
Domänenspezifische Modellierung von Automatisierungssystemen mit AutomationML

Masterarbeit zur Erlangung des akademischen Grades

Master of Science im Studiengang Allgemeiner Maschinenbau (Master)

an der Fakultät für Anlagen, Energie- und Maschinensysteme

der Technischen Hochschule Köln

Technology
Arts Sciences
TH Köln

Vorgelegt von Yifan Wang

Matrikel-Nr.: 11110890

Adresse: Osternohstraße 25
51647 Gummersbach
yifiwang@gmx.de

Eingereicht bei: Prof. Dr.-Ing. Dipl.-Wirt.Ing. Nicolas Pyschny

Zweitgutachter: M.Sc. Ben Rudat

Gummersbach, 31. Mai 2022

Kurzfassung

Titel: Domänenspezifische Modellierung von Automatisierungssystemen mit AutomationML

Gutachter:

- Prof. Dr.-Ing. Dipl.-Wirt.Ing. Nicolas Pyschny
- M.Sc. Ben Rudat

Zusammenfassung: Zeitaufwendige und fehleranfällige manuelle Übertragungsarbeiten sind das Ergebnis von einer Vielzahl an spezialisierten Engineering-Werkzeugen, welche aktuell in der Industrie genutzt werden. Um diese zu automatisieren, bietet sich das Datenformat AutomationML an. Trotz der Einführung des Formats im Jahr 2006 [1] wird es jedoch kaum in der industriellen Umgebung genutzt. Aus diesem Grund ist das Ziel der vorliegenden Arbeit das Potenzial von AutomationML (AML) für die domänenspezifische Modellierung von Automatisierungssystemen aufzuzeigen. Das Automatisierungssystem wird in einem Fallbeispiel durch ein Miniaturmodell einer modularen Anlage dargestellt. Das Fallbeispiel soll den Ablauf einer Anlagenplanung mit Nutzung des AML-Formats aufzeigen, um detaillierte Erkenntnisse zu Aufwand und Vorteilen zu gewinnen. Das Format ist zur Darstellung des digitalen Anlagenabbilds da. Um eine Domäne für die automatisierte Nutzung des digitalen Anlagenabbild zu involvieren, wird AML für eine Simulation in der Virtuellen Inbetriebnahme (VIBN) genutzt. Der evaluierte Ergebnisse von Aufwand und Vorteilen, belegt durch das Fallbeispiel, zeigen, dass die Nutzung sowohl von AML, als auch von VIBN eine markante Erhöhung des Planungsaufwands zur Folge hat, um eine kürzere Entstehungszeit beim Anlagenbau zu erreichen.

Stichwörter: AutomationML, Automatisierungssystem, Virtuelle Inbetriebnahme, modulare Anlagensysteme, modularer Anlagenbau

Datum: 31. Mai 2022

Abstract

Title: Domain specific modeling of automation systems with AutomationML

Reviewer:

- Prof. Dr.-Ing. Dipl.-Wirt.Ing. Nicolas Pyschny
- M.Sc. Ben Rudat

Abstract: Time-consuming and error-prone manual transfer work is the result of a large number of specialized engineering tools currently used in the industry. The AutomationML data format is ideal to automate the manual labour. Despite the format's introduction in 2006 [1], it is rarely used in the industrial environment. For this reason, the aim of this work is to show the potential of AutomationML (AML) for the domain specific modeling of automation systems. In a case study, the automation system is represented by a miniature model of a modular system. The case study is intended to show the process of plant planning using the AML format in order to gain detailed insights into the costs and benefits. The format exists to represent a digital twin of the plant. In order to involve a domain for a use case of the digital twin in which the data is retrieved automatically, AML is used for a simulation in the virtual commissioning (VC). The evaluated results of effort and benefits, documented by the case study, show that the use of both AML and VC results in a significant increase in planning effort in order to achieve a shorter development time in plant construction.

Keywords: AutomationML, automation system, virtual commissioning, modular plant systems, modular plant design

Date: May 31, 2022

Inhaltsverzeichnis

Tabellenverzeichnis	vii
Abbildungsverzeichnis	viii
Abkürzungsverzeichnis	x
1 Einleitung.....	1
2 Forschungsdesign	2
3 Grundlagen zu AutomationML.....	5
3.1 Über AutomationML	5
3.2 Architektur	6
3.3 Abstraktion von Daten zu Informationen	7
3.4 Objektmodelle	8
3.5 Beziehungsmodelle	12
3.6 AMLX Container für AutomationML Komponenten	14
3.7 AutomationML Editor	14
3.7.1 Aufbau der Benutzeroberfläche	15
3.7.2 Objekte erzeugen, modifizieren und löschen	16
3.7.3 Modellierung von Vererbungen.....	16
3.8 AutomationML Engine	17
3.9 AutomationML Datenaustausch Architektur.....	17
3.9.1 Implementierungskonzept für die „Transformation“ Ebene einer Exportschnittstelle	18
3.10 Product-Process-Resource (PPR) Konzept.....	19
4 Grundlagen zu Automatisierungssystemen und modularen Anlagensystemen.....	21
4.1 Über Automatisierungssysteme	21
4.2 Über modulare Anlagensysteme	22
4.2.1 Vor- und Nachteile.....	22
4.3 Entstehungsprozess einer modularen Anlage	23
4.3.1 Planungsphase einer modularen Anlage	24
5 Grundlagen zur Virtuellen Inbetriebnahme	26
5.1 Über Virtuelle Inbetriebnahme	26
5.2 Aufbau eines Virtuelle Inbetriebnahme Modells	27
5.3 Umgebung für die Virtuelle Inbetriebnahme: fe.screen-sim	28

6	Nutzung von VIBN und AML im Entstehungsprozess von modularen Anlagen.....	29
6.1	Nutzung von Virtueller Inbetriebnahme.....	29
6.2	Nutzung von AutomationML	30
7	Fallbeispiel: Demonstrator einer modularen Anlage für die Virtuelle Inbetriebnahme	31
7.1	Über den Demonstrator	31
7.1.1	Aufbau des physischen Demonstrators	31
7.1.2	Aufbau des digitalen Anlagenmodells.....	32
7.1.3	Simulationsablauf	33
7.1.4	Darstellung von Eigenschaften eines Automatisierungssystems.....	34
7.2	Nutzung des AML Formats	35
7.2.1	AML Tool.....	35
7.2.2	Softwarearchitektur.....	35
7.2.3	Benutzeroberfläche	36
7.3	Phase 1: Erzeugen des Domänenmodells.....	37
7.3.1	Mapping von Datenstrukturen auf AML-Strukturen	37
7.3.2	Strukturierung der Datenelemente.....	39
7.3.3	AML Vorlagenbibliotheken.....	42
7.4	Phase 2: Topologie Export	44
7.5	Phase 3: Inferenzmaschine	45
7.6	Phase 4: Importer und fe-Vorlagenmodelle	45
7.6.1	fe-Vorlagenmodelle	46
7.6.2	Importer.....	47
7.7	Phase 5: Simulator	47
7.7.1	Kommunikationsstruktur	48
8	Potenzial von AutomationML bei der domänenspezifischen Modellierung von Automatisierungssystemen	50
8.1	Modellierung für verschiedene Interessengruppen und Strukturierung der Daten durch das PPR-Konzept.....	50
8.2	AutomationML und Virtuelle Inbetriebnahme im Zusammenhang mit dem Entstehungsprozess des modularen Anlagenbaus.....	50
8.3	Erfahrungen mit der Nutzung der Programmierschnittstellen.....	51

9	Zusammenfassung und Fazit	52
10	Ausblick	54
	Literaturverzeichnis	55
	Anhang 1: Systemarchitektur vom AML_Tool	56
	Eidesstattliche Erklärung	58

Tabellenverzeichnis

Tabelle 1: Beziehungstypen in CAEX [2, S. 98, 2, S. 99, 2, S. 97, 2, S. 95, 2, S. 95]	13
Tabelle 2: Mapping der Datenstrukturen zu AML Strukturen nach [11]	37
Tabelle 3: Bedeutung der Datenelemente	39

Abbildungsverzeichnis

Abbildung 1: Fünf Phasen Modellierungsprozess	3
Abbildung 2: Phasen 1-3 des Modellierungsprozesses	3
Abbildung 3: Phase 4 des Modellierungsprozesses.....	4
Abbildung 4: Phase 5 des Modellierungsprozesses.....	4
Abbildung 5: Abdeckung von AutomationML [2, S. 20]	5
Abbildung 6: Architektur [2, S. 47]	6
Abbildung 7: Vier-Ebenen-Modell, Abstraktion des AutomationML Datenaustauschs [2, S. 28]	7
Abbildung 8: RoleClasses, SystemUnitClasses, InstanceHierarchy [3, S. 256]	9
Abbildung 9: BaseRoleClassLibrary	9
Abbildung 10: InterfaceClassLibrary	9
Abbildung 11: herstellerspezifische SystemUnitClassLibrary.....	10
Abbildung 12: herstellerspezifisches RoleClassLibrary.....	10
Abbildung 13: InstanceHierarchy.....	11
Abbildung 14: Beziehungstypen in CAEX [2, S. 93].....	12
Abbildung 15: AMLX Container [3, S. 276]	14
Abbildung 16: AML Editor Ansicht mit darin geöffnetem AML Dokument.....	15
Abbildung 17: AML Editor Sektionen der Benutzeroberfläche	15
Abbildung 18: Grundlegende Systemarchitektur von Datenschnittstellen [3, S. 42] ..	17
Abbildung 19: Externes Generierungskonzept einer Exportschnittstelle [3, S. 47]	19
Abbildung 20: PPR InstanceHierarchy [4, S. 71]	20
Abbildung 21: Anlagenmodul, bestehend aus Stahlrahmen mit vormontierten Prozesskomponenten, Messeinrichtungen, Rohrleitungen und elektrischer Verkabelung [6].....	22
Abbildung 22: Entstehungsprozess modularer Anlagensysteme in Anlehnung an [7]	23
Abbildung 23: Kumulierter Finanzfluss bei klassischem Engineering (blaue Linie) und beschleunigtem Engineering auf Basis modularer Anlagen (rote Linie) [8]	24
Abbildung 24: Modulares Engineering [8]	24
Abbildung 25: Virtuelle Inbetriebnahme im Entstehungsprozess eines Produktionssystems [7]	26
Abbildung 26: fe.screen-sim Softwarestruktur [10]	28
Abbildung 27: Einordnung der VIBN in den Entstehungsprozess von modularen Anlagen in Anlehnung an [7]	29

Abbildung 28: Einordnung des AML-Anlagenmodells in den Entstehungsprozess von modularen Anlagen.....	30
Abbildung 29: Einordnung des Fallbeispiels in den Entstehungsprozess eines modularen Anlagensystems	31
Abbildung 30: physische Miniaturanlage	32
Abbildung 31: Demonstrator als geometrische Repräsentation in der VIBN-Simulationsumgebung mit „productSources“ (farbiger Deckel) und „productSinks“ (weißer Deckel mit Lampe)	33
Abbildung 32: Benutzeroberfläche des AML Tools	36
Abbildung 33: Klassendiagramm Anlagentopologie.....	40
Abbildung 34: InstanceHierarchy Demonstrator	41
Abbildung 35: SystemUnitClassLibrary Demonstrator	42
Abbildung 36: RoleClassLibrary Demonstrator	43
Abbildung 37: InterfaceClassLibrary Demonstrator	44
Abbildung 38: Externes Generierungskonzept bei der Erzeugung der IH (Anlagentopologie) [3, S. 47].....	44
Abbildung 39: Externes Generierungskonzept bei der Erzeugung der IH (PPR-Konzept) nach [3, S. 47].....	45
Abbildung 40: "Motion Joint" Objekt	46
Abbildung 41: "Motion Joint" Objekt im Projektbaum	46
Abbildung 42: Logikdiagramm mit "Logic Object"	46
Abbildung 43: Import in Engineering Tools (FEE) [3, S. 45].....	47
Abbildung 44: Transformation Layer beim Import	47
Abbildung 45: Simulator	48
Abbildung 46: Systemarchitektur AML_Tool.....	57

Abkürzungsverzeichnis

AML	AutomationML
API	Application Programming Interface
Attr	Attribute (AutomationML Datenstruktur)
CAEX	Computer Aided Engineering Exchange (Datenformat)
COLLADA	Collaborative Design Activity (Datenformat)
EI	External Interface (AutomationML Datenstruktur)
GUID	Globally Unique Identifier
IC	InterfaceClass (AutomationML Objekttyp)
IE	Internal Element (AutomationML Datenstruktur)
IH	InstanceHierarchy (AutomationML Objekttyp)
IL	Internal Link (AutomationML Datenstruktur)
PPR	Product-Process-Resource (AutomationML Modellierungskonzept)
RC	RoleClass (AutomationML Objekttyp)
RFID	Radio Frequency identification
SUC	SystemUnitClass (AutomationML Objekttyp)
VC	Virtual Commissioning (dt. Virtuelle Inbetriebnahme)
VIBN	Virtuelle Inbetriebnahme
XML	Extensible Markup Language (Datenformat)
XSD	XML Schema Definition

1 Einleitung

In der heutigen globalisierten Welt herrscht ein großer Konkurrenzdruck in der Industrie. Aufgrund von steigender Produktkomplexität, werden gleichzeitig Fertigungsprozesse komplizierter. Dieser Komplexität muss mit einem erhöhtem Aufwand und somit einem erhöhtem Bedarf an Arbeitskraft entgegnet werden. Da die menschliche Arbeitskraft in hochindustrialisierten Staaten teuer ist, stellen Automatisierungssysteme eine wirtschaftliche Lösung dar, mit dem Nachteil, dass sie durch den Einsatz von automatisierten Maschinen einen erhöhten Planungsaufwand bei der Anlagenentstehung generieren. Für großen Produktionsanlagen mit Automatisierungssystemen, worin sich Anlagenbestandteile standardisieren und modularisieren lassen, ist das Konzept eines modularen Anlagensystems anwendbar. Dieses verkürzt die Entstehungszeit und rückt den Zeitpunkt der Fertigstellung nach vorne, wodurch die Gewinngenerierung früher stattfinden kann.

Ein weiteres Werkzeug, um die Wirtschaftlichkeit von komplexen Automatisierungssystemen zu steigern ist die Virtuelle Inbetriebnahme. Diese zielt darauf ab, durch Simulation eines digitalen Anlagenmodells den klassischen Inbetriebnahmeprozess zu verkürzen. Wenn es um digitale Prozesse im Anlagenbau geht, tritt nach heutigem Stand immer wieder ein markanter Flaschenhals in Engineering-Prozessen auf. Für jede einzelne Domäne im Anlagenbau existiert eine Vielzahl von spezialisierten Engineering-Werkzeugen. Diese bilden eine heterogene Landschaft an Werkzeugen und machen einen Informationsaustausch zwischen den einzelnen Werkzeugen sehr mühsam. Aufgrund von fehlenden Kompatibilitäten zwischen den Schnittstellen der Werkzeuge entstehen manuelle Übertragungsarbeiten, welche zum einen Zeit kosten und zum anderen fehleranfällig sind. AutomationML soll als Lösung dieser Problematik dienen, jedoch stellt sich die Frage, aus welchem Grund das Format, trotz seiner Einführung im Jahr 2006 [1], sich heute noch nicht als verbreiteter Industriestandard etabliert hat. Aus diesem Grund soll das Potenzial von AutomationML bei der domänenspezifischen Modellierung von Automatisierungssystemen herausgearbeitet werden.

2 Forschungsdesign

Diese Arbeit beschäftigt sich mit dem Potenzial von AutomationML bei der domänenspezifischen Modellierung von Automatisierungssystemen.

Um ein Verständnis für den Aufbau und die Nutzung der Modellierungssprache zu bekommen, wird im ersten Teil dieser Arbeit eine tiefgängige Grundlagenrecherche über AutomationML (AML) durchgeführt. AML soll als Modellierungssprache für Automatisierungssysteme dienen. Als Anwendungsbeispiel für ein Automatisierungssystem wird eine modulares Anlagensystem gewählt. Der Entstehungsprozess einer modularen Anlage ist bei der Nutzung von AML von zentraler Bedeutung, da die Modellierungssprache dazu fähig ist, ein digitales Abbild des Anlagenmodells darzustellen. Dieses ist für eine wirtschaftliche Nutzung der Virtuellen Inbetriebnahme (VIBN) nötig, welche im Anschluss vorgestellt wird. Dort werden die Vorteile von AML sichtbar, welche in der Möglichkeit des automatisierten Datentransfers zu Engineering-Werkzeugen liegen. Für Softwares, die das Format noch nicht für den Export oder Import unterstützen, wird die Programmierschnittstelle (API) Aml.Engine bereitgestellt. Diese kann dafür genutzt werden, um mithilfe von C# Programmierung einen eigenen Exporter und Importer zu schreiben.

Ein Fallbeispiel soll schließlich das Potenzial von AML zur Modellierung von modularen Anlagensystemen aufzeigen. Im Fallbeispiels gilt es, den Planungsprozess für modulare Anlagensysteme an einem vereinfachten Miniaturmodell durchzuführen. Die Anlagentopologie des Miniaturmodells stellt das Ergebnis einer Anlagenplanung dar. Diese soll mithilfe des AML-Formats digital abgebildet werden. Es soll alle nötigen Informationen für eine automatisierte Nutzung in der Virtuelle Inbetriebnahme bereitstellen.

AML hat den Zweck, den Datenaustausch zwischen verschiedenen Engineering-Werkzeugen zu standardisieren und automatisieren. Aus diesem Grund wird bei der Bearbeitung des Fallbeispiels der Fokus auf den Informationsweg der Engineering-Daten gelegt (s. Abbildung 1). In dem skizzierten Informationsweg sind alle benötigten Modelle mitsamt ihrer Schnittstellen dargestellt. Um die vielen Verarbeitungsschritte im Informationsweg vom Miniaturmodell bis zur VIBN zu strukturieren, wird die Modellerstellung in fünf Phasen aufgeteilt.

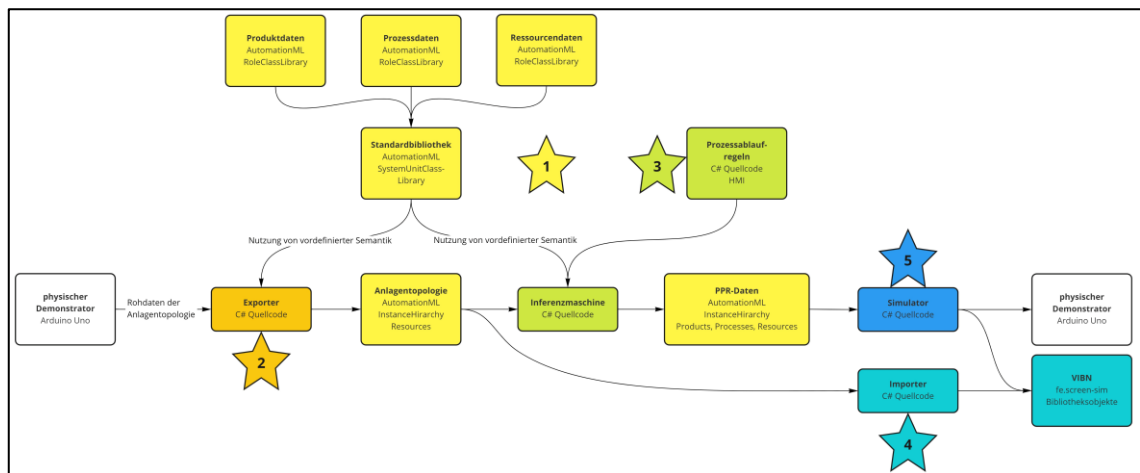


Abbildung 1: Fünf Phasen Modellierungsprozess

Phasen 1-3 beschreiben das Erzeugen der Anlagendaten in einer AML InstanceHierarchy (s. Abbildung 2):

- Phase 1 beinhaltet das Mapping der Anlagentopologie des Demonstrators auf AML Datenstrukturen und das Erzeugen des Domänenmodells
- Phase 2 beinhaltet das Programmieren eines Exporter zum Verarbeiten der Rohdaten der Anlagentopologie, um sie in das AML Format zu überführen
- Phase 3 beinhaltet das Programmieren einer Inferenzmaschine zum Erzeugen der verfügbaren Prozesse und Produkte innerhalb der Anlage, welche sich aus der Anlagentopologie ableiten lassen. Daraufhin werden Prozessablaufregeln auf die verfügbaren Prozesse angewandt, um die resultierende Prozesskette zu erzeugen. Mithilfe von Anlagentopologie, resultierender Prozesskette und Produktdaten sind alle Informationen und Daten für ein vollständiges Anlagenmodell vorhanden. Diese werden mithilfe des Product-Process-Resource Konzepts im AML Format modelliert.

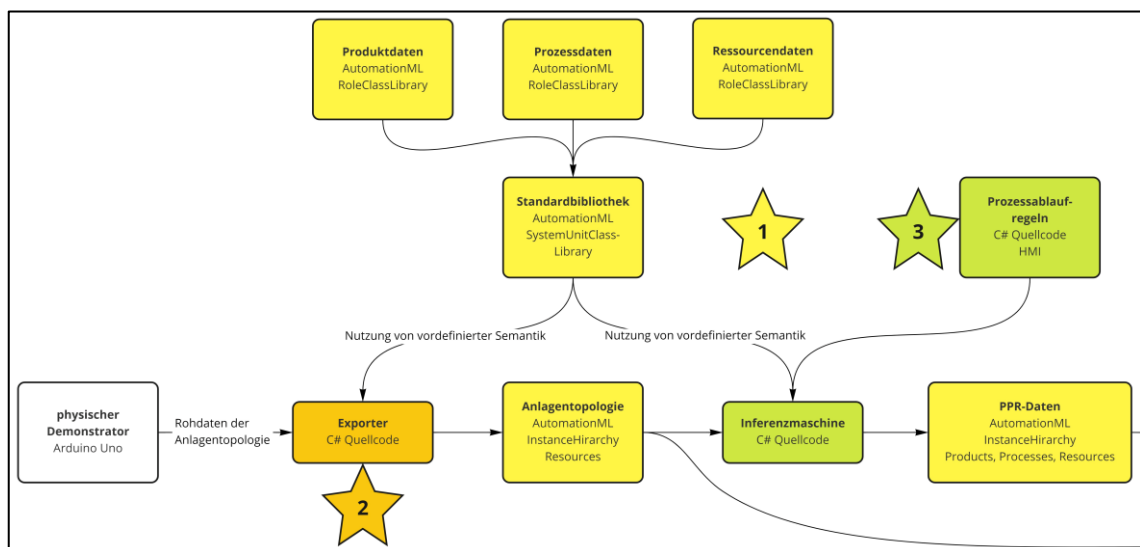


Abbildung 2: Phasen 1-3 des Modellierungsprozesses

Phase 4 beinhaltet die Erzeugung von fe-Vorlageobjekten. Diese werden darauffolgend von einem Importer genutzt, um eine automatisierte Erzeugung des VIBN-Modells zu ermöglichen. Die dazu nötigen Anlagendaten werden aus bereitgestellten PPR-Daten im AML Format gelesen (s. Abbildung 3).

