Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

# DEXPI process: Standardizing interoperable information for process design and analysis

David B. Cameron [a,*], Wilhelm Otten [b], Heiner Temmen [c], Monica Hole [d], Gregor Tolksdorf [e]

[a] University of Oslo, SIRIUS Centre, Department of Informatics, Gaustadalléen 23B, 0373, Oslo, Norway
[b] WOtten-Consulting, Nussbaumweg 21A, D-64839, Münster (Hessen), Germany
[c] DEXPI, Fürst-Salm-Straße 14, D-46414, Rhede, Germany
[d] Aibel AS, Hagaløkkveien, Asker, Norway
[e] Evonik Operations GmbH, Technology & Infrastructure, Paul-Baumann Straße 1, 45772 Marl, Germany

A R T I C L E I N F O

*Keywords:*
DEXPI
Digitalization
Interoperability
Process design
Standards

A B S T R A C T

DEXPI Process is a proposed standard for modelling information about process design, as it is presented on block flow and process flow diagrams. It was developed by the DEXPI+ working group and builds upon the DEXPI (Data Exchange in the Process Industry) standard for piping and instrumentation diagrams. Digitalization is making increasing demands on the exchange of information in the process facility lifecycle. Industry 4.0 methods require shared terminology and knowledge models to exchange information. Standards, such as ISO15926, CFIHOS and DEXPI, try to address this need. All these focus on the physical plant items, as shown on a PID or 3D model. There is a lack of standards for early-phase, top-down process design. DEXPI Process fills this gap. This paper presents the development of DEXPI Process in the context of knowledge modelling of process systems and previews how the model is a foundation for applications of automated reasoning and decision support for design and operations.

*Abbreviations*

| | |
|---|---|
| AAS | Asset Administration Shell. |
| API | Application Programming Interface |
| BFD | Block Flow Diagram. |
| CAE | Computer-Aided Engineering. |
| CAD | Computer-Aided Drafting. |
| CFIHOS | Capital Facilities Information Hand-Over Standards |
| DEXPI | Data Exchange in the Process Industry. |
| EPC | Engineering, Procurement and Construction. |
| IMF | Information Modelling Framework. |
| MDG | Model-Driven Generation. |
| MTP | Module Type Package. |
| NAMUR | Normenarbeitsgemeinschaft für Meß- und Regeltechnik in der chemischen Industrie. |
| OWL | Web Ontology Language. |
| PID | Piping & Instrumentation Diagram. We follow ISO 10628 in using this definition, instead of Process & Instrumentation diagram, as defined in ISO 15519. |
| PFD | Process Flow Diagram. |
| PPR | Product-Process-Resources model. |
| RDF | Resource Description Framework. |
| RDS | Reference Designation System (ISO/IEC81346). |
| STEP | Standard for the Exchange of Product model data. |
| SysML | Systems Modelling Language. |
| UML | Unified Modelling Language. |
| XMI | XML Metadata Interchange. |
| XML | Extended Mark-up Language. |

## 1. Introduction

### 1.1. Interoperability in the process industry facility lifecycle

The digitalization of the process industries depends on easy access to data. This data is spread, unfortunately, across many data sources and applications. Each application vendor and database designer, of necessity, uses their own data models and semantics. This means that it is laborious and difficult to collect information for use in optimization of design, construction, or operations. It is also hard to transfer information between applications and between different actors in the lifecycle. All

* Corresponding author.
*E-mail address:* davidbc@uio.no (D.B. Cameron).

https://doi.org/10.1016/j.compchemeng.2023.108564
Received 6 September 2023; Received in revised form 25 November 2023; Accepted 19 December 2023
Available online 23 December 2023

too much information occurs as documents, rather than machine-readable datasets. In addition, it is difficult to document requirements, trace them through the design process and verify that they are met in the built facility. Current ways of working with and managing engineering information can lead to this information about requirements, the rationale behind the design, being either lost or made inaccessible.

### 1.2. The need for the DEXPI process standard

This is not a new problem. These interoperability challenges are well known, and are the motivation for more than three decades of work on STEP standards: ISO 10303 (Nzetchou et al., 2019; Xiao et al., 2018) for manufacturing and ISO 15926 (Leal, 2005) for the process industry.

