

VEREIN
DEUTSCHER
INGENIEURE

VERBAND DER
ELEKTROTECHNIK
ELEKTRONIK
INFORMATIONSTECHNIK

INTERESSENGEMEINSCHAFT
AUTOMATISIERUNGSTECHNIK
DER PROZESSINDUSTRIE

Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie

Modellierung von Bedienbildern

VDI/VDE/
NAMUR 2658

Blatt 2

Entwurf

Automation engineering of modular systems in the process industry – Modeling of human-machine – interfaces

Einsprüche bis 2018-03-31

- vorzugsweise über das VDI-Richtlinien-Einspruchsportal
<http://www.vdi.de/einspruchsportal>
- in Papierform an
VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik
Fachbereich Industrielle Informationstechnik
Postfach 10 11 39
40002 Düsseldorf

Inhalt	Seite
Vorbemerkung	2
Einleitung	2
1 Anwendungsbereich.....	3
2 Normative Verweise.....	3
3 Abkürzungen.....	3
4 Bedienbilder	3
4.1 Grundkonzepte für die Bedienerschnittstellen modularer Anlagen ..	3
4.2 Bedienbilder als Aspekt im Module Type Package (MTP)	5
5 Modellierungsvorschriften zur Erstellung eines Bedienbilds.....	12
5.1 Allgemeine Modellierungsvorschriften	12
5.2 Modellierungsvorschriften für Verbindungen.....	12
Schrifttum	21

VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)
Fachbereich Industrielle Informationstechnik

VDI-Handbuch Informationstechnik, Band 1: Angewandte Informationstechnik
VDI/VDE-Handbuch Automatisierungstechnik

Vorbemerkung

Der Inhalt dieser Richtlinie ist entstanden unter Beachtung der Vorgaben und Empfehlungen der Richtlinie VDI 1000.

Alle Rechte, insbesondere die des Nachdrucks, der Fotokopie, der elektronischen Verwendung und der Übersetzung, jeweils auszugsweise oder vollständig, sind vorbehalten.

Die Nutzung dieser Richtlinie ist unter Wahrung des Urheberrechts und unter Beachtung der Lizenzbedingungen (www.vdi.de/richtlinien), die in den VDI-Merkblättern geregelt sind, möglich.

An der Erarbeitung dieser Richtlinie waren beteiligt:

Andreas Stutz, Karlsruhe

Christof Kotsch, Brühl

Henning Mersch, Verl

Jens Bernshausen, Leverkusen

Henry Bloch, Hamburg

Stephan Hensel, Dresden

Leon Ubras, Dresden (Vorsitz)

Mario Hoernicke, Ladenburg

Mathias Maurmaier, Karlsruhe

Thomas Holm, Minden

Thomas Ruschival, Esslingen

Jürgen Rudnick, Erkrath

Simon Kronemeier, Ludwigshafen

Tobias Nekolla, Essen

Klaus Erni, Langenfeld

Allen, die ehrenamtlich an der Erarbeitung dieser Richtlinie mitgewirkt haben, sei gedankt.

Eine Liste der aktuell verfügbaren Blätter dieser Richtlinienreihe ist im Internet abrufbar unter www.vdi.de/2658.

Einleitung

Der Fachausschuss „Zukünftige Architekturen in der Automation“ der VDI-Gesellschaft für Mess- und Automatisierungstechnik (GMA) hat sich zusammen mit der Namur und dem ZVEI mit der Erstellung dieser Richtlinie der Aufgabe angenommen, die Spezifikation von Modulschnittstellen, zur Verwendung in modularen Anlagen, zu definieren und syntaktisch, semantisch und pragmatisch zu beschreiben.

Modulare Anlagen werden in der Fertigungs- und Verfahrenstechnik vermehrt eingesetzt. Das Ziel hierbei ist die Planungszeit neuer Anlagen deutlich zu verkürzen und Umbauarbeiten an Anlagen zeitlich zu verkürzen. Hierdurch reduziert sich die

Stillstandzeit bzw. die Time-to-Market wird bei Neuanlagen deutlich verkürzt.

Da die Domänen „Fertigungstechnik“ und „Verfahrenstechnik“ hierbei sehr unterschiedliche Anforderungen an die Modularität stellen, wird in dieser Richtlinie vornehmlich die Verfahrenstechnik betrachtet.

Ausgehend von abgeschlossenen Projekten, wie F3 Factory [1], und bestehenden Empfehlungen und Anforderungen (veröffentlicht in der NAMUR NE 148) an verfahrenstechnische Module wird in dieser Richtlinie das Engineering der Automatisierungstechnik modularer Anlagen beschrieben. Hierbei wird sowohl das Modulengineering als auch das Anlagenengineering der Automatisierungstechnik betrachtet.

Zur Beschreibung der Modultypen wird das Module Type Package (MTP) verwendet, das die Schnittstellen und Funktionen der Automatisierungstechnik von Modulen definiert und beschreibt und letztlich die Integration von Modulen in eine Prozessführungsebene (PFE) ermöglicht.

Hierbei wird der Aspekt der Bedienbildbeschreibung in dieser Richtlinie mit folgenden Themen fokussiert:

- Beschreibungsmittel für Bedienbilder im Module Type Package
- Schnittstellen zwischen Manifest (VDI/VDE/NAMUR 2658 Blatt 1) und dem Bedienbildaspekt
- Modellierungsvorschriften für Bedienbilder im Module Type Package
- Verwendete Klassen und Strukturen zur Beschreibung von Bedienbildern

Weitere (geplante) Blätter der Richtlinienreihe greifen folgende Aspekte des automatisierungstechnischen Engineerings modularer Anlagen auf:

- Blatt 1: Allgemeines Konzept und Schnittstellen
- Blatt 3: Bibliothek für Datenobjekte
- Blatt 4: Modellierung von Moduldiensten
- Blatt 5: Laufzeit- und Kommunikationsaspekte
- Hinzu kommen Richtlinien zu den Themen: Diagnose, Alarmmanagement, funktionale Sicherheit sowie Validieren von MTP und Modulen.

Durch die zunehmende Vernetzung der Module werden weitere Themen hinzukommen, wie modulübergreifende funktionale Sicherheit, sichere Kommunikation zwischen Modulen und Nutzerrollenkonzepten.

AI HBe: VDI-Redaktion + Verweis auf ProcessNet
Whitepaper wie in Blatt 1

1 Anwendungsbereich

Diese Richtlinie definiert die Modellierungsvorschriften für die statischen Aspekte von Bedienbildern für prozesstechnische Module gemäß VDI/VDE/NAMUR 2658 Blatt 1. Die Objekte und Strukturen für die Dynamisierung von Bedienbildern werden in Blatt 3 spezifiziert.

Zielgruppen dieser Richtlinie sind die gleichen wie in VDI/VDE/NAMUR 2658 Blatt 1:

- Modulhersteller,
- Werkzeughersteller und
- Modulintegrator.

Anwendungsfälle und Definitionen entsprechen ebenfalls VDI/VDE/NAMUR 2658 Blatt 1.

AI HBe: mehr Text bei Schlussredaktion

2 Normative Verweise

Das folgende zitierte Dokument ist für die Anwendung dieser Richtlinie erforderlich:

VDI/VDE/NAMUR 2658 Blatt 1:2017-06 Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie; Allgemeines Konzept und Schnittstellen

3 Abkürzungen

In dieser Richtlinie werden die nachfolgend aufgeführten Abkürzungen verwendet:

MTP Module Type Package

PFE Prozessführungsebene

4 Bedienbilder

Die Bedienbildbeschreibung stellt einen Aspekt der Schnittstellen- und Funktionsmodellierung im MTP dar. Alle Aspekte werden wie in VDI/VDE/NAMUR 2658 Blatt 1 beschrieben innerhalb des Manifests (also der Organisationsdatei des MTP) verknüpft. Hierbei wird mittels eines Verzeichniseintrags im Manifest zur Beschreibung der Bedienbilder auf die **InstanceHierarchy** verwiesen. Als Meta-Modell wird für die Bedienbildbeschreibung ebenfalls AutomationML nach IEC 62714 verwendet. Die Modellierungskonzepte sind angelehnt an [2].

Wie in Bild 1 grau markiert, widmet sich diese Richtlinie der Modellierung von Bedienbildern innerhalb des MTP, es wird also dieser Aspekt fokussiert.

4.1 Grundkonzepte für die Bediener-schnittstellen modularer Anlagen

Die zentralen Herausforderungen der Bedien- und Beobachtbarkeit des über mehrere Module verteilten Prozesses sind sowohl die automatische Bedienbilderstellung als auch die Realisierung des nach NAMUR NE 148 formulierten einheitlichen „Look and Feel“ modularer Anlagen.

Da der Modulhersteller für die Planung, den Aufbau und die Programmierung des Moduls verantwortlich ist, fertigt er auch das oder die Bedienbild(er) des Moduls an. Eine Kenntnis der in industriellen Anlagenprojekten meist projektspezifisch verwendeten Bedienbildbibliothek der übergeordneten PFE hat er zu diesem Zeitpunkt jedoch nicht. Die Generierung des modulspezifischen Bedienbilds in der PFE der Gesamtanlage kann somit erst nach Integration des Moduls im PFE-Engineering erfolgen.

4.1.1 Bedienbilder im Modulengineering

Während des Modulengineering fertigt der Modulhersteller ein oder mehrere Bedienbilder des innerhalb des Moduls realisierten Verfahrensschritts an.

Dazu benötigte Informationen für das MTP sind:

- ungefähre Lageinformationen der im Bedienbild gezeigten Anlagenequipments
- verwendete Symbolik/Funktion (z.B. Ventil, Binärventil oder Regelventil)
- Verbindungsinformationen (Vorgänger- und Nachfolgerbeziehungen) des verwendeten Equipments
- Rohrleitungslage und deren Verbindungen zum Anlagenequipment
- Signale und Wirklinien sowie deren Verbindungen zum Anlagenequipment
- Tag-Namen der Symbole, um diese später den Kommunikationsobjekten zuzuweisen
- Anfangs- und Endpunkte innerhalb der Grafik (Sprungmarken oder Knöpfe)



Bild 1. Aspekt der Bedienbildbeschreibung im MTP

Diese Informationen beziehen sich immer auf die zu visualisierenden Teile des Moduls. Sollen Teile nicht visualisiert werden, so müssen diese auch nicht in der Bedienbildbeschreibung vorliegen.

Weitere Informationen, die im klassischen Bedienbildengineering benötigt werden, wie Liniendicke, Farb- oder Verhaltensinformationen der Symbole, werden nicht während des Modulengineerings festgelegt, sondern später für ein einheitliches „Look and Feel“ während des PFE-Engineerings bestimmt.