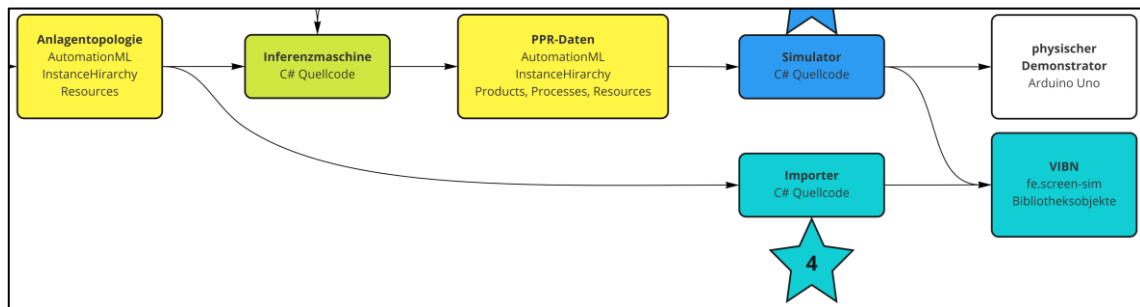


Abbildung 3: Phase 4 des Modellierungsprozesses

Phase 5 beinhaltet die Nutzung des AML-Formats für die Prozessausführung mithilfe eines Simulators, welche an der physischen Anlage und am VIBN-Modell durchgeführt werden (s. Abbildung 4).

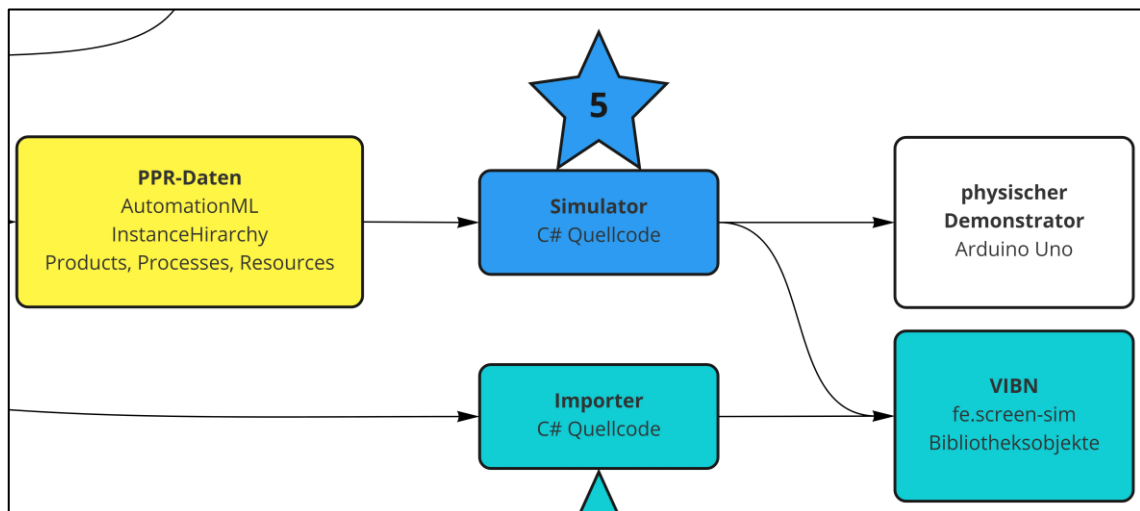


Abbildung 4: Phase 5 des Modellierungsprozesses

3 Grundlagen zu AutomationML

Eine Grundlagenrecherche zur Modellierungssprache ist nötig, um ein Verständnis für den Aufbau und die Nutzung erlangen. Angefangen mit der Architektur von AML wird mit der Abstraktion von Daten und der Erläuterung von Objektmodellen der Zweck der objektorientierten, hierarchischen Darstellungsart veranschaulicht. Mithilfe von Beziehungsmodellen lassen sich semantische (bedeutungsbeschreibende) Assoziationen unter den Objekten erstellen. Um Daten im AML-Format Format zu manipulieren wird der AML Editor genutzt. Zusätzlich sind für eine Anwendung der Modellierungssprache Konvertierungskonzepte, wie Import- und Exportkonzepte, wichtig.

3.1 Über AutomationML

AutomationML ist eine verständlich formulierte, XML-basierte und objektorientierte Modellierungssprache. Sie wird für Modellierung, Speicherung und Austausch von Modellen aus der Ingenieursdomäne genutzt. Das Ziel ist es, ein Werkzeug zu schaffen, welches die Modellierung und den Austausch von heterogenen, semantischen Engineering-Datenmodellen über alle Entwicklungsschritte eines Engineering-Prozesses ermöglicht. AutomationML deckt die Modellierung von Anlagenhierarchien, Geometrien, Kinematiken, Bewegungsplanung und Verhalten ab (s. Abbildung 5). [vgl. 2, S. 19-20]

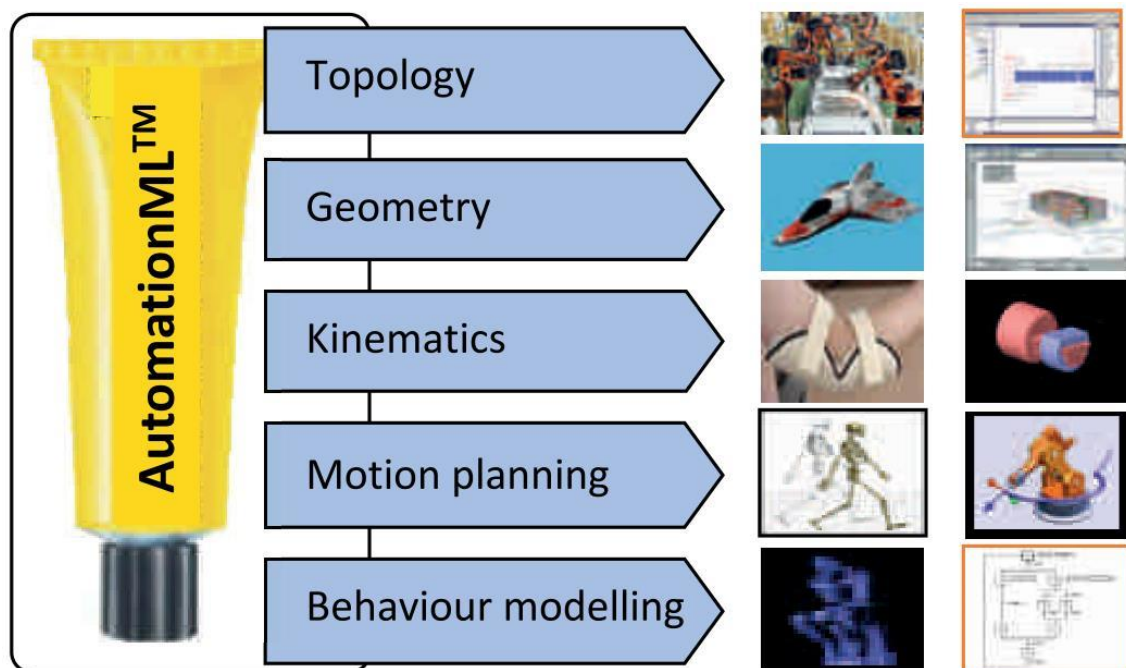


Abbildung 5: Abdeckung von AutomationML [2, S. 20]

3.2 Architektur

Abbildung 6 zeigt die grundlegende AML Architektur mit ihren Submodellen für Objekthierarchie-, Geometrie-, Kinematik- und Logikinformationen, bestehend aus unterschiedlichen Dateiformaten. In grau ist das AML-Dokument im CAEX-Format zu erkennen. Dieses beinhaltet eine Objekthierarchie, bestehend aus vereinzelt Objekten. Jedes Objekt in der Objekthierarchie kann aus einer Zusammensetzung von hierarchisch untergeordneten „Kind“-Objekten bestehen. Die Bezeichnung „Kind“ ist hierbei von einer „Eltern“-„Kind“ Beziehung abgeleitet und steht für eine untergeordnete hierarchische Zuordnung. „Eltern“-Objekte können jedoch auch selbst Teil einer größeren Komposition sein, in der sie selbst eine „Kind“-Rolle besitzen. Alle Objekte in ihrer Gesamtheit formen einen Objektbaum. Jedes Objekt kann Geometrien, Kinematiken, oder Verhaltensinformationen, welche außerhalb des CAEX-Dokuments im XML-Format existieren, referenzieren. Dies erlaubt eine domänenübergreifende Modellierung. [vgl. 2, S. 47]

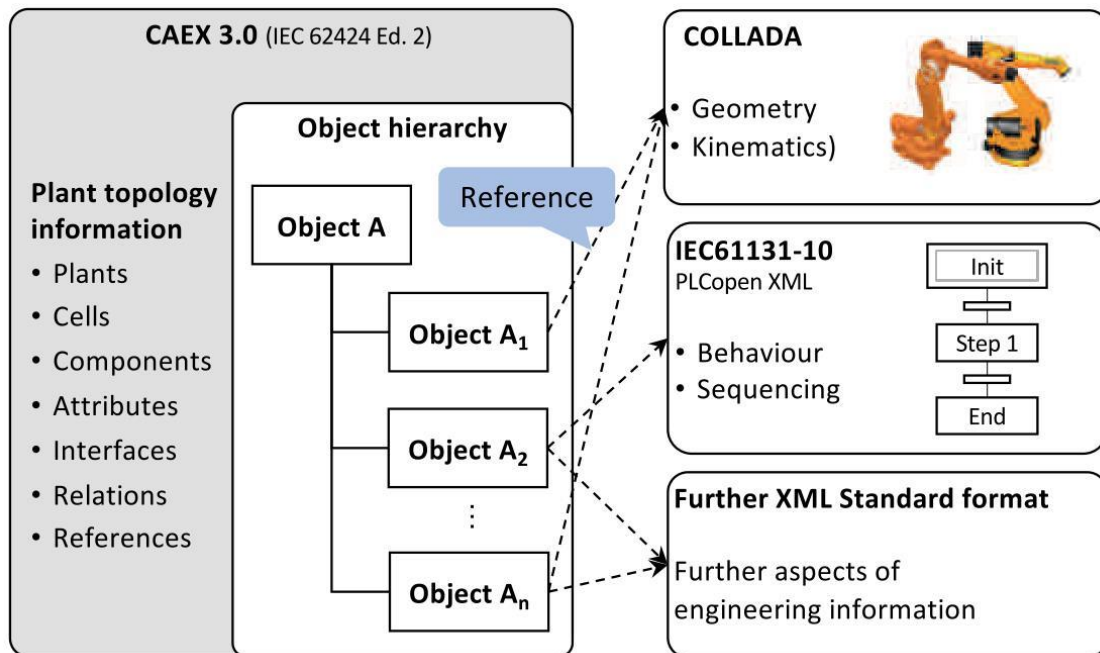


Abbildung 6: Architektur [2, S. 47]

Nachfolgend wird die Anlagentopologie oder Objekthierarchie auch „InstanceHierarchy“ genannt.

3.3 Abstraktion von Daten zu Informationen

Für die nachfolgende Diskussion über die Anwendung von AutomationML ist es nötig zwischen „Daten“ und „Informationen“ zu unterscheiden. Als Grundlage dafür dienen vier verschiedene Abstraktionsebenen: XML-Ebene, Objektebene, semantisch-domänen-spezifische Modellebene und Projektebene (s. Abbildung 7). [vgl. 2, S. 28-31]

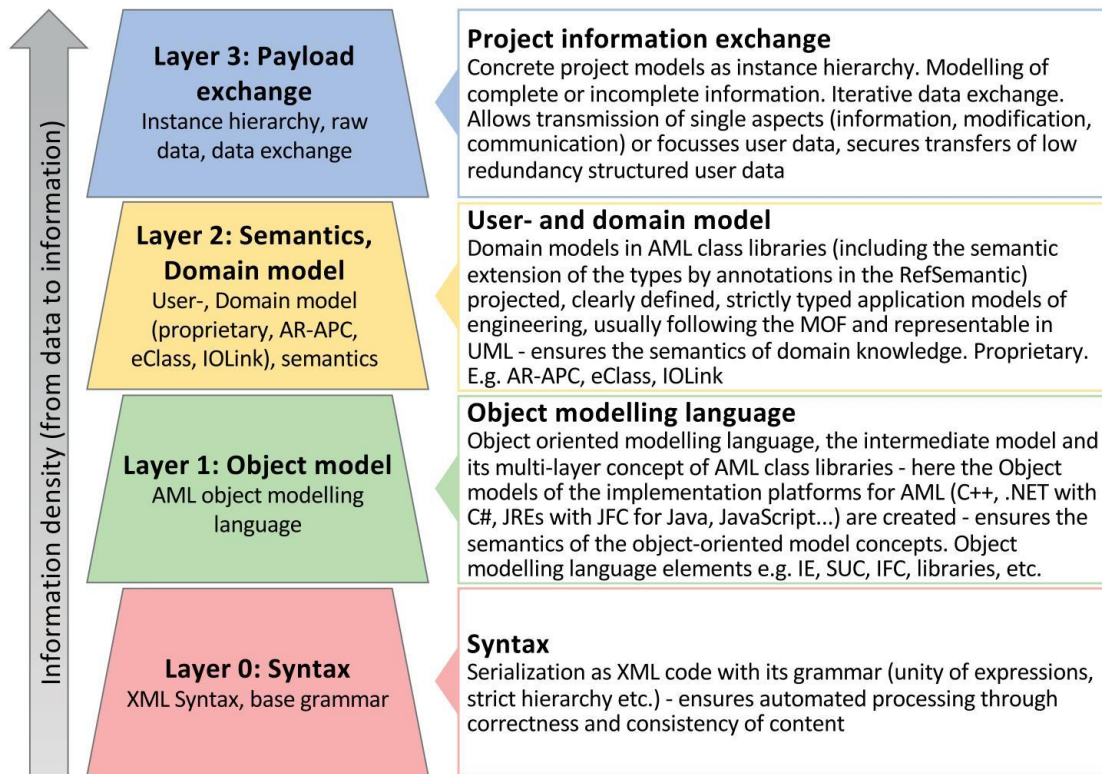


Abbildung 7: Vier-Ebenen-Modell, Abstraktion des AutomationML Datenaustauschs [2, S. 28]

Jede Ebene wird durch die Konzepte der jeweils darunterliegenden Ebene repräsentiert. Es ist somit nicht notwendig, auf die Repräsentation der darunterliegenden Ebene zu referenzieren. Beispielsweise ist es nicht nötig eine explizite XML-Syntax zu nutzen, wenn ein Konzept einer höheren Ebene diskutiert werden soll. Angefangen bei Ebene 0 erhöht sich die Informationsdichte auf dem Weg in die darüberliegende Ebene von „Daten“ bis hin zu „Informationen“. [vgl. 2, S. 28-31]

Ebene 0: Diese Ebene stellt eine reine XML-Syntax dar. Es handelt sich um eine serielle Ebene, eine Serie von ASCII-Zeichen, welche einer XML-Struktur folgen. In dieser existieren reine Daten und keine Informationen. Die Ebene ermöglicht eine automatisierte Datenverarbeitung und eine generelle Korrektheit und Konsistenz in Verbindung mit XML. Sie beinhaltet keine Semantik. [vgl. 2, S. 28-31]

Ebene 1: Basierend auf dem strukturellen Modell von XML, besteht die Ebene 1 aus einer objektorientierten Sprache auf CAEX Standard. Dieser Standard stellt ein Datenformat für die Modellierung von Objekthierarchien dar. Es definiert Objektelemente, wie Klassen, Attribute, Instanzen, Schnittstellen und Verbindungen. Es kann strukturelle Semantik und standardisierte Syntax darstellen. Aus diesem Grund ist der größte Teil des Datenmodells in Form von Daten vorzufinden. Dementsprechend ist der Anteil an Informationen gering. [vgl. 2, S. 28-31]

Ebene 2: Diese Ebene handelt von Semantik, welche in den domänenspezifischen Modellen genutzt wird. Sie besteht aus Sprachelementen der Ebene 1. Die domänenspezifischen Modelle, welche unabhängig von AutomationML existieren, können zunächst skizziert und dann in AutomationML elektronisch modelliert werden. Die Domänenmodelle können in Form von Unternehmensstandards oder auch in Form von öffentlichen Standards vorkommen. Domänenmodelle repräsentieren einen Informationstyp, jedoch enthalten sie keinen konkreten Daten zu den Informationen. [vgl. 2, S. 28-31]

Ebene 3: Diese Ebene handelt vom Informationsaustausch zwischen Engineering-Werkzeugen. Die Inhalte spezifischer Engineering-Daten können exportiert, ausgetauscht, importiert und gemanagt werden. In dieser Ebene handelt es sich um reine Informationen. Dementsprechend existieren keine Daten. [vgl. 2, S. 28-31]

3.4 Objektmodelle

Wenn im Zusammenhang mit AML von Objektmodellen die Rede ist, so sind Informationen der Abstraktionsebene 2 bis 3 (s. Kapitel 3.3) gemeint. Sie richten sich nach den Interessengruppen, für die sie modelliert werden. Es wird zwischen Metamodel, Type Model und Instance Model unterschieden (s. Abbildung 8). Der übliche Ablauf zur Erzeugung eines Anlagenmodells als „Instance Model“ sieht vor, dass mit der Definition eines generischen, allgemeingültigen Metamodells begonnen wird. Durch die Nutzung der generischen Komponentendefinitionen im Metamodel können herstellerspezifische Komponentenbibliotheken mit Vorlagenobjekten erstellt werden. Diese können schließlich dazu genutzt werden, um sie in einem Anlagenmodell zu instanzieren, d.h. die Vorlage zu einem vom Endkunden gewünschten Anlagenobjekt zu konkretisieren. Das Ergebnis ist ein Instance Model mit konkreter Anlagendefinition.

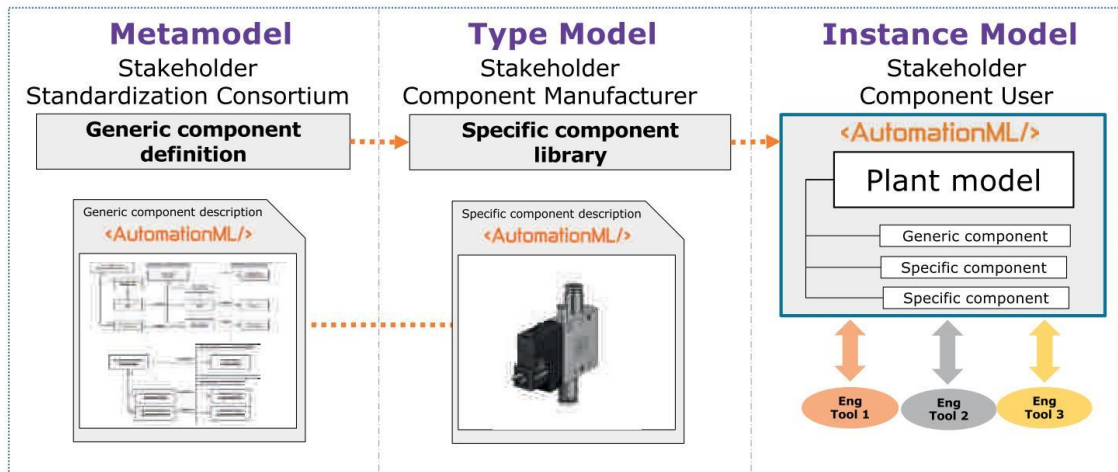


Abbildung 8: RoleClasses, SystemUnitClasses, InstanceHierarchy [3, S. 256]

Das Metamodel richtet sich nach einem Standardisierungskonsortium und stellt generische Komponentendefinitionen in Form von BaseRoleClassLibraries und InstanceClassLibraries für die Erzeugung von Komponentenbibliotheken zur Verfügung. Beispiele für generische RoleClasses (RC) und InterfaceClasses (IC) sind in Abbildung 9 und Abbildung 10 zu finden.

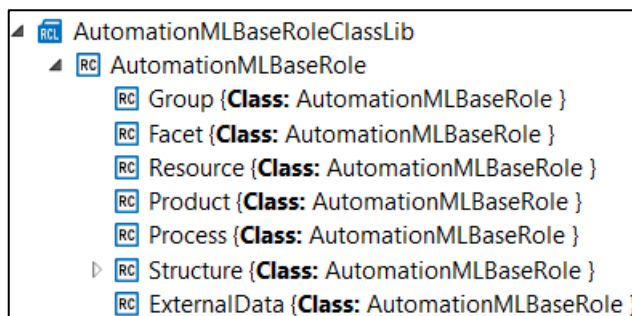


Abbildung 9: BaseRoleClassLibrary

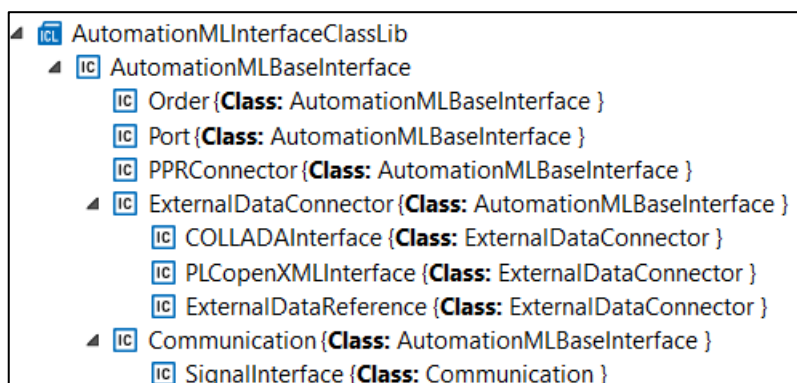


Abbildung 10: InterfaceClassLibrary

Generische RCs und ICs dienen zur standardisierten Klassifizierung von AML Objekten. Beispielsweise kann ein AML Objekt, welches einen Rohstoff oder ein Endprodukt darstellt, mit der RC „Product“ referenziert werden. Ein Beispiel für eine IC ist ein AML Objekt, welches eine Schnittstelle zu COLLADA-Geometriedaten darstellt. Diese wird mit einer IC „COLLADAInterface“ referenziert. Generische Klassifizierungen haben den

Zweck, AML-Modellierungsrichtlinien für bestimmte Rollen oder Schnittstellendefinitionen einzuhalten.

