen Kommunikationskanäle der in den Bedienbildern darzustellenden und zu manipulierenden Variablen des Moduls beschreibt.

```
<graph id="Visu_Temp" name="Visu_Temp" w="1920" h="1200">
        <!-- Each node is a picture element -->
        <node id="2a175c26-8071-4c66-86cf-a26c875dae44" tagname="L001" x="446" y="594"
                w="371" h="469" role="NA_TankLevel01">
                …
        </node>
        <node id="7369b780-9b9b-4cfb-9b29-5a95e56e1473" tagname="P001" x="937" y="1025"
                w="95" h="145" role="NA_Pump01">
                …
        </node>
        <node id="dd174dd1-c603-4260-a77d-fe80e8e34977" tagname="X001" x="914" y="354"
                w="106" h="153" role="NA_Valve01" ref="http://www.electropedia.org/iev/
                iev.nsf/display?openform&ievref=551-14-03">
                …
        </node>

        <!-- Each edge is a connecttion between picture elemtens e.g. a pipe -->
        <edge id="fd6f47d1-7491-4aa6-9446-8142633c0c22" name="Pipe_01" role="NA_Pipe01"
         source="2a175c26-8071-4c66-86cf-a26c875dae44" target="cde7ce82-e99e-4324-b3bf-
                46b578f1ec52" overlap="0">
                …
        </edge>
        <edge id="cde7ce82-e99e-4324-b3bf-46b578f1ec52" name="Pipe_02" role="NA_Pipe01"
         source="fd6f47d1-7491-4aa6-9446-8142633c0c22" target="7369b780-9b9b-4cfb-9b29-
```

```
            5a95e56e1473" overlap="0">
            …
      </edge>
      <edge id="2ec85233-29b7-49c1-8e02-49817e791f27" name="Pipe_03" role="NA_Pipe01"
       source="7369b780-9b9b-4cfb-9b29-5a95e56e1473" target="dd174dd1-c603-4260-a77d-
            fe80e8e34977" overlap="0">
            …
      </edge>
</graph>
```

Abbildung 52: Beschreibungen der Anlagentopologie eines Bedienbildes in GraphML

Jedes Bedienbildelemente wird durch das in GraphML spezifizierte XML-Element *node* repräsentiert, jede Verbindung zwischen Bedienbildelemente (Rohrleitung, Wirklinie, Hierarchie) wird durch das Kanten-Elemente *edge* beschrieben. Die die Bedienbildelemente repräsentierenden Knoten-Elemente werden durch nachfolgende skalare Attribute näher bestimmt:

id: diese für den Graphen eineindeutige, rechnererzeugte Identifikation erlaubt Querverweise, beispielsweise als Start- und Endpunkt einer Kante, dieses notwendige Attribut ist in GraphML definiert;

tagname: stellt eine notwendige MTP spezifische Erweiterung von GraphML dar und liefert eine kontextbezogene und eindeutige Kennung für den Bediener;

x,y,w,h: repräsentieren, optional als MTP spezifische Erweiterung von GraphML, Koordinaten und Abmaße des Bedienbild-Elementes im Quellengineeringsystem als Hinweis für das Layoutsystem des integrierenden Systems;

role: über eine eindeutige und weitgehend generische Rollenbezeichnung werden in den Engineering-Werkzeugen die zu platzierenden Bedienbildelemente ausgewählt. Dies ist ebenfalls eine notwendige, MTP-spezifische Erweiterung von GraphML;

ref: ist eine semantische Referenz auf ein standardisiertes Merkmal- oder Taxonomiesystem wie eClass und IEV zur weiteren Spezialisierung der in dem Attribut *role* definierten Rolle und ist eine optinale MTP-spezifische Erweiterung.

```
<node id="7369b780-9b9b-4cfb-9b29-5a95e56e1473"
        tagname="P001" x="937" y="1025" w="95" h="145" role="NA_Pump01">
      <data>
            <variable informationtag="NA_351-41-02"
              pvid="5de8891b-4319-4d29-84c4-6281588267e1"
              ref="http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=351-
            41-02"/>
      </data>
            </node>
```

Abbildung 53: Beschreibungen der statischen Layout-Parameter, der dynamischen Elemente und der Semantik eines Bedienbildelementes mit GraphML

Für die Beschreibung nicht-skalarer Elemente von Knoten sieht GraphML ein *data*-Element vor (Abbildung 53). In diesem Block werden die für den Knoten relevanten Prozessvariablen wie Istwert, Sollwerte, Grenzwerte als *variable*-Elemente aufgelistet. Jedes *variable*-Elemente muss mit folgenden Attributen ausgezeichnet werden:

informationtag: Rolleninformation entsprechend der Bibliothek;

pvid: Referenz auf die Kommunikations-Variable in dem *pconnection*-Element;

ref: semantischer Verweis auf ein standardisiertes Merkmal- oder Taxonomiesystem wie eClass und IEV.

Die mit *edge*-Elementen beschriebenen Rohrleitungen, Wirklinien und Teil-von-Beziehungen werden durch folgende Attribute näher spezifiziert:

target, source: sind in GraphML definiert und notwendig und beschreiben den Anfangs- und Endknoten einer Verbindung;

edgepath: beschreibt den Verlauf der Kante im Quellsystem als Hinweis für das Layout im Integrationssystem und ist eine optinale, MTP-spezifische Erweiterung von GraphML;

edgeio: beschreibt die Koordinaten von Anfang und Ende als Hinweis für das Layout im Integrationssystem und ist ebenfalls eine optionale, MTP-spezifische Erweiterung von GraphML;

ref: ist ein semantischer Verweis auf ein standardisiertes Merkmal- oder Taxonomiesystem wie eClass, IEV oder RDFS zur Bestimmung der Bedeutung einer Kante, wird keine Referenz angegeben, soll das integrierende System diese Kante wie eine unspezifizierte Rohrleitung behandeln und ist eine optionale, MTP-spezifische Erweiterung von GraphML.

Wie das *node*-Element kann das *edge*-Element mit beliebigen weiteren, anwendungsspezifischen Attributen und einem *data*-Element erweitert werden. Relevante Erweiterungen sind Hinweise zur Auszeichnung von statischen Rohrleitungsklassen oder Variablen zur Visualisierung von dynamischer Routing-Information. Dabei gilt zu beachten, dass bereits mit den wenigen vorgestellten Elementen ein Großteil der statischen und dynamischen Elemente verschiedener Arten von Bedienbildern beschrieben und im look-and-feel des Integrationssystems instanziiert werden kann.

Ein kritischer Punkt für die Akzeptanz dieses Ansatzes ist sicherlich der Aufwand für die (einmalige) Implementierung in den Engineeringsystemen und der zusätzliche Aufwand für die Modulingenieure zur automatischen Generierung der Instanz-Modelle. Im Projekt DIMA wurden Exportschnittstellen für CodeSys als Modulengineering-Werkzeug und Importschnittstellen für PCS 7 als Integrationsengineeringwerkzeug prototypisch implementiert. Mit den verfügbaren kommerziellen und open-source-Libraries für XML und GraphML waren Export und Import schnell realisiert und das

Entwicklungsteam konnte sich auf die werkzeugspezifischen Datenstrukturen und Application Programming Interfaces konzentrieren. Da die Datenstrukturen des Modul-HMI-Engineering-Systems nicht verändert werden sollten, wurde die für den Export notwendige Rolleninformation als Präfix des Symbolnamens angegeben, beispielsweise *NA_Pump01%P001* für das in Abbildung 53 aufgeführte Bedienbild-Element einer Pumpe.

Die automatische Erstellung der Kanten der Hierarchie-Relationen zwischen den Elementen erfordert größere Eingriffe in das Modul-HMI-Engineeringsystem, wenn diese nicht automatisch aus dem Kennzeichnungssystem rekonstruiert oder über Objektverweise auf eine technologische Hierarchie entnommen werden können. Da diese Relationen erst dann benötigt werden, wenn sich Bildschirme und Bedienbildelemente der Quell- und Zielsysteme bezüglich der Formfaktoren deutlich unterscheiden und auf dem Zielsystem ein automatischer Layoutlauf vorgenommen werden muss, wurden diese Beziehungen als Elemente des HMI-Graphen im Sinne einer 20:80-Lösung als optional klassifiziert.

Zusammenfassung & Bewertung

Die Überführung der Modul-Informationsmodellierung in einen Informationsträger im Rahmen des Projektes DIMA erfolgte zunächst mit Hilfe verschiedener, zum Teil proprietärer Formate. Die Machbarkeit konnte bereits in einem Prototyp nachgewiesen werden (siehe Abschnitt 5.3). Die Abbildung der untersuchten Integrationsaspekte: Prozesswerte, Verhalten und HMI konnten erfolgreich dargestellt werden. Für die Beschreibung der Bedienbilder kam das Format GraphML zu Anwendung. Durch die Kapselung der einzelnen Integrationsaspekte in einem Container mit unterschiedlichen Beschreibungsmitteln, können diese in Zukunft flexibel voneinander und unabhängig weiterentwickelt werden oder sogar gegen andere Beschreibungsmittel getauscht werden. Im Rahmen der NAMUR MTP Entwicklung wurde dieser Ansatz aufgegriffen und einzelne Beschreibungsmittel bereits aufgrund der besseren Akzeptanz der Anwender erfolgreich durch AutomationML (Bernshausen et al., 2016; Holm et al., 2016) ersetzt.

4.4.3 OPC UA Ansatz[3]

Die in den letzten beiden Kapiteln vorgestellten Ansätze beschränken sich auf eine Abbildung des Informationsmodells in einer, getrennt vom eigentlichen Modul, separaten Datei. Im Folgenden soll das, zur Integration des Moduls notwendige Informationsmodell, direkt aus dem Modul geladen werden können (siehe Abbildung 54) Als Basis dient hier die Weiterentwicklung des DIMA MTP - Ansatzes (Holm et al., 2014; Obst et al., 2015a) aus Abschnitt 4.4.2 durch den Namur-Arbeitskreis AK 1.12, in Zusammenarbeit mit verschiedenen Ad-Hoc-Arbeitskreisen der Namur und dem ZVEI (Bernshausen et al., 2016; Holm et al., 2016), bei der das Informationsmodell des Modul-Informationsträgers auf AutomationML abgebildet wird. Der im folgenden Abschnitt dargelegte Ansatz untersucht, inwieweit

[3] Das in Kapitel 4.4.3 dargestellte Konzept zur Abbildung von eines Modul Informationsträgers in OPC UA wurden erstmals in der vom Autor betreuten studentischen Abschlussarbeit (Wassilew (2015)) veröffentlicht.

sich dieses Modell anschließend auf ein OPC UA Informationsmodell überführen lässt, da OPC UA als Schlüsseltechnologie im Rahmen von Industrie 4.0 (acatech, 2013; Drath & Koziolek, 2015) gilt und bereits zahlreiche Steuerungen und deren Informationsmodelle über OPC UA zugänglich sind.