Bild 2 zeigt ein Beispiel eines Bedienbilds im Engineeringwerkzeug mit allen hierfür notwendigen Informationen über die Visualisierung eines Moduls. Weitere Informationen als die oben aufgezählten sind nicht notwendig.

4.1.2 Bedienbilder im PFE-Engineering

Während des PFE-Engineerings werden die Bedienbildbeschreibungen der verwendeten Module aus deren MTPs ausgelesen. Die Konformität mit dieser Richtlinie stellt sicher, dass diese automatisch in die PFE importierbar sind.

Dabei integriert die PFE die einzelnen Bedienbilder in ein HMI für die modulare Anlage. Es ist Aufgabe der PFE für die Bedienbilder ein einheitliches „Look and Feel“ sicher zu stellen. Dazu gehören beispielsweise Interaktionskonzepte, Darstellungsformen von Symbolen, Farbschemata und Linieneigenschaften wie Farbe und Linienstärke. Für das konkrete Layout werden je nach PFE-Engineeringwerkzeug die Lageinformation der Objekte innerhalb des Bedienbilds verwendet und/oder die Verbindungsinformation zwischen den zu visualisierenden Objekten genutzt.

Das Zusammenführen von Modulen unterschiedlicher Hersteller in der PFE steht außerdem vor der Herausforderung, dass die Anzeige auf unterschiedlichen Endgeräten stets nutzerfreundlich bleiben soll. Dazu reicht die Definition eines Farb-

schemas und der Linienstärke nicht aus. Das Einführen einer räumlichen Gruppierung der Elemente nach ihrer semantischen Bedeutung kann das automatische Layout unterstützen. Damit wird auch ermöglicht, dass Unterschiede wie Abstände von zusammengehörenden Elementen beim Engineering von Modulherstellern angeglichen werden können.

Zur Umsetzung der modulspezifischen Bedienbilder in projektspezifisch vereinheitlichten Bedienbildelementen müssen die Bedienbilder in einer darstellungsunabhängigen Beschreibungsform vorliegen. Diese Beschreibung enthält die Information zum Typ des darzustellenden Prozessequipments, dessen Lage- und Größeninformation, Verbindungsinformation zwischen den Elementen, optionale Zugehörigkeitsinformation zu einer semantischen Gruppe und gegebenenfalls weiterer dynamischer Information.

Diese Informationen sind durch einen Algorithmus auswertbar, der die projektabhängigen Bedienbildelemente in gewünschter Darstellung und Lage auf das Bedienbild in der PFE setzt und mit den entsprechenden Variablen für die Kommunikation mit der Modulsteuerung verknüpft.

4.1.3 Lokale Bedienstationen des Moduls

Neben der Visualisierung innerhalb der PFE, ist es vorstellbar, dass Module lokale Bedienstationen mitbringen. Diese können z.B. für Diagnosezwecke verwendet werden oder eine lokale Bedienung ermöglichen und können ebenfalls im MTP beschrieben werden.

Sollen diese später nicht in die PFE integriert werden, sondern lediglich die Vorortbedienung ermöglichen, müssen diese nicht im MTP beschrieben werden. Sollen diese jedoch in der PFE repliziert werden, können die gleichen Methoden wie für dezentrale Bedienbilder verwendet werden.

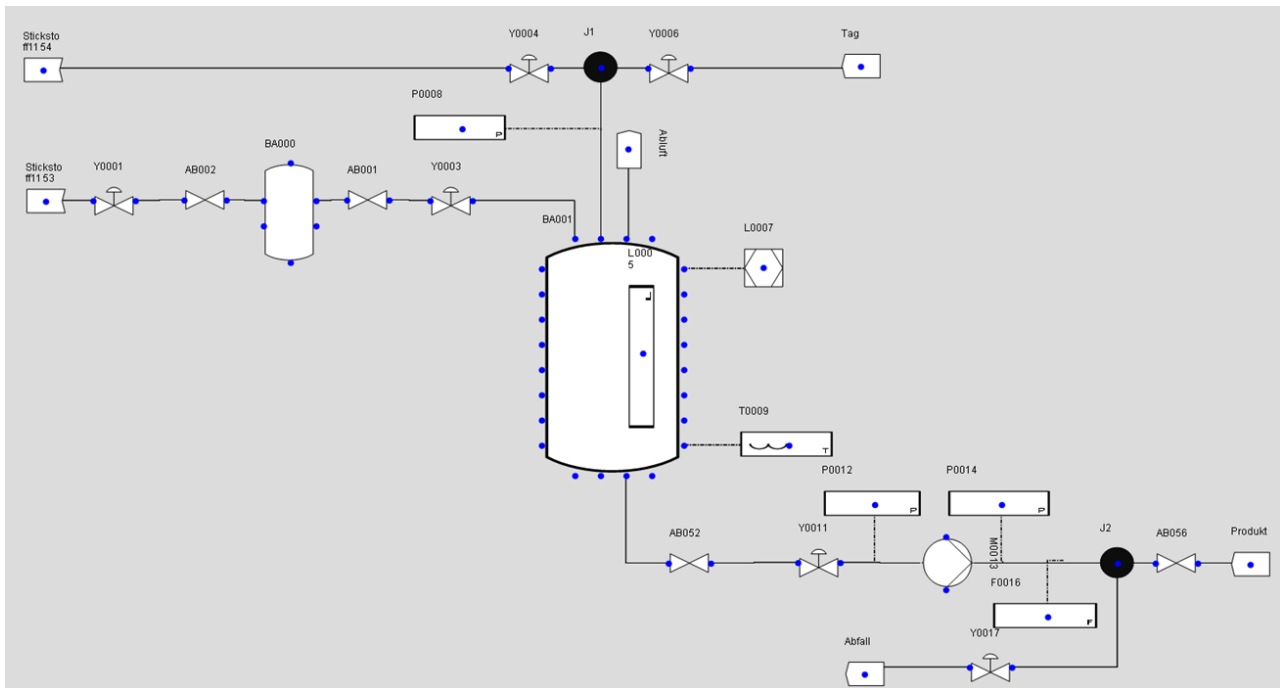


Bild 2. Beispiel eines Bedienbilds im Engineeringwerkzeug des Modulherstellers

Die Zugriffsmethodik der lokalen Bedienstationen, z.B. über eine Webschnittstelle, bleibt dem Modulhersteller überlassen und wird in dieser Richtlinie nicht weiter spezifiziert.

4.2 Bedienbilder als Aspekt im Module Type Package (MTP)

Die Darstellung des Aspekts der Bedienbildbeschreibung erfolgt als Erweiterung der Richtlinienserie, wie in Blatt 1 beschrieben. Hierfür wird die Bibliothek **MTPHMISUCLib** verwendet. Das zugehörige UML-Modell der notwendigen **SystemUnitClasses** ist in Bild 3 dargestellt, wobei aus Gründen der Übersichtlichkeit Bibliotheken und **SystemUnitClasses** aus anderen Blättern der Richtlinienserie ausschnittsweise dargestellt sind, die einen direkten Bezug für dieses UML-Modell besitzen. In der in diesem Blatt definierten Bibliothek wird das **HMISet** als Ableitung vom abstrakten **MTPSet** definiert, das den Ausgangspunkt für die Beschreibung der Bedienbilder darstellt.

Das **HMISet** kann beliebig viele Bedienbilder (**Pictures**) besitzen. Diese besitzen jeweils Information über die Ursprungsgröße des Bedienbilds. Hierfür werden die Attribute **Width** und **Height** verwendet, die die Ursprungsgröße in

Pixeln enthalten. Diese Information kann verwendet werden, um die Bedienbildbeschreibung auf andere Monitorauflösungen und Formate zu skalieren.

Zur Modellierung des Bedienbilds werden fünf Basisklassen in der **MTPHMISUCLib** bereitgestellt:

- **VisualObject** – wird verwendet, um ein Symbol im Bedienbild zu platzieren. Es ist vom **LinkedObject** abgeleitet. Ein **VisualObject** kann sowohl als statisches als auch dynamisches Objekt realisiert sein. Beispiele für statische Objekte sind Tanks oder manuelle Armaturen; Beispiele für dynamische Objekte sind hingegen Ventile, Pumpen oder Messstellen.
- **PortObject** – Diese Klasse und deren Ableitungen **LogicalPort**, **MeasurementPoint** und **Nozzle** werden verwendet, um Verbindungen zwischen Objekten herstellen zu können. Verbunden werden Objekte immer über **InternalLinks**, die die Verbindung zweier **ExternalInterfaces** bilden. Die **ExternalInterfaces** dürfen immer nur an **PortObject**-Instanzen hierarchisch unterhalb angehängt werden.