Das Type Model richtet sich an Komponentenhersteller, welche dem Nutzer eine spezifische Komponentenbibliothek in Form von herstellerspezifischen SystemUnitClassLibraries und RoleClassLibraries zur Verfügung stellen. Beispiele für herstellerspezifische SystemUnitClasses (SUC) und RCs sind in Abbildung 12 und Abbildung 11 zu finden.

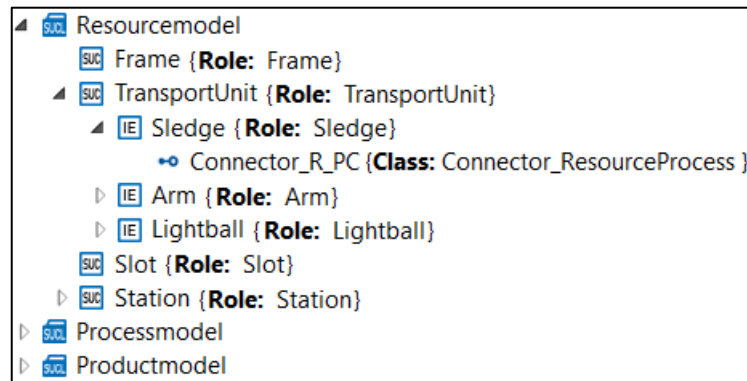


Abbildung 11: herstellerspezifische SystemUnitClassLibrary

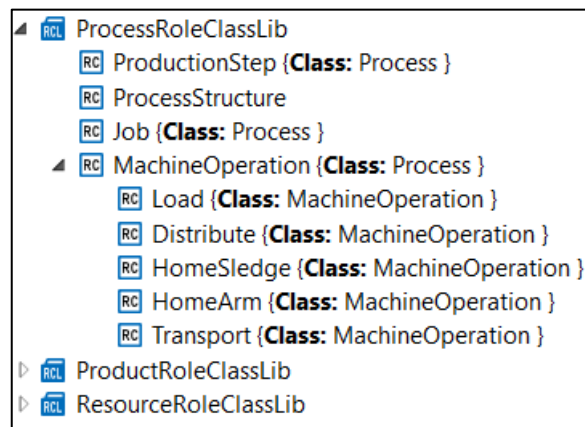


Abbildung 12: herstellerspezifisches RoleClassLibrary

Herstellerspezifische SUCs dienen zur spezifizierten Klassifizierung von Objektvorlagen, welche genutzt werden, um konkrete AML Objekte eines Anlagenmodells zu erzeugen. Am Beispiel der SUC „TransportUnit“ lässt sich die Funktionsweise der Objektvorlagen erklären. Das Vorlagenobjekt setzt sich aus den Bestandteilen „Sledge“, „Arm“- und „Lightball“-Objekt zusammen. Diese werden im Fall einer Instanzerzeugung der Eltern-Klasse „TransportUnit“ mitinstanziiert. Herstellerspezifische RCs dienen zur Erweiterung der generischen RCs und sind somit dafür zuständig einem Komponentenhersteller die Möglichkeit zu geben eigene Rollendefinitionen für domänenspezifische Klassifizierungen zu erstellen. So kann ein Komponentenhersteller beispielsweise Maschinenoperationen, wie „Load“, „Distribute“ oder „Transport“, definieren. Diese können zur Beschreibung von bestimmten Prozessoperationen von Anlagenstationen dienen.

Das Instance Model richtet sich hierbei an die Interessengruppe der Komponentennutzer und beschreibt das Anlagenmodell in Form einer InstanceHierarchy. Ein Beispiel für eine InstanceHierarchy ist in Abbildung 13 zu finden.

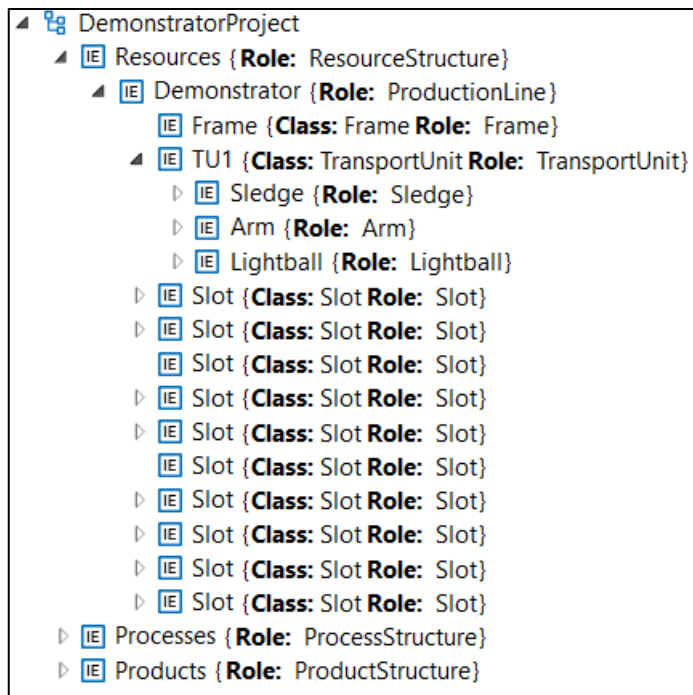


Abbildung 13: InstanceHierarchy

Beispielsweise wurde hier eine Instanz der Vorlagenklasse „TransportUnit“ instanziiert, welche die AML Objekte „Sledge“, „Arm“ und „Lightball“ beinhaltet. In der Abbildung ist die SUC Referenz mit dem Kürzel „Class:“ dargestellt. Diese bekommt in der Instance-Hierarchy eine Bezeichnung „TU1“. Um das AML Objekt trotz benutzerdefinierter Bezeichnung noch als Transporteinheit eindeutig zu identifizieren, ist eine Rollenreferenz vorhanden, in diesem Fall die Referenz zur RC „TransportUnit“. In der Abbildung ist die RC Referenz mit dem Kürzel „RC:“ dargestellt.

3.5 Beziehungsmodelle

Um technische Systeme modellieren zu können verlangt es nach Mechanismen, welche Datenobjekte zueinander in Beziehung stellen. Eine Beziehung stellt physische oder logische Assoziationen zwischen Objekten dar. In Abbildung 14 sind die Beziehungstypen in CAEX beispielhaft in einer Objekthierarchie eingebettet worden.

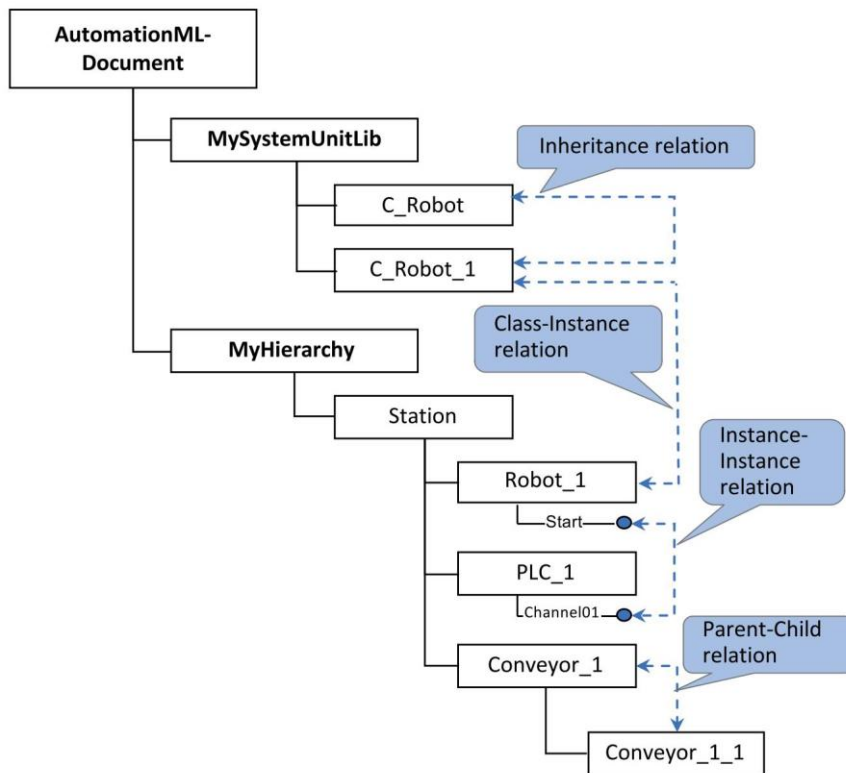


Abbildung 14: Beziehungstypen in CAEX [2, S. 93]

Die in der Abbildung aufgeführten Beziehungstypen werden in Tabelle 1 mithilfe von besonderen Eigenschaften des jeweiligen Typs näher erläutert.

Beziehungstyp	Eigenschaften
Parent-Child	<ul style="list-style-type: none"> • Hierarchische Objektstrukturen • Keine Vererbung • Dient nicht zur Modellierung von Bedeutungen • Dient zur Darstellung von Benutzerdefinierten Hierarchien
Inheritance	<ul style="list-style-type: none"> • Definiert durch die Referenz RefBaseClassPath • Identisches Konzept für InterfaceClasses, RoleClasses, SystemUnitClasses und AttributeTypes • Eine Kind-Klasse darf nur von einer Eltern-Klasse erben • Eine Eltern-Klasse darf an beliebig viele Kind-Klassen vererben • Gemischte Vererbungen sind nicht erlaubt. Beispiel: Eine SystemUnit-Class kann nur von einer anderen SystemUnitClass erben • Überschreiben der vererbten Daten ist durch Neudefinition dieser möglich
Class-Instance	<ul style="list-style-type: none"> • Definiert durch die Referenz RefBaseSystemUnitPath, RefBaseRoleClassPath, RefInterfaceClassPath oder RefAttributeType • Kopie einer Klasse inklusive ihrer Architektur und vererbten Informationen welche in einem InternalElement abgelegt wird • Das Bearbeiten innerhalb der Kopie hat keine Änderungen in der ursprünglichen Klasse zur Folge
Instance-Instance	<ul style="list-style-type: none"> • Modelliert durch ExternalInterfaces, welche mit InternalLinks verbunden sind • Alle AML kompatible Schnittstellen müssen direkt oder indirekt von einer AML standard interface class abgeleitet werden • „mirror“ Konzept zur Mehrfachnutzung von Objekten, Schnittstellen und Attributen mit einem „master“ Objekt

Tabelle 1: Beziehungstypen in CAEX [2, S. 98, 2, S. 99, 2, S. 97, 2, S. 95, 2, S. 95]

3.6 AMLX Container für AutomationML Komponenten

In Kapitel 3.2 wurde bereits erwähnt, dass in AML Referenzen zu externen Dateiformaten existieren können. Der Vorteil der Nutzung eines AMLX Containers ist, dass es möglich ist, alle mit dem AML Dokument verbundenen Dateien kombiniert in einem Container abzulegen. Es können insgesamt drei Anwendungsfälle unterschieden werden, bei denen AMLX Container genutzt werden [3, S. 276]:

- Austausch von vereinzelt AML Komponenten
- Austausch von Bibliotheken von AML Komponenten
- Austausch von Komponenten, welche im Detail spezifiziert sind und ein bestimmtes Profil erfüllen

In Abbildung 15 sind die Anwendungsfälle „Austausch von einzelnen AML Komponenten“ und „Austausch von Bibliotheken von AML Komponenten“ zu sehen.

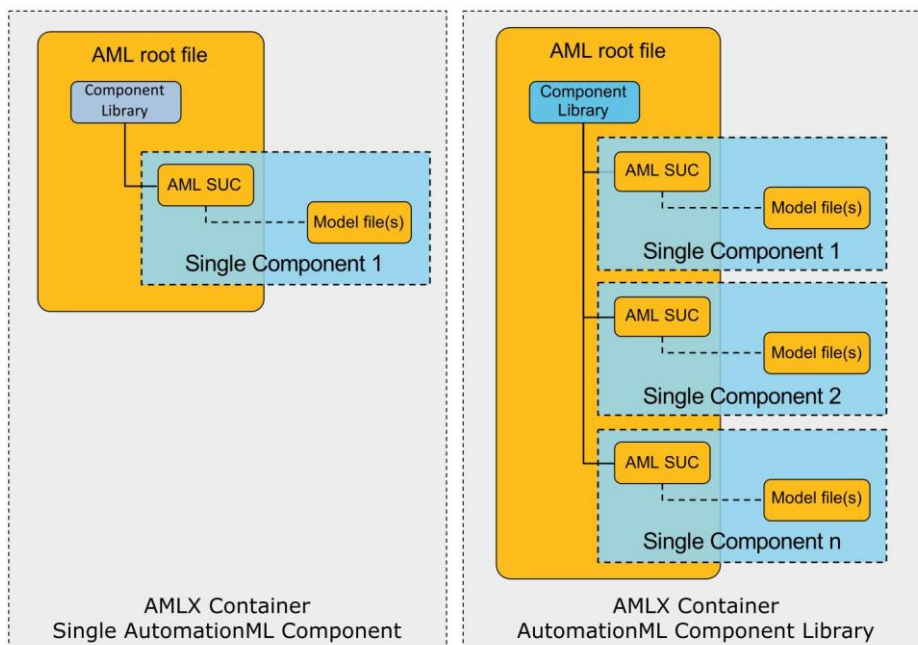


Abbildung 15: AMLX Container [3, S. 276]

3.7 AutomationML Editor

Der AML Editor wurde von der AML Vereinigung entwickelt, um die wichtigsten AML Konzepte sichtbar zu machen und um AML Dateien betrachten, erzeugen und bearbeiten zu können [2, S. 237]. In Abbildung 16 ist eine typische Ansicht des Editors mit einem darin geöffneten AML Dokument dargestellt.

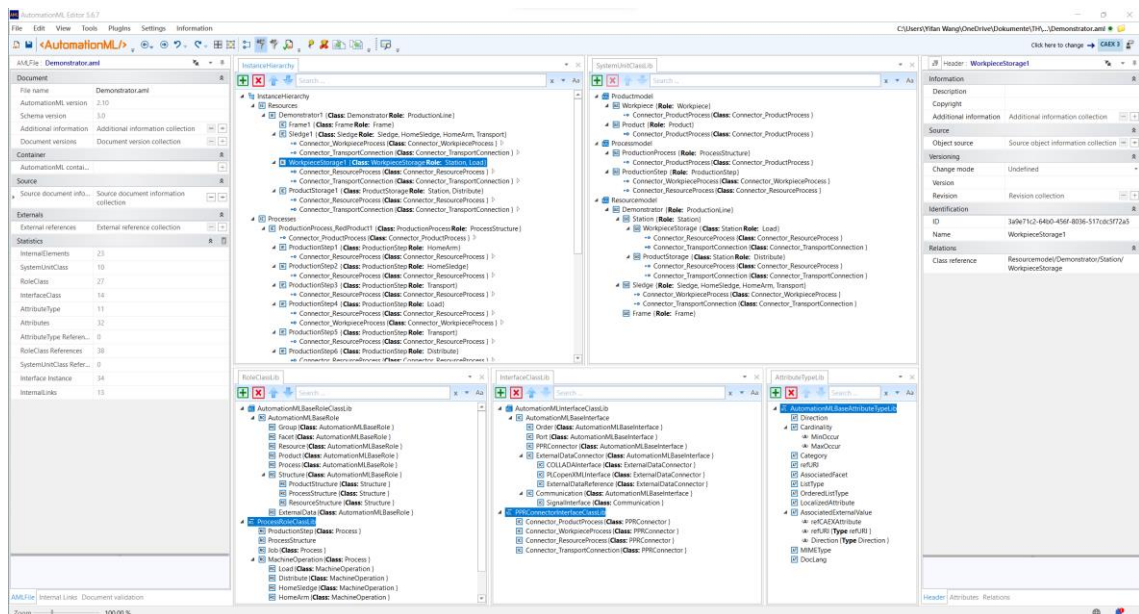


Abbildung 16: AML Editor Ansicht mit darin geöffnetem AML Dokument

3.7.1 Aufbau der Benutzeroberfläche

Mit dem Editor ist es möglich Informationen auf den Abstraktionsebenen 2 und 3 (s. Kapitel 3.3) zu modellieren. Dazu dienen Objektbausteine aus der Abstraktionsebene 1, welche InstanceHierarchy, SystemUnitClasses, RoleClasses und InstanceClasses repräsentieren. Auf der Benutzeroberfläche des Editors befinden sich Sektionen für verschiedene Arten von Dokumentinformationen, welche in Abbildung 17 zu erkennen sind. Insgesamt besteht sie aus 7 Sektionen.

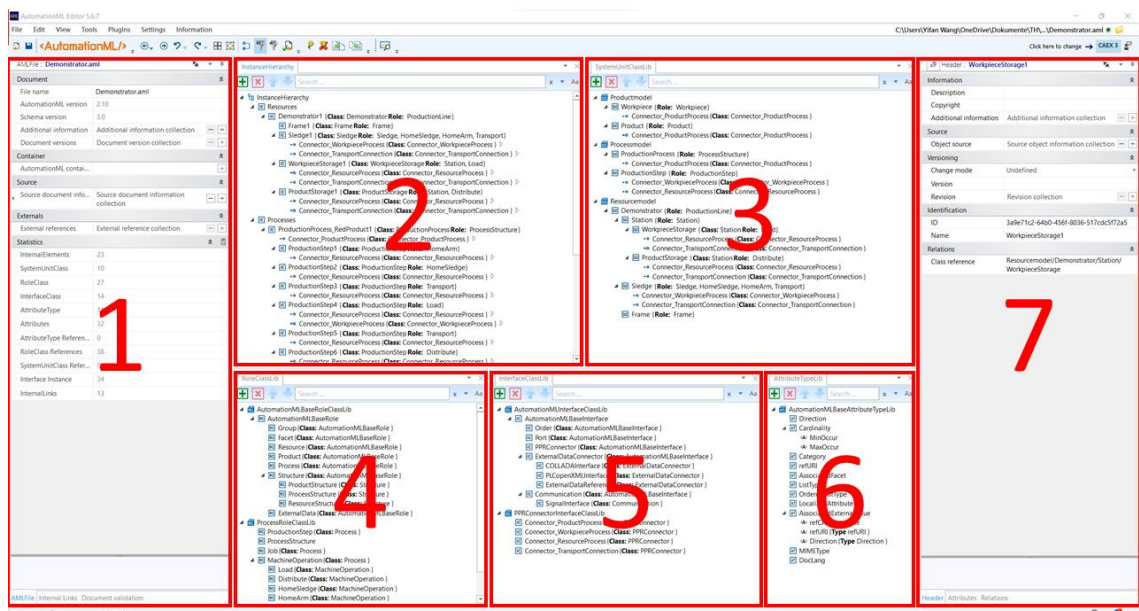


Abbildung 17: AML Editor Sektionen der Benutzeroberfläche

Sektion 1 beschreibt die Dokumentinformationen mithilfe von drei Tabs:

- AMLFile: Dieser Tab enthält editierbare Metainformationen über das AML Dokument
- Internal Links: Dieser Tab enthält alle modellierten internen Beziehungen des AML Dokuments
- Document Validation: In diesem Tab kann eine Validierung des AML Dokuments ausgeführt werden.

Sektion 2 stellt den InstanceHierarchy (IH) Editor dar. Dieser bietet eine Objektbaumansicht der IHs des AML Dokuments und stellt die Basisoperationen zum Bearbeiten der Objekte dieser IHs bereit.

Sektion 3, 4, 5 und 6 haben dieselbe Funktion wie Sektion 2 mit dem Unterschied, dass sie von SystemUnitClassLibraries, RoleClassLibraries, InterfaceClassLibraries und AttributeTypeLibraries, anstatt von InstanceHierarchies handeln.

Sektion 7 stellt den Properties Editor dar. Dieser enthält drei Tabs:

- Header: In diesem Tab können die Identifikationsinformationen des ausgewählten Objekts aus Sektion 2 – 6 dargestellt und bearbeitet werden.
- Attributes: In diesem Tab können alle Attribute des ausgewählten Objekts aus Sektion 2 – 6 dargestellt und bearbeitet werden
- Relations: In diesem Tab werden alle Beziehungen des ausgewählten Objekts aus Sektion 2 – 6 dargestellt.

3.7.2 Objekte erzeugen, modifizieren und löschen

Für die Sektionen 2-4 existiert im Editor eine eigene Toolbar zum Erstellen und Löschen von Objekten. Diese sind in Form von grünen und roten Buttons mit den Symbolen „+“ für das Erzeugen und „-“ für das Löschen von Objekten zu finden. Der Typ des erzeugten Objekts hängt von dem zum Erzeugungszeitpunkt angewählten Objekt ab. Sollten mehrere alternative Typen zu Auswahl stehen, so werden diese in einem Selektionsdialog mit angeboten. Die gebräuchlichsten Typen können auch direkt über das Kontextmenü mit einem Rechtsklick erreicht werden. Das Modifizieren von Eigenschaften erfolgt, wie es häufig in modernen Programmierungsumgebungen zu finden ist, über einen Properties Editor. [2, S. 241]

3.7.3 Modellierung von Vererbungen

Die einfachste Art Vererbungen zu modellieren ist, die Eltern-Klasse per „drag and drop“ auf die Kind-Klasse zu ziehen. Nach diesem Vorgang schlägt der AML Editor mehrere mögliche Ausführungsmöglichkeiten vor. Dabei soll „Class inheritance“ gewählt werden. Eine andere Möglichkeit Vererbungsbeziehungen zu erzeugen ist, die Elternklasse in „Relations reference“ oder „Class reference“ unter dem „Header“-Tab als vollständigen

Pfad einzutragen. Diese Eintragung kann entweder manuell oder auch per „drag and drop“ Funktion erfolgen, indem die Elternklasse auf das Eingabefeld gezogen wird. [2, S. 243]

3.8 AutomationML Engine

AutomationML Engine (Aml.Engine) stellt eine Softwarekomponente dar welche eine API bereitstellt, um AML Dokumente basierend auf der C#.NET Sprache zu verarbeiten. Die Implementierung basiert auf den .NET Standardbibliotheken „System.Xml.Linq“. Diese ermöglichen eine Verarbeitung von XML-Dokumenten, indem es das gesamte Dokument in den Zwischenspeicher lädt. Mithilfe der Integration von Linq ist es möglich Abfragen am XML-Dokumente durchzuführen, um benötigte Elemente und Attribute herauszufiltern. [3, S. 28]

Die wichtigsten Funktionalitäten werden im Folgenden aufgelistet [3, S. 28]:

- Laden von XML-Dokumenten von Dateien oder Streams
- Serialisierung von XML zu Dateien oder Streams
- Erzeugen von neuen XML-Dokumenten
- Abfragen von XML-Dokumenten
- Manipulierung der Zwischengespeicherten XML-Elemente
- Validierung von XML-Dokumenten mittels XSD

3.9 AutomationML Datenaustausch Architektur

Die grundlegende Systemarchitektur einer AML-basierten Datenschnittstelle besteht aus mehreren Ebenen. Diese ist in Abbildung 18 zu finden.