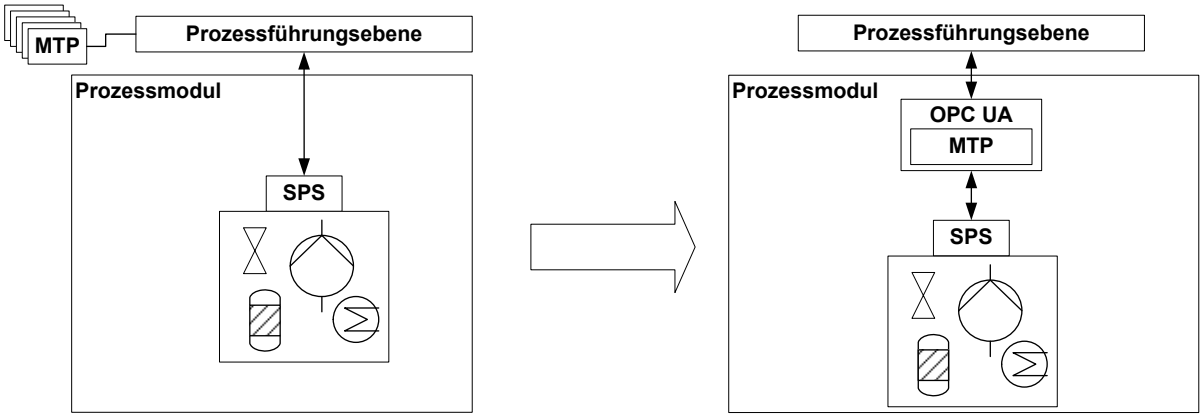


Abbildung 54: Vergleich der Integrationsansätze (Wassilew, 2015)

Das hier vorgestellte Konzept baut auf dem von (Henßen & Schleipen, 2014) entwickelten Ansatz auf, der eine allgemeine Abbildung von AutomationML auf OPC UA formulieren. Für ein besseres Verständnis wird dieser Ansatz hier zunächst kurz skizziert. In Abbildung 55 und Abbildung 56 wird dargestellt, dass die wesentlichen Kernelemente von AutomationML in OPC UA modelliert werden können. Die erste Abbildung zeigt dabei die Beziehungen zwischen den Klassenbibliotheken, während in der zweiten die Beziehungen der Bibliotheken mit den Schnittstellen aufgezeigt werden.

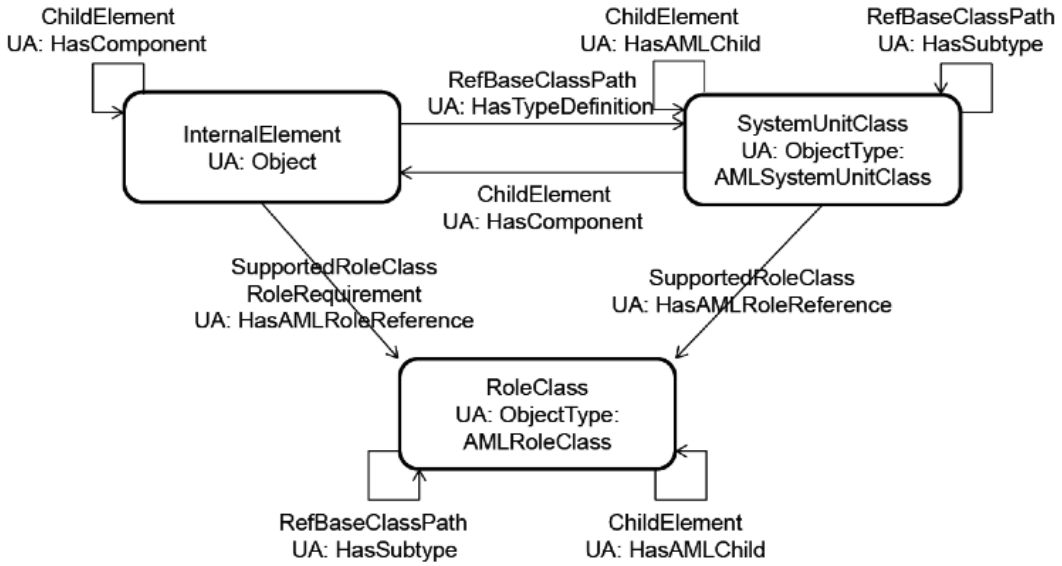


Abbildung 55: Abbildung von AutomationML auf OPC UA - Teil 1 (Henßen & Schleipen, 2014)

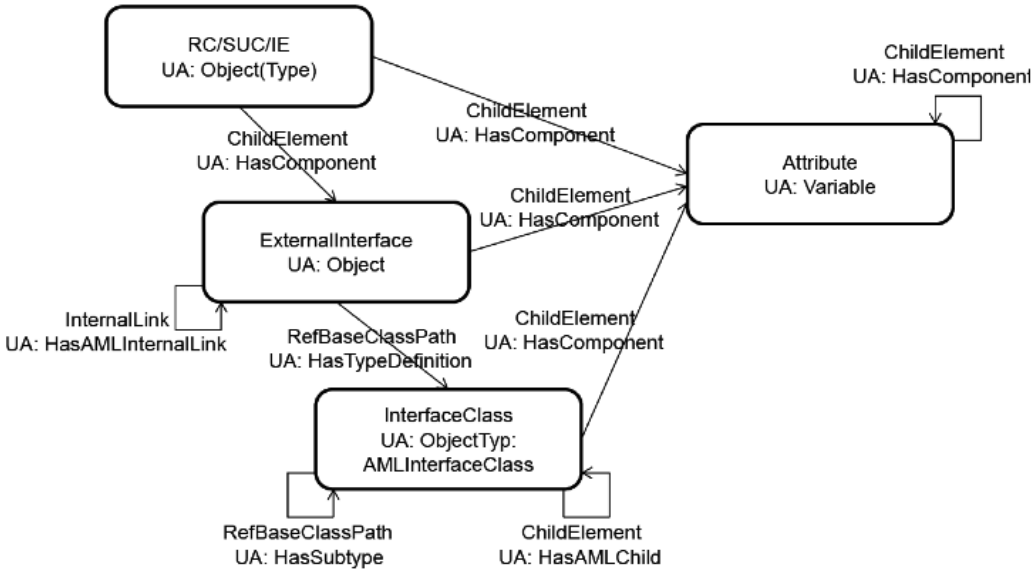


Abbildung 56: Abbildung von AutomationML auf OPC UA - Teil 2 (Henßen & Schleipen, 2014)s

Jeder AutomationML-Beziehung ist darunter die korrespondierende OPC UA-Beziehung zugeordnet. Einige dieser OPC UA-Beziehungen mussten neu erstellt werden, um AutomationML korrekt abbilden zu können. Diese sind in Abbildung 57 aufgeführt und werden anschließend erläutert.

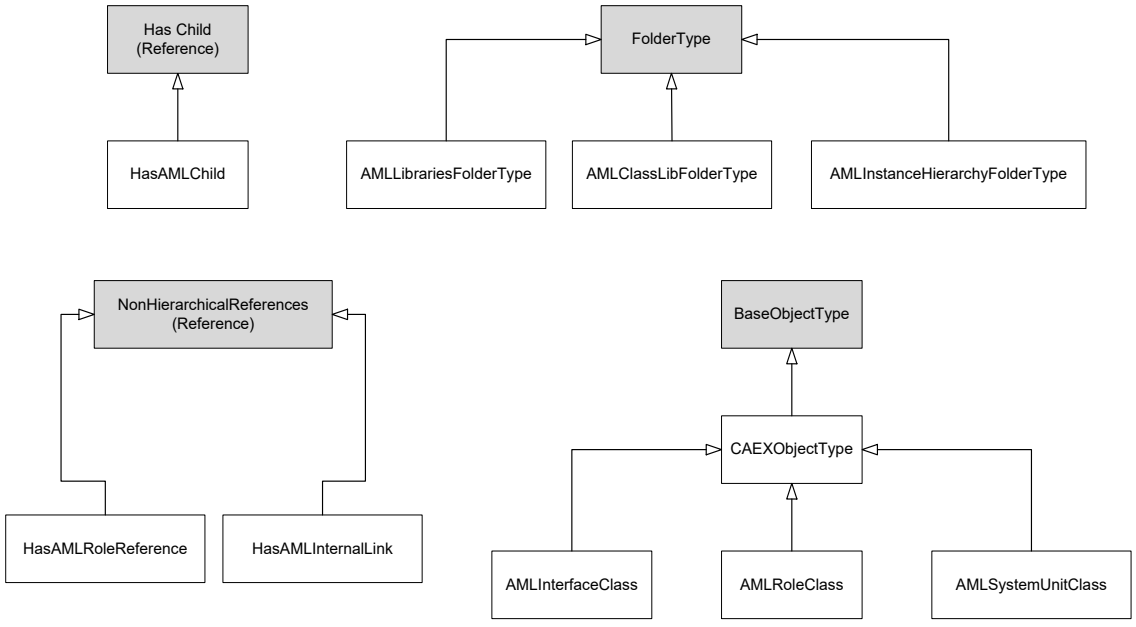


Abbildung 57: OPC UA Typen (Wassilew, 2015)

- *HasAMLChild*: Realisiert hierarchische Relationen innerhalb von AutomationML
- *HasAMLRoleReference*: Realisiert SupportedRoleClass-Beziehung
- *HasAMLInternalLink*: Realisiert *InternalLink*-Beziehung zwischen zwei Schnittstellen (*ExternalInterface*)
- *AMLLibrariesFolderType*: Ordner für AutomationML Bibliotheken, die das gleichzeitige Abbilden mehrerer verschiedener AutomationML Dateien ermöglicht
- *AMLClassLibFolderType*: Typ für die Abbildung der drei AutomationML Bibliotheken.

- *AMLInstanzHierarchyFolderType*: Typ für die Abbildung der *InstanceHierarchy*.
- *CAEXObjectType*: Dieser Objekttyp bildet mit seinen Sub-Typen die grundlegenden Klassen *RoleClass*, *InterfaceClass* und *SystemUnit*- Class ab.

Die drei Referenztypen *HasAMLChild*, *HasAMLRoleReference* und *HasAMLInternalLink* sind dabei von Standard-OPC UA-Referenztypen abgeleitet.

Wie diese Ausführungen zeigen, existieren für die wesentlichen Zusammenhänge in AutomationML entsprechende Pendants in OPC UA oder können relativ einfach geschaffen werden. Dennoch ist eine direkte Übernahme des Modells nicht sinnvoll, da spezielle Bestandteile des Modells günstiger dargestellt werden können. Als Grundlage ist es jedoch geeignet, weil so spätere Erweiterungen des Namur-MTP einfach auf OPC UA abgebildet werden können. Für die Überführung eines AutomationML basierenden Informationsträgers, kann das Vorgehen entsprechend Abbildung 58 angewendet werden.

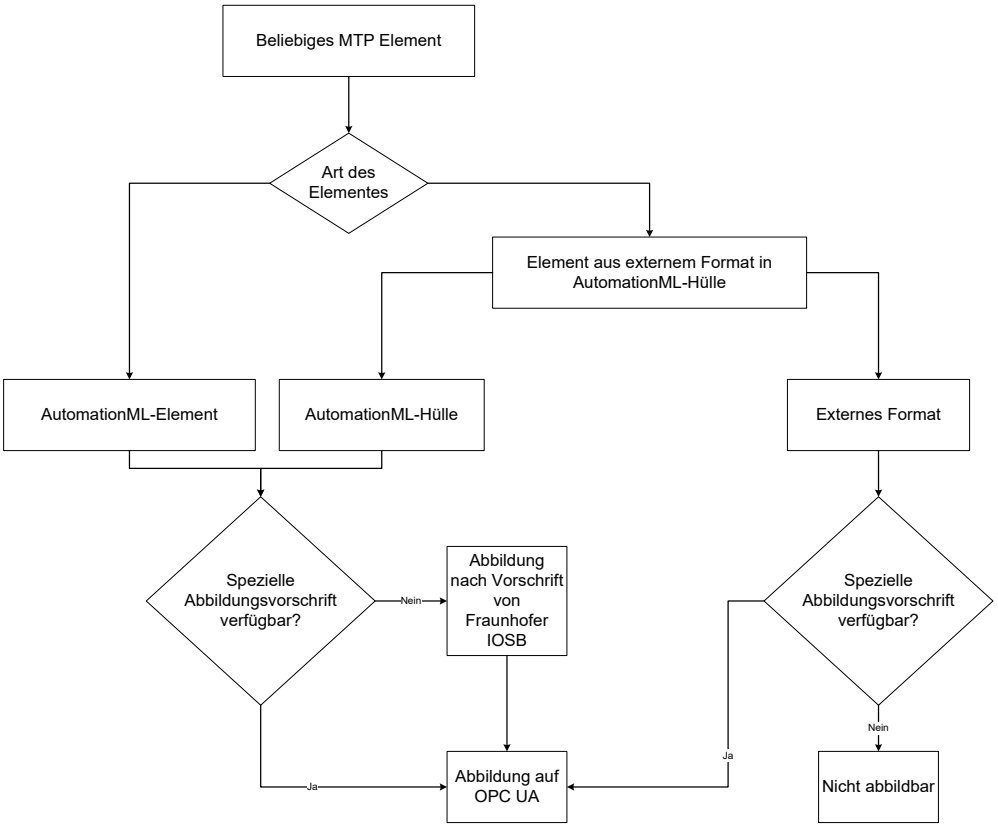


Abbildung 58: Vorgehen bei Abbildung von AutomationML auf OPC UA (Wassiliew 2016)

Zunächst gibt es zwei Arten von Elementen im Informationsträger: Reine AutomationML-Elemente und AutomationML-Elemente, die als eine Art Hülle fungieren und Informationen sowie Referenzen auf externe Formate besitzen können. Sämtliche Elemente werden dabei auf eine spezielle Abbildungsvorschrift geprüft. Existiert keine Vorschrift, so wird für AutomationML-Elemente die Variante nach (Henßen & Schleipen, 2014) angewandt. Im Falle externer Formate ist eine spezielle Abbildungsvorschrift Voraussetzung, sonst ist die Information nicht in OPC UA abbildbar. Ein detaillierte Beschreibung der Abbildungsvorschriften kann (Wassilew, 2015) entnommen werden. Im

Folgenden wird auszugsweise die spezifische Abbildung des Teilbereiches für die Prozessführung erläutert.