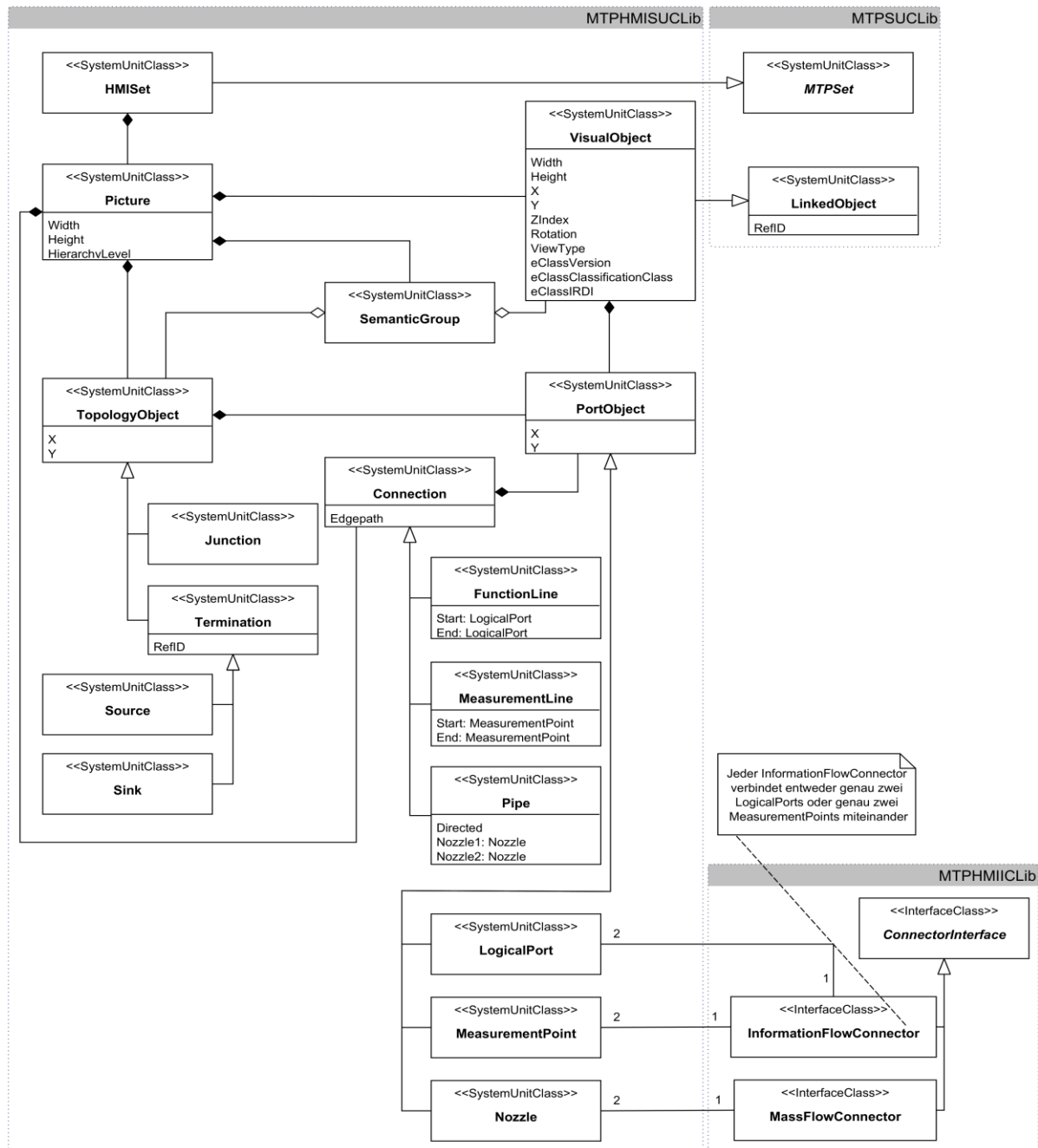


Bild 3. SystemUnitClasses für die Beschreibung von Bedienbildern

- Connection** – Diese Klasse und deren Kindklassen **FunctionLine**, **MeasurementLine** und **Pipe**, stellen Verbindungen zwischen Objekten des Bedienbilds her. Sie werden verwendet, um jeweils zwei **PortObjects** miteinander zu verbinden. Die Klasse **Connection** bietet hierzu ein Attribut **Edgepath**, das Start- und Endpunkt einer Verbindung sowie Zwischenpunkte (Eckpunkte der Linie) beinhaltet. Die spezialisierten Klassen bilden Rohrleitungen, Signale oder Zusammenhänge zwischen einem Messpunkt und einem Sensor ab. Diese

beinhalten bereits die notwendigen Anschlusspunkte und innerhalb der Anschlusspunkte die zu verwendenden **ExternalInterfaces**. Die **ExternalInterfaces** sind Instanzen der **InterfacesClasses**, wie in der **MTPHMIICLib** festgelegt. Die Basisklasse **ConnectorInterface** wird hierbei nicht verwendet und dient lediglich der Verallgemeinerung der spezialisierten Klassen **MassFlowConnector** und **InformationFlowConnector**. **MassFlowConnector** wird verwendet, um zwei Anschlusspunkte des Typs **Nozzle** zu verbinden, und stellt damit

eine Verbindung im Materialfluss dar. **InformationFlowConnector** wird verwendet, um eine Verbindung im Informationsfluss zu modellieren, und kann damit genutzt werden, um die Verbindung zweier Anschlusspunkte des Typs **LogicalPort** oder des Typs **MeasurementPoint** zu verbinden. Für Elemente, die immer eine definierte Anzahl von Verbindungen im Materialfluss aufweisen dürfen, werden die **Nozzle**-Objekte bereits in der **SystemUnitClass** vordefiniert, z.B. hat eine Rohrleitung immer zwei Verbindungspunkte.

- **TopologyObject** – und deren Ableitungen werden verwendet, um einerseits Grenzen des Bedienbilds zu beschreiben (Sprungmarken bzw. Anfang und Ende) und andererseits Verzweigungen von **Connection**-Instanzen zu erlauben.
- **SemanticGroup** – Diese Klasse wird dazu verwendet, um innerhalb eines Bedienbildes semantische Gruppen zu beschreiben, wodurch eine Zusammengehörigkeit von den beinhalteten verschiedenen Symbolen ausgedrückt wird.

Die **SystemUnitClasses** definieren einige AutomationML-Attribute, die zur Beschreibung des Bedienbilds verwendet werden. Diese sind in Tabelle 1 zusammenfassend und für jede **SystemUnitClass** aufgelistet. Attribute von Elternklassen werden bei abgeleiteten Klassen nicht nochmals in der Tabelle aufgelistet.

4.2.1 Bedienbilder (Picture)

Ein Element des Typs **Picture** bildet das oberste Element der **InstanceHierarchy** und beinhaltet die Daten der Referenzgrafik. Die ursprüngliche Höhe und Breite und damit das Seitenverhältnis wird durch die entsprechenden Attribute abgebildet. Da innerhalb eines MTP mehrere Grafiken definiert werden dürfen – z.B. eine Übersichtsgrafik und eine Detailansicht –, wird im **Picture**-Objekt der jeweilige **HierarchyLevel** (die Detailierungstiefe des HMI) mit angegeben. Hierzu wird eine Pfadangabe verwendet, wobei die einzelnen Pfadelemente durch Punkt getrennt werden, z.B. „1.2.3“. Damit können beliebig viele Hierarchieebenen codiert werden, wobei das erste Pfadelement die oberste Hierarchieebene („1“) beschreibt, das zweite Element die zweite („1.2“) usw.. Die einzelnen Pfadelemente können dabei alphanumerisch sein. Falls nur eine Grafik definiert sein sollte, kann der **HierarchyLevel** weglassen werden.

4.2.2 Symbole (VisualObject)

Im MTP können statische und dynamische Symbole, die Zusammenhänge des im Modul realisierten Prozesses darstellen, modelliert werden. Statische Symbole sind all diese, die keine dynamische Änderung der Werte benötigen, um ihre Funktionsweise darzustellen, z.B. ein Behälter oder ein Wärmetauscher. Dynamische Symbole werden verwendet, um die Funktionsweise von Apparaten darzustellen, die durch eine Änderung des Symbols oder des angezeigten Werts beschrieben werden, z.B. ein Ventil oder eine Pumpe.

Beide Symbolarten werden gleich modelliert. Es wird hierarchisch unterhalb des **Picture** Wurzelements eine Instanz der **SystemUnitClass VisualObject** angehängt. Dieses wird durch die Koordinaten (Attribute **x** und **y**) der oberen linken Ecke und die Höhe und Breite (Attribute **Width** und **Height**) des orthogonal ausgerichteten, minimal umgebenden Rechtecks des Symbols beschrieben. Hinzu kommt ein Attribut für die Drehung des Symbols, **Rotation**, das die Rotation des Objekts im mathematisch positiven Sinne (gegen den Uhrzeigersinn) in Grad spezifiziert. 0 Grad entspricht hierbei einem Material- oder Informationsfluss von links nach rechts. Es wird das fertige Bedienbild beschrieben, d.h. die Objekte sind bereits rotiert. Darüber hinaus wird über das Attribut **ZIndex** beschrieben auf welcher Ebene das Objekt liegt. Hierbei entspricht ein Wert von 0 dem Hintergrund. Werte größer als 0 beschreiben Ebenen, die weiter vorne liegen. Hierbei sind die Ebenen relativ zueinander zu verstehen, so dass ein Objekt mit **ZIndex** = 2 vor einem Objekt mit **ZIndex** = 1 liegt. Hierdurch wird eine Überlagerung von Objekten ermöglicht.

Zusätzlich können für dynamische Symbole spezialisierte Ansichten für das spezifische Bedienbild angegeben werden. Diese Ansichten verwenden eine eCl@ss-Referenz, um den genauen Typ des darzustellenden Equipments festzulegen, modelliert nach [4]. Für statische Symbole müssen eCl@ss-Referenzen angegeben sein.

Das statische Symbol besitzt ein AutomationML-Attribut **RefId**, das nicht belegt ist. Das dynamische Symbol verwendet dieses, um auf die zugehörige **DataAssembly**-Instanz zu referenzieren.

Diese Art der Symbole darf logisch mit anderen Symbolen verbunden werden. Hierfür werden als Kindobjekte Instanzen von **PortObject** gebildet, die wiederum Verbindungen enthalten.

Um die dynamischen Symbole des Bedienbilds zu modellieren, wird das Konzept der

LinkedObjects, wie in VDI/VDE/NAMUR 2658 Blatt 1 beschrieben, verwendet.

Die **InternalElements** des Bedienbilds und das Manifest sind durch eine eindeutige Identifikation miteinander verbunden. Verbundene Elemente verwenden dafür das Attribut **RefID**, um dies sicherzustellen.

Bild 4 zeigt wie die Objekte der Visualisierung mit den Datenobjekten im Manifest verbunden sind. Hierfür erhält das Symbol die gleiche **RefID** wie das Datenobjekt.

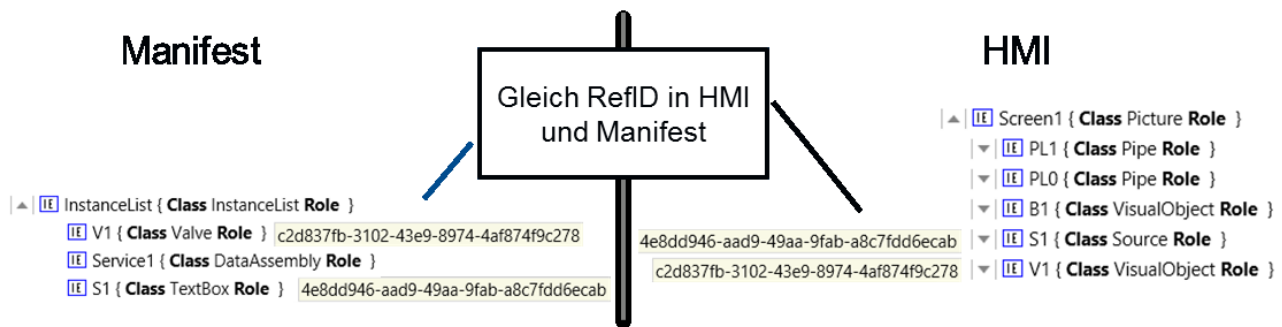


Bild 4. Verknüpfung Manifest und HMI

4.2.3 Symbolverbindungen (Connection, PortObject, Junction)

Zur Herstellung einer Verbindung zwischen Symbolen (**VisualObject**) und/oder **TopologyObjects** werden **PortObject** und **Connection** verwendet.