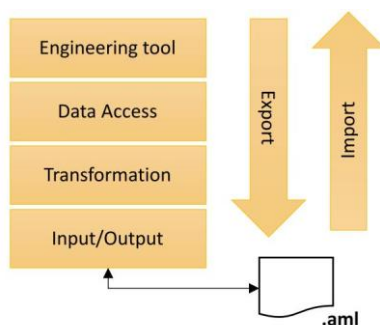


Abbildung 18: Grundlegende Systemarchitektur von Datenschnittstellen [3, S. 42]

Es folgt eine Beschreibung der in der Abbildung aufgeführten Ebenen [vgl. 3, S. 42]:

- *Engineering tool*: Diese Ebene stellt die domänenspezifischen Daten innerhalb des Engineering-Software dar.

- *Data Access*: Voraussetzung für einen AML-basierten Export oder Import ist der Zugang zu den Daten des Engineering Werkzeugs. Diese ist in Form einer Datei oder einer Softwareschnittstelle zu finden.
- *Transformation*: Die Funktion der „Transformation“ Ebene besteht darin, gelesene Daten während eines Exports mit Informationen anzureichern, sodass beim Import in derselben Ebene die transferierten Daten interpretiert werden können. Für das Anreichern der Daten mit Informationen (Semantik) werden RCs genutzt. Engineering-Daten müssen eine objektorientierte Datenmodellierung mit Instanzen und Klassen unterstützen, um eine Übersetzung oder Rückübersetzung eines spezifischen Datenformats in ein AML Format zu ermöglichen. Sollten diese objektorientierten Informationen nicht vorhanden sein, müssen sie in der „Transformation“ Ebene erzeugt werden. Zusätzlich müssen alle Datenobjekte mit einzigartigen IDs versehen werden. Die IDs entsprechen dem GUID-Standard und ermöglichen eine eindeutige Identifizierung der Daten für einen zuverlässigen Datenaustausch.
- *Input/Output*: Diese Ebene wird dazu genutzt die Daten zu transferieren. Sie generiert die AutomationML Dateien.

3.9.1 Implementierungskonzept für die „Transformation“ Ebene einer Exportschnittstelle

Bei den Implementierungskonzepten für die „Transformation“ Ebene einer Exportschnittstelle wird zwischen internem Generierungskonzept und externem Generierungskonzept unterschieden. In dieser Arbeit ist nur das externe Generierungskonzept von Bedeutung, welches im Folgenden erläutert wird.

Das externe Generierungskonzept (s. Abbildung 19) nutzt SUCs als Blaupause für die Generierung der AML Objekte. Im „Generator“ können AML Objekte als Instances aus den SUCs instanziiert werden. Nach der Instanziierung können die dazugehörigen Attributwerte und Semantiken transferiert werden. Die zur Auswahl stehenden RCs werden auf die den SUCs zugewiesenen SupportedRoleClasses beschränkt. Der „Interpreter“ entscheidet, welche SUCs welchem Datenobjekt zugeordnet werden.

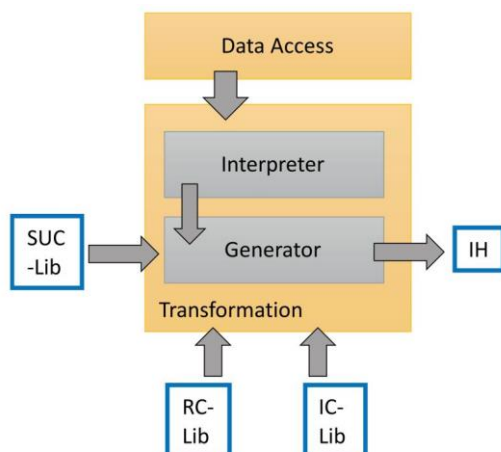


Abbildung 19: Externes Generierungskonzept einer Exportschnittstelle [3, S. 47]

3.10 Product-Process-Resource (PPR) Konzept

Um komplexe Anlagendaten strukturieren zu können, hat sich eine Aufteilung in Produkte, Prozesse und Ressourcen als praktikabel herausgestellt.

In einer produktzentrierten Sicht steht das Produkt im Fokus der Strukturierung. Unter „Produkt“ wird in AML ein Endprodukt, ein Zwischenprodukt oder ein Rohmaterial verstanden. Es stellt somit nicht zwangsweise ein finales Element in der Herstellung dar. Zur Beschreibung von Produkten gehören Testergebnisse, Produktdaten und die dazugehörigen Dokumentationen. Produkte werden ihren Prozessen, welche diese nutzen, und ihrer Ausstattung, welche für die Manipulation dieser genutzt werden, mithilfe von Konnektoren zugeordnet. [4, S. 66]

In einer prozesszentrierten Sicht stellen Prozesse die zentralen Elemente des Modells dar. Ein Prozess in AML repräsentiert den Produktionsprozess mitsamt der Subprozesse. Bestandteile eines Prozesses sind Prozessparameter, Prozesskette und Prozessplanung. Im technischen Sinne verändern Prozesse Produkte mithilfe der dafür benötigten Ressourcen. Prozesse haben also Verbindungen zu Ressourcen und Produkten, genauso wie auch umgekehrt. [4, S. 66]

In einer ressourcenzentrierten Sicht sind Ressourcen die zentralen Bestandteile des Modells. Sie führen Prozesse aus und bearbeiten Produkte. Auftauchen können sie in Form eines Produktions- oder Softwaresystems. In AML werden Ressourcen typischerweise als Hierarchie modelliert, welche die Anlagentopologie widerspiegelt. [4, S. 66]

In Abbildung 20 ist eine beispielhafte InstanceHierarchy mit PPR Konzept in AML dargestellt.

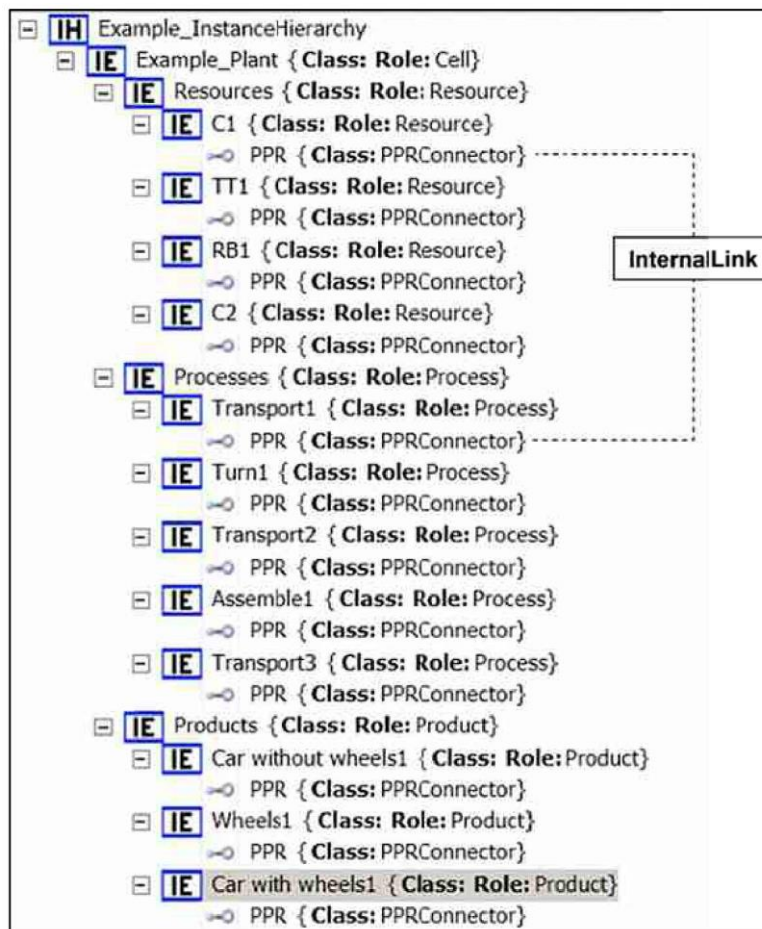


Abbildung 20: PPR InstanceHierarchy [4, S. 71]

4 Grundlagen zu Automatisierungssystemen und modularen Anlagensystemen

Bei dem in dieser Arbeit entwickeltem Fallbeispiel handelt es sich um ein Automatisierungssystem, welches repräsentativ durch ein modulares Anlagensystem dargestellt wird. Zur Klärung der Verwendung der zwei Begriffe „Automatisierungssystem“ und „modulares Anlagensystem“, werden zunächst ihre Begriffsdefinitionen gebildet. Um den Zusammenhang zwischen dem genutzten Fallbeispiel und modularen Anlagensystem aufzuzeigen, wird als nächstes der Entstehungsprozess von modularen Anlagen vorgestellt. Dieser dient dazu, die Nutzung der im Fallbeispiel auftauchenden Domänen inhaltlich einzuordnen.

4.1 Über Automatisierungssysteme

Automatisierungssysteme stellen Systeme dar, welche automatisierte Prozessabläufe einsetzen. In der industriellen Fertigung werden sie oft für komplexe Produktionsprozesse eingesetzt, bei denen hohe Anforderungen an zuverlässige Qualität, Präzision und Geschwindigkeit erfüllt werden müssen. So werden Aufgaben, welche bei einer manuellen Bearbeitung fehleranfällig wären oder zu langsam ablaufen würden, durch automatisierte Maschinen ausgeführt. Somit wird der Mensch vor z.B. eintönigen, feinmechanisch anspruchsvollen oder körperlich belastenden Aufgabenbereichen bewahrt. Automatisierungssysteme sind dafür da, einen großen Anteil an ausführender Arbeit in einer industriellen Fertigung zu übernehmen, wobei der Mensch folglich verstärkt für dispositive Aufgaben, wie z.B. Planungs- und Wartungsaufgaben, zuständig sein wird. [vgl. 5]

Mit zunehmendem Grad an Digitalisierung tendiert die Industrie zu einer Aufrüstung von Maschinen mit umfangreicheren Funktionalitäten. Damit werden steigende Produktkomplexitäten, resultierend aus dem hohen Konkurrenzdruck der modernen, globalisierten Welt, entgegnet. Aus einer Funktionserweiterungen von automatisierten Maschinen ergibt sich ein erhöhter Aufwand hinsichtlich des Informationsmanagements. Die Kosten für menschliche Arbeitskraft sind besonders in den führenden Industriestaaten hoch und erschweren ein einfaches Hochskalieren der Mitarbeiterzahl. Um sich diesem Aufwand wirtschaftlich anzupassen, werden verstärkt Computerhardware (SPS) und -software (PPS) zur automatisierten Verarbeitung der zusätzlich anfallenden Informationen genutzt. Die Folge daraus sind komplexere Anlagen mit erhöhter Anzahl von Schnittstellen unterschiedlicher Art und somit ein erhöhter Aufwand in der Anlagenplanung. Eine Möglichkeit diesem zu entgegnen ist ein modulares Anlagenkonzept zu nutzen. Im folgenden Abschnitt wird der Begriff eines „modularen Anlagensystems“ näher erläutert.

4.2 Über modulare Anlagensysteme

In einem modularen Anlagensystem werden Prozesskomponenten, Messeinrichtungen, Rohrleitungen und elektrische Verkabelung in einem Stahlrahmen montiert. Die montierte Einheit wird als Modul bezeichnet. Jedes Modul ist eine in sich geschlossene Prozesseinheit, welche in der Regel vor der Anlieferung an den Anlagenstandort vormontiert wird. Eine modulare Anlage kann aus mehreren Modulen bestehen, welche am Anlagenstandort zu einem großen Anlagensystem zusammengefügt werden. [6]



Abbildung 21: Anlagenmodul, bestehend aus Stahlrahmen mit vormontierten Prozesskomponenten, Messeinrichtungen, Rohrleitungen und elektrischer Verkabelung [6]

Die Modulbauweise erfordert mehr Material als die herkömmliche Bauweise, da jedes Modul so konstruiert und ausgelegt werden muss, dass es eigenständig einen sicheren Stand bietet und Beanspruchungen beim Transport, Anheben und Aufstellen standhält. Somit müssen sie strukturell stärker aufgebaut sein als Einheiten, welche vor Ort zusammengebaut werden. [6]

4.2.1 Vor- und Nachteile

Modulare Bauweisen werden aufgrund ihrer Vielzahl an Vorteilen genutzt. Dabei sind die folgenden Punkte von zentraler Bedeutung [6]:

- **Baustellensicherheit:** Der Bau von Anlagen findet üblicherweise unter Risiken und Gefahren, wie Arbeiten in extremen Höhen, Nutzung von schweren Gerätschaften und elektrischen Hochspannungsumgebungen, statt. Oft findet der Bau sogar unter freiem Himmel statt, was wetterbezogene Schäden als Risiko miteinschließt. Modulare Anlagenkomponenten werden hingegen in sicherer und kontrollierter Innenumgebung vormontiert.
- **Qualität:** Die Entwicklung und Fertigung werden meist am selben Standort durchgeführt, sodass eine bessere Kommunikation zwischen den jeweiligen

zuständigen Teams stattfinden kann. So wird eine hohe Zuverlässigkeit der Qualität sichergestellt.

- **Effiziente Zeitplanung:** Für Märkte mit kritischen time-to-market Anforderungen ist eine kurze Zeitspanne zwischen Konzept und fertigem Produkt wichtig. Im modularen Anlagenbau besteht die Möglichkeit neben den Baustellen- und Fundamentarbeiten auch die Fertigung der Anlagenkomponenten durchzuführen. Zusätzlich werden durch die Fertigung im Innenraum Bauverzögerungen durch Wettereinflüsse minimiert.
- **Flexibilität:** Eine Anlage, bestehend aus vielen kleinen Produktionseinheiten, bietet eine größere Flexibilität in der Anordnungsplanung. Es ist somit möglich sowohl alle Module zentralisiert, als auch geografisch verteilt zu betreiben. Eine Zentralisierung hat den Vorteil der besseren Skalierbarkeit. Ein verteilter Betrieb hat den Vorteil, die Module durch geringfügige Anpassungen an unterschiedliche Rohstoff- oder Endprodukthanforderungen anzupassen.
- **Minimale Arbeit vor Ort:** Die Bestandteile einer modularen Anlage erreichen die Baustelle vormontiert, getestet und elektrisch verdrahtet. Somit wird nur ein geringer Aufwand für die Verbindung der Module und die Inbetriebnahme benötigt.

Trotz der Vielzahl an Vorteilen sind die folgenden Nachteile zu berücksichtigen [6]:

- **Transport:** Die Beschränkungen und Kosten beim Transport stellen aufgrund der großen Abmaße eines Anlagenmoduls eine Herausforderung dar. Es muss auf Höhen- und Breitenbeschränkungen bestimmter Routen geachtet werden. Außerdem ist der Zustand der Fahrbahn und die Wetterbedingungen während des Transports zu beachten.
- **Upfront engineering:** Eine modulare Bauweise erfordert ein hohes Maß an vorangehender Planung. Es muss ein detailliert ausgestaltetes Modell bestehen, bevor ein Anlagenmodul gefertigt werden kann. Somit tauchen zusätzliche Anforderungen hinsichtlich Planung, Kommunikation, Koordination und Projektmanagement unter mehreren Interessengruppen auf. Dazu zählen z.B. der Endnutzer, der Bauunternehmer und die Modulhersteller.

4.3 Entstehungsprozess einer modularen Anlage

Die Entstehung einer industriellen Anlage kann durch fünf grundlegende Prozesse beschrieben werden: Planung, Entwicklung, Fertigung, Montage und Inbetriebnahme (s. Abbildung 22).



Abbildung 22: Entstehungsprozess modularer Anlagensysteme in Anlehnung an [7]

Dabei werden bis zum Ende des Inbetriebnahmeprozesses Kosten verursacht, welche durch einen negativen Cashflow beschrieben werden können. Darauffolgend setzt die Gewinngenerierung ein, welche als positiver Cashflow dargestellt werden kann. Mithilfe der oben genannten Vorteile eines modularen Anlagenkonzepts ist es möglich eine weitgehende Parallelisierung von Fertigung und Montage zu realisieren und die Inbetriebnahmedauer zu verkürzen. Somit ist ein früheres Einsetzen des Zeitpunkts der Gewinngenerierung die Folge. Gleichzeitig wird durch einen erhöhten Bedarf an vorangehender Planung und Entwicklung ein verstärkter negativer Cashflow im vorderen Zeitabschnitt der Anlagenentstehung zu beobachten sein (Abbildung 23).

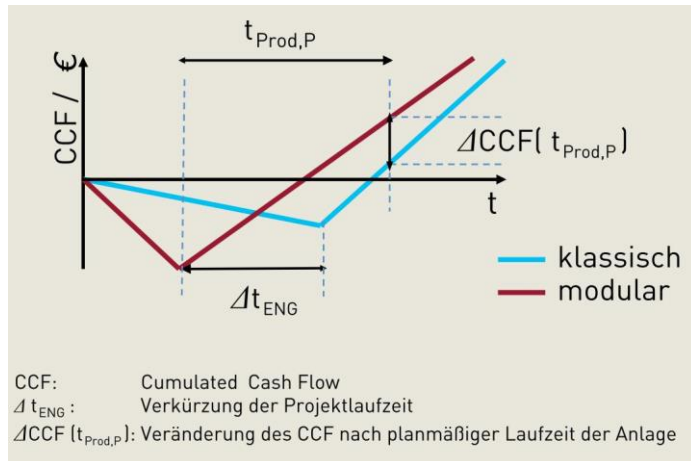


Abbildung 23: Kumulierter Finanzfluss bei klassischem Engineering (blaue Linie) und beschleunigtem Engineering auf Basis modularer Anlagen (rote Linie) [8]

Die Entstehung des Entwurfs der Anlagentopologie ist in dieser Arbeit von besonderer Bedeutung. Um diesen zu generieren ist eine vorangehende Planungsphase nötig, welche im Folgenden näher erläutert wird.

4.3.1 Planungsphase einer modularen Anlage

Die Planungsphase einer modularen Anlage beinhaltet den *modularen Konzeptentwurf*, aus dem ein Konzept für die zu nutzenden Modulbausteine folgt, sowie drei weitere Schritte zur detaillierteren Planung der Modulbausteine (s. Abbildung 24).

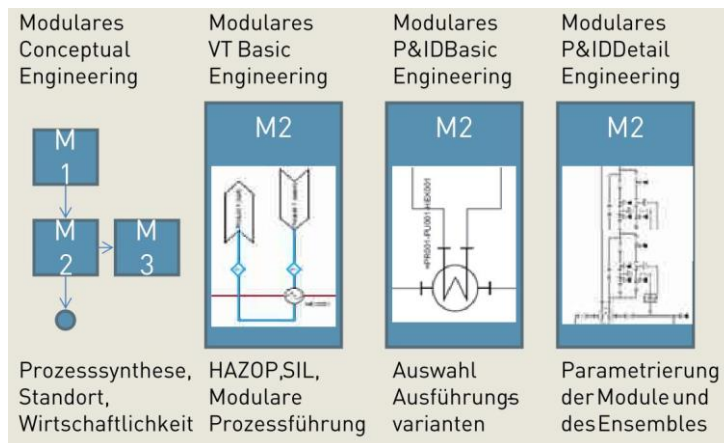


Abbildung 24: Modulares Engineering [8]

Im *modularen Konzeptentwurf* wird die Anordnung der benötigten Produktionsprozesse zusammengestellt. Die Prozessanordnungen werden hinsichtlich Kosten, Produktqualität und Energieverbrauch auf ihre Wirtschaftlichkeit geprüft. Jedem Produktionsprozess kann dabei ein Modulbaustein zugeordnet werden, welcher seinen jeweiligen Prozess ausführen soll.

Für jeden Modulbaustein folgt daraufhin eine *sicherheitstechnische Entwurfsplanung* der Prozessführung mittels Risikobewertung der funktionalen Sicherheit. Darin geht es um eine systematische Identifizierung von potenziellen Gefahren, welche durch Systeme, Prozesse oder Anlagenteile verursacht werden können und die Funktionsfähigkeit der Anlage beeinflussen. Aus der *Auswahl der Ausführungsvarianten* und der *Parametrisierung der Module* folgt ein vollständig geplantes Modul.

Als Ergebnis der Planungsphase entsteht ein Entwurf der Anlagentopologie.

5 Grundlagen zur Virtuellen Inbetriebnahme

Die Virtuelle Inbetriebnahme (VIBN) ist ein Konzept, welches eine effiziente Modellierung, Programmierung, Visualisierung und Validierung eines Produktionssystems oder -prozesses in einer virtuellen Umgebung ermöglicht. Dies verkürzt die Dauer der klassischen Inbetriebnahme und führt zu einem früheren Produktionsstart von industriellen Anlagen. Für die Simulation von VIBN-Modellen wird in dieser Arbeit die Software fe.screen-sim verwendet.