Dienste und Dienste-Typen

Die Darstellung des Dienste-Objektes bzw. der Klassen und deren Unterobjekte erfolgt zunächst entsprechend der allgemeinen Abbildungsvorschrift. Die bereits in der Dienste-Rollenklasse definierten Schnittstellen Read und Write müssten als OPC UA - Object abgebildet werden. An dieser Stelle erfolgt eine Modifizierung. Bei der Schnittstelle der Modulsteuerung zum OPC UA Server handelt es sich um eine Menge von Datenwerten deren passendes OPC UA-Äquivalent der Variable-Knotentyp ist. Alle Schnittstellen, die als Module OPCVariableInterface bezeichnet werden, dienen der Kopplung mit diesen Datenwerten der Modulsteuerung. Read und Write sind ebenfalls von diesem Typ. Es bietet sich daher an, statt der Abbildung als Object den Variable - Knotentyp zu verwenden.

Dienste-Parameter und Dienste-Parameter-Typen

Dienste-Parameter bzw. deren Typen werden analog zu den Diensten abgebildet. Anschließend erfolgen zwei Modifikationen. Zuerst werden die Schnittstellen zur Modulsteuerung als Variable-Knoten dargestellt und nicht als Object. Dies geschieht aus denselben Gründen wie mit den Dienste-Schnittstellen Read und Write und auch analog dazu, da es sich um denselben Schnittstellentyp handelt. Die andere Modifikation betrifft die Darstellung, insbesondere der durch Dienste-Parameter-Typen hinzukommenden Attribute. Ein solcher Typ, beispielsweise ein begrenzter Float-Wert, kann die Attribute Min und Max als Entsprechungen für untere und obere Grenzwerte ergänzen. Diese zusätzlichen Attribute sind nicht durch den *Variable*-Knoten nach (IEC 62541-1) abbildbar. Um sicherzustellen, dass diese Grenzwerte beispielsweise bei Werteingabe eingehalten werden, muss eine Prüfung des Wertes erfolgen, der entweder ignoriert und durch einen anderen gültigen Wert ersetzt werden muss. Zur Lösung dieses Problems wird eine OPC UA Methode mit der entsprechenden Funktionalität erstellt und an die Klasse des Parameter-Typen gekoppelt, der diese Methode verwendet. Der OPC Server ruft diese bei jeder Wertänderung eines Knotens, der die *HasAMLRoleReference*-Referenz zu dem entsprechenden Typen besitzt, auf.

Zusammenfassung der Abbildung von AutomationML auf OPC UA

Die Darstellung in OPC UA erfolgt individuell für jedes MTP-Element. Es existieren zwei Formen von Abbildungsvorschriften für AutomationML-Elemte: Eine spezifische und eine allgemeine. Letztere kommt immer zum Einsatz, falls ein in AutomationML vorliegendes MTP-Element keine spezifische Vorschrift besitzt. Diese allgemeine Vorschrift basiert auf einem vom Fraunhofer IOSB ausgearbeitetem Konzept (Henßen & Schleipen, 2014). Mit der online Verfügbarkeit des Informationsträges aus dem Modul heraus und der damit verbundenen definierten Kommunikationsfähigkeit des Moduls über seine Daten und Funktionen, kann dieser Ansatz bereits ein Schritt in Richtung einer Verwaltungsschale einer Industrie 4.0 Komponente darstellen, wie sie in Abschnitt 2.4 vorgestellt wurde.

5 Praktische Untersuchungen

Für die Überprüfung der vorgestellten Methodik muss diese an verschiedenen Stellen implementiert werden. Betrachtet man die aufgestellte Architektur (Abschnitt 4.1), so ist eine durchgängige Implementierung an drei Punkten notwendig. Zum einen muss der Informationsträger selber erstellt werden, zum anderen muss dieser aus dem Engineering Informationen über das Modul erhalten und kann anschließend in die Prozessführungsebene integriert werden. Im Folgenden wird zunächst die Integrationsfähigkeit verschiedener Leitsysteme untersucht. An einem Demonstrator wird exemplarisch der Informationsträger erstellt und in die Prozessführungsebene integriert.

5.1 Bewertung der Integrationsfähigkeit der aktuellen Prozessleitsysteme

Für die Umsetzung des aufgestellten Konzeptes ist eine Integration desselben in verschiedene Prozessleitsystem als Realisierung der Prozessführungsebene notwendig. Hierfür wurden im Rahmen des Projektes DIMA verschiedene Prozessleitsystemanbieter hinsichtlich ihrer Anwendbarkeit überprüft, deren Ergebnis wurden erstmals in (Holm et al., 2016) veröffentlicht. Dazu wurden in einem ersten Schritt zunächst die, sich aus dem Konzept geforderten technischen Anforderungen, mit den Funktionen der PLS-Produkte verglichen. Die untersuchte Basis der Prozessleitsystemanbieter ergibt sich zum großen Teil aus der Übersicht aus (SPS-MAGAZIN, 2014), diese wurde um Information ergänzt, die über die Web-Präsenz (i-need.de, 2016) zu erhalten sind. Aufgrund der Aktivitäten aller großen Prozessleitsystemanbieter in den bereits laufenden Namur MTP Aktivitäten (Bernshausen et al., 2016; Holm et al., 2016), wurden diese Systeme im Rahmen der Umfrage nicht näher betrachtet. Insgesamt wurden über 60 PLS Anbieter kontaktiert. Ein Teil der Anbieter lehnten die Befragung an bzw. wurden aus technischen Gründen ausgeschlossen, da sie veraltete und nicht mehr weiterentwickelte Produkte verwendeten, eine starke Fokussierung auf eine Anwendungsdomäne außerhalb der Prozessindustrie aufwiesen oder ausschließlich SCADA Systems ohne Batch-Funktion im Einsatz hatten. Im Ergebnis der Umfrage wurde das System zenon[4] der Firma Copa-Data ausgewählt.

5.2 Beschreibung des Versuchsszenarios

Zur Überprüfung der Softwarekomponenten wurde im Rahmen des Projekts DIMA ein Anlagendemonstrator entwickelt, der die Anforderungen der Prozessindustrie im Labormaßstab umsetzt und auf der Messe *sps ipc drives* Ende November 2015 vorgeführt werden konnte (siehe Bild 7). Die Ergebnisse wurden erstmals in (Holm et al., 2016) veröffentlicht. Die Anlage besteht aus 4 verschiedenen Modulen:

1. ein Mischmodul, welches Stoffe aus drei Behältern in einem parametrierbaren Verhältnis vermischt,

[4] *www.copadata.com/de/**zenon**/*

2. ein Destilliermodul,
3. ein Filtermodul mit Partikelfilter,
6. ein Abfüllmodul zum Abfüllen von Flüssigkeiten in Behälter.

Zudem ist ein Backbone entwickelt worden, der die benötigte Spannungs- und Druckluftversorgung, sowie die Netzwerkinfrastruktur zur Kommunikation zwischen Modulen und Prozessführungsebene bereitstellt.

Die Steuerungslogik und die Bedienbilder aller Module wurden in *e!Cockpit* entwickelt. Jedes Modul stellt verschiedene Dienste zur Verfügung, sämtliche Modulinformationen wurden in generierten MTPs hinterlegt. In einem ersten Schritt sollte der in Abbildung 59 dargestellte Prozess realisiert werden.

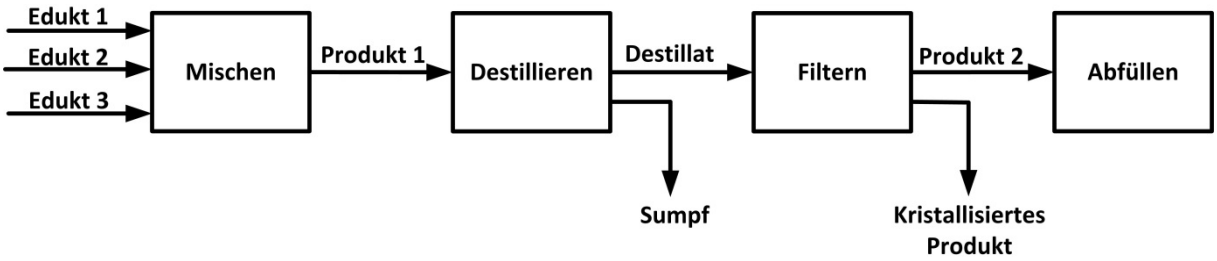


Abbildung 59: Grundprozess des Anlagendemonstrators (Holm et al., 2016)

Drei Edukte werden im Mischmodul im Verhältnis 1:1:1 miteinander vermengt. Hierfür stellt das Modul einen Dienst „Mischen“ zur Verfügung, in dem das Mischverhältnis von der Prozessführungsebene parametriert werden kann. Das entstandene Produkt wird anschließend destilliert (Dienst „Destillieren“ des Destilliermoduls), durch den Dienst „Filtern“ im Filtermodul vom Farbstoff befreit und anschließend abgefüllt (Dienst „Abfüllen“ im Modul Abfüllen). Die Erstellung des benötigten Rezeptes erfolgte in *zenon* und entspricht einem gleichzeitigen parallelen Aufruf der genannten Dienste. Zum Initialisieren der Module können entsprechende Grundfunktionen der Module genutzt werden, die ebenfalls im MTP als Dienste vorgehalten werden.

In einem Szenario soll das Filtermodul aus der Anlage entfernt werden. Ein möglicher Grund hierfür wäre bspw. der Bedarf, das Modul in einer anderen Anlage zu verwenden, in Verbindung mit der Feststellung, dass es in diesem Prozess für die aktuell geforderte Produktqualität verzichtbar ist. Zur Entnahme des Moduls wird zunächst der Prozess durch Stoppen des Rezeptes beendet. Um das Filtermodul entfernen und wiederverwenden zu können, muss es gereinigt und entleert werden. Hierfür bietet das Modul die Dienste „Abkoppeln“ und „Entleeren“ an. Dafür wird ein entsprechendes Rezept erstellt und ausgeführt. Nun kann das Modul sowohl physikalisch als auch aus der Prozessführungsebene entfernt werden. Anschließend wird eine Rohrverbindung zwischen dem Destilliermodul und dem Abfüllmodul hergestellt. Der Dienst „Destillieren“ wird aus dem Rezept im PLS entfernt und die Anlage kann wieder betrieben werden.