Hierfür werden in der **InstanceHierarchy** unterhalb des Wurzelobjekts des Typs **Picture** die nötigen Instanzen von **Connection** erzeugt. Sollen zwei Symbole durch ein Rohrleitungssymbol verbunden werden, so wird ein **InternalElement** vom Typ **Pipe** instanziiert. Das **InternalElement** vom Typ **Pipe** hat zwei Anschlusspunkte vom Typ **Nozzle**. Diese können mit anderen Anschlusspunkten vom Typ **Nozzle** verbunden werden. Ist das Attribut **Directed** wahr, wird eine Vorzugsrichtung von Anschlusspunkt **Nozzle1** zu **Nozzle2** angedeutet.

Am dynamischen oder statischen Symbol ist deshalb als Kind **InternalElement** eine Instanz vom Typ **Nozzle** anzuhängen.

Die Objekte vom Typ **Nozzle** besitzen jeweils ein **ExternalInterface** vom Typ **MassFlowConnector**, die über einen **InternalLink** in AutomationML miteinander verbunden werden. Der **InternalLink** wird an

So erhält man eine eindeutige Zuordnung zwischen den Bedienbildern und der Kommunikationschnittstelle.

Um die Zusammengehörigkeit auch lesbar darzustellen, wird hier empfohlen, die Namen der Symbole in der Bedienbildbeschreibung entsprechend den Tag-Namen der Datenobjekte im Manifest zu wählen. Das heißt: Der Name des Symbols **InternalElements** sollte ebenfalls der Tag-Name des zugehörigen Equipments sein.

das Wurzelobjekt in der **InstanceHierarchy** angehängt und ist somit für alle **InternalElements**, die hierarchisch unterhalb hängen, erreichbar.

Werden **InternalElements** im Informationsfluss miteinander verbunden, so wird mit diesen analog verfahren, wobei die Verbindung vom Typ

- **MeasurementLine** immer dann zum Einsatz kommt, wenn ein Messpunkt an ein dynamisches oder statisches Symbol (Rohrleitung zu Sensor) angebracht werden soll.
- **FunctionLine** immer dann zum Einsatz kommt, wenn eine logische Verbindung zwischen zwei Symbolen (Sensor zu Regler) hergestellt werden soll.

Die Vererbungshierarchie von **PortObject** und **Connection** sowie die Vererbungshierarchie der verwendeten **InterfaceClasses** wird in Bild 3 dargestellt.

Grundsätzlich besitzen die abgeleiteten **SystemUnitClasses** von **Connection** bereits die nötigen **PortObject**-Instanzen. Außerdem dürfen nur **PortObject**-Instanzen des gleichen Typs miteinander verbunden werden. Alle **PortObject SystemUnitClasses** besitzen bereits ein **ExternalInterface** des erlaubten Typs und deren Instanzen dürfen auch keine weiteren erhalten.

Die **PortObject**-Instanzen unterhalb eines **VisualObject** werden auf der Anzeigenebene, die über den **ZIndex** des **VisualObject** angegeben wird, angezeigt. Alle **PortObject**-Instanzen, die sich nicht unterhalb eines **VisualObject** befinden, werden im Hintergrund angezeigt.

Die verwendeten **PortObject**-Instanzen die miteinander verbunden werden, liegen immer auf dem gleichen grafischen Punkt (gleiche *x,y*-Koordinate). Sie erhalten ein **ExternalInterface**, das mit dem **ExternalInterface** des Partnerobjekts durch einen AutomationML-konformen **Internallink** verbunden wird. Die **Internallinks**, die im Beispiel benötigt werden, werden dann am Wurzelement angehängt.

Für eine Auflistung aller Modellierungsvorschriften für Verbindungen sei auf Abschnitt 4.3.7.2 verwiesen.

4.2.3.1 Rohrleitungen (Pipe)

Als Beispiel zur Modellierung von Rohrleitungen werden in Bild 5 zwei Symbole durch eine Rohrleitung verbunden. In Bild 5 werden die **InternalLinks**, die sich als Subelemente im **InternalElement RuehrreaktorHmi** befinden an den Pfeilen, welche die Verbindung der Elemente symbolisieren dargestellt. Außerdem werden die Koordinaten der Verbindungspunkte am jeweiligen Verbindungspunkt dargestellt.

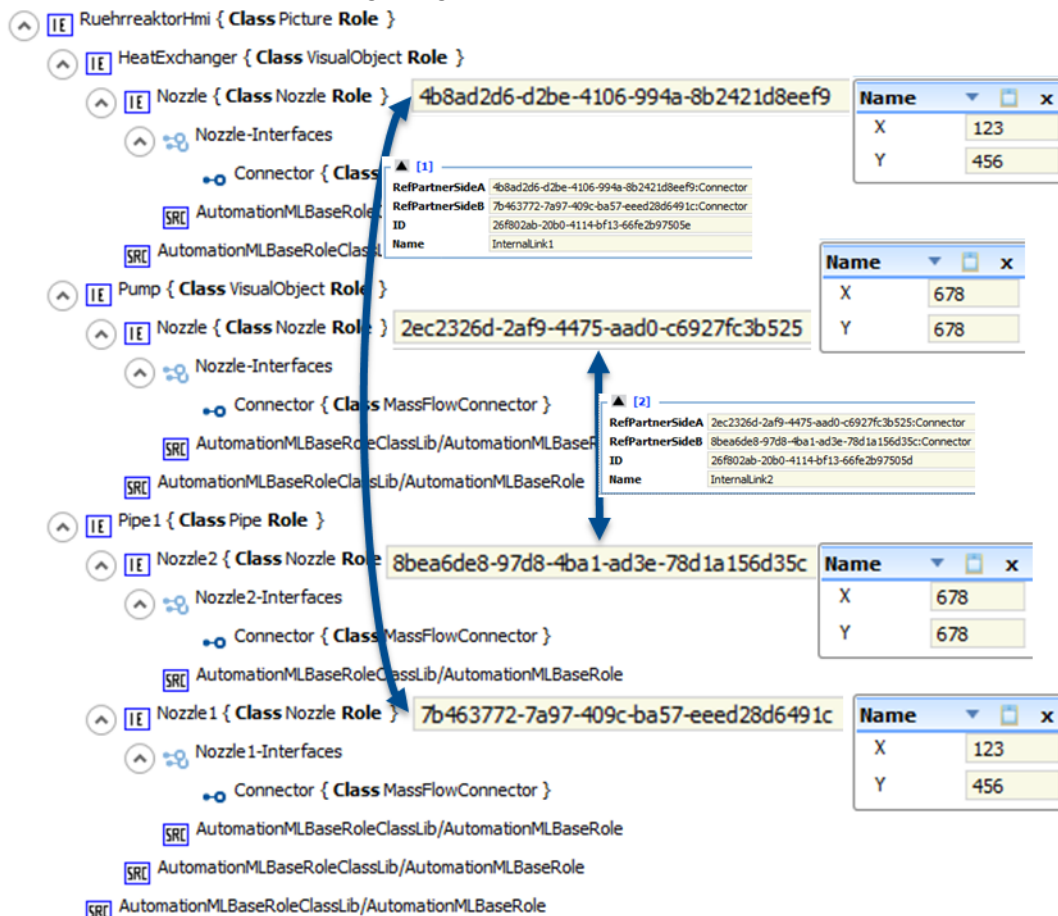


Bild 5. Beispiel einer modellierten Rohrleitung

4.2.3.2 Signale (FunctionLine, MeasurementLine)

Signale werden wie die Rohrleitungen über **Port-Object**-Instanzen verbunden. Allerdings gibt es zwei verschiedene Arten von Signalen, funktionale Zusammenhänge (**FunctionLine**) und Messsignale (**MeasurementLine**).

- Funktionale Zusammenhänge werden verwendet, wenn zwei Equipments ein Signal

austauschen, also z.B. ein Sensor mit einem Regler logisch verbunden ist und dies im Bedienbild dargestellt werden soll.

- Messsignale werden verwendet, wenn ein Messwert und dessen Messstelle im Materialfluss zusammenhängend dargestellt werden sollen. Also wenn z. B. eine Durchflussanzeige mit einer Linie am entspre-

chenden Rohr, an dem gemessen wird, dargestellt werden soll.

Im Beispiel in Bild 6 wird ein funktionaler Zusammenhang dargestellt. Hier wird ein Stellsignal einer Pumpe von einem Regler ausgegeben. Dieser Zusammenhang soll im Bedienbild modelliert sein und muss deshalb auch im MTP entsprechend vorgesehen werden. Genau wie Rohrleitungen werden **PortObject**-Instanzen, in diesem Fall vom Typ **LogicalPort**, verwendet. Es können auch hier immer nur zwei Ports des gleichen Typs miteinander verbunden sein. Diese erhalten wie bei Rohrleitungsverbindungen die gleichen Koordinaten und werden über einen **InternalLink** zwischen den beinhalteten **ExternalInterfaces** miteinander verbunden.

In Bild 6 werden die **InternalLinks**, die sich als Subelemente im **InternalElement RuehrreaktorHmi** befinden an den Pfeilen, welche die Verbindung der Elemente symbolisieren dargestellt. Außerdem werden die Koordinaten der Verbindungspunkte am jeweiligen Verbindungspunkt dargestellt.

Bild 7 stellt die Modellierung eines Messsignals dar. Hierfür werden **InternalElements** des Typs **MeasurementLine** verwendet. Der Unterschied zu den funktionalen Zusammenhängen stellt

sich dadurch dar, dass die Ports Instanzen von **MeasurementPoint** sind.

In Bild 7 werden die **InternalLinks**, die sich als Subelemente im **InternalElement RuehrreaktorHmi** befinden an den Pfeilen, welche die Verbindung der Elemente symbolisieren dargestellt. Außerdem werden die Koordinaten der Verbindungspunkte am jeweiligen Verbindungspunkt dargestellt.

MeasurementPoint-Instanzen dürfen immer da verwendet werden, wo das Signal vom Typ **MeasurementLine** angeschlossen wird, im Beispiel an einer Rohrleitung.

Das heißt, zusätzlich zu den vorhandenen Verbindungsstellen des Typs **Nozzle** dürfen beliebig viele Messpunkte an einer Rohrleitung hinzugefügt werden. Grundsätzlich dürfen bei Instanzen von **SystemUnitClasses**, die bereits **PortObject**-Instanzen besitzen, keine weiteren hinzugefügt bzw. keine weggelassen werden. Die Rohrleitung bildet hierbei die Ausnahme.