5.1 Über Virtuelle Inbetriebnahme

VIBN kann je nach Einsatzzweck in verschiedenen Szenarien angewandt werden. Es kann bspw. eine visuelle Repräsentation oder Validierung von Sicherheitsfunktionen zum Ziel haben. Wenn VIBN im vollen Umfang der Steuerung modelliert, simuliert und getestet wird, so ist das Resultat ein digitales Abbild des implementierten Systems. [9, S. 1254]

Zeitlich lässt sich der Beginn der VIBN im Modellierungsprozess einordnen (s. Abbildung 25). So können CAD-Daten genutzt werden um sie in ein Simulationsmodell einzupflegen, worin Anlagenbestandteile in Echtzeit angesteuert werden können. Der Prozess der VIBN verläuft parallel zum klassischen Entstehungsprozess einer industriellen Anlage, welche aus Planung, Entwicklung, Montage und Inbetriebnahme besteht. Dadurch, dass entwickelte Konzepte frühestmöglich getestet werden, wird die Dauer der klassischen Inbetriebnahme verkürzt. Somit unterstützt die VIBN das Prinzip des „Frontloadings“. Der Begriff „Frontloading“ ist ein Gestaltungsprinzip aus dem Lean Development und besagt, dass der größte Anteil an Produktkosten im frühen Stadium der Produktentwicklung festgesetzt werden. Aus diesem Grund soll ein erhöhter Planungsaufwand für nachhaltigere Entscheidungen und Fehlervermeidung sorgen, denn je später ein Fehler in einem Produktionsprozess erkannt wird, desto mehr Kosten werden von diesem verursacht.

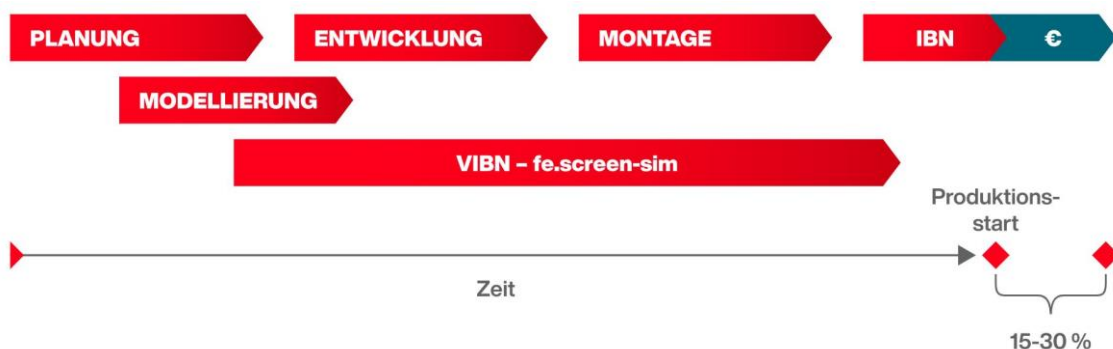


Abbildung 25: Virtuelle Inbetriebnahme im Entstehungsprozess eines Produktionssystems [7]

5.2 Aufbau eines Virtuelle Inbetriebnahme Modells

Ein VIBN-Modell kann in drei Bereiche eingeteilt werden: Physiksimulation, Verhaltenssimulation und Automation.

Wie schon im Kapitel 5.1 erwähnt, sind vorhandene CAD-Daten die Voraussetzung für die Erstellung eines Modells für die Physiksimulation. Diese werden als Geometriemodell in die Simulationsumgebung importiert. Daraufhin muss das Geometriemodell kinematisiert werden. Zur Kinematisierung werden üblicherweise Gelenke genutzt. Diese definieren Randbedingungen der Bewegung in Richtung verschiedener Freiheitsgrade. Danach folgt die Definition von Sensoren und Aktoren. Beispiele für Sensoren sind Kollisionssensoren, welche Lichtschranken darstellen können, oder Sensoren zum Erfassen von Parametern der kinematisierten Objekte, wie Position oder Geschwindigkeit. Beispiele für Aktoren sind Transportflächen, inverse Kinematiken, Positionsregler und Geschwindigkeitsregler.

Als nächstes erfolgt eine Verhaltenssimulation, welche die Schnittstelle zwischen einer Steuer- und Regelungseinheit (z.B. SPS) und der Physiksimulation darstellt. In dieser werden die definierten Sensoren und Aktoren über Verhaltensmodelle angesteuert, welche in einer realen Anlage beispielsweise das Anfahr- und Ausbremsverhalten von Antriebsmotoren einer Transportfläche darstellen können. Verhaltensmodelle sind aus verschiedenen Logikbausteinen zusammengesetzt. Diese verarbeiten Ein- und Ausgangssignale in beide Schnittstellenrichtungen.

Der Bereich der Automation wird durch eine automatisierte Regelung der Sensoren und Aktoren mit einer Steuer- und Regelungseinheit ermöglicht. Je nach Anwendungsfall kann die Einheit in Form von Hardware (z.B. SPS oder Mikrocontroller) oder Software (API-Anbindung) vorkommen. Je nach Art der Ansteuerung spricht man entsprechend von einer „Hardware in the loop“-Simulation oder einer „Software in the loop“-Simulation.

5.3 Umgebung für die Virtuelle Inbetriebnahme: fe.screen-sim

Die Software fe.screen-sim ist eine VIBN-Anwendung, welche auf einer modularen Softwarestrukturaufbaut (s. Abbildung 26). Als Herzstück dient der „Core“, auf dem das Simulationsmodell in Echtzeit ausgeführt wird. Während die Simulation läuft, ist es möglich sich mithilfe eines „Clients“ mit dem „Core“ zu verbinden, um das Simulationsmodell zu bearbeiten. Dazu stehen dem „Client“ verschiedene Werkzeuge zur Verfügung, welche den Import von CAD-Modellen, die Erstellung von Verhaltensmodellen (Logic Creator), die Nutzung von Vorlagenobjekten und die direkte Steuerung des Simulationsmodells ermöglichen. Des Weiteren besitzt die Anwendung auch Schnittstellen zur Softwareerweiterung (API), welche es ermöglicht, das Simulationsmodell im „Core“ mithilfe von selbst entwickelter Software zu bearbeiten.

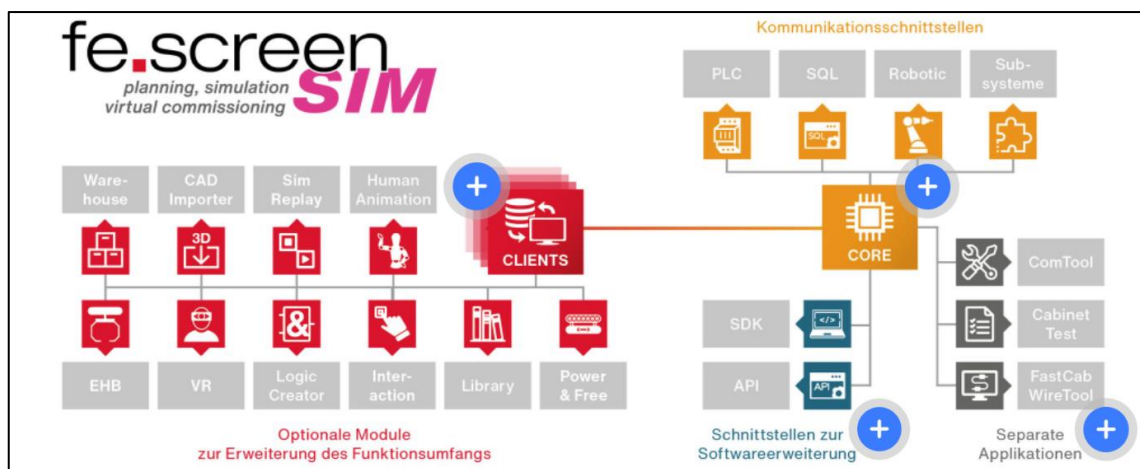


Abbildung 26: fe.screen-sim Softwarestruktur [10]

6 Nutzung von VIBN und AML im Entstehungsprozess von modularen Anlagen

Die Phasen des Entstehungsprozesses von modularen Anlagen wurden bereits in Kapitel 4.3 erläutert und bestehen aus Planung, Entwicklung, Fertigung, Montage und Inbetriebnahme. Das Konzept der Modularisierung in der Anlagenplanung erreicht eine Verkürzung des Entstehungsprozesses durch die Parallelisierung von Fertigung und Montage. Zusätzlich wird durch den Einsatz von bereits getesteten Teilfunktionen der vormontierten Module eine zuverlässigere Inbetriebnahme gewährleistet.

Mithilfe der Virtuellen Inbetriebnahme ist es möglich die klassische Inbetriebnahmedauer auf ein erforderliches Minimum zu verkürzen. Um umfangreiche Engineering Prozesse, wie das Erzeugen eines VIBN Simulationsmodells nicht wiederholt manuell durchführen zu müssen, ist es nötig ein automatisierbares Format zur Darstellung, Speicherung und Transfer der Anlagendaten zu nutzen. Für diesen Zweck wird die Modellierungssprache AutomationML genutzt.

6.1 Nutzung von Virtueller Inbetriebnahme

Die Virtuelle Inbetriebnahme kann ablauftechnisch hinter der Planungsphase eingeordnet werden (s. Abbildung 27). Mit ihr ist es möglich frühzeitig innerhalb der Gesamtanlage Automatisierungsprozesse auf ihre Funktionalität zu prüfen, Prozessabläufe zu konfigurieren und Fehlerszenarien zu simulieren.

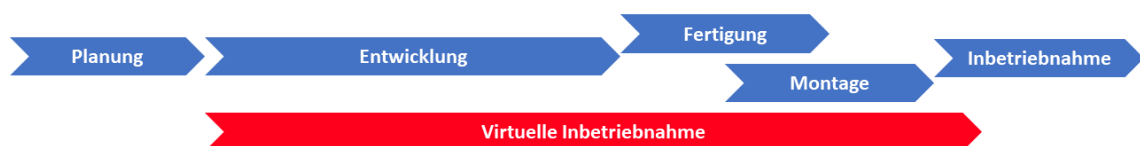


Abbildung 27: Einordnung der VIBN in den Entstehungsprozess von modularen Anlagen in Anlehnung an [7]

Voraussetzung für die wirtschaftliche Nutzung der VIBN sind wiederverwendbare Vorlagemodelle. Diese sind bei der Planung von modularen Anlagen bereits vorhanden, womit sie einen guten Anwendungsfall für die VIBN darstellen. Sollten die Vorlagen keine Kinematik- und Verhaltensmodellinformationen enthalten, müssen sie noch mit diesen angereichert werden. Die angereicherten Modelle werden danach in die VIBN Simulationsumgebung importiert. Um die vereinzelt Modelle in der Simulationsumgebung zu einer funktionsfähigen digitalen Gesamtanlage zu verbinden, werden vordefinierte Schnittstellen in den Verhaltensmodellen miteinander verknüpft. Nach der erfolgreichen Erstellung des digitalen Abbilds der Gesamtanlage, kann es schließlich für ein frühzeitiges Testen der Anlagenbestandteilen genutzt werden, um die klassische Inbetriebnahmedauer zu verkürzen.

6.2 Nutzung von AutomationML

Die Modellierung des Anlagenmodells im AML Format reiht sich unmittelbar hinter der Fertigstellung des Entwurfs der Anlagentopologie ein, welcher sich in der Planungsphase im Entstehungsprozess modularer Anlagen befindet. Das Anlagenmodell im AML-Format ist die Voraussetzung für eine automatisierte Modellierung des VIBN-Simulationsmodells und stellt somit die Schnittstelle zwischen der Planungsphase und der VIBN-Phase dar.



Abbildung 28: Einordnung des AML-Anlagenmodells in den Entstehungsprozess von modularen Anlagen

Für eine automatisierte Nutzung des AML Formats werden durch die AML API „Aml.Engine“ erreichbare Quelldaten benötigt, aus denen Informationen für einen AML-Export bereitgestellt werden. Da Aml.Engine die Programmiersprache C# nutzt, müssen die Quelldaten dementsprechend ebenfalls mit C# abrufbar sein. Ebenso muss die Zielumgebung, mit einer C# Anwendung manipulierbar sein, um einen Import von Informationen aus einem AML-Dokument zu ermöglichen. In dem kommenden Fallbeispiel wird eine serielle Schnittstelle der Anlage zur Übertragung der Anlagentopologie genutzt. Als Software für die VIBN wird fe.screen-sim genutzt. Sowohl die serielle Schnittstelle, als auch die VIBN-Software besitzen eine C# Schnittstelle, welche somit die Voraussetzungen für die Nutzung des AML Formats erfüllen.

Beim Exportvorgang wird die Anlagentopologie über die serielle Schnittstelle übergeben. Dort werden die darin enthaltenen Modulinformationen in Form einer PPR-Struktur (s. Kapitel 3.10) in ein AML-Dokument übertragen. Der Exporter setzt hierbei auf das externe Generierungskonzept (s. Kapitel 3.9.1) bzw. nutzt AML-Vorlagenmodelle in Form von SUCs für die Modulgenerierung. Neben den Topologieinformationen werden nach demselben Prinzip auch Prozess- und Produktinformationen in die PPR-Struktur eingebettet. Das daraus entstandene PPR-Anlagenmodell kann für die VIBN genutzt werden. Um die vereinzelt Module beim Import in die VIBN-Software zu identifizieren, werden SUCs und RCs aus den Vorlagenmodellen genutzt. Nachdem der automatisierte Import in die Simulationsumgebung ausgeführt wurde, folgt die automatisierte Simulation der Produktionsprozesse. Indem wieder SUCs und RCs aus den Vorlagenmodellen genutzt werden, wird eine Befehlsliste zur Ansteuerung des Simulationsmodells erzeugt. Diese kann mithilfe eines Kommunikationsprotokolls abgearbeitet werden, um die Simulation durchzuführen.

7 Fallbeispiel: Demonstrator einer modularen Anlage für die Virtuelle Inbetriebnahme

Ausgangspunkt des Fallbeispiels ist eine Anlagentopologie aus der modularen Anlagenplanung. Diese wird vereinfacht durch eine modulare Miniaturanlage repräsentiert, welche im Folgenden „Demonstrator“ genannt wird. Die Anlagentopologie lässt sich mithilfe von steckbaren Stationen benutzerdefiniert ändern. Sobald diese feststeht, wird sie in das AML-Format konvertiert und für die Simulation von Transportprozessen genutzt. Die Simulation kann sowohl auf dem physischen Demonstrator (Inbetriebnahme), als auch in der Simulationsumgebung von fe.screen-sim (VIBN) ablaufen. In Abbildung 29 ist dieser Prozess schematisch dargestellt.

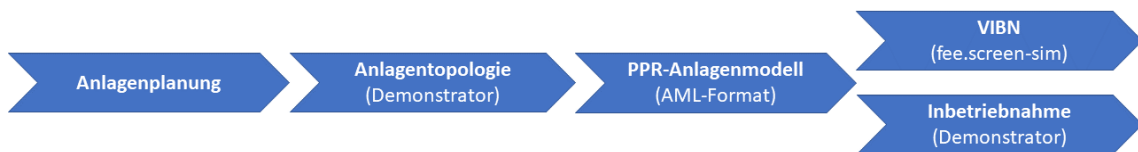


Abbildung 29: Einordnung des Fallbeispiels in den Entstehungsprozess eines modularen Anlagensystems

Durch ein simultanes Durchführen von Inbetriebnahme und VIBN lässt sich die Nähe des virtuellen Simulationsmodells gegenüber der physischen Anlage veranschaulichen.

7.1 Über den Demonstrator

7.1.1 Aufbau des physischen Demonstrators

Gegeben ist ein vereinfachtes Miniaturmodell eines modularen Anlagensystems (s. Abbildung 30), welches in einem studentischen Projekt entwickelt wurde. Die Modularität wird im Miniaturmodell durch steckbare, farbige Stationen symbolisiert. Neben der Farbe lassen sich die Stationen zusätzlich noch vom Typ „productSource“ und „productSink“ unterscheiden. Die verfügbaren Stationen lassen sich beliebig auf zehn verfügbaren Steckplätzen verteilen. Durch eingebaute RFID-Tags können die Stationen durch eine Transporteinheit auf Typ und Farbe identifiziert werden. Aufgrund der Aufteilung der Stationstypen in „productSource“ und „productSink“ lassen sich Transportprozesse zwischen zwei Stationen gleicher Farbe simulieren. Um einen Materialfluss für den Nutzer des Demonstrators vor Augen zu führen, leuchtet beim Anfahren einer „productSource“ Station die LED am Schwenkarm einer Transporteinheit auf. Das Leuchten soll dabei die Belegung der Transporteinheit durch ein Produkt der jeweiligen Farbe darstellen. Stationen vom Typ „productSink“ leuchten bei erfolgreicher Belieferung einer „productSink“ Station auf, um dem Nutzer zu signalisieren, dass die jeweilige Station nun belegt ist. Die Aktoren der Transporteinheit und die LEDs werden durch zwei Mikrocontroller angesteuert. Die Mikrocontroller werden wiederum über eine serielle Schnittstelle bzw. USB durch ein C# Skript angesteuert. Dieses kann Befehlssets zum Scannen der

Anlagentopologie und zum Ausführen der daraus resultierenden Transportprozesse zum Demonstrator senden.



Abbildung 30: physische Miniaturanlage

7.1.2 Aufbau des digitalen Anlagenmodells

Um den Demonstrator für die VIBN zu nutzen wird ein digitales Anlagenmodell benötigt, welches Anlagentopologie und Prozessabläufe abbildet, speichert und für eine automatisierte Verarbeitung bereitstellt. Zu diesem Zweck wird die Modellierungssprache AML genutzt. Mithilfe des digitalen Anlagenmodells soll in der VIBN die Anlagentopologie automatisiert aufgebaut und simuliert werden. In Abbildung 31 ist die geometrische Repräsentation des Demonstrators in der Simulationsumgebung zu sehen. Damit die Komplexität des automatisierten Modellaufbaus in der VIBN-Umgebung überschaubar bleibt, werden vorgefertigte fe-Bibliotheksvorlagen genutzt.

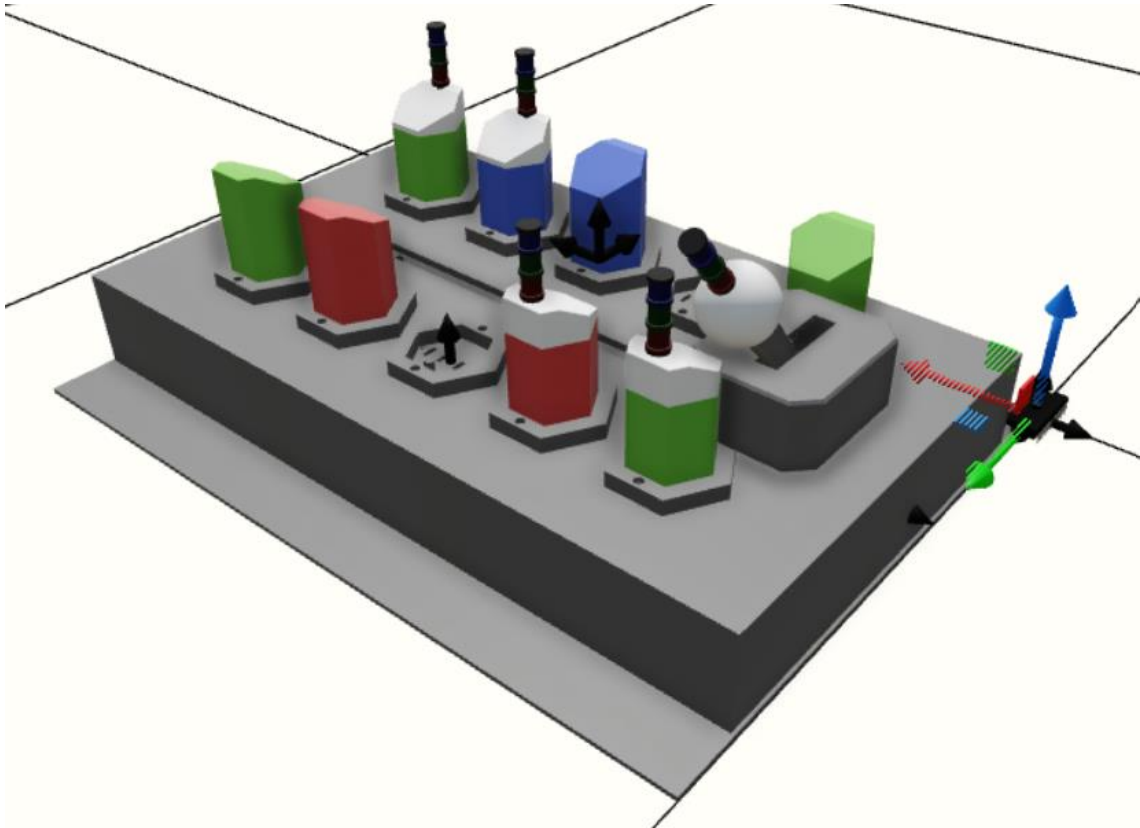


Abbildung 31: Demonstrator als geometrische Repräsentation in der VIBN-Simulationsumgebung mit „productSources“ (farbiger Deckel) und „productSinks“ (weißer Deckel mit Lampe)

7.1.3 Simulationsablauf

Vor dem Simulationsablauf ist ein Scannen aller Stationen nötig, um ihren Typ und Farbe zu identifizieren. So wird sichergestellt, dass nach jeder Neuordnung der Anlagentopologie das digitale Anlagenmodell entsprechend den vollzogenen Änderungen aktualisiert wird. Die Aktualisierung des Anlagenmodells besteht im ersten Schritt aus der Erstellung der neuen Anlagentopologie. Diese wird im darauffolgenden Schritt mit Produkt- und Prozessinformationen angereichert. Als Ergebnis wird ein vollständiges Anlagenmodell nach PPR-Konzept erzeugt.

Anschließend erfolgt der Simulationsablauf, welcher die Prozesse im PPR-Anlagenmodell nutzt. Der Ablauf beginnt mit einer Initialisierung. Indem das Simulationsmodell eine Referenzstellung anfährt, werden konsistente Simulationsergebnisse sichergestellt. Danach folgen, je nach definierter Ablaufreihenfolge, Transportaufträge, welche einen Materialfluss von „productSource“ Stationen nach „productSink“ Stationen der gleichen Farbe veranschaulichen sollen. Dazu fährt eine Transporteinheit die „productSource“ Station einer bestimmten Farbe an und signalisiert durch das Aufleuchten einer Lampe an der Transporteinheit, dass eine Produktaufnahme stattgefunden hat. Nach der Produktaufnahme verfährt die Transporteinheit zum „productSink“ der gleichen Farbe und signalisiert dort durch das Abschalten der Lampe an der Transporteinheit, dass eine Produktabgabe stattgefunden hat. Die „productSink“ Station signalisiert daraufhin durch das Aufleuchten ihrer eingebauten Lampe ihre Belegung.