5.3 Überführung der Methodik in die Praxis mit DIMA

Um die Anwendbarkeit des Ansatzes und den Nutzen des MTP als Modulbeschreibung aufzeigen zu können, wurde im Rahmen des DIMA-Projektes eine Umsetzung anhand eines Anlagendemonstrators vorgenommen (Holm et al., 2016). Die Umsetzung beinhaltet sowohl das Modul-Engineering als auch das Prozessführungsebenen-Engineering. Dabei wird die Erstellung der Automatisierungssoftware in gewohnter IEC 61131-3-Umgebung vorgenommen. Aus den im Engineering-Tool erzeugten Informationen wird das MTP vollautomatisch generiert. Die Integration findet anhand eines Prozessleitsystems statt, in welches das MTP eingelesen werden kann. Bedienbilder werden automatisch anhand der Beschreibung im MTP erzeugt, die Dienste als Batch-Phasen zur Verfügung gestellt. Diese können dann zum Gesamtprozess als Rezept orchestriert werden.

Die Umsetzung wurde hierbei mit zwei verschiedenen Software-Werkzeugen realisiert:

1. dem Engineering-Werkzeug *e!Cockpit*[5] der Firma WAGO zur Durchführung des Modulengineerings und zur Erstellung von MTPs;
2. dem Prozessleitsystem *zenon*[6] der Firma COPA-DATA zur Integration mehrerer Module mittels MTP, sowie zur Orchestrierung der Modul-Dienste im PFE-Engineering.

Des Weiteren wurde ein Anlagendemonstrator geplant und aufgebaut, der die Vorteile des Konzeptes aufzeigt und die Anforderungen der Anwender heute schon abbildet. Alle Teile der bisherigen Umsetzung sollen im Folgenden aufgezeigt werden.

5.3.1 Modulautomatisierung mit *e!Cockpit*

Aufgabe des Modul-Engineerings ist es, die Modullogik in sich geschlossen zu entwerfen, entsprechende Schnittstellen in Form von Diensten bereitzustellen, sowie nötige Informationen zur Visualisierung des Moduls zu erzeugen. Diese Tätigkeiten wurden im Rahmen des DIMA-Projektes mit dem Wago Engineering Werkzeug *e!Cockpit* durchgeführt. Die Applikationsentwicklung der Modul-Steuerung findet in den Programmiersprachen der IEC 61131-3 statt. Dienste werden als Funktionen oder Funktionsbausteine in einem hierfür vorgesehenen Ordner „Services“ erstellt (siehe Abbildung 60).

[5] http://www.wago.de/produkte/neuheiten/uebersicht/engineering-software.jsp
[6] *www.copadata.com/de/**zenon**/*

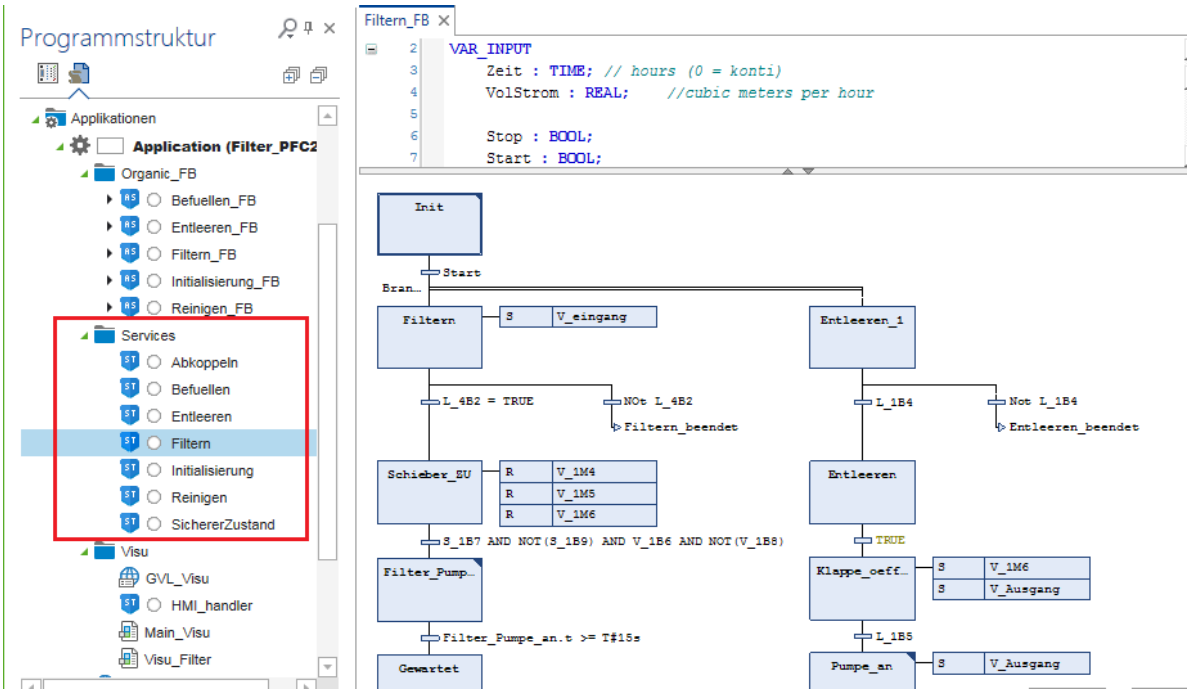


Abbildung 60: Dienste im Ordner Services (links) in e!Cockpit (Holm et al., 2016)

Die Dienste enthalten ihre Phasenlogik, Parametervariablen zur Parametrierung, sowie Funktionsbaustein- und Funktionsaufrufe. Die aufgerufenen Funktionsbausteine und Funktionen enthalten die eigentliche Steuerlogik. Nach der Erstellung der Steuerapplikation und der Dienste, die das Modul anbieten können soll, wird die Bedienoberfläche entwickelt. Bedienbilder werden unter Nutzung von Templates erarbeitet und mit Variablen aus der Applikation verknüpft. (vgl. Abbildung 61 - links). Damit ist die Erstellung des Programmcodes der Modul-Steuerung abgeschlossen. Die automatische Generierung des MTP schließt das Modul-Engineering ab. Hierfür beschreibt der Benutzer die Metadaten des MTP (Hersteller, Version, Bild, kurze Beschreibung) und wählt die Dienste und Bedienbilder aus, die im MTP enthalten sein sollen. Gerade letzteres ermöglicht den Aufbau eines Variantenmanagements beim Modulhersteller. Er kann alle denkbaren Dienste im Programmcode vorbereiten und nur solche durch den MTP publizieren, die auch von einem Kunden bestellt wurden.

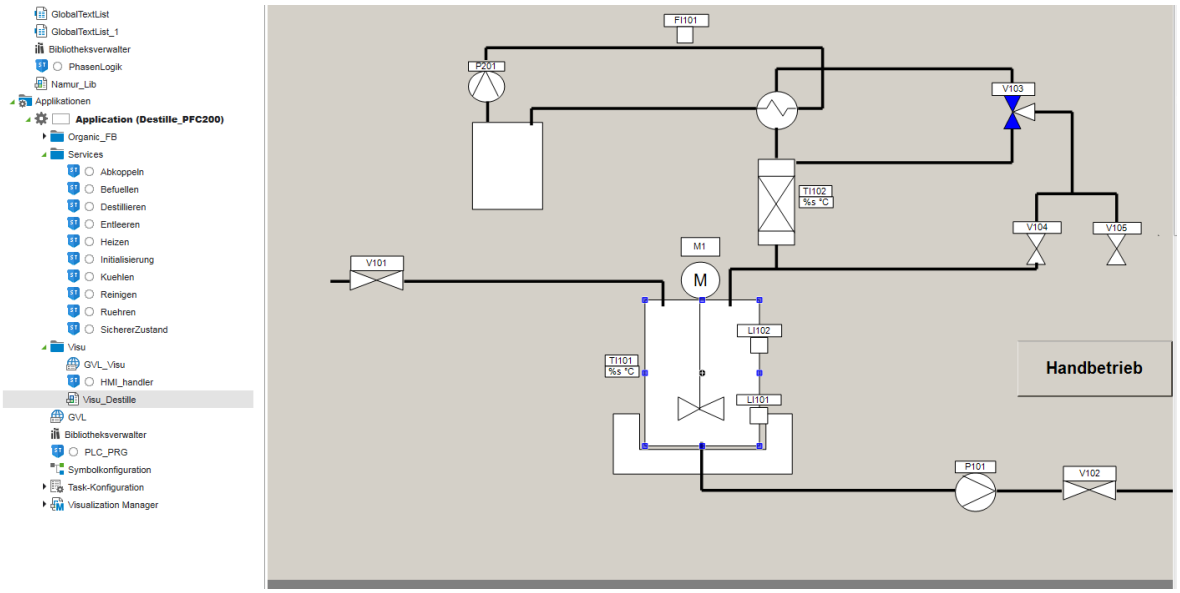


Abbildung 61: Darstellung der Bedienbilderstellung (Holm et al., 2016)

5.3.2 Integrationsengineering in der Prozessführungsebene mit dem PLS *zenon*

Das PFE Engineering wird durch den Anlagenbauer/-betreiber durchgeführt. Dieser nutzt die vom Modulhersteller in *e!Cockpit* generierten MTPs zur Integration in das Prozessleitsystem (PLS). Im Rahmen dieses Projektes wurde mit dem PLS zenon von COPA-DATA ein flexibles, offenes und skalierbares PLS ausgewählt. Die Integration von Modulen geschieht hier ausschließlich über die MTPs. Dazu wurde ein MTP-Wizard entworfen, über den die MTPs eingelesen werden. Der Wizard legt alle benötigten Variablen und Bedienbilder, sowie deren Verknüpfungen gemäß den Informationen im MTP an, wodurch eine direkte Kommunikation zwischen Leitsystem und Modul-Steuerung aufgebaut werden kann.

Die Darstellung der Bedienbilder (siehe Abbildung 62) ist hierbei leitsystemspezifisch, da die Bildobjekte aus einer vorab erstellten Template-Bibliothek des PLS entnommen werden. Die Informationen bzgl. Namen, Größe, Position und Variablenanbindung entstammen aus dem MTP. Somit ist gewährleistet, dass Bedienbilder von Modulen unterschiedlicher Hersteller, die durch unterschiedliche MTPs eingelesen wurden, im Leitsystem dem kundenspezifischen „Look and Feel“ entsprechen.

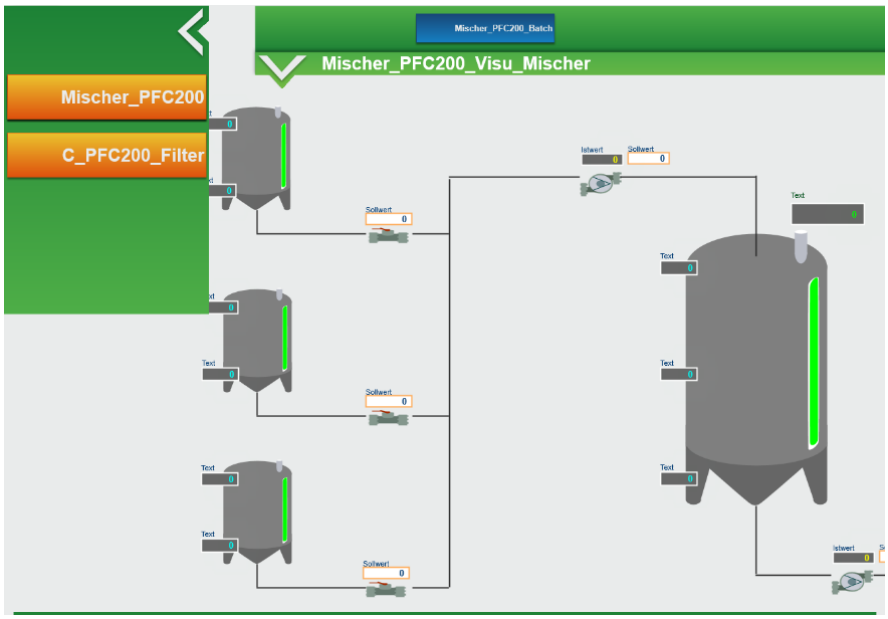


Abbildung 62: Bedienbild eines Moduls nach MTP-Import (Holm et al., 2016)

Zusätzlich zur Erstellung und Anbindung von Variablen und Bedienbildern werden die im MTP enthaltenen Dienste als Grundfunktionen im Batch-Werkzeug (vgl. Abbildung 63) des Leitsystems angelegt. Die Grundfunktionen (Abbildung 63, rechts) können, wie erforderlich parametrisiert und in Form von Rezepten (Abbildung 63, links) miteinander verknüpft werden. Nach diesem Schritt ist der Produktionsprozess der Anlage ablauffähig. Bei Bedarf können durch den angelegten Variablenhaushalt modulübergreifende Verriegelungen im Leitsystem implementiert, sowie zusätzliche Meldungen angelegt werden.