Auch die Messsignale werden wie die funktionalen Zusammenhänge und die Rohrleitungen über deren **PortObject**-Instanzen und die darin befindlichen **ExternalInterfaces**, in diesem Fall

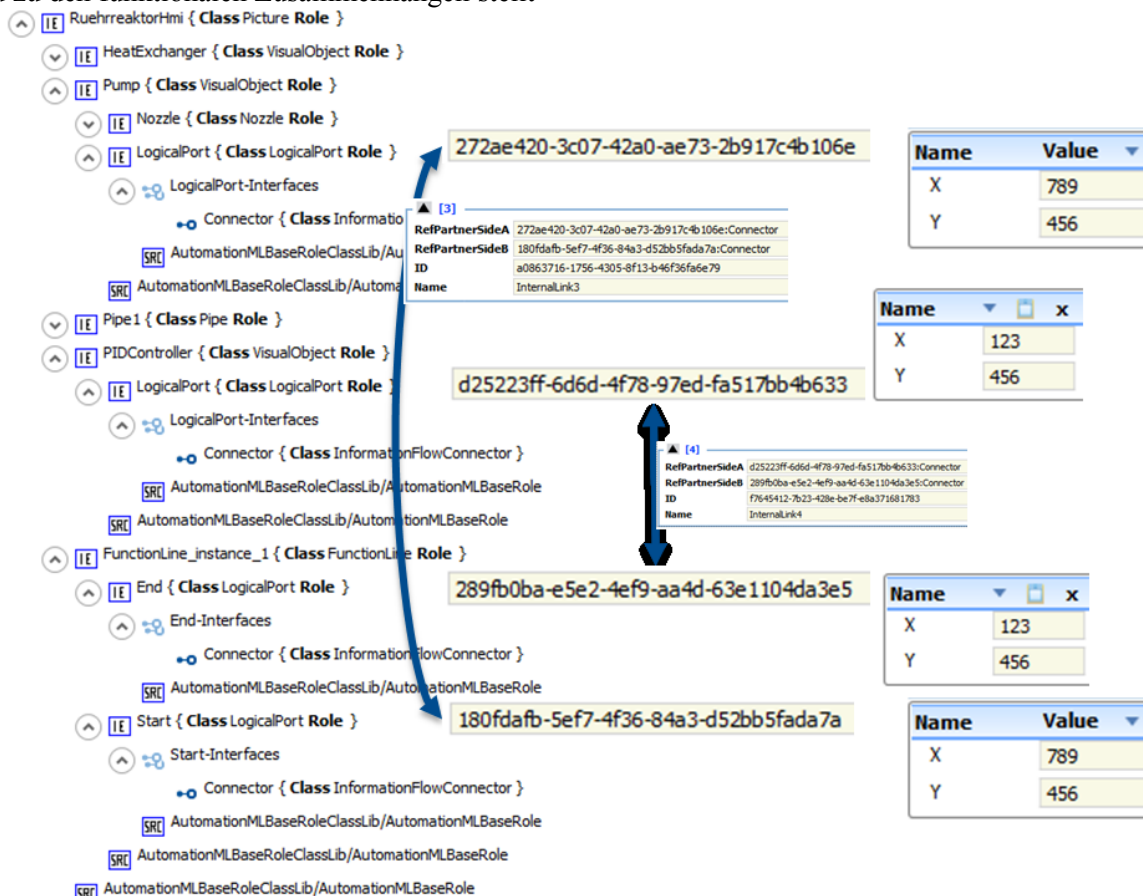


Bild 6. Beispiel eines modellierten Signals

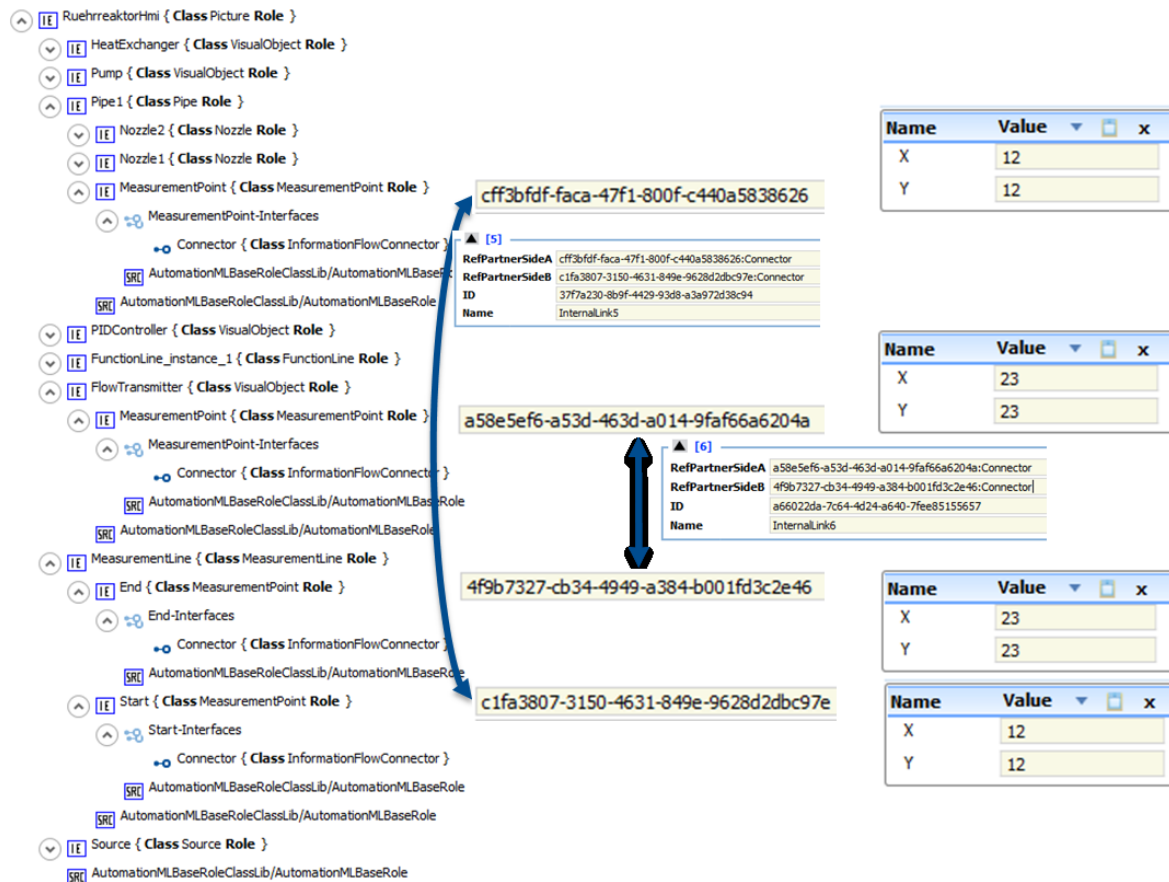


Bild 7. Beispiel einer modellierten Messstelle

vom Typ **InformationFlowConnector**, verbunden.

4.2.3.3 Verzweigungspunkte (Junction)

Zusätzlich zu den Symbolverbindungen ist es möglich, Abzweigungen in Verbindungen (**SystemUnitClass Connection**) zu modellieren. Hierfür steht die **SystemUnitClass Junction** zur Verfügung. Diese **Junction** ist vom Typ **TopologyObject** abgeleitet (Bild 3).

Eine **Junction** stellt einen Verzweigungspunkt mit drei angeschlossenen Verbindungssegmenten vom gleichen Typ dar. Dazu besitzt die **Junction** drei Instanzen des gleichen Typs von einer Ableitung des Typs **PortObject**, z.B. **Nozzle** für Rohrleitungen (**Pipe**). Außer einer x,y-Koordinate besitzt sie keine weiteren AutomationML-Attribute.

Wie im Beispiel in Bild 8 dargestellt, wird jeweils ein Port jeder zu verbindenden Rohrleitung mit einem Port des Verzweigungspunkts verbunden. Der verbleibende Port kann – wie zuvor beschrieben – mit einem anderen Symbol oder auch einem weiteren Verzweigungspunkt verbunden

sein. Die Verbindungen werden wie bei herkömmlichen Rohrleitungsverbindungen über die **ExternalInterfaces** und die entsprechenden **InternalLinks** zwischen diesen abgebildet.

4.2.4 Bedienbildgrenzen (Termination)

Letztlich können zur Modellierung der Bedienbilder Sprungmarken verwendet werden.

Sprungmarken stellen die Grenzen des Bedienbilds dar und können sowohl eingehende als auch ausgehende Grenzen sein. Hierfür werden in diesem Modellierungsverfahren drei Klassen vom Typ **TopologyObject** abgeleitet, die ungerichtete Grenzen (**Termination**), eingehende Grenzen (**Source**) und ausgehende Grenzen (**Sink**) sein können (Bild 3).

Aus diesen lassen sich Instanzen bilden, die im Bedienbild entsprechend als Sprungmarken oder Textfelder interpretiert werden können.

Die Bedienbildgrenzen können genau wie die dynamischen und statischen Symbole mit anderen Symbolen verbunden werden. Hierfür werden auch wie für normale Symbole **PortObject**-Instanzen verwendet. Die Einschränkung hier ist, dass Be-

Bedienbildgrenzen immer nur ein **PortObject** als Kindobjekt besitzen dürfen.



Bild 8. Beispiel einer modellierten Rohrleitungsverzweigung, Modellierung von Bediengrenzen

Um Sprungmarken verschiedener Bedienbilder miteinander zu verbinden, erhält jede Sprungmarke ein AutomationML-Attribut **RefID**, das die ID des **InternalElement** der Partnersprungmarke enthält.

4.2.5 Semantische Gruppen (SemanticGroup)

Ein weiteres Modellierungskonzept für Bedienbilder sind semantische Gruppen. Sie zeigen eine Zusammengehörigkeit verschiedener Symbole an. Dies kann helfen, wenn Layoutalgorithmen verwendet werden sollen, um das Bedienbild in der PFE besser darzustellen.

So können beispielsweise Messgeräte und Aktoren einem Behälter zugeordnet werden. Damit können sie unabhängig von Auflösung und Darstellungsform der Symbole nahe am Behälter platziert werden. Ein weiteres Beispiel wäre die Zuordnung von Absperrventilen und Pumpen zueinander.

Zur Modellierung semantischer Gruppen steht die **SystemUnitClass SemanticGroup** zur Verfügung. Instanzen dieser Klasse können wie Symbole oder Verbindungen direkt unter dem Wurzelement hierarchisch eingeordnet werden. Diese Instanzen bilden dann einen weiteren Knoten an den hierarchisch die Symbole und Verbindungen eingeordnet werden dürfen, sind also eine Art Wurzelknoten.

Der Unterschied zum „echten“ Wurzelement besteht darin, dass eine semantische Gruppe keine **InternalLinks** beinhalten darf. Sie darf auch selbst keine Verbindungen haben und hat keine Position auf dem Bedienbild. Sie bildet also lediglich eine Zusammengehörigkeit verschiedener Symbole logisch ab.