7.1.4 Darstellung von Eigenschaften eines Automatisierungssystems

Um die Erfahrungen aus dem Fallbeispiel auf reale Automatisierungssysteme übertragbar zu gestalten, wird die bewusste Auswahl der Bestandteile und des Entwurfsprozess des Demonstrators erläutert. Das Ziel ist es mithilfe von Teilaspekten des Demonstrators Eigenschaften eines Automatisierungssystems darzustellen.

7.1.4.1 Aufbau des Demonstrators angelehnt an Bestandteilen eines Automatisierungssystems

Wie schon im Kapitel über den Aufbau erläutert (s. Kapitel 7.1.1), existieren zehn Steckplätze am Demonstrator, worin unterschiedliche Stationstypen untergebracht werden können. Diese sind nummeriert und stellen Modulplatzhalter eines modularen Automatisierungssystems dar. Die Anlagenmodule des Automatisierungssystems mit ihren Moduleigenschaften sollen mithilfe von „productSource“ und „productSink“ Stationen dargestellt werden, welche jeweils einen RFID Chip als Speichermedium für ihre Eigenschaften besitzen. Die Transporteinheit soll eine vereinfachte Form eines fahrerlosen Transportsystems darstellen. Diese beherbergt einen Schwenkarm mit beleuchtetem Kopf, welcher ein vereinfachtes Spiegelbild zu einem industriellen Greifarm veranschaulichen soll. Der Materialtransport wird dem Betrachter anstatt durch das Weitergeben von physischen Produkten mithilfe von einer Farbübergabe über LED-Lampen präsentiert.

7.1.4.2 Entwurfsprozess der Anlagentopologie des Demonstrators angelehnt am Planungsprozess eines Automatisierungssystems

Im Fallbeispiel wird das Automatisierungssystem durch ein modulares Anlagensystem repräsentiert. Aus diesem Grund dient der Planungsprozess eines modularen Anlagensystems als Referenz für die Generierung der Anlagentopologie. Dieser ist in Kapitel 4.3.1 näher erläutert. Zusammengefasst lässt sich der Planungsprozess einer modularen Anlage mit zwei Phasen darstellen. In der ersten Phase werden erforderliche Produktionsprozesse hinsichtlich einer Vielzahl von Kriterien ausgewählt und somit Modulbausteine festgelegt. In der zweiten Phase erfolgt die Detailauslegung der Modulbausteine. Das Ergebnis des Planungsprozesses ist schließlich der Entwurf einer modularen Anlagentopologie. Dieser Planungsprozess wird am Demonstrator vereinfacht durch das Stecken von Stationen verbildlicht. Die gescannte Stationsanordnung soll den Entwurf der Anlagentopologie darstellen. Im Unterschied zum Vorgehen bei einer realen modularen Anlage, müssen beim Demonstrator zunächst Prozesse und Prozessablauf in einer Inferenzmaschine von der Anlagentopologie abgeleitet werden. Die Prozessinformationen werden durch die Daten über Stationstyp und -farbe bestimmt. Als Ergebnis soll ein Ablauf entstehen, bei dem bei gegebener Farbreihenfolge ein Produkt von der „productSource“ Station einer jeweiligen Farbe zur „productSink“ Station der gleichen Farbe transportiert wird. Nach der Prozessableitung werden Prozesse und Produkte der Anlagentopologie hinzugefügt, sodass ein vollständiges Anlagenmodell nach PPR-Konzept entsteht. Bei der Planung einer realen Anlage würde der Prozessablauf bereits feststehen und könnte direkt mit der Anlagentopologie zu einem vollständigen PPR-

Anlagenmodell ergänzt werden. Mit dem vollständigen PPR-Anlagenmodell folgt schließlich die VIBN und die Inbetriebnahme an der physischen Anlage.

7.2 Nutzung des AML Formats

Um das AML-Format automatisiert zu Nutzen wird eine Software namens „AML Tool“ in der Programmiersprache C# erstellt. Als Programmierumgebung wird Visual Studio 2022 genutzt. Das AML Tool enthält eine Benutzeroberfläche zur Bedienung aller Funktionalitäten. Um die Funktionalitäten der Software für zukünftige AML-Projekte nutzbar zu machen, wird die Softwarearchitektur objektorientiert gestaltet. Es werden für Import-, Export- und Simulationsfunktionalitäten getrennte Klassen erstellt, welche Attribute enthalten, die zur Überwachung der Funktionsfähigkeit beitragen.

7.2.1 AML Tool

Um den Export von Daten ins AML-Format und Import von Daten aus dem AML-Format zu ermöglichen, wird die Aml.Engine genutzt, welche in C# bereitgestellt wird. Für den automatisierten Aufbau und die automatisierte Simulation des VIBN-Modells, wird die fe.screen-sim API genutzt. Diese wird ebenfalls in C# bereitgestellt.

Alle Konvertierungs- und Simulationsfunktionalitäten werden mit einer Benutzeroberfläche zur Bedienung in einer Software untergebracht. Diese wird im Folgenden „AML Tool“ genannt.

7.2.2 Softwarearchitektur

Die Softwarearchitektur richtet sich nach einem objektorientierten Aufbau mit Klassen für die Hauptfunktionen zur Verarbeitung des AML Formats (Exporter, Importer) und einer Klasse zur Simulation von Anlagenprozessen (Simulator). Um die Schnittstellen zum physischen Demonstrator und zur VIBN-Software zu überwachen, werden die Klassen „SerialCommunication“ und „FeeAPI“ genutzt. Generelle Funktionalitäten zur Fehlerdiagnose bei der Entwicklung, zur Darstellung von Nutzerfeedback und zum Management des Verbindungsstatus bei Schnittstellenkommunikationen werden mithilfe der Klassen „ObservableObject“ und „StatusInformation“ ermöglicht. Sie werden durch Vererbungen an andere Klassen weitergegeben, um die in ihnen enthaltenen Funktionen nicht mehrfach definieren zu müssen.

Jedes der genannten Klassen wird in einer Klassenbibliothek mit dem Namen „AMLConversion_ClassLibrary“ zusammengefasst, welche die Funktionalitäten zur Verarbeitung des AML Formats von Funktionalitäten der Benutzeroberfläche trennen sollen. Durch die Trennung ist es möglich bei zukünftigen Projekten die gekapselte Klassenbibliothek für neue Anwendungsfälle zu nutzen oder zu erweitern.

Die Softwarearchitektur wurde im UML-Format erstellt, um eine Übersicht über die Inhalte der Klassen und der Beziehungen der Klassen untereinander zu veranschaulichen. Um sie in lesbarer Form darzustellen, wird sie im Anhang dargestellt.

7.2.3 Benutzeroberfläche

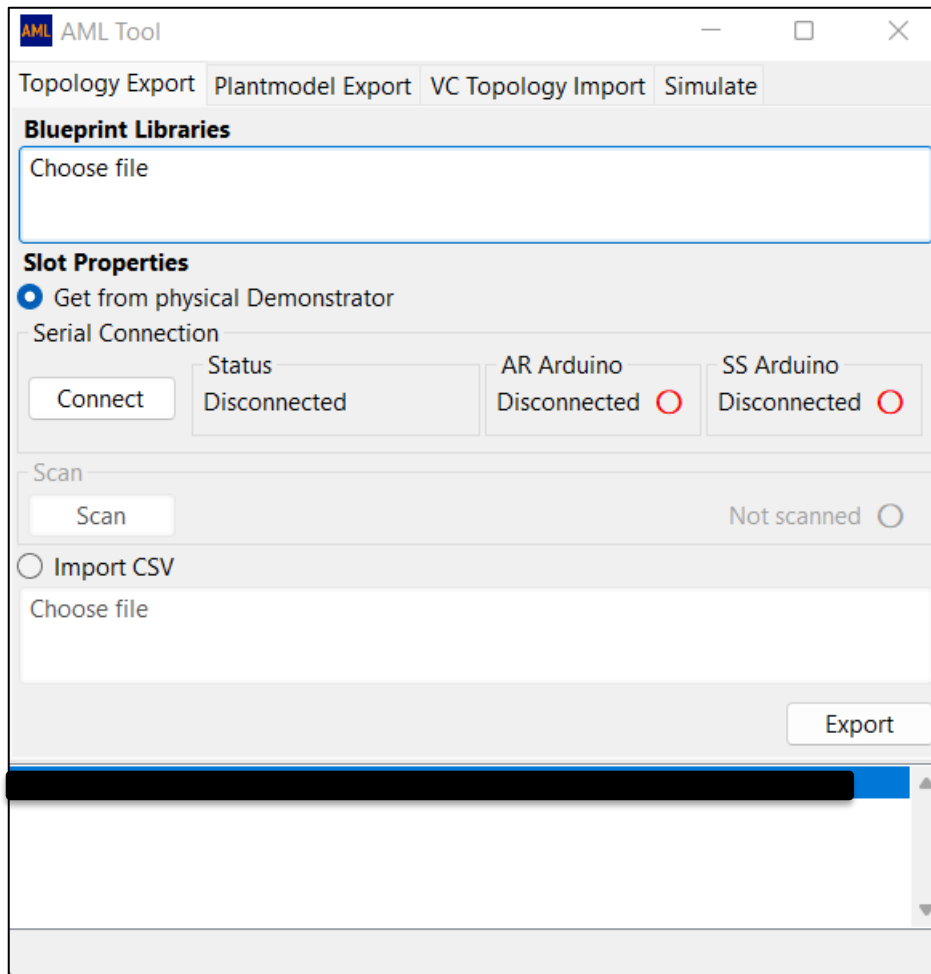


Abbildung 32: Benutzeroberfläche des AML Tools

Die Benutzeroberfläche des AML Tools ist in Abbildung 32 zu erkennen. Sie teilt sich in 4 Tabs mit den Hauptfunktionalitäten der Software auf: „Topology Export“, „Plantmodel Export“, „VC Topology Import“ und „Simulate“. Diese wurden nach der angedachten Nutzungsreihenfolge von links nach rechts angeordnet, um eine intuitive Bedienung zu bieten. Unabhängig von den Tabs befindet sich unter ihnen eine Textbox, welche die vergangenen Aktionen der Software dokumentiert. Unter der Textbox befindet sich eine Leiste, in welcher auf der linken Seite die Bezeichnung der aktuell ausgeführten Aktion und auf der rechten Seite den Fortschritt der aktuell ausgeführten Aktion angezeigt werden.

Zur Vermeidung von fehlerhaften Nutzereingaben genutzt, welche vordefinierte Relativpfade in Ordnerauswahldialoge zum Erstellen von Speicher- und Importpfaden nutzen. Dies soll den Ordnerauswahlprozess beschleunigen.

Um bei einem großen Funktionsumfang einen überladenen Eindruck zu vermeiden, werden nicht benötigte Funktionen deaktiviert, d.h. ausgegraut dargestellt.

Für eine reaktionsfähige Oberfläche werden asynchrone Methoden genutzt, welche eine parallele Bedienung dieser bei gleichzeitiger Ausführung umfangreicher Aufgaben, wie z.B. Importprozesse von VIBN-Modellen, ermöglichen.

Ungewünschte simultane Operationen, wie das Anstoßen einer weiteren Simulation, während eine vorherige Simulation noch in Ausführung ist, werden durch das Deaktivieren des „Simulieren“-Knopfes während einer laufenden Simulation verhindert.

7.3 Phase 1: Erzeugen des Domänenmodells

In Phase 1 wird das Zielformat des Anlagenmodells in AML definiert. Dieses setzt die Anforderungen für Exporter und Importer fest. Um ein Verständnis dafür zu gewinnen, welche Datenelemente existieren und wie diese mit AML-Strukturen dargestellt werden können, wird ein Mapping Prozess durchgeführt. Darauf folgt eine Strukturierung der Datenelemente, welche die Anordnung der Elemente untereinander nach dem PPR-Konzept bestimmt. Nach der Strukturierung ist es möglich ein Zielformat für das Anlagenmodell festzulegen. Auf Basis dieses Formats werden Vorlagenbibliotheken abgeleitet. Zu den Vorlagenbibliotheken zählen „SystemUnitClassLibrary“, „RoleClassLibrary“ und „InterfaceClassLibrary“.

7.3.1 Mapping von Datenstrukturen auf AML-Strukturen

Um die Datenstrukturen innerhalb von AML mit Semantik anzureichern ist ein Mapping dieser auf AML Strukturen nötig. Dazu sind in Tabelle 2 die essenziellen Datenelemente und deren Mapping dargestellt.





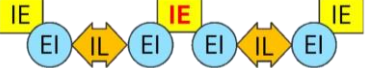
Daten-element	AutomationML Struktur	Bsp. am Demonstrator
Hierarchieebene, Objekt	InternalElement (IE) 	Ressource, Prozess, Produkt, „besteht aus“ Beziehungen
Eigenschaft	Attribute (Attr) 	Stationstyp, Farbe, ID
Klassifizierung	Referenz zu RoleClass (RC) oder SystemUnit-Class (SUC) 	Konkrete Prozesstypen, Produkttypen, Maschinenoperationen (als SUC), generische ... (als RC)
Einfache Beziehung	ExternalInterfaces (EI), welche mit einem Internal-Link (IL) verbunden sind 	PPR Beziehungen
Beziehung mit Daten	ExternalInterface mit InternalElement für Bezugsdaten 	PPR Beziehungen

Tabelle 2: Mapping der Datenstrukturen zu AML Strukturen nach [11]

Hierarchieebenen und Objekte werden in AML mit InternalElements (IE) modelliert. Diese können beim Demonstrator Teile der Anlagentopologie, Prozessschritte oder Produkte darstellen.

Eigenschaften werden in AML mit Attributes (Attr) modelliert. Mit ihnen können Eigenschaften des Demonstrators, wie z.B. Stationsnummer, -farbe oder Lagermenge abgebildet werden.

Klassifizierungen werden in AML mit Referenzen zu RoleClasses (RC) oder SystemUnitClasses (SUC) modelliert. Um beim Export und Import von AML-Modellen die digitalen Rohdaten mit realen Objekte und Prozessen zu assoziieren, dienen Klassifizierungen der Daten als Hilfsmittel für die Identifikation.

Beziehungen werden in AML mit ExternalInterfaces (EI) und dazugehörigen InternalLinks (IL) modelliert. Jegliche für das PPR Konzept (s. Kapitel 3.10) benötigten Beziehungen werden in AML mithilfe von EIs der InterfaceClass (IC) „PPRConnector“ verknüpft. Dazu gehören kausale Zusammenhänge wie z.B. die Verknüpfung eines Anlagenobjekts mit dem dazugehörigen auszuführenden Prozess in der Prozesskette.

7.3.2 Strukturierung der Datenelemente

Über den AML-Editor wird eine InstanceHierarchy im gewünschten Zielformat erzeugt. Das Zielformat soll die Anforderungen zur Weiterverarbeitung in der VIBN Simulationssoftware abdecken und somit nach dem PPR-Konzept (s. Kapitel 3.10) strukturiert. Jedem Datenelement in AML kommt bei der Modellierung eine Bedeutung zu. Diese ist für das jeweilige AML Datenelement in der folgenden Tabelle zusammengefasst (s. Tabelle 3).

PPR-Bereich	Datenelement	Bedeutung
Resources	SystemUnitClassReference (Class)	Vorlagenattribute und Architektur aus der SystemUnitClass
	RoleClassReference (Role)	Übertragen aus der SystemUnitClass, dient der Klassifizierung und Identifizierung des Ressourcentyps
	PPR-Connector (Resource_Process)	Ressourceninstanz, in welcher eine bestimmte Prozessinstanz genutzt wird / Prozessinstanz, In welcher die Ressourceninstanz genutzt wird
Processes	SystemUnitClassReference (Class)	Vorlagenattribute und Architektur aus der SystemUnitClass
	RoleClassReference (Role)	Übertragen aus der SystemUnitClass, dient der Klassifizierung und Identifizierung der Maschinenoperationen
	PPR-Connector (Resource_Process)	Ressourceninstanz, in welcher eine bestimmte Prozessinstanz genutzt wird / Prozessinstanz, In welcher die Ressourceninstanz genutzt wird
	PPR-Connector (Product_Process)	Prozessinstanz, welche genutzt wird um Produktinstanz herzustellen / Produktinstanz welche aus einer bestimmten Prozessinstanz hergestellt wird
Products	SystemUnitClassReference (Class)	Vorlagenattribute und Architektur aus der SystemUnitClass
	RoleClassReference (Role)	Übertragen aus der SystemUnitClass, dient der Klassifizierung des Produkttyps
	PPR-Connector (Product_Process)	Prozessinstanz, welche genutzt wird um Produktinstanz herzustellen / Produktinstanz welche aus einer bestimmten Prozessinstanz hergestellt wird

Tabelle 3: Bedeutung der Datenelemente

Der Aufbau der Anlagentopologie, welcher sich im „Resources“-Teil des Anlagenmodell wiederfindet, wird nach Vorbild der objektorientierten Programmierung hierarchisch angeordnet. In Abbildung 33 ist ein Klassendiagramm im UML-Format zu sehen, in dem die Objektbeziehungen untereinander veranschaulicht werden.

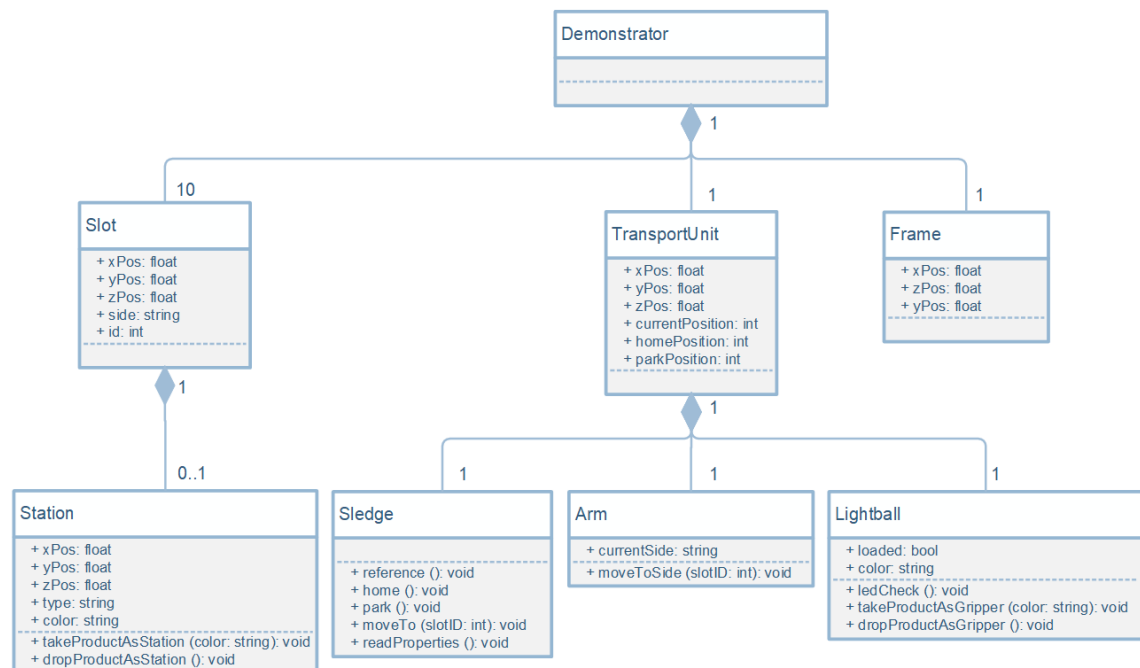


Abbildung 33: Klassendiagramm Anlagentopologie

Die aus dem PPR-Konzept entstandene InstanceHierarchy ist in Abbildung 34 zu finden. Die Bezeichnung „Role:“ stellt dabei die jeweilige RoleClassReference und die Bezeichnung „Class:“ die jeweilige SystemUnitClassReference dar.

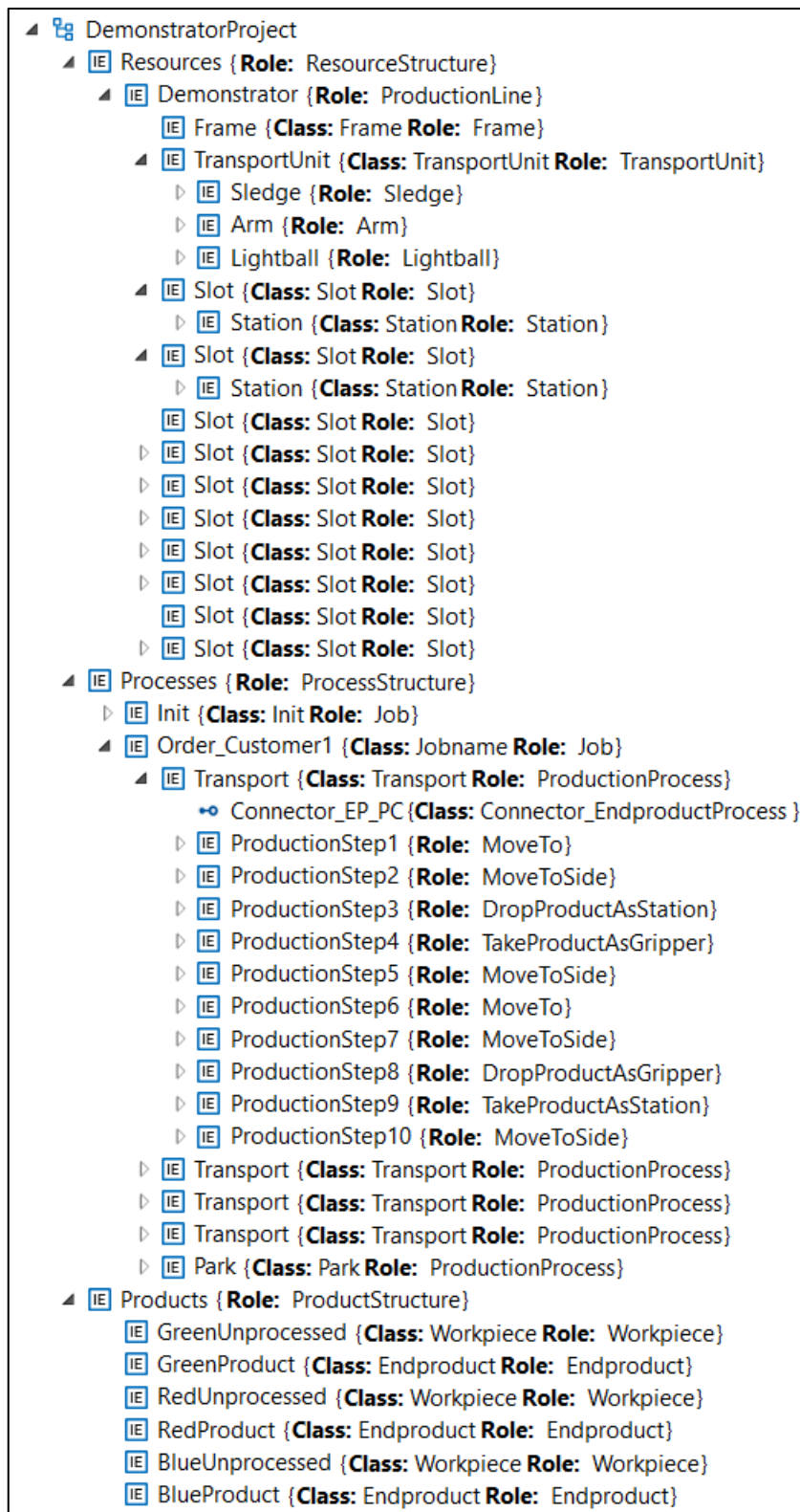


Abbildung 34: InstanceHierarchy Demonstrator

7.3.3 AML Vorlagenbibliotheken

Mithilfe der SystemUnitClassLibrary (SUCLib) (s. Abbildung 35) lassen sich konkrete Klassifizierungen in Form von Anlagenmodulen erstellen. Die SystemUnitClasses (SUCs) dienen als Modulvorlagen für die Gesamtanlage. Durch Instanzieren der SystemUnitClasses lassen sich Modulkomponenten in der InstanceHierarchy erzeugen, welche spezifische Attributwerte besitzen. Mithilfe der Bezeichnung „Role:“ werden die RoleClassReferences dargestellt, welche beim Instanzieren der SystemUnitClass mit übertragen werden. Beispielsweise wird aus der „Resourcemodel“ SystemUnitClassLibrary die SystemUnitClass „Slot“ zehn Mal instanziiert mit jeweils eigenem „id“ Attribut, welches die Arealnummerierung darstellen soll.