In the process industries, recent initiatives have worked on simplification of ISO 15926 for specific use cases. Thus, the CFIHOS initiative (https://www.jip36-cfihos.org) has created a data model, with semantic reference data, for process equipment, with focus on petroleum processes. This allows the creation of machine-readable specification data sheets and the corresponding equipment data sheets. Similarly, the DEXPI initiative (https://dexpi.org) has produced a standard for representing piping & instrumentation diagrams (PID)[1] in a standard, machine-readable format. This allows PIDs to be shared between different actors, for example, contractors and clients, in a vendor-neutral format.

The current DEXPI data model has been described in reference (Wiedau et al., 2019). It represents the equipment, piping and control-system functions shown in the PID. This reference positions DEXPI within the plant design phase of the asset lifecycle, as shown in Fig. 1.

The DEXPI data model is semantic: It defines classes for equipment that build on part 4 of the ISO 15926 standard (ISO TC 184 2019). Because of this, there is a large overlap between DEXPI and CFIHOS classes. This offers immediate opportunities for consolidation, where aligned CFIHOS and DEXPI classes can be used to build a common data model of a plant. This model can then be used to generate consistent PIDs and data sheets.

In its current form, DEXPI models the *plant*: the specification of physical artefacts that form a processing facility. The PID shows symbols as placeholders for these artefacts and their topology. It is two-dimensional representation of a more-detailed, three-dimensional (3D) model. However, there is usually a one-to-one correspondence between an artefact in the PID, the 3D model, items in a main equipment list and CFIHOS specification documents.

PIDs are not the only schematic drawings used in process facility design. They are only the final form of a top-down design activity that began with conceptual design. Block Flow Diagrams (BFD) and Process Flow Diagrams (PFD) document this activity. These diagrams are important documents for design and operation. They provide engineers with a high-level, easy to grasp, overview of the process and its rationale. At present, these diagrams are usually drawings, without links between the graphical content and engineering design. They would benefit from representation as a machine-readable data model.

Here we have identified a gap in the standards. The STEP standards focus on the equipment in the as-built plant. They contain neither the concepts nor the reference data needed to support abstract, early-phase process design. For this reason, the DEXPI initiative formed DEXPI+, a working group, to define a data model for these diagrams and the underlying design activity. The data model is called DEXPI Process. This paper presents the results of this work and puts it into the context of ongoing work on knowledge representation for process facility design (ISO 2014).

In this work, we aim to support the engineering lifecycle that was identified in the ENPRO project (Wiedau et al., 2019) and the Norwegian IMF project (Cameron et al., 2022). The view of the engineering lifecycle developed in these projects is combined and summarised in Fig. 1. The ENPRO project presented a lifecycle where process design was a preliminary to plant design. The IMF work emphasised the need to keep functional requirements for process separate from requirements for equipment. If this is done, we can expect easier access to engineering data in operational contexts, as the process requirements are made available for verification and optimization of operational behaviour. Fig. 1 also shows the scope of current important standards. These will be discussed in Section 2.1. However, at this point, we see that they do not provide support for the process design activity.

### 1.3. Overview of this paper

This paper begins with an assessment of the standards landscape as applied to plant and process modelling. Here we identify limitations in previous work and select promising approaches to modelling process design knowledge. This provides the technical basis of the DEXPI Process model.

This discussion is followed by a presentation of the model itself. We describe the elements of the model and present a demonstration example of how the model applies to the Tennessee Eastman example process, as described in (Downs and Vogel, 1993).

We conclude with an exploration of how model can be used to stimulate more effective process engineering and operations. We believe that the proposed data model is not merely a standard way of representing BFDs and PFDs. It offers a graph-based, semantically informed data model. It supports better data and requirements management in the top-down, early-phase conceptual and design processes. It can simplify and automate the configuration and management of results from process simulation tools. The graph model also opens opportunities for applying automatic reasoning and graph-based data science methods to solve design, safety, and operational issues.