5 Modellierungsvorschriften zur Erstellung eines Bedienbilds

Um den automatischen Übertrag der während des Modulengineerings erstellten Bedienbilder in die PFE zu gewährleisten, müssen die im Folgenden beschriebenen Modellierungsvorschriften eingehalten werden.

5.1 Allgemeine Modellierungsvorschriften

Die in Tabelle 2 abgebildeten Vorschriften sind bei der Modellierung des Bedienbilds zu beachten. Die **SupportedRoleClass** – die für alle **SystemUnitClasses** die gleiche ist – wird der Übersicht wegen für die Elemente der Beispiele nicht gezeigt.


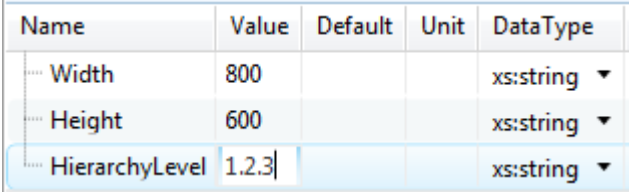
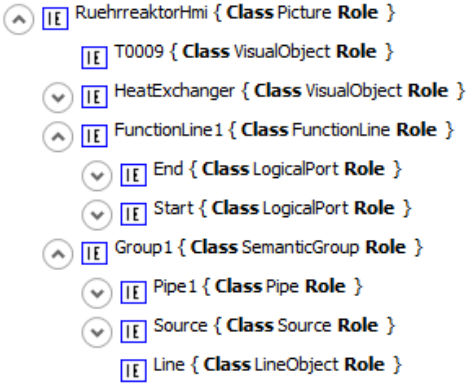
5.2 Modellierungsvorschriften für Verbindungen

Die in Tabelle 3 gelisteten Modellierungsvorschriften sind für Verbindungen einzuhalten.

Tabelle 1. Definierte AutomationML-Klassen

Lib	SUC/IC	Parent SUC	Attributes	Attribute data type
MTPHMISUCLib	PortObject	–	X Y	xs:integer xs:integer
	LogicalPort	PortObject	–	
	MeasurementPoint	PortObject	–	
	Nozzle	PortObject	–	
	TopologyObject	–	X Y	xs:integer xs:integer
	Junction	TopologyObject	–	
	Termination	TopologyObject	RefID	xs:ID
	Source	Termination	–	
	Sink	Termination	–	
	Connection	–	Edgepath	xs:string (formatted)
	Pipe	Connection	Directed	xs:boolean
	FunctionLine	Connection	–	
	MeasurementLine	Connection	–	
	Picture	–	Width Height HierarchyLevel	xs:integer xs:integer xs:string (formatted)
	SemanticGroup	–	–	
	VisualObject	LinkedObject	Width Height X Y ZIndex Rotation eClassVersion eClassClassifica- tionClass eClassIRDI	xs:integer xs:integer xs:integer xs:integer xs:integer xs:integer xs:string xs:string xs:string
MTPHMIICLib	ConnectorInterface	–	–	
	InformationFlow- Connector	ConnectorInterface	–	
	MassflowConnector	ConnectorInterface	–	

Tabelle 2. Modellierungsvorschrift Bedienbild

ID	Regel	Beispiel
1	Alle InternalLinks werden am PictureObjekt des jeweiligen HMI abgelegt.	
2	Die Zuordnung zwischen Symbol und dem zugehörigen Datenobjekt wird über das LinkedObject -Konzept, beschrieben in VDI/VDE/NAMUR 2658 Blatt 1, modelliert.	
3	x,y-Koordinaten werden absolut, bezogen auf den Ursprung, links oben im Bedienbild, beschrieben. Die x-Richtung ist von links nach rechts zunehmend. Die y-Richtung ist von oben nach unten zunehmend. <i>Anmerkung:</i> Dies gilt auch für Instanzen von PortObject	
4	Es wird eine InstanceHierarchy für die Bedienbilder erstellt. Diese enthält alle Bedienbilder des Moduls.	
5	Für das Bedienbild wird ein InternalElement der Klasse Picture erzeugt. Dieses beinhaltet die Informationen aus dem Referenzbild und erhält als Namen den Namen des Bedienbilds. Werden mehrere Bedienbilder modelliert, beschreibt das Attribut HierarchyLevel die hierarchische Sicht auf die verschiedenen Bedienbilder. Die Formatierung der Pfadangabe ist: „<Erste Ebene>.<Zweite Ebene>.<Dritte Ebene>“, wobei die Anzahl der Hierarchieebenen beliebig ist. So beschreibt z. B. „UnitA.Module5“ die Einordnung eines Bedienbilds in die zweite Hierarchieebene als Detailbild zu „UnitA“. „UnitA.Module5.SubmoduleX“ beschreibt die Einordnung in die dritte Hierarchieebene als Detailierung des im vorherigen Beispiel genannten Bedienbilds.	
6	Unterhalb des Picture -Objekts dürfen Instanzen folgender SystemUnitClasses und deren Spezialisierungen enthalten sein: VisualObject, TopologyObject, Connection, SemanticGroup	
7	Unterhalb semantischer Gruppen dürfen Instanzen folgender SystemUnitClasses und deren Spezialisierungen enthalten sein: VisualObject, Connection, TopologyObject Des Weiteren dürfen semantische Gruppen auch aus einer oder mehreren anderen semantischen Gruppen aufgebaut werden.	
8	Semantische Gruppen dürfen nicht verbunden werden. Semantische Gruppen besitzen keine PortObjects .	

ID	Regel	Beispiel																																	
9	Objekte die logisch verbunden werden, erhalten Instanzen der SystemUnitClass PortObject oder deren Spezialisierungen.	<div><div>▲</div><div>IE</div><div>PL0 { Class Pipe Role }</div></div> <div><div>▼</div><div>IE</div><div>Nozzle1 { Class Nozzle Role }</div></div> <div><div>▼</div><div>IE</div><div>Nozzle2 { Class Nozzle Role }</div></div> <div><div>▲</div><div>IE</div><div>B1 { Class VisualObject Role }</div></div> <div><div>▼</div><div>IE</div><div>B1_N0 { Class Nozzle Role }</div></div>																																	
10	Jeder Port besitzt genau ein Interface vom Typ MassFlowConnector oder InformationFlowConnector . Diese sind in der MTPHMIICLib definiert.	<div><div>▲</div><div>IE</div><div>B1 { Class VisualObject Role }</div></div> <div><div>▲</div><div>IE</div><div>B1_N0 { Class Nozzle Role }</div></div> <div><div>▲</div><div>B1_N0-Interfaces</div><div><div>●</div><div>Connector { Class MassFlowConnector }</div></div></div>																																	
11	SystemUnitClasses mit vordefinierten Ports dürfen keine weiteren Ports erhalten. Die Ausnahme bildet die Rohrleitung Pipe , an die beliebig viele MeasurementPoints zusätzlich instanziiert werden dürfen.	<div><div>▲</div><div>IE</div><div>B1 { Class VisualObject Role }</div></div> <div><div>▲</div><div>IE</div><div>B1_N0 { Class Nozzle Role }</div></div> <div><div>▲</div><div>B1_N0-Interfaces</div><div><div>●</div><div>Connector { Class MassFlowConnector }</div></div></div>																																	
12	Symbole werden mithilfe einer eCI@ss-Referenz näher spezifiziert. Diese wird am VisualObject wie in [4] beschrieben modelliert. Für statische Symbole ist die eCI@ss-Referenz zwingend anzugeben. Für dynamische Objekte ist diese optional.	<table><tr><th>Name</th><th>Value</th><th>DataType</th></tr><tr><td>Width</td><td>12</td><td>xs:integer ▼</td></tr><tr><td>Height</td><td>12</td><td>xs:integer ▼</td></tr><tr><td>X</td><td>1234</td><td>xs:integer ▼</td></tr><tr><td>Y</td><td>5678</td><td>xs:integer ▼</td></tr><tr><td>ZIndex</td><td>0</td><td>xs:integer ▼</td></tr><tr><td>Rotation</td><td>90</td><td>xs:integer ▼</td></tr><tr><td>eClassVersion</td><td>9.1</td><td>xs:string ▼</td></tr><tr><td>eClassClassificationClass</td><td>27200490</td><td>xs:string ▼</td></tr><tr><td>eClassIRDI</td><td></td><td>xs:string ▼</td></tr><tr><td>RefID</td><td>4e97836d-0f09-4eea-ab82-166fcd3782be</td><td>xs:ID ▼</td></tr></table>	Name	Value	DataType	Width	12	xs:integer ▼	Height	12	xs:integer ▼	X	1234	xs:integer ▼	Y	5678	xs:integer ▼	ZIndex	0	xs:integer ▼	Rotation	90	xs:integer ▼	eClassVersion	9.1	xs:string ▼	eClassClassificationClass	27200490	xs:string ▼	eClassIRDI		xs:string ▼	RefID	4e97836d-0f09-4eea-ab82-166fcd3782be	xs:ID ▼
Name	Value	DataType																																	
Width	12	xs:integer ▼																																	
Height	12	xs:integer ▼																																	
X	1234	xs:integer ▼																																	
Y	5678	xs:integer ▼																																	
ZIndex	0	xs:integer ▼																																	
Rotation	90	xs:integer ▼																																	
eClassVersion	9.1	xs:string ▼																																	
eClassClassificationClass	27200490	xs:string ▼																																	
eClassIRDI		xs:string ▼																																	
RefID	4e97836d-0f09-4eea-ab82-166fcd3782be	xs:ID ▼																																	
13	Das Edgepath -Attribut der SystemUnitClass Connection enthält alle Eckpunkte einer Verbindung als Zeichenkette in folgender Form: x1, y1; x2, y2; ...; xn, yn; Start- und Endpunkt sind in der Zeichenkette enthalten. Das erste Koordinatenpaar entspricht dem Startpunkt, das letzte dem Endpunkt.	<div><div>▼</div><div>IE</div><div>PL1 { Class Pipe Role }</div></div> <div><div>Name</div><div>Edgepath</div><div>208,251;356,251;356,254;</div></div>																																	
14	Der Name eines Internalelement entspricht, sofern möglich, immer dem Tag-Namen des zugehörigen Objekts.	<div><div>IE</div><div>T0009 { Class VisualObject Role }</div></div>																																	
15	Bei VisualObjects ohne Dynamik bleibt das AutomationML-Attribut RefId leer.																																		
16	VisualObjects dürfen ihrem Typ entsprechend beliebig viele Anschlusspunkte besitzen.																																		
17	Höhe, Breite und x,y-Koordinaten werden in Pixeln angegeben.	<table><tr><td>Width</td><td>40</td></tr><tr><td>Height</td><td>40</td></tr><tr><td>X</td><td>123</td></tr><tr><td>Y</td><td>456</td></tr><tr><td>ZIndex</td><td>0</td></tr><tr><td>Rotation</td><td>0</td></tr></table>	Width	40	Height	40	X	123	Y	456	ZIndex	0	Rotation	0																					
Width	40																																		
Height	40																																		
X	123																																		
Y	456																																		
ZIndex	0																																		
Rotation	0																																		
18	Symboldrehungen werden in dem Rotation -Attribut in Grad angegeben. Begonnen wird bei 0° in Material- oder Informationsflussrichtung von links nach rechts und mathematisch positiv gedreht. Beispiel: Eine Pumpe pumpt von links nach rechts, ergibt 0°. Eine Pumpe pumpt von unten nach oben, ergibt 90°.	<table><tr><td>Width</td><td>40</td></tr><tr><td>Height</td><td>40</td></tr><tr><td>X</td><td>123</td></tr><tr><td>Y</td><td>456</td></tr><tr><td>ZIndex</td><td>0</td></tr><tr><td>Rotation</td><td>0</td></tr></table>	Width	40	Height	40	X	123	Y	456	ZIndex	0	Rotation	0																					
Width	40																																		
Height	40																																		
X	123																																		
Y	456																																		
ZIndex	0																																		
Rotation	0																																		