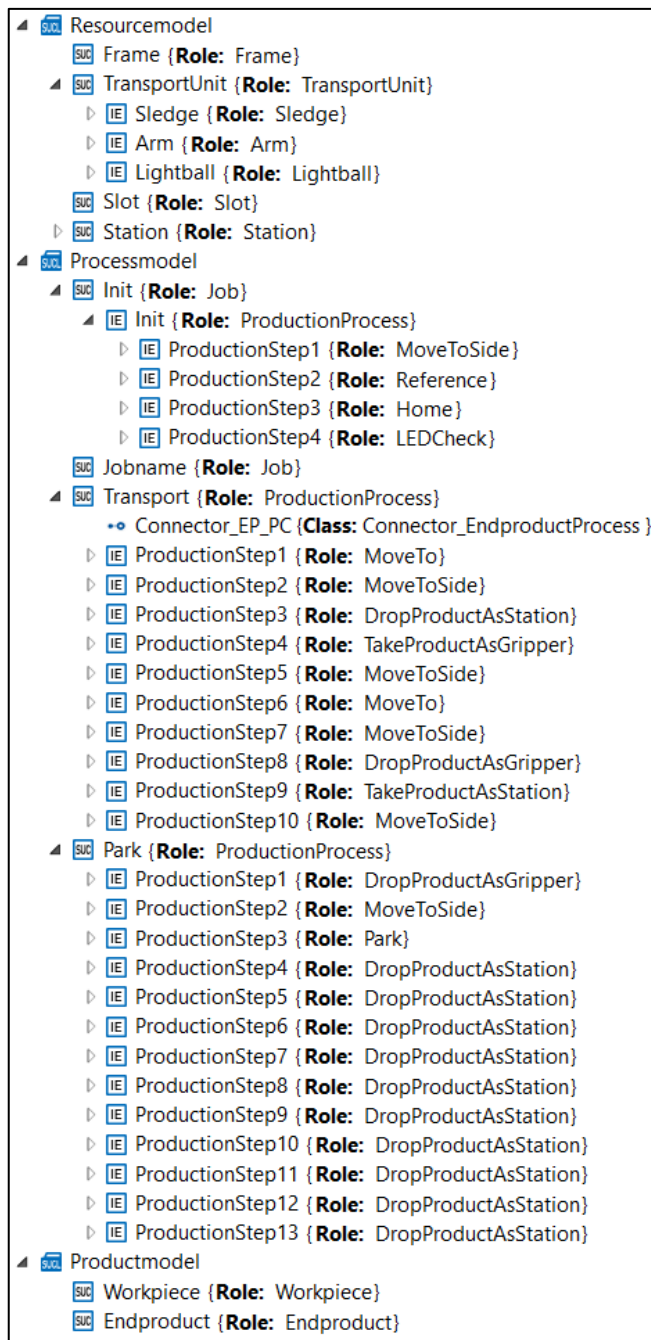


Abbildung 35: SystemUnitClassLibrary Demonstrator

RoleClassLibraries (RCLib) (s. Abbildung 36) dienen der generischen Klassifizierung und dienen in diesem Projekt der automatisierten Identifikation von AML Objekten. Die Bezeichnung „Class:“ besitzt im Gegensatz zur gleichnamigen Bezeichnung in der InstanceHierarchy die Bedeutung, dass von einer anderen Klasse des gleichen Typs (hier: RoleClass) geerbt wird. Durch die Zuweisung einer RoleClassReference wird ein AML Objekt mit einer Funktion versehen. Beispielsweise befinden sich unter der RoleClass (RC) „ProductionStep“ Maschinenoperationen, welche durch eine RoleClassReference automatisiert ausgelesen und in einer VIBN Simulation ausgeführt werden können.

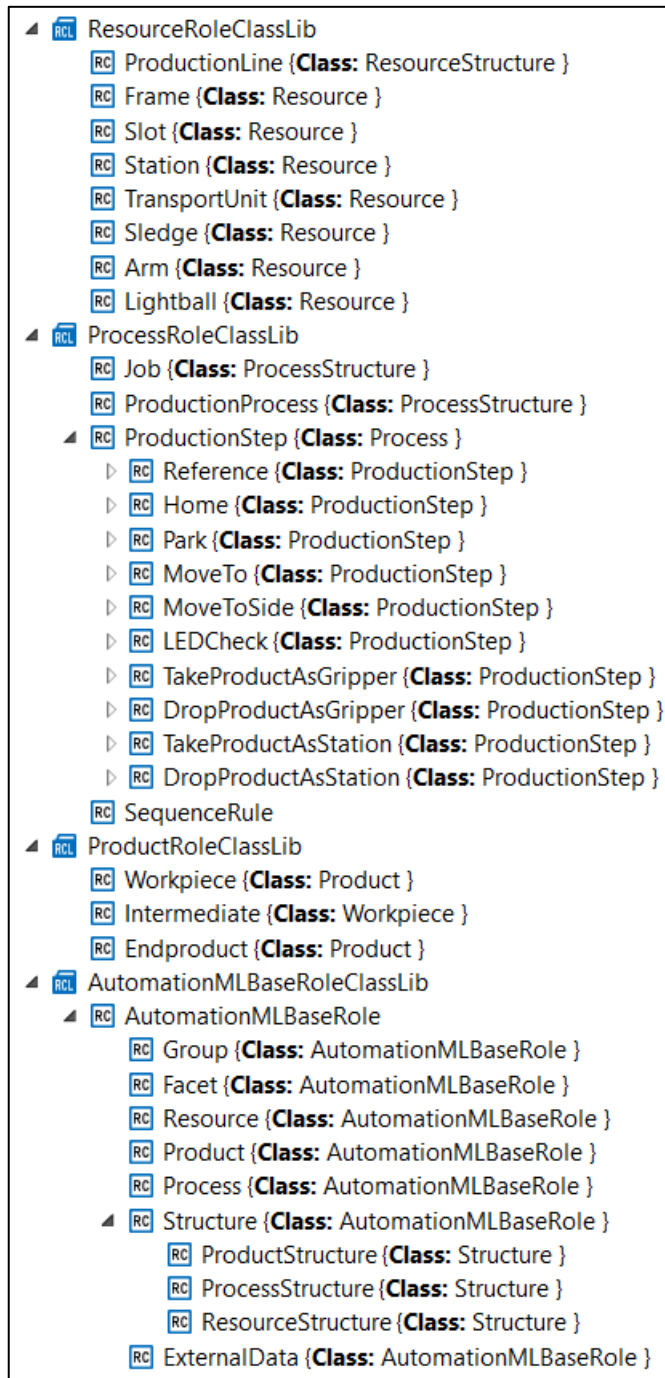


Abbildung 36: RoleClassLibrary Demonstrator

In den InterfaceClassLibraries (ICLib) (s. Abbildung 37) befinden sich InterfaceClasses (ICs), welche Verbindungspunkte für verschiedenste Schnittstellen darstellen sollen. Wie bei den RCLibs auch, bedeutet hier die Bezeichnung „Class:“ eine Vererbung von einem anderem Klassenobjekt des gleichen Typs. Unter anderem sind dort PPR Konnektoren zu finden, welche dazu genutzt werden, PPR-Beziehungen nach dem PPR-Konzept zu erzeugen.

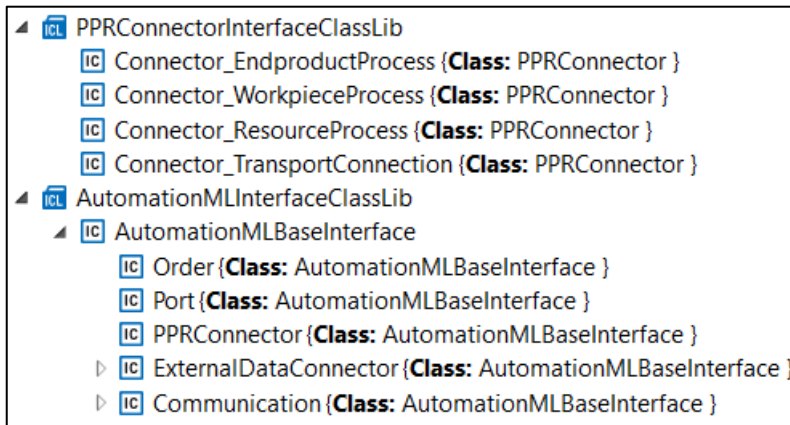


Abbildung 37: InterfaceClassLibrary Demonstrator

7.4 Phase 2: Topologie Export

Phase 2 stellt die Erstellung der ersten Softwarefunktionalität dar. Diese beinhaltet die Ausarbeitung eines Exporters, welcher für die Konvertierung von Topologieinformationen in das AML Format zuständig ist. Dabei werden die Topologieinformationen vom physischen Demonstrator übergeben. Für den Exportvorgang das externe Generierungskonzept genutzt, welches in Kapitel 3.9.1 näher beschrieben wird. In Abbildung 38 sind die Bestandteile des Konzepts schematisch dargestellt.

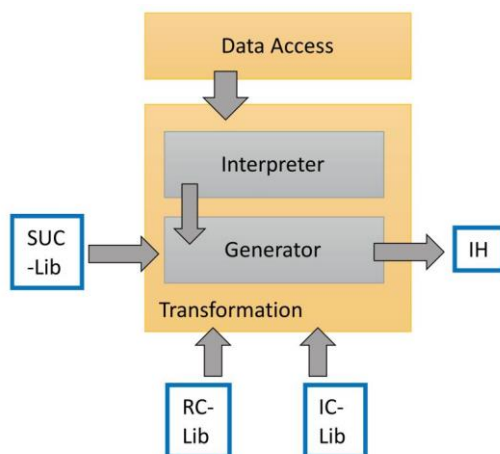


Abbildung 38: Externes Generierungskonzept bei der Erzeugung der IH (Anlagentopologie) [3, S. 47]

Aus dem „Data Access“ Block fließen Rohdaten der Anlagentopologie in Form einer Liste zum „Transformation“ Block. Dieser interpretiert die Rohdaten im „Interpreter“ und verknüpft diese mit Anlagenmodulvorlagen (SUCs). Die verknüpften SUCs werden vom

Generator instanziiert und zu einer Anlagentopologie in Form einer IH zusammensetzt. Die SUCs beinhalten dabei bereits Referenzen zu RCs und ICs, welche zur Anreicherung von Semantik dienen und für die spätere Identifizierung durch einen Importer benötigt werden.

7.5 Phase 3: Inferenzmaschine

Die Inferenzmaschine dient dem Anreichern der Topologieinformationen mit Prozess- und Produktinformationen. Diese bilden zusammen ein vollständiges digitales Anlagenmodell nach dem PPR-Konzept. Der Export des digitalen Anlagenmodells erfolgt wie in Phase 2 nach dem externen Generierungskonzept. In Abbildung 39 sind die Bestandteile der Inferenzmaschine dargestellt.

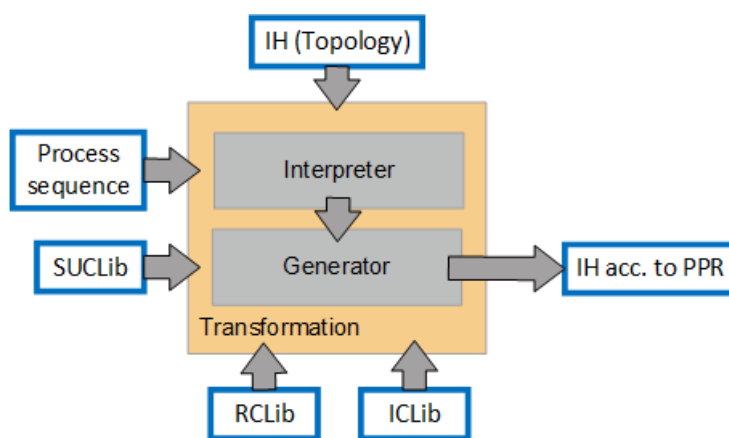


Abbildung 39: Externes Generierungskonzept bei der Erzeugung der IH (PPR-Konzept) nach [3, S. 47]

Die Anlagentopologie wird in Form einer AML IH zur Bereitstellung von Informationen wie z.B. Stationentyp und Stationenfarbe genutzt. In Kombination mit einer benutzerdefinierten Ablaufreihenfolge werden vom „Interpreter“ Prozesse und Produkte konfiguriert und zusammengestellt, welche vom „Generator“ schließlich in die IH eingetragen werden. SUCs dienen als Vorlagenmodelle, welche RCs und ICs zur Identifizierbarkeit durch einen Importer beinhalten. Zusätzlich hat der „Generator“ die Aufgabe Beziehungen zwischen Produkten, Prozessen und Ressourcen nach dem PPR-Konzept (s. Kapitel 3.10) herzustellen. Dazu werden PPR-Konnektoren aus der ICLib genutzt.

7.6 Phase 4: Importer und fe-Vorlagenmodelle

Phase 4 hat das Ziel, das Anlagenmodell in der VIBN-Simulationsumgebung automatisiert erzeugen zu lassen. Um die Komplexität des Erzeugungsvorgangs verständlich und überschaubar zu gestalten, werden dazu fe-Vorlagenmodelle genutzt, welche vorher erstellt werden. Danach folgt ein Importvorgang, welcher Informationen der Anlagentopologie aus AML-Daten nutzt, um Vorlagen aus der fe-Vorlagenbibliothek in die Simulationsumgebung zu importieren. Nach dem Import der Modelle erfolgt eine Verknüpfung

der in ihnen enthaltenen Verhaltensmodelle, wodurch ein vollständiges Simulationsmodell entsteht.

7.6.1 fe-Vorlagenmodelle

Die Bestandteile der fe-Vorlagenmodelle lassen sich in drei Kategorien unterteilen: Geometrie, Kinematik und Verhaltensmodelle.

Die Geometrieinformationen werden durch den Import aus dem STEP-Format erlangt. Bei den importierten Geometrien ist auf ein einheitliches Konzept zur Bestimmung des Koordinatenursprungs zu achten, da diese ist für eine definierte Positionierung bei der Nutzung der Vorlage von zentraler Bedeutung ist.

Kinematiken werden in fe.screen-sim durch „Motion Joint“ Objekte dargestellt (s. Abbildung 40). Durch hierarchische Eltern-Kind-Beziehungen wird der kinematische Freiheitsgrad des „Motion Joints“ allen darunterliegenden Kindern zugewiesen. In Abbildung 41 sind die „Kinematic Joints“ einer Transporteinheit zu sehen. Die „Linearachse“ wird dabei auf die Schwenkachse und den Schlitten angewandt. Die „Schwenkachse“ wird auf den Schwenkarm angewandt.

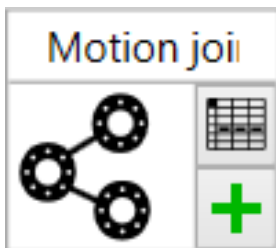


Abbildung 40: "Motion Joint" Objekt

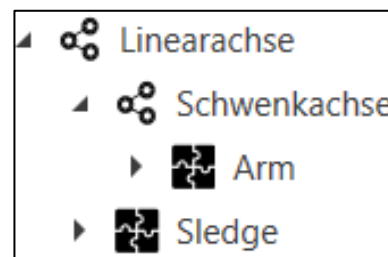


Abbildung 41: "Motion Joint" Objekt im Projektbaum

Verhaltensmodelle werden mithilfe von „Logic Objects“ dargestellt, welche mit steuerbaren Parametern von Objekten wie „Kinematic Joints“ oder „Segmented Lamp“ verknüpft werden. Mit einer Änderung der Signale an den Eingängen von „Logic Objects“ werden die Ausgänge der Verhaltensmodelle neu kalkuliert und somit Parameter in der Simulation angesteuert und z.B. eine segmentierte Lampe auf eine bestimmte Farbe geschaltet. In Abbildung 42 ist ein Logikdiagramm mit „Logic Object“ zu erkennen. Die Signalverarbeitung innerhalb der „Logic Objects“ wird mithilfe von C# Programmcode gesteuert.



Abbildung 42: Logikdiagramm mit "Logic Object"

7.6.2 Importer

Der Importvorgang von fe-Vorlagenmodellen ist schematisch in Abbildung 44 dargestellt.

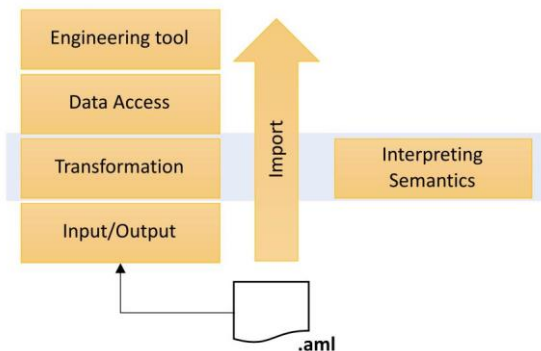


Abbildung 43: Import in Engineering Tools (FEE) [3, S. 45]

Der Importer nutzt bei diesem Vorgang Informationen aus einer AML-Datei, welche als „Input“ dient. Die Informationsverarbeitung, welche die Interpretation von Semantik und die Generierung von Daten im „Engineering tool“ beinhalten, wird im „Transformation“ Schritt durchgeführt. In Abbildung 44 wird der „Transformation“ Schritt detaillierter aufgeführt.

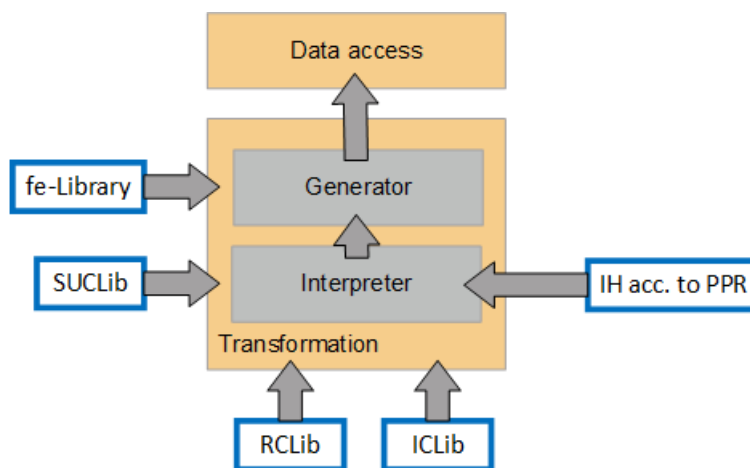


Abbildung 44: Transformation Layer beim Import

Der „Interpreter“ nutzt im „Transformation“ Schritt die bereitgestellte IH nach PPR-Konzept und gleicht die darin enthaltenen Elemente mit RCs ab, um so die Anlagentopologie zu identifizieren. Der „Generator“ nutzt dann schließlich fe-Vorlagenmodelle aus der „fe-Library“ um die passenden Anlagenmodule über die Programmierschnittstelle in den Simulationsbereich zu importieren. Nach dem Importvorgang werden noch „Symbolic Variables“ erzeugt, welche zur Verknüpfung der Verhaltensmodelle und zur Ansteuerung der Modelle über die API dienen.

7.7 Phase 5: Simulator

Phase 5 beinhaltet die Simulation der Anlagenprozesse, sowohl am physischen Demonstrator, als auch am VIBN-Modell. Zunächst wird der Aufbau der Simulators

vorgestellt, welcher das generelle Interpretieren und Ausführen von Prozessbefehlen übernimmt. Der Simulator setzt den Aufbau einer soliden Kommunikationsstruktur voraus, welche die simultane Simulation von physischer und digitaler Anlage ermöglicht. Um dies sicherzustellen werden spezielle Konzepte für die seriellen Kommunikation, wie auch für die Kommunikation über die fee.screen-sim API benötigt, welche in den folgenden Kapiteln erläutert werden.

7.7.1 Kommunikationsstruktur

Die Kommunikationsstruktur, welche innerhalb des Simulators genutzt wird, ist in Abbildung 45 zu sehen.