## 2. Functional modelling of process facilities

### 2.1. DEXPI process: a novel approach to process information management

DEXPI Process provides a novel, and we believe necessary, approach to managing engineering design information. It work builds on previous work that applied aspect-oriented modelling to the representation of BFDs and PFDs (Cameron et al., 2022). This previous work used the SysML modelling language (Hernandez et al., 2016), a dialect of the UML modelling language (Rumbaugh et al., 2004) adapted to systems engineering, to create system objects that represented the process steps shown on these diagrams. We follow this work in distinguishing between the **process** in a facility and the **plant** that performs the process. BFDs and PFDs represent process design, not plant design. This also means that the linkage between objects on a BFD or PFD to a plant item, as shown in a PID, is indirect.

### 2.2. What do objects on block flow diagrams and process flow diagrams represent?

As noted above, the BFD and PFD are important design documents for a facility. They give process engineers and operators an overview of the logical structure and functional behaviour of the facility. However, the existing international standards for these diagrams: ISO 10628 (ISO 2014) (ISO 2012) and ISO 15519 (ISO 2010), are unclear on what a symbol in a BFD or PFD represents.

ISO 10628 (§4.3) says that a symbol represents equipment, and the

[1] Standards differ on the use of the name *Piping* & Instrumentation Diagram or *Process* & Instrumentation Diagram. ISO 15519 (ISO 2010) specifies the latter whereas ISO 15926-4 (RDS329894), CFIHOS (700292) and ISO 10628 (ISO 2014) specify the former. Given the argument in this paper, we believe that Piping & Instrumentation Diagram is the best term for these drawings, as they are concerned with the plant structure, not the process.