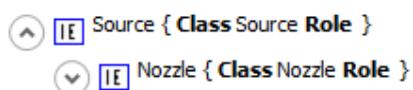
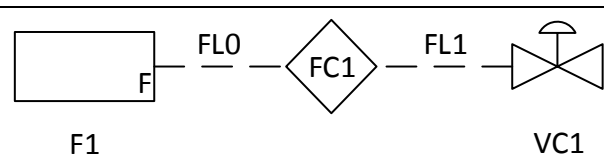
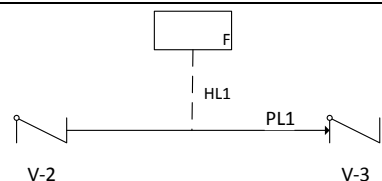
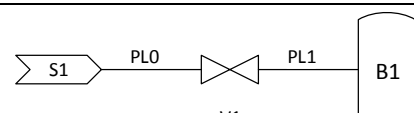
ID	Regel	Beispiel
19	Die Lage eines VisualObject in der Anzeigenebene wird über das Attribut ZIndex festgelegt. Hierbei bezeichnet ein Wert von 0 den Hintergrund. Je höher der Wert ist, umso weiter vorn liegt das VisualObject . Die Werte des ZIndex sind relativ zueinander zu interpretieren, sodass ein Object mit ZIndex = 2 vor einem Object mit ZIndex = 1 liegt. PortObjects , die Anschlüsse eines VisualObject repräsentieren werden in der gleichen Anzeigenebene wie das Eltern- VisualObject angezeigt.	
20	Termination und deren Spezialisierungen besitzen immer genau einen Anschlusspunkt, entweder vom Typ Nozzle oder LogicalPort .	

Tabelle 3. Modellierungsvorschriften für Verbindungen

ID	Regel	Beispiel
21	FunctionLine wird eingesetzt, um eine logische Funktion zu visualisieren, z. B. eine Regelschleife oder Verbindung zwischen Regler und Regelventil.	 <p> IE FLO { Class FunctionLine Role } IE F1 { Class VisualObject Role } </p>
22	MeasurementLine wird eingesetzt, um einen Zusammenhang zwischen Materialfluss und Informationsfluss zu signalisieren, z. B. bei Messstellen an Rohrleitungen oder Behältern.	 <p> IE HL1 { Class MeasurementLine Role } </p>
23	Pipe wird eingesetzt, um Zusammenhänge im Materialfluss zu beschreiben z. B., die Verbindung zwischen einem Ventil und einem Behälter über eine Rohrleitung.	 <p> IE PL1 { Class Pipe Role } IE PLO { Class Pipe Role } </p>
24	Verbindungen über Pipe : Werden verwendet, um Equipments mittels Rohrleitungen zu verbinden. Die zu verbindenden Objekte erhalten beide jeweils einen Port vom Typ Nozzle . Diese erhalten jeweils eine ExternalInterface -Schnittstelle vom Typ MassFlowConnector . Die Schnittstellen werden über einen InternalLink miteinander verbunden.	<p> IE PLO { Class Pipe Role } IE Nozzle1 { Class Nozzle Role } IE Nozzle2 { Class Nozzle Role } IE S1 { Class Source Role } IE Nozzle { Class Nozzle Role } IE V1 { Class VisualObject Role } IE V1_N1 { Class Nozzle Role } IE V1_N0 { Class Nozzle Role } </p>
25	Verbindungen über FunctionLine : Werden verwendet, um logische Verbindungen im Informationsfluss darzustellen. Die zu verbindenden Objekte erhalten beide jeweils einen Port vom Typ LogicalPort . Diese erhalten jeweils eine ExternalInterface -Schnittstelle vom Typ InformationFlowConnector . Die	<p> IE FLO { Class FunctionLine Role } IE FC1_LP { Class LogicalPort Role } IE F1_LP { Class LogicalPort Role } IE FC1 { Class VisualObject Role } IE FLO_LP { Class LogicalPort Role } IE F1 { Class VisualObject Role } IE FLO_LP { Class LogicalPort Role } </p>

	Schnittstellen werden über einen Internal-Link miteinander verbunden.	
26	Verbindungen über MeasurementLine : Werden verwendet, um einen Sensor mit einem sich im Materialfluss befindlichen Objekt (z. B. Rohrleitung oder Behälter) zu verbinden. An der Messstelle im Materialfluss wird ein Port vom Typ MeasurementPoint zur Verbindung verwendet. Am Sensor wird ebenfalls ein MeasurementPoint zur Verbindung verwendet. Die Ports erhalten jeweils eine ExternalInterface -Schnittstelle vom Typ InformationFlowConnector . Die Schnittstellen werden über einen Internal-Link miteinander verbunden.	
27	An ein PortObject darf genau ein Port-Object angebunden werden. Dieses muss vom gleichen Typ sein.	
28	Ein PortObject besitzt immer genau ein ExternalInterface zur Verbindung.	
29	Durch ConnectorInterfaces verbundene PortObjects besitzen die gleichen x,y-Koordinaten.	
30	Ein TopologyObject und seine PortObjects besitzen die gleichen x,y-Koordinaten.	

5.2.1 Beispiel einer Bedienbildbeschreibung

Im Folgenden wird ein Beispielbedienbild gezeigt, welche zunächst innerhalb eines Modulengineering Werkzeugs entwickelt wurde, Bild 15, anschließend ein MTP generiert wurde und in eine PFE importiert wurde, Bild 16.

Bild 17 zeigt einen Ausschnitt und die zugehörige AutomationML Modellierung. Im Beispiel wird ein Regelventil über ein Binärventil mit einem Behälter verbunden.

Für jedes darzustellende Objekt wird zunächst eine Instanz des zugehörigen Typs aus den **SystemUnitClasses** der HMI-Bibliothek, **MTPHMISUCLibrary** erstellt. Behälter und Ventile sind für das HMI nicht weiter spezifiziert und werden deshalb als **VisualObject** in die **InstanceHierarchy** bzw. das Wurzelobjekt, eingefügt. Jede Instanz von **VisualObject** besitzt Attribute zur Beschreibung der Lage und Größe des Symbols. Zusätzlich ist es möglich, eine Referenz auf die zugehörige eCl@ss-Beschreibung [3; 4] einzufügen. Hierdurch kann das anzuzeigende Symbol näher spezifiziert werden; im Fall des Ventils Y0001 aus dem Beispiel wird als eCl@ss-

Referenz auf die Klasse „37-01-02-03 Regulier-ventil“ verwiesen.

Neben den Objekten werden die Rohrleitungen als Instanzen des entsprechenden Typs dargestellt, im Fall des Beispiels sind dies zwei Rohrleitungen: PL2 und PL3. Jede Rohrleitung wird durch deren Eckpunkte beschrieben, die in das Attribut **Edgepath** eingetragen werden.

Verbindungen zwischen zwei Objekten werden durch Verwendung AutomationML-konformer **InternalLinks** abgebildet. Jeder Anschlusspunkt im Material- oder Informationsfluss besitzt ein **ExternalInterface** der zugehörigen Klasse. Diese können durch die **InternalLinks** miteinander verbunden werden. Alle **InternalLinks** werden am **Picture**-Objekt gespeichert, da dieses als Wurzel-**InternalElement** von allen darunterliegenden erreichbar ist. Im Beispiel werden vier **InternalLinks** verwendet, um die Rohrleitungen PL2 und PL3 (jeweils zwei **InternalLinks**) mit den jeweiligen Symbolen logisch zu verbinden. Das Beispiel wird in Anhang C als AutomationML bereitgestellt.