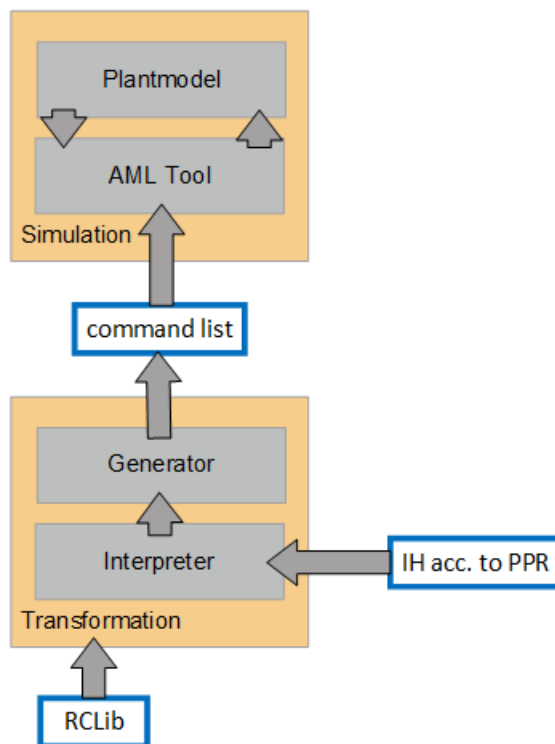


Abbildung 45: Simulator

Zunächst wird eine Befehlsliste erzeugt, welche alle Befehle beinhalten soll, welche an das Anlagenmodell gesendet werden sollen. Um diese aufzubauen, wird im „Interpreter“ die Prozessstruktur aus der IH auf eine Übereinstimmung mit einer Prozess RC geprüft. Alle gefunden Prozesse werden mithilfe des „Generators“ mit einer festgelegten Kommunikationssyntax in eine Befehlsliste geschrieben. Diese Befehlsliste wird schließlich chronologisch im „Simulation“ Schritt abgearbeitet. Das AML Tool sendet den Befehl zum Anlagenmodell und wartet nach Sendung auf eine Antwort. Das Anlagenmodell verarbeitet den gesendeten Befehl und sendet dem AML Tool nach Ausführung des Befehls den selbigen Befehl mit Suffix „f“ zurück. Sobald das AML Tool die Antwort verarbeitet hat, sendet es den nächsten Befehl an das Anlagenmodell.

7.7.1.1 Besonderheiten bei der seriellen Kommunikation

Während der Nutzung der seriellen Kommunikation gilt es zwei Besonderheiten zu beachten. Zum Einen werden bereits unmittelbar nach dem Verbindungsaufbau Daten vom

USB Gerät versendet. Diese dürfen nicht im „Simulation“ Schritt verarbeitet werden. Die zweite Besonderheit ist, dass die Dauer des Verbindungsaufbaus von Gerät zu Gerät unterschiedlich ist. Da im physischen Demonstrator zwei unterschiedliche Mikrocontroller als Kommunikationsgeräte verbaut sind, brauchen sie unterschiedlich lang, um die Verbindung aufzubauen. Damit gewährleistet wird, dass beide Mikrocontroller die Verbindung aufgebaut haben und kommunikationsbereit sind, wird beim Verbindungsaufbau so lange wiederholt ein Befehl zur Identifizierung des USB-Geräts gesendet, bis dieses seine Bezeichnung zurückgesendet hat. Dies dient auch zusätzlich zur Zuweisung der Mikrocontroller, da jeder Controller für unterschiedliche Funktionalitäten des Demonstrators zuständig ist.

7.7.1.2 Besonderheiten bei der Kommunikation über die fe.screen-sim API

Um das Senden von Signalen an das VIBN-Simulationsmodell zu simulieren werden „Symbolic Variables“ genutzt. Für jeden Mikrocontroller existiert eine „Symbolic Variable“ mit dem Namen „command“ und eine mit dem Namen „commandState“. Über die Variable „command“ werden Befehle an das „Logic Object“ des Mikrocontrollers weitergegeben. Die Variable „commandState“ wird bei ausgeführtem Befehl auf den „command“-Wert mit Suffix „f“ gesetzt. Durch die wiederholte Überprüfung der Variable „commandState“ durch das AML Tool kann schließlich der Abschluss des Befehls identifiziert werden.

Dieses Konzept der Kommunikation beinhaltet jedoch die Einschränkung, dass nicht derselbe Befehl mit der gleichen Syntax hintereinander gesendet werden kann. Bei zweiter Ausführung des Befehls würde das AML Tool somit fälschlicherweise eine sofortige Antwort zum Abschluss des Befehls erhalten.

Außerdem ist bei der Verwendung der API eine mangelhafte Dokumentation hervorzuheben. In der Dokumentation existieren keine detaillierten Beschreibungen zur Verwendung der Parameter von Methoden und manche Verweise zu Parameterbeschreibungen existieren noch gar nicht. Zusätzlich ist noch anzumerken, dass sich Objektbezeichnungen in der visuellen Benutzeroberfläche bei der manuellen Modellierung von Objektbezeichnungen derselben Objekten in der API unterscheiden. Beispielsweise wird ein Begriff bestehend aus zwei Worten in der einen Umgebung mit Leerzeichen und in der anderen Umgebung ohne Leerzeichen geschrieben.

8 Potenzial von AutomationML bei der domänenspezifischen Modellierung von Automatisierungssystemen

AutomationML bietet ein weites Spektrum an Werkzeugen für die Modellierung von Automatisierungssystemen.

8.1 Modellierung für verschiedene Interessengruppen und Strukturierung der Daten durch das PPR-Konzept

Eines der Werkzeuge für die Modellierung von Automatisierungssystemen in AML ist die Betrachtung von verschiedenen Interessengruppen innerhalb der Wertschöpfungskette des Anlagenbaus. Durch die Definition von Metamodel, Type model und Instance model wird eine Standardisierung von Daten vom Komponentenhersteller bis zum Anlagenbauer ermöglicht. Für Komponentenhersteller bietet es sich an ihre physischen Komponenten parallel mit einem digitalen Abbild zu vertreiben. Dies gibt ihnen ein schlagkräftiges Verkaufsargument für innovative Anlagenbauer, da so mit wenig Aufwand die vor-modellierten digitalen Einzelkomponenten zu einem digitalen Zwilling der Gesamtanlage zusammensetzen können. Bleibt jedoch die gesamte Modellierungsarbeit beim Anlagenbauer, weil die Komponentenhersteller kein Potenzial in der digitalen Modellierung ihrer Produkte sehen, so ist nach den Erkenntnissen beim Demonstrator klar geworden, dass es gerade im initialen Entstehungsprozess viel Aufwand kostet, Komponenten durch Mapping und PPR-Strukturierung im AML Format zu modellieren. Es gilt abzuwägen, ob der hohe Aufwand zur Modellerstellung gegenüber den Vorteilen der Nutzung eines digitalen Anlagenmodells gerechtfertigt ist. Die VIBN, welche hier zur Anwendung kommt, ist nur eine von vielen Möglichkeiten, um das digitale AML-Anlagenmodell zu nutzen. Weitere Möglichkeiten wären z.B. eine Validierung der Komponenten durch automatisierte FEM oder CFD Simulationen, welche die Komponentenhersteller dazu nutzen könnten um für Funktionalitäten oder Effizienz ihrer Produkte zu werben. Der Anlagenbetreiber kann die Daten dazu nutzen, um alternative Maschinenbelegungen bei Prozess- und Ressourcenausfällen zu ermitteln. Dazu müssen zusätzlich Transportwege in Form von Konnektoren in das PPR-Konzept implementiert werden.

8.2 AutomationML und Virtuelle Inbetriebnahme im Zusammenhang mit dem Entstehungsprozess des modularen Anlagenbaus

Des Weiteren bietet AML die Möglichkeit Daten aus der Anlagenplanung für die Weiterverarbeitung in z.B. VIBN-Simulationswerkzeugen zu nutzen. Die VIBN ermöglicht eine Verringerung der klassischen Inbetriebnahmedauer, erhöht jedoch den Planungsaufwand, da zuvor ein digitales Abbild der Anlage in einer zusammenhängenden

Darstellung von Topologie, Prozessen, Verhalten und Produkten erstellt werden muss. Sie begünstigen somit ein Frontloading der Verantwortungen der klassischen Inbetriebnahme. Ein wirtschaftlicher Weg aufwändig modellierte und digitalisierte Komponenten mehrfach wiederzuverwenden, ist die Nutzung automatisierter Konvertierungsvorgänge. Mithilfe der Konvertierungsvorgänge ist es möglich von dem Entwurf der Anlagentopologie zu einem digitalen Anlagenmodell im AML Format zu gelangen. Dazu wird die AML API genutzt. Da viele Engineering Werkzeuge nach heutigem Stand keine AML Schnittstelle besitzen, sind für die Nutzung des Formats Kompetenzen in der C# Programmierung nötig, um mithilfe der AML API eine eigene Anwendung zur Konvertierung zu erzeugen. Sollte die Anwendung jedoch für ein erstes Projekt erfolgreich erstellt worden sein, so ist ihre Nutzung in weiteren Projekten mit deutlich geringerem Aufwand verbunden. Durch einen objektorientierten Aufbau der Modellierungssprache und C# Anwendung zur Konvertierung ist eine Anpassung auf neue Anlagenumgebungen einfach, denn die Eigenschaften der objektorientierten Programmierung sind gute Erweiterbarkeit und Skalierbarkeit.

8.3 Erfahrungen mit der Nutzung der Programmierschnittstellen

Bei der Arbeit mit der AML API ist positiv aufgefallen, dass sie eine gute Dokumentation besitzt. Es sind viele Beispiel Quellcodes frei zugänglich, welche als Referenzlösungen bei der Entwicklung dienen können. Die API auf Seiten des genutzten VIBN-Werkzeugs fe.screen-sim ist hingegen nur sehr oberflächlich dokumentiert. Bei manchen Befehlen bestehen Inkonsistenzen in der Bezeichnung der Methoden. Beispielsweise unterscheiden sich manche Objektbezeichnungen in der visuellen Benutzeroberfläche bei einer manuellen Modellierung von den Objektbezeichnungen innerhalb der API. Dies ist möglicherweise darauf zurückzuführen, dass Engineering Softwares noch wenig Erfahrungen mit dem Einsatz von Programmierschnittstellen haben. Es wird vermutet, dass erst mit dem Aufleben von Industrie 4.0 eine weitgehende Digitalisierung im Anlagenbau begonnen wurde und somit ein Bedarf nach Schnittstellen zu automatisierter Informationsverarbeitung entstanden ist.

9 Zusammenfassung und Fazit

Zu Beginn der Arbeit wurde die Frage nach dem Potenzial von AutomationML bei der domänenspezifischen Modellierung von Automatisierungssystemen gestellt.

Bevor die Modellierungssprache an einem Fallbeispiel angewandt werden konnte, musste zunächst ein Verständnis für Aufbau und Nutzung von AML durch eine Grundlagenrecherche aufgebaut werden. Als nächstes folgte eine Definition von Automatisierungssystemen und ihrer Repräsentation durch ein modulares Anlagensystem. Danach wurde die VIBN als eine Möglichkeit zur Optimierung des Entstehungsprozesses des modularen Anlagenbaus vorgestellt. Dabei wurde AML als Verbindungsglied zwischen der modularen Anlagenplanung und der VIBN eingeordnet.

Der Verlauf, beginnend mit modularer Anlagenplanung, weitergeführt mit Nutzung von AML als Verbindungsglied und endend mit der Simulation in der VIBN, wurde an einem Fallbeispiel, dem „Demonstrator“, angewandt. Nach einer Vorstellung der Eckdaten des Demonstrators wurde die Programmierung einer C# Anwendung zur Konvertierung von AML-Daten vorgestellt. Danach folgten 5 Phasen zur Erzeugung der Anwendung, welche sich nach dem Informationsfluss der Anlagendaten richteten. Die erste Phase stellte dabei das Festlegen eines Zielformats in AML dar. Zur Strukturierung der Daten wurde das PPR-Konzept genutzt. Die zweite Phase erläuterte die Exportarchitektur, mit welcher eine Anlagentopologie in das AML-Format überführt werden konnte. Phase drei enthält die Erstellung einer Inferenzmaschine zur Anreicherung der AML-Topologiedaten mit Prozess und Produktdaten, damit ein PPR-Anlagenmodell entsteht. Phase vier beschäftigt sich mit der Anwendung des AML-Formats für die VIBN. Sie beschrieb den Aufbau eines Importers, welche Anlagentopologie und Verhaltensmodelle der Module innerhalb dieser in die Simulationsumgebung importieren. Die fünfte Phase beinhaltet die Beschreibung eines Simulators, welcher für die automatisierte Prozesssimulation in der VIBN zuständig ist. Dazu wurde eine Kommunikationsstruktur zum Verwalten der Schnittstelle zur VIBN-Software fe.screen-sim und der seriellen Schnittstelle der physischen Anlage vorgestellt. Diese ist in der Lage, sowohl in einer vereinzelter Umgebung, als auch in beiden Umgebungen simultan, einen Datentransfer für eine Simulation zu ermöglichen.

Letztendlich wurde eine Diskussion über das Potenzial von AML zur Beantwortung der Forschungsfrage durchgeführt. Sie beinhaltet die Vorteile der Aufteilung der AML-Modellierung für verschiedene Interessengruppen und verdeutlicht den hohen Aufwand zur Modellierung der Anlagendaten nach dem PPR-Konzept. Durch eine Einordnung von AML in den Entstehungsprozess einer modularen Anlage wurden die Zusammenhänge zwischen der Modellierungssprache, der modularen Anlagenplanung und der VIBN klar gestellt. Es wurden Vorteile wie eine verringerte Inbetriebnahmezeit und erhöhte Planungsaufwände zur Erreichung dieser bei der Nutzung der VIBN festgestellt. Ebenfalls wurde die Architektur von Modellierungssprache und C# Anwendung angesprochen, welche aufgrund ihres objektorientierten Konzepts Vorteile wie eine gute Erweiterbarkeit

und Skalierbarkeit bieten. Die Erfahrungen mit der Nutzung der Programmierschnittstellen von der Aml.Engine sind aufgrund einer ausführlichen Dokumentation als positiv anzusehen, wohingegen die API der Software fee.screen-sim aufgrund einer mangelhaften Dokumentation und Inkonsistenzen Bezeichnungen einen unreifen Eindruck hinterlassen hat.

Der Einsatz von VIBN und AML im modularen Anlagenbau ermöglicht insgesamt eine frühere Gewinngenerierung, welche durch einen Mehraufwand bei der Planung ausgeglichen werden muss. Dies stellt eine Verschiebung der Verantwortungsbereiche auf die frühen Phasen der Anlagenentstehung dar, welche dem Frontloading-Prinzip des Lean Development folgt.

Die Vorteile durch Nutzung von AML lassen sich durch das Prinzip des Lean Development, dem „Einsparen“ von Verschwendungen, nicht passend definieren. Aus dem Grund, dass die Modellierung des digitalen Abbilds der Anlage, sowie ihre Anwendungen in der VIBN, parallel neben dem klassischen Entwicklungsprozess durchgeführt werden, lassen sich die Vorteile in Form einer verbesserten Zuverlässigkeit und Planbarkeit im Anlagenbau durch Inanspruchnahme eines „erforderlichen Mehraufwands“ besser beschreiben. Alles in allem wird die Optimierung des Entstehungsprozesses im modularen Anlagenbau nicht durch „Einsparpotenziale“, sondern durch Fehlervermeidung in der Planung passender definiert.

10 Ausblick

Mit dem jetzigen Stand des Fallbeispiels ist noch nicht das gesamte Potenzial der Modellierungssprache AML ausgeschöpft. Beim Import der Anlagendaten in die VIBN-Simulationsumgebung wurden bisher fe-Vorlagenmodelle benutzt. Die Bestandteile des Vorlagenmodells, wie CAD-Daten und Verhaltensmodelle können in Zukunft ebenfalls vollautomatisiert innerhalb des Importers generiert und verknüpft werden. CAD-Daten können in Form von Referenzen im AML Dokument vorhanden sein und mithilfe der API bei Bedarf in die VIBN-Software importiert werden.

Ebenfalls besteht die Möglichkeit das PPR-Konzept zur Strukturierung der AML-Daten zu erweitern. Mit ihr können Transportwege in Form von Transportkonnektoren dargestellt werden. Diese können dazu genutzt werden, um alternative Maschinenbelegungen bei Maschinen- oder Prozessausfällen zu ermitteln.

Unabhängig vom Fallbeispiel gilt es die werkseitige Kompatibilität von Engineering-Werkzeugen zum AML Format zu beobachten. Sollte sich das Format langfristig durchsetzen, um Anlagendaten zu speichern, so wird ein großer Teil an Mehraufwand zur Programmierung einer eigenen Anwendung zur Konvertierung in AML und von AML eingespart.

Literaturverzeichnis

- [1] *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
- [2] Rainer Drath, „AutomationML: A Practical Guide“ in *AutomationML*, R. Drath, Hg., De Gruyter, 2021, doi: 10.1515/9783110746235-202.
- [3] Rainer Drath, „AutomationML: The Industrial Cookbook“ in *AutomationML*, R. Drath, Hg., De Gruyter, 2021, doi: 10.1515/9783110745979-202.
- [4] pauly, „Whitepaper AutomationML Edition 2.1: Part 1 - Architecture and General Requirements“, 2018.
- [5] Deutscher Medien Verlag GmbH, *Automatisierungssysteme: Allgemeine Informationen*. [Online]. Verfügbar unter: <https://www.industrystock.de/de/unternehmen/Automatisierungstechnik/Automatisierungssysteme#:~:text=Automatisierungssysteme%20sind%20Systeme%2C%20welche%20einen,gewissen%20Geschwindigkeit%20gefertigt%20werden%20m%C3%BCssen>. (Zugriff am: 27. Mai 2022).
- [6] *Consider modular plant design*, 2017.
- [7] F.EE, *Vergleich: traditionelle Inbetriebnahme vs. Virtuelle Inbetriebnahme: Virtuelle Inbetriebnahme*. [Online]. Verfügbar unter: <https://www.fescreen-sim.de/fescreen-sim/was-ist-virtuelle-inbetriebnahme.html> (Zugriff am: 26. Mai 2022).
- [8] Leon Urbas, Stefan Bleuel, Tobias Jäger, Stephan Schmitz, Lars Evertz und Tobias Nekolla, „Automatisierung von Prozessmodulen: Von Package-Unit-Integration zu modularen Anlagen“, *atp*, S. 44–53, 2012.
- [9] A. Albo und P. Falkman, „A standardization approach to Virtual Commissioning strategies in complex production environments“, *Procedia Manufacturing*, Jg. 51, S. 1251–1258, 2020, doi: 10.1016/j.promfg.2020.10.175.
- [10] F.EE, *Softwaremodule und -struktur*. [Online]. Verfügbar unter: <https://www.fescreen-sim.de/module.html> (Zugriff am: 12. Mai 2022).
- [11] Björn Grimm, „Data modelling with AutomationML: understand - structure - classify“, 14. Okt. 2015.

Anhang 1: Systemarchitektur vom AML_Tool

Der Anhangsinhalt folgt auf der nächsten Seite.

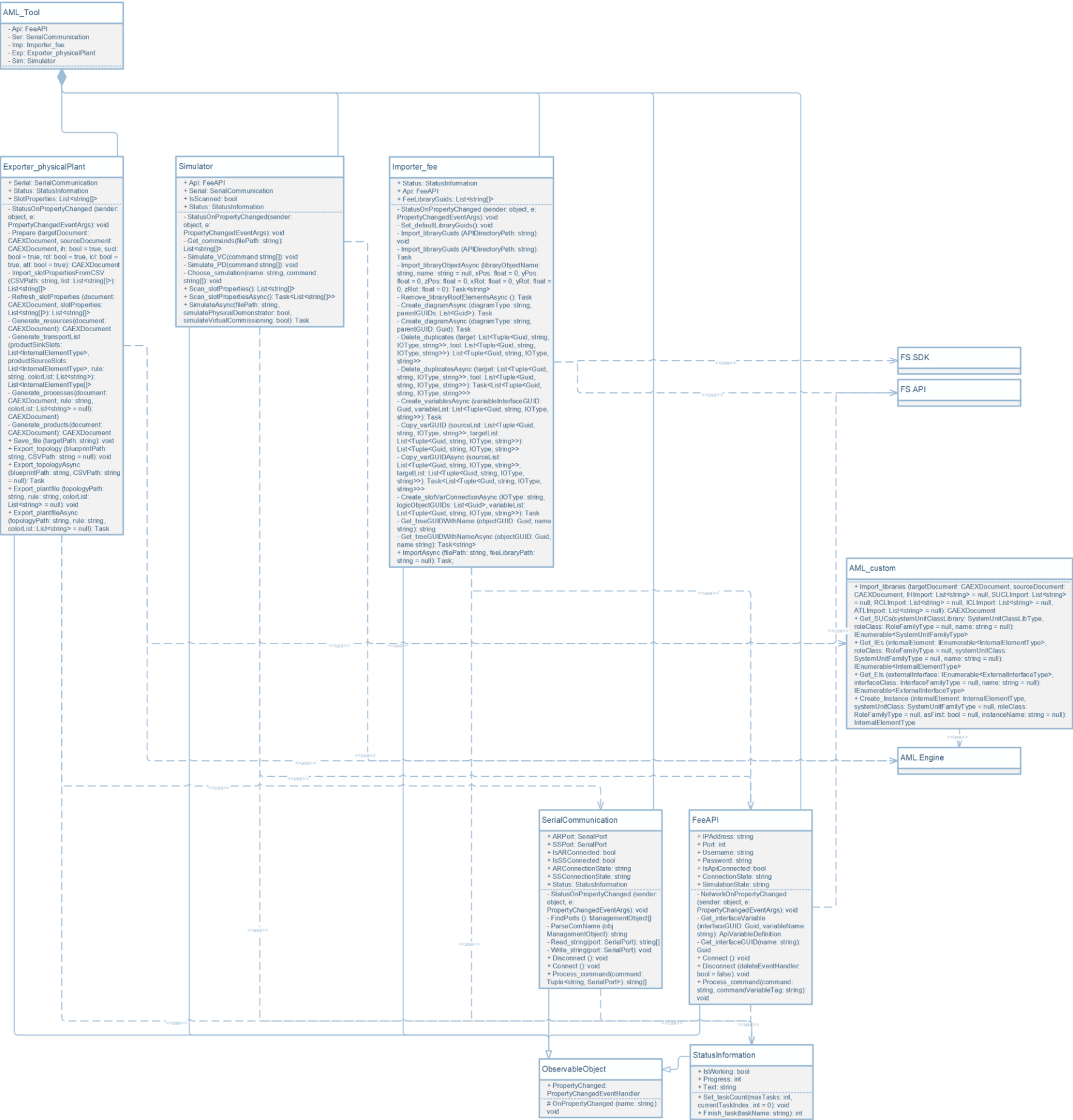


Abbildung 46: Systemarchitektur AML_Tool

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum

Rechtsverbindliche Unterschrift