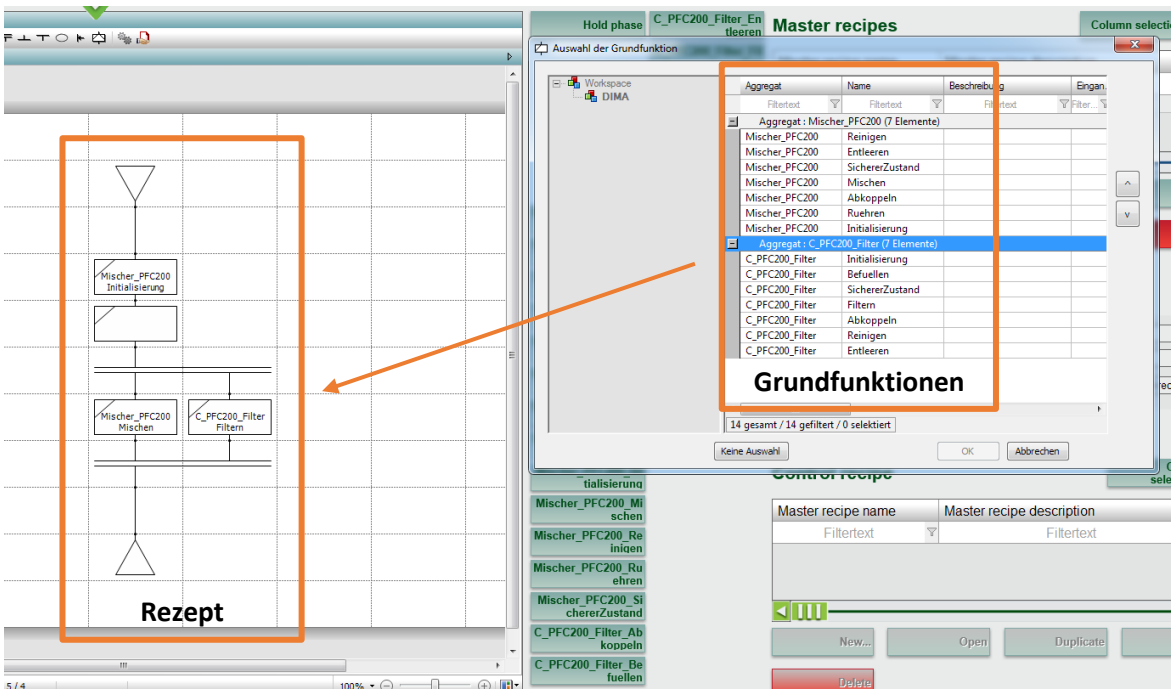


Abbildung 63: Orchestrieren der Dienste im Batch-Werkzeug nach (Holm et al., 2016)

5.4 Überführung der Methodik in die Praxis mit OPC UA

Die Überführung des OPC UA Ansatzes erfolgte prototypisch an der modularen Forschungsanlage der Professur für Prozessleittechnik im Rahmen der vom Autor betreuten Diplomarbeit von Herrn Wassilew (Wassilew, 2015). Als Basis für den Prototyp ist die Wahl auf open62541 (*open62541,* 2016) gefallen. Die Kommunikation zwischen Server und Steuerung (SPS) findet in diesem Fall über einen OPC UA Client statt, da die Steuerungen der Module bereits mit einem eigenen einfachen OPC UA Server versehen ist, der die relevanten Ein- und Ausgänge der Steuerung bereitstellt.

Die im Folgenden vorgestellten Implementierungen des OPC UA Servers basieren auf dem open62541 OPC UA Stack[7] (*open62541,* 2016). Ausschlaggebend waren der offen vorliegende Quellcode, sowie die unmittelbare Umsetzung der IEC 62541. Im Abschnitt 4.4.3 wurde die automatische Abbildung eines AutomationML basierenden Modul Informationsträges (Bernshausen et al., 2016) auf ein OPC UA Informationsmodell erläutert, sowie die allgemeine Kopplung eines solchen OPC UA Servers mit dem Modul. In diesem Abschnitt soll nun die spezifische Realisierung mit dem open62541-OPC UA Server beschrieben werden.

Die technische Realisierung des Prototypens (siehe Abbildung 64) beinhaltetet eine SPS mit eingebautem OPC UA Server, dessen Informationsmodell intern nicht verändert werden konnte und somit nur ein Zugriff auf die Steuerungsdaten, entsprechend der Konfiguration durch das Engineering System, möglich war. In dem speziellen Fall des Prototypens handelt es sich um einen OPC UA Client, welcher innerhalb des Servers als Plugin bzw. Kommunikationsmodul und nicht als separates Programm behandelt wird. Dadurch muss der Client nur eine Verbindung mit dem OPC UA Server der Modulsteuerung aufrechterhalten und kann innerhalb des Programmes Daten mit dem nach außen für den Modulnutzer sichtbaren OPC UA Server austauschen. Das Auslesen der Statussignale erfolgt mit Hilfe von Subscriptions und deren Benachrichtigungen bei Wertänderung.

[7] Genutzt wurde die open62541-Implementierung vom master-branch des 09.09.2015.

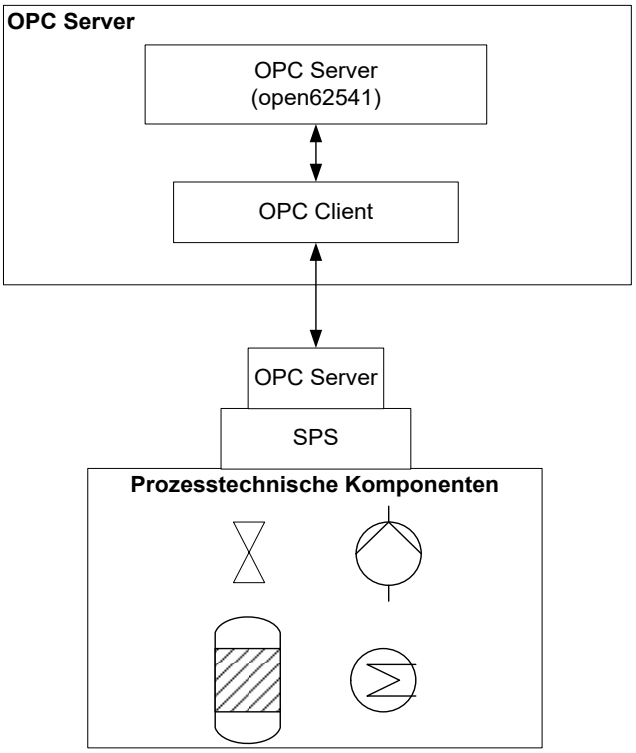


Abbildung 64: Kommunikation zwischen OPC UA Server und Modulsteuerung im Prototyp (Wassilew, 2015)

Der Prototyp ist am Reaktormodul der modularen Forschungsanlage zum Einsatz gekommen. Bei entsprechender Erweiterung kann davon ausgegangen werden, dass ein auslesbarer MTP zum Zeitpunkt des Initialisierungsvorgangs eine dynamische Generierung der OPC UA Struktur ermöglicht. Die erstellten Knoten werden ebenfalls selbstständig mit der unterliegenden Steuerung verknüpft, somit ist ein direktes Steuern des Moduls über den OPC UA Server möglich. Im Vergleich zu einer Implementierung mit einem Datei-basierten Informationsträgers (z.B. Projekt DIMA, Namur MTP) ergeben sich folgende Vorteile.

I. Der Informationsträger muss der Prozessführungsebene (PFE) nicht separat zugänglich gemacht werden, sondern kann innerhalb des Moduls verbleiben und direkt durch die PFE beim Ankoppeln ausgelesen werden.

II. Der OPC UA Server dient als alleinige Kommunikationstechnologie zwischen PFE und Modul. Eine Unterstützung weiterer Technologien von Seiten der PFE ist nicht mehr erforderlich. Gleichzeitig bildet das eine Grundlage für eine mögliche zukünftige dezentrale Kommunikation zwischen Modulen.

5.5 Zusammenfassung der praktischen Realisierung

Im Rahmen der praktischen Untersuchung konnten zwei Wege der Integration eines Moduls in die Prozessführungsebene demonstriert werden. Die Realisierung im Rahmen des Projektes DIMA stellt eine erfolgreiche Umsetzung eines dateibasierten Informationsträgers dar. Es konnte sowohl die automatische Generierung des Informationsträgers im Engineeringsystem für das Modul, als auch der automatische Import des Informationsträgers im Automatisierungssystem der Gesamtanlage erfolgreich implementiert werden. Damit steht ein grundsätzliches Vorgehen zu Erstellung eines Modulinformationsträgers zu Verfügung, das bereits durch NAMUR und ZVEI in der weiteren Umsetzung befindet (Bernshausen et al., 2016).

Der zweite Weg zeigt eine grundsätzliche Möglichkeit auf, den Informationsträger innerhalb eines OPC UA Informationsmodells abzubilden und damit einen Schritt in Richtung der Implementierung einer Industrie 4.0 Verwaltungsschale darzustellen.

Beide Implementierungen sind zunächst prototypisch erfolgt. Für eine umfassende Bewertung vor allem der zweiten Implementierung müssen noch weitere Integrationsaspekte abgedeckt werden. Die anlagentechnische Umsetzung erfolgte an Laborforschungsanlagen und stellen damit keinen industriellen Maßstab dar. Mit der Umsetzung mit industriell genutzten Automatisierungssystemen konnte dennoch die grundsätzliche Machbarkeit nachgewiesen werden.

6 Zusammenfassung

Die Integration eines Moduls in ein übergeordnetes Leitsystem ist mit den heute verfügbaren Informationsmodellen und Werkzeugen mit großen manuellen Aufwänden verbunden. Verschiedene Aspekte der Automatisierung wie Bedienbilder, Zustände von Abläufen oder Verriegelungen müssen für die Visualisierung und Führung des Moduls in einem übergeordneten Leitsystem manuell nachgebildet werden. Jedoch sind heutige Leitsysteme nicht dafür vorbereitet, das geforderte flexible Führen einer aus verschiedenen Modulen aufgebauten Anlage zu ermöglichen. Hierzu ist eine modulare Plug-and-Produce Methodik notwendig. Für diese Methodik wird eine durchgängige Informationsmodellierung, beginnend bei einem modularen, funktionsorientierten, integrierten Engineering benötigt.

Diese Arbeit setzte sich zum einen mit der Auswahl der Integrationsaspekte, der Modellierung im Einzelnen und deren Integration in die Prozessführungsebene auseinander. Zum anderen soll die konkrete Auswahl eines oder mehrerer Beschreibungsmittel unterstützt werden.