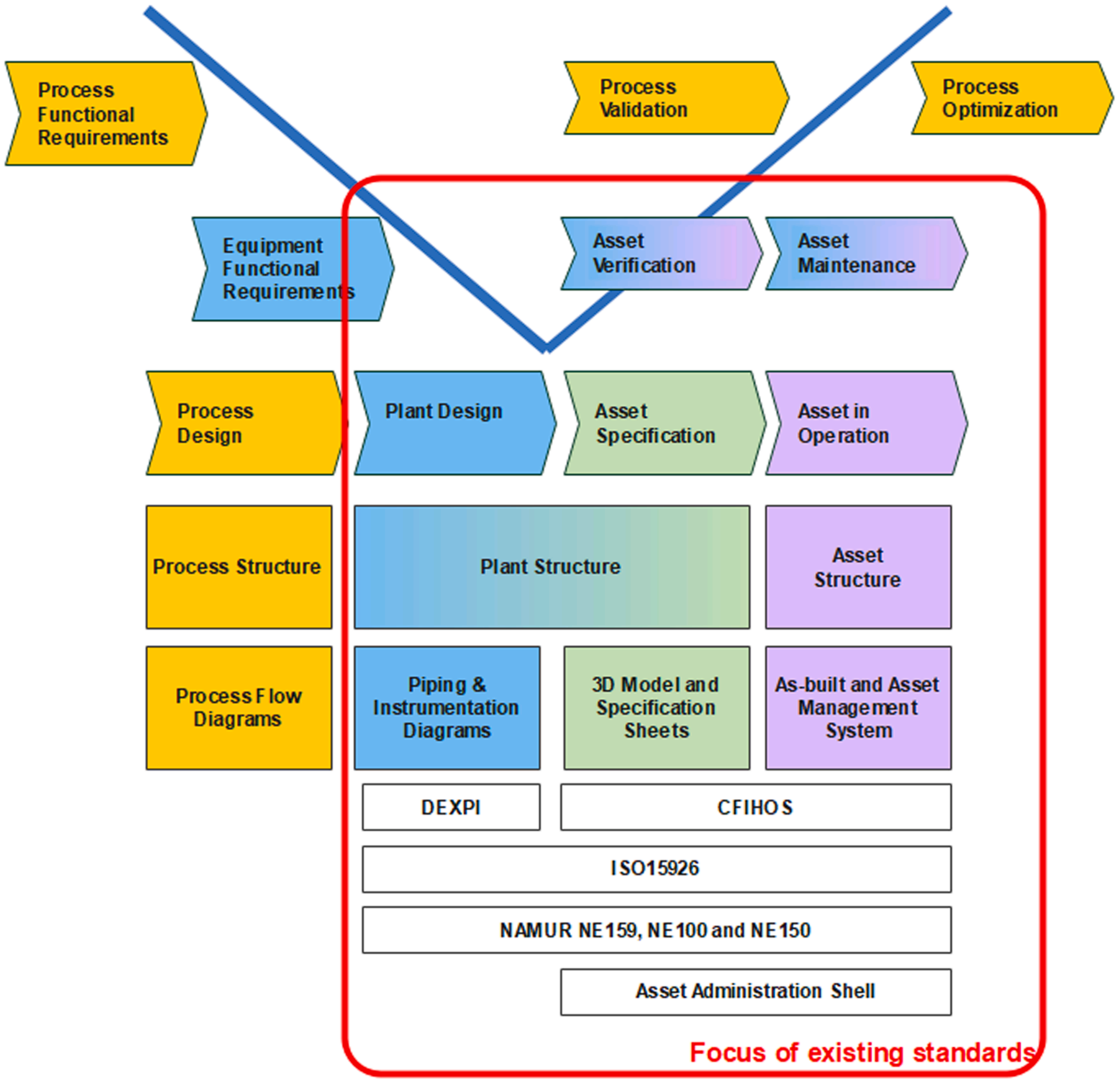

Process Functional Requirements
Process Validation
Process Optimization
Equipment Functional Requirements
Asset Verification
Asset Maintenance
Process Design
Plant Design
Asset Specification
Asset in Operation
Process Structure
Plant Structure
Asset Structure
Process Flow Diagrams
Piping & Instrumentation Diagrams
3D Model and Specification Sheets
As-built and Asset Management System
DEXPI
CFIHOS
ISO15926
NAMUR NE159, NE100 and NE150
Asset Administration Shell
Focus of existing standards

**Fig. 1.** An overview of the asset lifecycle, adapted from (Wiedau et al., 2019) with the V-model presented in (Cameron et al., 2022).

lines represent flows of mass, energy, or energy carriers. ISO 15519 (ISO 2010) defines a PFD to be a “diagram illustrating the configuration of a process system or process plant by means of graphical symbols”. This standard has the advantage of being informed by the systems thinking and aspects described in ISO/IEC 81346–1 (IEC 2022).

ISO 10628 is unclear on what the BFD or PFD represents. A block can represent process steps, unit operations or equipment. A symbol in the PFD represents equipment. Both ISO 10628 and ISO 15519 state that the BFD is a representation of the process [system] or process plant. This ambiguity between process and plant is common and natural. For example, a “distillation column” symbol is used on a PFD. Does it represent the distillation equipment or the distilling process?

We argue here that objects on a BFD and a PFD represent **process systems**, and only represent equipment indirectly. Thus, the “distillation column” symbol on a PFD always represents a *distilling process*. Here we have an object that is viewed through what ISO/IEC 81346 calls the **function aspect**: what the object does. The standards, however, agree about the meaning of the lines on the diagrams. They represent flows of mass/material (also called streams) or energy. We supplement this with flows of information. These are needed when a PFD shows control and safety functions.

Here we return to a fundamental concept in chemical engineering: the unit operation. This idea, introduced by Arthur D. Little in 1916, transformed the design process by abstracting processes from the equipment that perform the process (Flavell-White, 2011). It is worth quoting his definition: “Any chemical process … may be resolved into a coordinate series of what may be termed ‘unit operations’, as pulverising, dyeing, roasting, crystallising, filtering, evaporation, electrolysing and so on. The number of these basic unit operations is not large and relatively few of them are involved in any particular process. The complexity of chemical engineering results from the variety of conditions as to temperature, pressure etc., under which the unit operations must be carried out in different processes, and from the limitations as to material of construction and design of apparatus imposed by the physical and chemical character of the reacting substances.”

This description describes the essence of DEXPI Process. We define a small number of process steps and then provide a set of parameters that allow the designer to create and maintain structured information about the variety of conditions and constraints that determine the design of the plant.