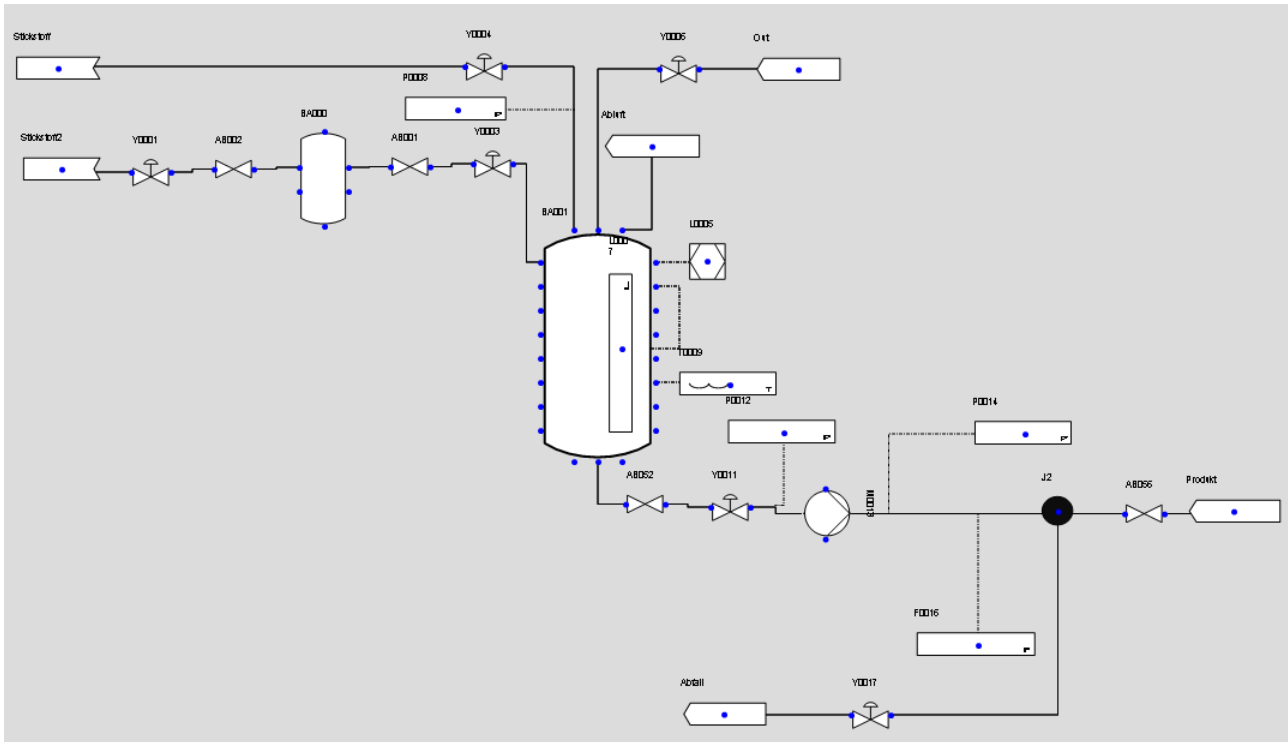


Bild 9. Bedienbild eines Moduls im Modulengineering Werkzeug zum Export als MTP

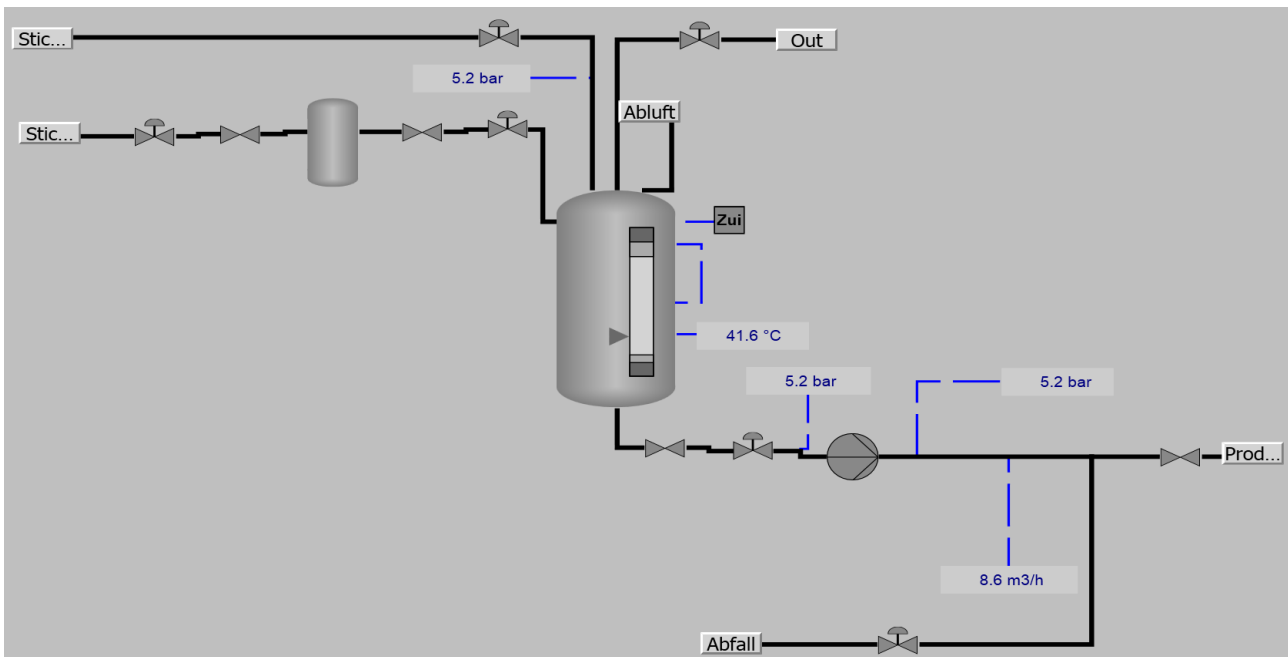


Bild 10. Bedienbild mit dynamischen Werten in der PFE, importiert aus einem MTP

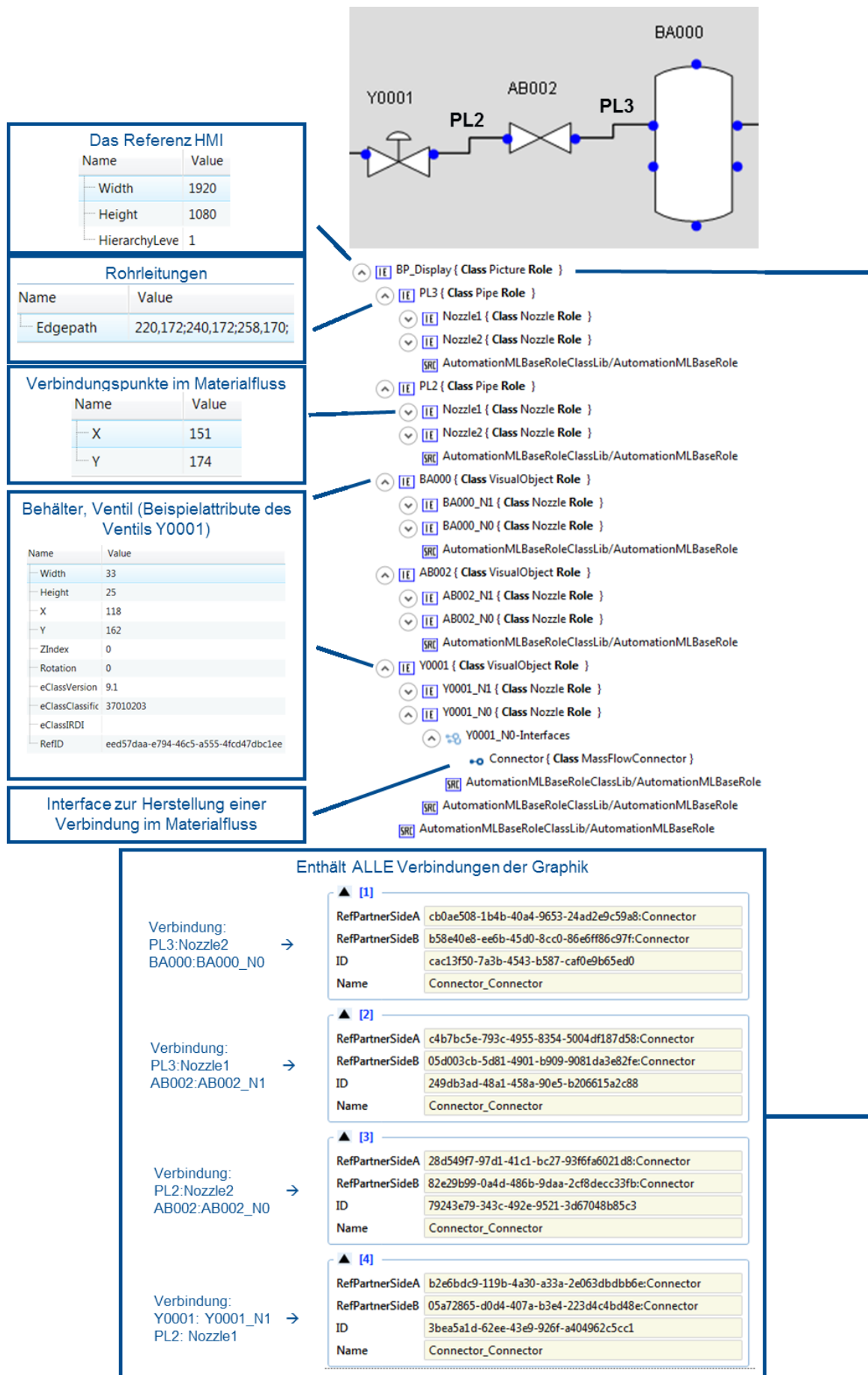


Bild 11. Modellierungsbeispiel des Beispielbedienbilds (Ausschnitt)

Unrestricted

Schrifttum

Technische Regeln

IEC 62714*CEI 62714 Engineering data exchange format for use in industrial automation systems engineering (Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme). Berlin: VDE Verlag

NAMUR NE 148:2013-10 Anforderungen an die Automatisierungstechnik durch die Modularisierung verfahrenstechnischer Anlagen (Automation Requirements relating to Modularisation of Process Plants). Leverkusen: NAMUR – Interessengemeinschaft Automatisierungstechnik der Prozessindustrie

NAMUR NE 150:2014-10 Standardisierte NAMUR-Schnittstelle zum Austausch von Engineering-Daten zwischen CAE-System und PCS-Engineering-Werkzeugen (Standardised NAMUR-Interface for Exchange of Engineering-Data between CAE-System and PCS Engineering Tools). Leverkusen: NAMUR – Interessengemeinschaft Automatisierungstechnik der Prozessindustrie

VDI 1000:2017-02 VDI-Richtlinienarbeit; Grundsätze und Anleitungen (VDI Standardisation Work; Principles and procedures). Berlin: Beuth Verlag

VDI/VDE/NAMUR 2658 Blatt 1:2017-06 (Entwurf) Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie; Allgemeines Konzept und Schnittstellen. (Automation engineering of modular systems in the process industry; General concept and interfaces). Berlin: Beuth Verlag

VDI/VDE/NAMUR 2658 Blatt 3 Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie;

Bibliothek für Datenobjekte (Automation engineering of modular systems in the process industry; Library for data objects) (in Vorbereitung)

VDI/VDE 3697 Blatt 1:2017-06 Empfehlung zur technischen Umsetzung des Datenaustauschs zwischen den Engineering-Systemen für PLT und PCS; Datenaustausch von PLT-Stellen gemäß NE 150 mit AutomationML (Recommendation for the technical implementation of data exchange between engineering systems for PCE and PCS; Data exchange between PCS objects in accordance with NE 150 using AutomationML). Berlin: Beuth Verlag

Literatur

- [1] *Urbas, L.; Bleuel, S.* et al: Automatisierung von Prozessmodulen; Von Package-Unit-Integration zu modularen Anlagen. atp edition. 54(1-2)44-52, 2012.
- [2] *Hoernicke, M.; Messinger, C.; Arroyo, E.; Fay, A.*: Topologiemodelle in AutomationML; Objektorientierte Grundlage für die Automatisierung der Automatisierung. atp edition. 58(5):28-41, 2016.
- [3] eCI@ss: <https://www.eClass.eu/> (zuletzt abgerufen am 09. November 2017)
- [4] White Paper: AutomationML and eCI@ss integration. Nov. 2015.
https://www.automationml.org/o.red/uploads/dateien/1448438009-20151030_WP_AutomationML_and_eClass_integration_v1.0_neu.pdf
(zuletzt abgerufen am 09. November 2017)