Zunächst wurden Beschreibungsmittel hinsichtlich ihrer Eignung zur Modellierung der zuvor diskutierten Integrationsaspekte analysiert. Diese Betrachtung zeigt auf, dass spezifische Formate für ihren jeweiligen Integrationsaspekt sehr gut anerkannt sind und bereits existierenden Schnittstellen in den Prozessleitsystemen genutzt werden können. Jedoch bedarf es, für jeden Integrationsaspekt eines anderen Beschreibungsmittels, sowie eines geeigneten Containers. Allgemeinere Formate decken einen großen Bereich der Integrationsaspekte ab, sind jedoch in der Modellierungstiefe der Integrationsaspekte allgemein gehalten. Die zukünftige Weiterentwicklung der verwendeten Komponenten bei der Integration eines Moduls, wie die eingesetzte Software und Hardware in Prozessleitsystem und im Modul, machen die Verwendung eines flexibel erweiterbaren Formates, entweder als Container mit mehreren spezifischen Formaten oder eines einheitlichen erweiterbaren Formates notwendig. Diese Erweiterbarkeit muss auch auf Ebene der Integrationsaspekte möglich sein, so dass in Zukunft noch weitere Aspekte berücksichtigt werden können.

Weiterhin wurde eine einheitliche Integrationsarchitektur beschrieben, mit der eine Integration eines Moduls in eine Prozessführungsebene ermöglicht wird. Grundgedanke einer solchen Integrationsarchitektur ist die Minimierung des Aufwands während der Integration eines Moduls in die Anlage. Dies kann durch die Verlagerung geeigneter Aufwände in das Engineering des Modullieferanten erreicht werden. Alle Informationen die in einem solchen Modul-Engineering erarbeitet und während des Gesamtanlagen-Engineerings benötigt werden, sind in einem Informationsträger zu hinterlegen.

Dafür war es zunächst wichtig den Informationsumfang der einzelnen Integrationsaspekte für einen solchen Informationsträger zu beschreiben. Näher betrachtet wurden dabei die Apekte Verriegelungen und Abläufe, die zustandsorientierte Prozessführung, sowie das Bedienen und Beobachten.

Für die Einordnung der einzelnen Integrationsaspekte in einen strukturierten Ablauf wurde darufhin einen Integrationsprozess vorgestellt, der einen Überblick über die notwendigen Integrationsschritte eines Moduls in die Prozessführungsebene gibt.

Bei der konkreten Modellierung eines Moduls als Informationsträger, können eine Vielzahl von Beschreibungsmittel zum Einsatz kommen. In dieser Arbeit wurden exemplarisch drei Ansätze vorgestellt. Aufgrund der verfügbaren Schnittstellen wurden daraufhin zwei Ansätze prototypisch umgesetzt. Für eine umfassende Bewertung vor allem der zweiten Implementierung müssen noch weitere Integrationsaspekte abgedeckt werden. Die anlagentechnische Umsetzung erfolgte an Laborforschungsanlagen und stellen damit keinen industriellen Maßstab dar. Mit der Umsetzung mit industriell genutzten Automatisierungssystemen konnte dennoch die grundsätzliche Machbarkeit nachgewiesen werden.

Mit dieser Arbeit wurde aufgezeigt, dass mit der Vielzahl verfügbarer Beschreibungsmittel in den einzelnen Integrationsaspekten eine Abbildung in einem Informationsträger möglich ist. Dabei gilt es zu unterscheiden, ob für jeden Aspekt eine separate Abbildung gewählt wird, wie es beispielweise in der zweiten praktischen Implementierung mit GrapML gewählt wurde, oder ob für den gesamten Informationsträger ein einheitliches Format genutzt wird. Die Bewertung, im Rahmen der Betrachtungen zum Stand der Technik der verfügbaren Beschreibungsmittel, legt die Verwendung von AutomationML nahe. Die praktische Umsetzung und Untersuchung erfolgen hier bereits im Rahmen der Namur MTP Entwicklungen und konnten daher nicht mehr in dieser Arbeit untersucht werden. Für die zu betrachtenden Integrationsaspekte existieren in der Literatur hinreichende Anhaltspunkte. Für die wichtigsten Aspekte Bedienen und Beobachten, sowie Prozessführung, wurden detaillierte Informationsmodellierungen vorgestellt und im Rahmen der Implementierung überprüft. Für die Auswahl des oder der Beschreibungsmittel(s) wurden zwei verschiedene Möglichkeiten vorgestellt und diskutiert. Um die Anforderung der flexiblen Erweiterbarkeit zu ermöglichen, ist es unabhängig vom konkret gewählten Format von Vorteil, ein Beschreibungsmittel zu wählen, bei dem die Integrationsaspekte voneinander getrennt beschrieben werden.

Für die zukünftige Entwicklung der Prozessindustrie ist die Nutzung modularer Anlangen im Bereich der Spezialchemie und Pharmaindustrie sowie anderen stark konjunkturabhängigen Branchen essentiell, um bei zunehmender Flexibilität der Märkte und der geforderten schnelleren Vermarktung neuer Produkte zu bestehen. Im Hinblick auf die Anforderungen der Industrie 4.0 nach Vernetzung und Durchgängigkeit aller beteiligten Komponenten über den gesamten Lebenszyklus, ist eine einheitliche Schnittstelle zwischen den beteiligten Automatisierungssystemen dringend notwendig. Dieser Arbeit

liefert dafür die ersten Bausteine. Dabei ist dies nicht nur auf die Prozessindustrie beschränkt, sondern kann auch auf den Bereich der Fertigungsindustrie angewendet werden.

Für den weiteren Ausbau einer flexiblen einheitlichen Schnittstelle zwischen Prozessführungsebene und Modul muss diese einen breiten Konsens in der Anwendung finden. Dazu muss diese Schnittstelle nicht nur standardisiert werden, was zurzeit bereits mit den Aktivitäten der NAMUR und des ZVEI in Arbeit ist, es bedarf auch ersten Umsetzungen im Rahmen von Pilotanwendungen, um die industrielle Anwendbarkeit zu demonstrieren.

Literaturverzeichnis

Acatech (Kagermann, H., Wahlster, W. & Helbig, J., Hrsg.). (2013). *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Abschlussbericht des Arbeitskreises Industrie 4.0.* Zugriff am 20.12.2016. Verfügbar unter https://www.bmbf.de/files/Umsetzungsempfehlungen_Industrie4_0.pdf

Alznauer, R. (1999). *Semantisches Informationsmodell für ein ganzheitliches, rechnergestütztes Engineering am Beispiel der Prozessleittechnik*. Dissertation, RWTH Aachen. Aachen.

Baldwin, C. Y. & Clark, K. B. (2000). *Design rules*. Cambridge, Mass., MIT Press.

Bernshausen, J., Haller, A., Holm, T., Hoernicke, M., Obst, M. & Ladiges, J. (2016). Namur Modul Type Package – Definition. Beschreibungsmittel für die Automation modularer Anlagen. *atp edition - Automatisierungstechnische Praxis, 58* (1-2), S. 82–90. http://dx.doi.org/10.17560/atp.v58i01-02.554

Birk, J. (1999). *Prozeßführung*. München: Oldenbourg.

Bott, T. & Schembecker, G. (2009). *Die 50 % - Idee Vom Produkt zur Produktionsanlage in der halben Zeit*, Weinheim. Zugriff am 02.10.2014.

Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M. & Marshall, M. S. (2002). GraphML Progress Report Structural Layer Proposal. In G. Goos, J. Hartmanis, J. van Leeuwen, P. Mutzel, M. Jünger & S. Leipert (Hrsg.), *Graph Drawing* (Lecture Notes in Computer Science, Bd. 2265, S. 501–512). Berlin: Springer-Verlag. http://dx.doi.org/10.1007/3-540-45848-4_59

Brandl, D. (2007). *Design patterns for flexible manufacturing*. Research Triangle Park, N.C.: ISA.

Braune, A. & Hennig, S. (2007). Technologieunabhängiges HMI-Engineering für technische Prozesse. In *Automation 2007* .

Brendenberger, M. & Scherwietes, T. (2015). Engineering. In K. F. Früh (Hrsg.), *Handbuch der Prozessautomatisierung. Prozessleittechnik für verfahrenstechnische Anlagen* (5. Aufl.). München: Oldenbourg Industrieverl.

Brown, P., Estefan, J. A., Laskey, K., McCabe, F. G. & Thornton, D. (2012). *Reference Architecture Foundation for Service Oriented Architecture Version 1.0. Committee Specification 01* (OASIS Open, ed.). Zugriff am 20.12.2016. Verfügbar unter http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf

Buchholz, S. (2012). *F³ Factory: Eine flexible, modulare Produktionsplattform.* Zugriff am 20.12.2016. Verfügbar unter http://www.chemanager-online.com/themen/anlagenbau-komponenten/f-factory-eine-flexible-modulare-produktionsplattform

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *INTERACTING WITH COMPUTERS, 15,* 289–308.

Christen, D. S. (2010). *Praxiswissen der chemischen Verfahrenstechnik. Handbuch für Chemiker und Verfahrensingenieure* (VDI-Buch, 2. Aufl.). Berlin: Springer-Verlag.

DIN EN 61512-1 (2000). *Chargenorientierte Fahrweise Teil 1: Modelle und Terminologie*.

DIN EN 62714-2 (2015). *Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme – AutomationML – Teil 2: Rollenbibliotheken*.

Diedrich, C. (2015). Integration intelligenter Feldgeräte in PLS. In K. F. Früh (Hrsg.), *Handbuch der Prozessautomatisierung. Prozessleittechnik für verfahrenstechnische Anlagen* (5. Aufl.). München: Oldenbourg Industrieverl.

DIN/DKE (Hrsg.). (2015). *Deutsche Normungs-Roadmap Industrie 4.0.* Zugriff am 04.12.2015. Verfügbar unter https://www.dke.de/de/std/documents/nr_industrie%204.0_v2_de.pdf

Doherr, F., Urbas, L., Drumm, O. & Franze, V. (2011). Bedienbilder auf Knopfdruck. Modellbasierte Erzeugung von Fließbilddarstellungen. *atp edition - Automatisierungstechnische Praxis, 53* (11), 30–39.

Drath, R. (2010). *Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen XML und COLLADA* (VDI-Buch). Heidelberg: Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-04674-2

Drath, R. & Koziolek, H. (2015). Industrie 4.0. Im Spannungsfeld zwischen dem Machbaren und Sinnvollen. *atp edition - Automatisierungstechnische Praxis, 57* (1-2), 28–35.

ECl@ss. (2014). *eCl@ss – eine Zusammenfassung,* eCl@ss. Zugriff am 23.01.2015. Verfügbar unter http://www.eclass.eu/eclasscontent/standard/overview.html.de

Fay, A. (2009). Effizientes Engineering komplexer Automatisierungssysteme. In E. Schnieder & T. Ständer (Hrsg.), *Wird der Verkehr automatisch sicherer? Beschreibungsmittel, Methoden und Werkzeuge des integrierten Systementwurfs zur Fahrzeug- und Verkehrsautomatisierung.* Braunschweig: iVA.

Fay, A., Drumm, O., Eckardt, R., Gutermuth, G., Krumsiek, D., Löwen, U. et al. (2014). Anforderungen an Leitsysteme durch Industrie 4.0. In *Automation 2014* .

Fittler, H. (2015). Automatisierung von Chargenprozessen. In K. F. Früh (Hrsg.), *Handbuch der Prozessautomatisierung. Prozessleittechnik für verfahrenstechnische Anlagen* (5. Aufl.). München: Oldenbourg Industrieverl.

Frey, G., Minas, M. & John, K.-H. (2002). Steuerungsentwurf mit Petrinetzen. *SPS-Magazin* (4/5), S. 44–47. Zugriff am 11.02.2016.

Früh, K. F. (Hrsg.). (2015). *Handbuch der Prozessautomatisierung. Prozessleittechnik für verfahrenstechnische Anlagen* (5. Aufl.). München: Oldenbourg Industrieverl.

George, J. (2013). *Engineering von Anlagen der Prozessleittechnik mittels Austausch maschinenlesbarer Daten. Klassen und Merkmalleisten als Basis für einen integrierten Datenaustausch innerhalb der Lebenszyklusphasen von Anlagen.* Zugriff am 28.11.2016. Verfügbar unter http://files.messe.de/abstracts/51438_091030_George_Pepperl_und_Fuchs.pdf

Gonsior, T. (2008). *Standardisierung vs. Differenzierung. Beschaffungsorientierte Betrachtung der Modularisierung entlang der Wertschöpfungskette* (Beiträge zum Beschaffungsmarketing, Bd. 24). Köln: Fördergesellschaft Produkt-Marketing e.V.