## 2.3. Background in existing standards

In this section we will examine some existing approaches to modelling processes and plants. This expands on the perspective and review given by Wiedau et al. (Wiedau et al., 2021). The review examines the standards listed in Fig. 1: CFIHOS, ISO 15926, NAMUR, and the Asset Administration Shell. In addition, we discuss a promising German standard: VDI/VDE 3682. We also examine another formalism for representing process design, namely the SFILES language (Vogel et al., 2022).

### 2.3.1. *CFIHOS*

The CFIHOS and DEXPI data models have concentrated on plant items: equipment and piping. Thus, the core data item in CHIFOS is a tagged plant item (CFIHOS 2023), represented by a **TAG** object and an **EQUIPMENT** object. The **TAG** object contains the specifications and requirements for a plant item. The **EQUIPMENT** object contains the properties of the equipment that meets these specifications. Both **TAG** and **EQUIPMENT** objects are typed by **TAG CLASS** and **EQUIPMENT CLASS** definitions. Thus, a **TAG** with type *alternating current generator* will be fulfilled by an **EQUIPMENT** object the same type. Other, more complex fulfilment patterns are possible. Several pieces of **EQUIPMENT** can fulfil the specifications of a single **TAG,** or a more specific type of **EQUIPMENT** can fulfil the specifications of a more general **TAG**. For example, CFIHOS allows that a blow down valve **TAG** can be fulfilled by **EQUIPMENT** with type ball control valve, butterfly control valve, gate control valve, globe control valve, or plug control valve. Note that the **TAG** and **EQUIPMENT** classes are identified by nouns, the names of physical artefacts.

An extract of relevant parts of the CFIHOS data model is shown in Fig. 2.

The data objects in CFIHOS reflect aspects of the design. Thus, an **AREA** or **SITE** models a geographical location whereas a **PLANT, CONSTRUCTION ASSEMBLY** or **TAG** models a physical artefact. This allows us to organize the **TAG** objects in a model into aspect systems.

The **PROCESS UNIT** object in CFIHOS is relevant for DEXPI Process, as it corresponds to a grouping of process functions, as shown on a BFD or PFD. Traceability between a **TAG** and the **PROCESS UNIT** is obtained by assigning a collection of **TAG** object identifiers to a **PROCESS UNIT**. However, the **PROCESS UNIT** object functions merely as metadata for a **TAG**.

The CFIHOS data model also defines a set of process-oriented classes that allow a **TAG** to be related to a **PROCESS ACTIVITY** and a **PROCESS STREAM**. The model also provides mechanisms for linking properties of a **PROCESS ACTIVITY** or **PROCESS STREAM** to the property of a **TAG**. It is not possible to model a BFD or PFD using these objects. In addition, the current version of CFIHOS does not provide reference data for classes of **PROCESS ACTIVITY** or **PROCESS STREAM**. We will return to how DEXPI Process relates to this model in Section 6.3.1.

### 2.3.2. *DEXPI and ISO 15926*

Similarly, ISO 15296 and its reference data has concentrated on plant items. Thus, Kim et al. (Kim et al., 2017; Kim et al., 2020) explore the use of ISO 15296 to exchange plant 3D CAD data and integrate engineering data with maintenance data. Their modelling is totally focused on the plant item, represented as a functional physical object. A system in this approach is then, merely, an assembly of physical objects.

A plant item is represented by three “anchor” objects: a main object, an object that represents the function of the item (cf. the CFIHOS **TAG** object) and an object that represents the physical aspect of the item (cf. the CFIHOS **EQUIPMENT** object). Representations of the plant item: symbols on a PID, renderings in a 3D model and specification sheets are representations of so-called “temporal parts” of these anchor objects.