Göpfert, J. (1998). *Modulare Produktentwicklung. Zur gemeinsamen Gestaltung von Technik und Organisation* (Gabler Edition Wissenschaft : Markt- und Unternehmensentwicklung). Wiesbaden: Dt. Univ.-Verl.

Göpfert, J. & Steinbrecher, M. (2000). Modulare Produktentwicklung leistet mehr. Warum Produktarchitektur und Projektorganisation gemeinsam gestaltet werden müssen. *Harvard Business Manager* (3), 20–31.

GraphML Working Group. (2006). *GraphML Specification.* Zugriff am 20.12.2016. Verfügbar unter http://graphml.graphdrawing.org/specification.html

Hady, Ł. & Wozny, G. (2012). Multikriterielle Aspekte der Modularisierung bei der Planung verfahrenstechnischer Anlagen. *Chemie Ingenieur Technik, 84* (5), S. 597–614. http://dx.doi.org/10.1002/cite.201100175

Hankel, M. (ZVEI - Zentralverband Elektrotechnik- und Elektronikindustrie e. V., Hrsg.). (2015). *Industrie 4.0: Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0).* Zugriff am 20.12.2016. Verfügbar unter http://www.zvei.org/Downloads/Automation/ZVEI-Faktenblatt-Industrie4_0-RAMI-4_0.pdf

Hawkins, W., Brandl, D. & Boyes, W. (2010). *Applying ISA-88 in discrete and continuous manufacturing* (WBF book series, v. 2). New York: Momentum Press.

Henßen, R. & Schleipen, M. (2014). Online-Kommunikation mittels OPC-UA vs. Engineering-Daten (offline) in AutomationML. In *Automation 2014* (S. 59–74). Zugriff am 04.12.2015.

Hertwig, K. & Martens, L. (2007). *Chemische Verfahrenstechnik. Berechnung, Auslegung und Betrieb chemischer Reaktoren*. München: Oldenbourg Verlag.

Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E. D., O'Connor, E. et al. (2014). *HTML5. A vocabulary and associated APIs for HTML and XHTML*. W3C Recommendation 28 October 2014 (W3C, Hrsg.). Zugriff am 23.02.2016. Verfügbar unter http://www.w3.org/TR/2014/REC-html5-20141028/

Hodgson, R., Keller, P. J., Hodges, J. & Spivak, J. (2014). *QUDT - Quantities, Units, Dimensions and Types.* Zugriff am 23.01.2015. Verfügbar unter http://www.qudt.org/

Hoernicke, M., Christiansen, L. & Fay, A. (2014). Anlagentopologien automatisch erstellen. Modelle aus der Mensch-Maschine-Schnittstelle erzeugen. *atp edition - Automatisierungstechnische Praxis, 56* (04), S. 28–40.

Hoffmeister, M. (2015). *Die Industrie 4.0-Komponente,* ZVEI. Zugriff am 15.09.2016. Verfügbar unter http://www.zvei.org/Downloads/Automation/Industrie%204.0_Komponente_Download.pdf

Holm, T. (2016). *Aufwandsbewertung im Engineering modularer Prozessanlagen*. Dissertation, Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg. Hamburg.

Holm, T., Obst, M., Fay, A., Urbas, L., Albers, T., Kreft, S. et al. (2014). Dezentrale Intelligenz für modulare Automation. Lösungsansätze für die Realisierung modularer Anlagen. *atp edition - Automatisierungstechnische Praxis, 56* (11), S. 34–43. http://dx.doi.org/10.17560/atp.v56i11.353

Holm, T., Obst, M., Ladiges, J., Urbas, L., Fay, A., Albers, T. et al. (2016). Namur Modul Type Package - Implementierung. Anwendung des Namur-MTP für Prozessanlagen. *atp edition - Automatisierungstechnische Praxis, 58* (1-2), S. 72–81. http://dx.doi.org/10.17560/atp.v58i01-02.555

I-need.de. (2016). *Prozessleitsysteme und SCADA.* Zugriff am 07.03.2016. Verfügbar unter http://www.i-need.de/?Produktkatalog=69

IEC 60050 (2014). *International Electrotechnical Vocabulary*.

IEC 62541-1 (2008). *OPC Unified Architecture - Part 1: Overview and Concepts*.

ISA 106 (2013). *Procedure Automation for Continuous Process Operations – Models and Terminology*.

John, D., Danzer, B., Riedl, M. & Zipper, H. (2013). Selbsterklärende Geräte. Selbsterklärende Geräte Abbildung semantischer Information auf Parameter mit EDD. *atp edition - Automatisierungstechnische Praxis, 55* (10), S. 56–61. Zugriff am 15.06.2015.

Kainz, G., Keddis, N., Pensky, D., Buckl, C., Zoitl, A., Pittschellis, R. et al. (2013). AutoPnP – Plug-and-produce in der Automation. Wandelbare Fabrik als cyber-physisches System. *atp edition - Automatisierungstechnische Praxis, 55* (4), S. 42–49.

Kastens, U. & Kleine Büning, H. (2005). *Modellierung. Grundlagen und formale Methoden*. München: Hanser.

Kühnel, A. (2008). *Visual C# 2008. Das umfassende Handbuch* (Galileo computing, 4. Aufl.). Bonn: Galileo Press.

Kumpfmüller, H.-G. & Lange, R. (2010). FDI Device Integration – Best of Both Worlds. *atp edition - Automatisierungstechnische Praxis, 52* (6), S. 16–19.

Lee, Y. T. (1999). Information Modeling: From Design to Implementation. In *Proceedings of the Second World Manufacturing Congress* (315--321). Zugriff am 17.03.2016. Verfügbar unter http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.4007&rep=rep1&type=pdf

Lier, S. (2013). *Entwicklung einer Bewertungsmethode für die Modularisierung von Produktionssystemen in der Chemieindustrie*. Dissertation, Ruhr-Universität Bochum. Aachen.

Mahnke, W., Gossling, A., Graube, M. & Urbas, L. (2011). Information modeling for middleware in automation. In *ETFA 2011 - Proceedings of 2011 IEEE 16th International Conference on Emerging Technologies & Factory Automation* .

Meier, H. & Uhlmann, E. (2012). Hybride Leistungsbündel – ein neues Produktverständnis. In H. Meier & E. Uhlmann (Hrsg.), *Integrierte Industrielle Sach- und Dienstleistungen: Vermarktung, Entwicklung und Erbringung hybrider Leistungsbündel* (S. 1–22). Springer Berlin Heidelberg.

Morozov, A. & Janschek, K. (2011). Dual Graph Error Propagation Model for Mechatronic System Analysis. In *World Congress* (IFAC proceedings volumes, S. 9893–9898). IFAC, Elsevier.

NA 35 (2003). *Abwicklung von PLT-Projekten*.

NE 100 (2006). *Nutzung von Merkmalleisten im PLT-Engineering-Workflow*.

NE 148 (2013). *Anforderungen an die Automatisierungstechnik durch die Modularisierung verfahrenstechnischer Anlagen*.

Obst, M., Drumm, O., Doherr, F., Bauer, C. & Urbas, L. (2012). Integriertes HMI-Engineering - Konzeption, Entwicklung und Untersuchung der Integration früher Phasen der Bedienbildgenerierung in CAE-Systemen. In P. Adolphs, U. Jumar & N. Kuschnerus (Hrsg.), *Automation 2012* (S. 227–230). Düsseldorg: VDI-Verlag.

Obst, M., Hahn, A. & Urbas, L. (2014). Package unit integration for process industry — A new description approach. In *ETFA 2014 - Proceedings of 19th IEEE Emerging Technology and Factory Automation* (S. 1–8).

Obst, M., Hahn, A. & Urbas, L. (2014b). Package-Unit-Integration in der Prozessindustrie. Was fehlt für Plug-and-produce? *atp edition - Automatisierungstechnische Praxis, 56* (01-02), 56–65.

Obst, M., Holm, T., Bleuel, S., Clausnitzer, U., Evertz, L., Jäger, T. et al. (2013). Automatisierung im Life Cycle modularer Anlagen. Welche Veränderungen und Chancen sich ergeben. . *atp edition - Automatisierungstechnische Praxis, 55* (1/2), 24–31.

Obst, M., Holm, T., Urbas, L., Fay, A., Kreft, S., Hempen, U. et al. (2015a). Beschreibung von Prozessmodulen. Ein weiterer Schritt zur Umsetzung der NE 148. *atp edition - Automatisierungstechnische Praxis, 57* (1-2), 48–59. http://dx.doi.org/10.17560/atp.v57i01-02.473

Obst, M., Holm, T., Urbas, L., Fay, A., Kreft, S., Hempen, U. et al. (2015). Semantic description of process modules. In *ETFA 2015 - Proceedings of 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation* (S. 1–8).

open62541. an open source and free implementation of OPC UA. Zugriff am 21.12.2016. Verfügbar unter http://open62541.org/

Pahl, G. & Beitz, W. (1997). *Konstruktionslehre:* Springer Verlag.

Pine, B. J. (1994). *Massgeschneiderte Massenfertigung. Neue Dimensionen im Wettbewerb* (Manager-Magazin-Edition, 1. [Dr.]. Wien: Wirtschaftsverl. Ueberreuter.

PI-Step Consortium. (1994). *Process Plant Engineering Activity Modell*. Zugriff am 05.07.2013. Verfügbar unter http://homepages.rya-online.net/matthew-west/pistep/Documents/PPEAM-A4.pdf

Plattform Industrie 4.0 (2016). *Cyber Physical Systems (CPS).* Zugriff am 20.12.2016. Verfügbar unter http://www.plattform-i40.de/I40/Navigation/DE/Service/Glossar/Functions/glossar.html?cms_lv2=157702

PLCopen. (2009). *Technical Committee 6: XML Formats for IEC 61131-3*. Zugriff am 20.05.2015.

ProcessNet. (2009). *Die 50% Idee – Vom Produkt zur Produktionsanlage in der halben Zeit. Thesen Tutzing.* Zugriff am 02.10.2014.

VDI/VDE 3699-1 (2005). *Prozessführung mit Bildschirmen Blatt 1 - Begriffe*.

VDI/VDE 3699-2 (2005). *Prozessführung mit Bildschirmen Blatt 2 - Grundlagen*.

VDI/VDE 3699-3 (1999). *Prozeßführung mit Bildschirmen Blatt 3 - Fließbilder*.

Puerta, A. & Eisenstein, J. (2002). *XIML: A Universal Language for User Interfaces:* XIML.org. Zugriff am 23.02.2016.

Reuther, A. (2003). *UseML. Systematische Entwicklung von Maschinenbediensystemen mit XML* (Fortschritt-Berichte / Pak, Bd. 8 : Mensch-Maschine-Interaktion, Als Ms. gedr). Kaiserslautern: Univ., PAK.

Schaudel, D. (2009). Prozessleittechnik im Informationsverbund des Unternehmens. In K. F. Früh, W. Ahrens & D. Schaudel (Hrsg.), *Handbuch der Prozessautomatisierung: Prozessleittechnik für verfahrenstechnische Anlagen.* Oldenbourg Industrieverlag.

Schmitz, S. (2010). *Grafik- und Interaktionsmodell für die Vereinheitlichung grafischer Benutzungsschnittstellen der Prozessleittechnik* (Fortschrittberichte VDI : Reihe 8, Meß-, Steuerungs- und Regelungstechnik, Nr. 1176, Als Ms. gedr). Düsseldorf: VDI-Verl.

Schnieder, E. (1999). *Methoden der Automatisierung. Beschreibungsmittel, Modellkonzepte und Werkzeuge für Automatisierungssysteme ; mit 56 Tabellen* (Studium Technik). Braunschweig: Vieweg.

Schuh, G. (2005). *Produktkomplexität managen. Strategien - Methoden - Tools* (2., überarb. und erw. Aufl.). München: Hanser.

Schüller, A., Scholz, A., Tauchnitz, T., Drath, R. & Scherwietes, T. (2015). Speed-Standardisierung am Beispiel der PLT-Stelle. Datenaustausch mit dem Namur-Datencontainer. *atp edition - Automatisierungstechnische Praxis, 57* (01-02), 37–46. Zugriff am 19.02.2016.

Siemens AG (2010). *SIMATIC Safety Matrix - Das Management Tool für alle Phasen des Safety Lifecycle*. Zugriff am 24.08.2016. Verfügbar unter https://www.industry.siemens.com/topics/global/de/safety-integrated/process-safety/safety-integrated-systems/safety-engineering-software/Documents/br_safetymatrix_de.pdf

Souchon, N. & Vanderdonckt, J.(2003). A Review of XML-compliant User Interface Description Language. In J. Jorge, N. J. Nunes & J. Falcao e Cunha (Hrsg.), *Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS* (Bd. 2844, S. 377–391). Berlin: Springer-Verlag. Zugriff am 23.02.2016.

SPS-MAGAZIN. (2014). *Produkt-Katalog Prozessleitsysteme und SCADA (SPS-Special/2014).* Zugriff am 07.03.2016. Verfügbar unter http://www.sps-magazin.de/?inc=mues/formmue&mue=69&s=0&l=104

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Wien, New York: Springer-Verlag.

Stern, W. (1935). *Allgemeine psychologie auf personalistischer Grundlage* (Bd. 1): M. Nijhoff.

Tauchnitz, T. (1996). Die „neuen Prozeßleitsysteme“ - wohin geht die Reise? *atp - Automatisierungstechnische Praxis, 38* (11), 12–23.

Theurich, S., Stoss, M., Wollschlaeger, M. & Urbas, L. (2012). Communication and information engineering of FDI equipment packages. In *ETFA 2012 - Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation* (S. 1–8).

Ulrich, A. (2009). *Entwicklungsmethodik für die Planung verfahrenstechnischer Anlagen* (Fortschritt-Berichte VDI / 20, Bd. 425). Düsseldorf: VDI-Verl.

Urbas, L. (2012). *Process Control Systems Engineering*. München: Oldenbourg Industrieverl. GmbH.

Urbas, L. (2014). *Zentrales Leitsystem ja oder nein?,* Vogel Business Media GmbH & Co. KG. Zugriff am 27.07.2014. Verfügbar unter http://www.process.vogel.de/automatisierung_prozessleittechnik/articles/451796/?cmp=nl-98

Urbas, L., Bleuel, S., Jäger, T., Schmitz, S., EvertzERTZ, L. & Nekolla, T. (2012). Automatisierung von Prozessmodulen. Von Package-Unit-Integration zu modularen Anlagen. *atp edition - Automatisierungstechnische Praxis, 53* (1-2), 44–53.

Urbas, L. & Doherr, F. (2011). autoHMI: a model driven software engineering approach for HMIs in process industries. In *CSAE 2011 -Proceedings of 2011 IEEE International Conference onComputer Science and Automation Engineering* (Bd. 3, S. 627–631).

Urbas, L., Doherr, F., Krause, A. & Obst, M. (2012). Modularisierung und Prozessführung. *Chemie Ingenieur Technik, 84* (5), 615–623. http://dx.doi.org/10.1002/cite.201200034

Urbas, L., Hennig, S., Hager, H., Doherr, F. & Braune, A. (2011). Towards context adaptive HMIs in process industries. In *INDIN 2011 - Proceedings of 9th IEEE International Conference on Industrial Informatics* (S. 244–249). Verfügbar unter http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6034881

Voigt, T. & Kather, A. (2005). Weihenstephaner Standards für BDE-Systeme. Industrieseminar verdeutlicht die große Akzeptanz in der Branche. *Getränkeindustrie, 59* (9), 158–161. Zugriff am 02.02.2016. Verfügbar unter http://www.weihenstephaner-standards.de/index.php?eID=tx_nawsecuredl&u=0&g=0&t=1454532792&hash=bc5e248c01781bd33c6187fea7e9ddce5966825f&file=fileadmin/pdf/veroeffentlichungen/GETRAENKEINDUSTRIE_9_2005.pdf

Wassilew, S. (2015). *Modellierung und Einbindung eines Modul Informationsträgers (MTP) in OPC UA*. Diplomarbeit, TU Dresden. Dresden.

Weyrich, M. & Klein, P. (2012). Modulbasiertes Engineering von Produktionsanlagen - Wissensbasierte Konzeption basierend auf funktionsorientierter Modularisierung. *wt Werkstattstechnik online, 102* (9), 592–597. Zugriff am 03.12.2015. Verfügbar unter http://www.werkstattstechnik.de/wt/get_article.php?data[article_id]=68995

Yokogawa Europe. (2015). *ProSafe-PLC.* Zugriff am 24.08.2016. Verfügbar unter http://www.yokogawa.com/eu/iss/systems/eu-plc-overview2.htm#anchor-top

Verzeichnis der Veröffentlichungen des Autors

Begutachtete Journalartikel

Bernshausen, J.; Haller, A.; Holm, T.; Hoernicke, M.; **Obst, M.**; Ladiges, J. (2016) Namur Modul Type Package–Definition, atp edition - Automatisierungstechnische Praxis 58(01-02), S. 72-81.

Holm, T.; **Obst, M.**; Ladiges, J.; Urbas, L.; Fay, A.; Hempen, U.; Albers, T. (2016) Namur Modul Type Package–Implementierung, atp edition - Automatisierungstechnische Praxis 58(01-02), S. 82-90.

Obst, M.; Holm, T.; Urbas, L.; Fay, A.; Kreft, S.; Hempen, U.; Albers, T. (2015). Beschreibung von Prozessmodulen - Ein weiterer Schritt zur Umsetzung der NE 148. atp edition - Automatisierungstechnische Praxis 57(1-2), S.48-59.

Holm, T.; **Obst, M**.; Fay, A.; Urbas, L.; Albers, T.; Kreft, S.; Hempen, U. (2014). Dezentrale Intelligenz für modulare Automation - Lösungsansätze für die Realisierung modularer Anlagen. atp edition - Automatisierungstechnische Praxis 56(11), S. 34-43.

Obst, M.; Hahn, A.; Urbas, L. (2014). Package Unit Integration in der Prozessindustrie. Was fehlt für Plug-and-Produce? atp edition - Automatisierungstechnische Praxis 56(1-2), S. 56-65.

Ohle, A.; **Obst, M.**; Mollekopf, N.; Urbas, L. (2014). Modularisierung von Gaswäschern für die CO2-Entfernung aus Biogas. Chemie Ingenieur Technik 86(5), S. 640-648, doi:10.1002/cite.201300163.

Obst, M.; Doherr, F.; Urbas, L. (2013). Wissensbasiertes Assistenzsystem für modulares Engineering. at - Automatisierungstechnik 61(2), S. 103-108, DOI:10.1524/auto.2013.0011.

Obst, M.; Holm, Th.; Bleuel, St.; Claussnitzer, U.; Evertz, L.; Jäger, T.; Nekolla, T.; Pech, St.; Schmitz, St.; Urbas, L. (2013). Automatisierung im Life Cycle modularer Anlagen. Veränderungen und Chancen. atp edition - Automatisierungstechnische Praxis 55(1-2), S. 24-31.

Urbas, L.; Doherr, F.; Krause, A.; **Obst, M**.; (2012). Modularisierung und Prozessführung. Chemie Ingenieur Technik 84(5), S. 615-623, DOI: 10.1002/cite.201200034.

Pfeffer, A.; Hahn, A.; **Obst, M.**; Urbas, L.; (2015) Systemfremde Steuerungen in modulare Anlagen integrieren. atp edition - Automatisierungstechnische Praxis 57(07-08), S. 56-61.

Begutachtete Konferenzbeiträge

Obst, M.; Holm, T.; Urbas, L.; Fay, A.; Kreft, S.; Hempen, U.; Albers, T. (2015) Semantic description of process modules. In: Proceedings of 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), DOI: 10.1109/ETFA.2015.7301440.

Holm, T.; **Obst, M.**; Fay, A.; Urbas, L.; Hempen, U.; Albers, T.; Kreft, S. (2015) Engineering method for the integration of modules into fast evolving production systems in the process industry. In Proceedings of 2015 IEEE International Conference on Automation Science and Engineering (CASE), DOI: 10.1109/CoASE.2015.7294236.

Obst, M.; Hahn, A.; Urbas, L. (2014). Package Unit Integration for Process Industry - A new description approach. In: Proceedings of 19th international IEEE Conference on Emerging Technologies & Factory Automation (ETFA), DOI:10.1109/ETFA.2014.7005159.

Obst, M.; Runde, S.; Wolf, G.; Urbas, L. (2013). Integration Requirements of Package Units - A Description Approach with FDI. In: Proceedings of 18th international IEEE Conference on Emerging Technologies & Factory Automation (ETFA), DOI: 10.1109/ETFA.2013.6647974.

Urbas, L., **Obst, M**., Stöß, M. (2012). Formal Models for High Performance HMI Engineering. In: I. Troch, F. Breitenecker (Hrsg.) Proceedings of MathMod 2012, S. 854-859. IFAC-online. DOI:10.3182/20120215-3-AT-3016.00151

Obst, M.; Drumm, O.; Doherr, F.; Bauer, C.; Urbas, L. (2012) Integriertes HMI-Engineering - Konzeption, Entwicklung und Untersuchung der Integration früher Phasen der Bedienbildgenerierung in CAE-Systemen. In: Tagungsband Automation Bd. 2171, S. 227-230

Krause, A.; **Obst, M.**; Urbas, L. (2012) Towards safe and reliable allocation of functions in modular process plants. In Proceedings of 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference 2012.

Krause, A.; **Obst, M.**; Urbas, L. (2012) Extraction of safety relevant functions from CAE data for evaluating the reliability of communications systems. In: Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA). DOI: 10.1109/ETFA.2012.6489784

Patenanmeldungen

Obst, M.; Holm, T.; Malken, J.; Kreft, S. (2016) Module for a Process Engineering System and method for controlling a Process Engineering System, United States Patent Application 20160124410.

Holm, T.; Kreft, S.; Malken, J.; **Obst, M.** (2014) Modul für eine prozesstechnische Anlage und Verfahren zur Steuerung einer prozesstechnischen Anlagen, DE102014222508A1

Betreute studentische Arbeiten

Albert, M. (2016) Zustandsorientierte Prozessführung für modulare Konti-Prozesse, Diplomarbeit.

Noack, S. (2015) Entwicklung einer Methodik zur Selbstorganisation der PLT-Funktionen eines Moduls für die zustandsorientierte Prozessführung, Diplomarbeit.

Wassiliew, S. (2015) Modellierung und Einbindung eines Modul Informationsträgers (MTP) in OPC UA, Diplomarbeit.

Altman, P. (2015) Erweiterung des Human-Maschine-Interface-Aspektes eines Modul Informationsträgers (MTP), Diplomarbeit.

Albert, M. (2015) Wiederinbetriebnahme und Integration eines verfahrenstechnischen Moduls unter Berücksichtigung der NE148, Studienarbeit.

Hahn, A. (2014) Vereinfachte Leitsystemintegration von Package Units, Diplomarbeit.

Noack, S. (2014) Entwurf und Aufbau eines Kühlsystems für modulare Forschungsanlage, Studienarbeit.

Nguyen, Q. T. (2013) Weiterentwicklung eines Assistenzsystems zur Auslegung von Modulen, Masterarbeit.

Yang, C. (2012) Funktionsextraktion von Planungsdaten für erweiterte Analysen, Studienarbeit.

Hou, B. (2012) Integrierte Planung von verfahrens- und prozessleittechnischen Optionen bei Modulen für modulare Anlagen, Studienarbeit.

Heldt, A. (2011) Funktionsorientierter Entwurf eines mobilen und autark agierenden Handlingsystems unter Anwendung mechatronischer Modulkonzepte, Diplomarbeit.