This model works where we are specifying and representing a single plant item. However, it becomes unwieldy when we are dealing with early-phase functional design. Here, the tight alignment between function and physical realization limits our ability to model abstract functional systems. The functional object in this model is a placeholder for a diverse set of process and equipment requirements for the plant item. It

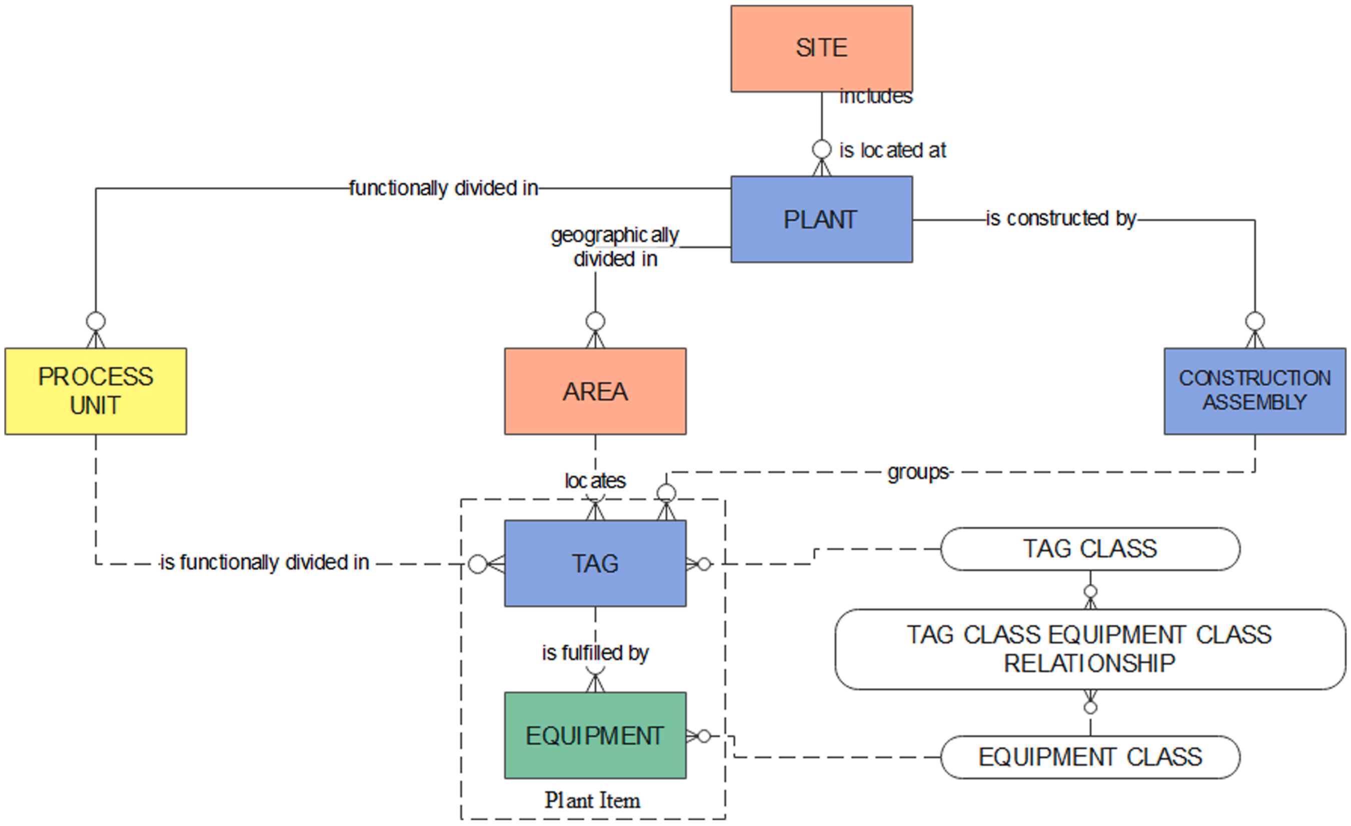


**Fig. 2.** Part of the CFIHOS data model (CFIHOS 2023), showing the place of TAG and EQUIPMENT objects. The objects are coloured to indicate the aspect used to classify the object: blue is a specified physical object, yellow is a functional aspect, red is a location aspect and green is a supplied physical object.

does not correspond to a process step in our model or the ENPRO model.

The existing DEXPI standard for PIDs shares reference data with ISO 15926–4 and CFIHOS. The DEXPI standard defines a set of plant items (or physical objects) and their topological arrangement. A DEXPI object can contain values of design parameters that are typically shown with the symbol on a PID.

### 2.3.3. NAMUR

The NAMUR organization (https://www.namur.net/en/index.html) develops standards for automation in the chemical industries. Three of these standards of direct relevance to this work:

- NE100. Lists of Properties and their Use in Process Control Engineering Workflows (NAMUR 2021).
- NE150 Standardised NAMUR-Interface for Exchange of Engineering-Data between CAE-System and PCS Engineering Tools (NAMUR 2014).
- NE159. Standardized NAMUR interface for data exchange between CAE systems for Process Design and CAE systems for PCT Hardware Planning (NAMUR 2018).

The first two, NE100 and NE150, define standard property names and interfaces to support interoperability of instrumentation specifications and automation system configurations between engineering and automation tools. The information described and transferred relates to instruments, control system artefacts and configuration blocks in the automation system.

NE159 provides a data model for transferring information about the constraints imposed by *process* and plant design from an engineering system into the system that specifies components in the control system. This transfer is done for a specific actuator or valve or sensor. The process data needed is sorted in four categories (NAMUR 2018), p7:

1. General properties of process and medium, e.g., the medium and its difficult or dangerous properties, its general state (solid, liquid, or gaseous), pressure and temperature.
2. Process-related design constraints and requirements, for example upper limit and lower limit pressure and temperature.
3. Operation case data: volume flow, density, viscosity, and composition for one or more operational cases.
4. Piping and location data, for example, the piping class, the nominal diameter, and nominal maximum pressure specification of piping.

NE159 provides an XML schema for transferring this data. Data in category four can be extracted from DEXPI data models, as it relates to plant items: piping artefacts, apparatus, and machines. However, data in categories 1 to 3 are related to process design and could be provided using the DEXPI Process model. At present, however, use of NE159 requires the use of a custom exporter for each engineering database.

A further relevant initiative of NAMUR is the Module Type Package (MTP) standard (VDI/VDE/NAMUR 2658), as described in (Tauchnitz, 2022). MTP defines a concept for building up a process facility from self-contained modules of process equipment, safety functions and automation. The facility is then built by connecting and orchestrating these modules. This break-down of complex processes into simpler process modules is like the top-down design that we are trying to support.

### 2.3.4. Asset administration shell

The last standard shown in Fig. 1 is the Asset Administration Shell (Wagner et al., 2017; Grüner et al., 2023) (AAS). This structures information around *assets*: entities “owned by or under the custodial duties of an organization, having either a perceived or actual value to the organization.” (Plattform Industrie 4.0, 2023). In practice, the entities that have been provided with AAS data have been manufactured artefacts or documents. However, we see no limitations on using AAS to represent data about more abstract assets like a definition of a process step in an early-phase design.

The hierarchical, object-oriented data model used in the AAS could be used as an implementation of the DEXPI Process data model. Each object type in the model would then have an AAS definition and a model would be built by configuring and parameterising these AAS objects. AAS could also be used to manage metadata, where a DEXPI Process data model is embedded in an AAS as a blob of data in some other format, such as XML or AutomationML. In this case, the AAS acts as an envelope for the data, ensuring that a vendor and purchaser use consistent metadata about a model that is exchanged.

### 2.3.5. VDI/VDE 3682 formalized process modelling

Finally, a German standard, VDI/VDE 3682 (VDI/VDE 2015a) (VDI/VDE 2015b), presents a formal, graph-based model for chemical processes. The model is built around a **Process Operator** object that processes **Product, Energy,** and **Information** objects, as shown in Fig. 3. A **Technical Resource** can then realize the **Process Operator**.

VDI/VDE supports hierarchical design, where a simpler, high-level system is decomposed into more complex, lower-level systems. This is done by aligning the **Product, Energy,** and **Information** nodes on system boundaries at the higher level with the lower level.

As the reader will see below, the data model we propose has many similarities with the VDI/VDE 3682 model. However, before we detail the model, we need to discuss how a process model relates to the objects on a BFD or PFD.

### 2.3.6. SFILES

SFILES is a text-based notation for process flowsheets (Vogel et al., 2022). It provides a compact, textual representation of the structure of a PFD. Process steps are represented by strings and connections are implied by the sequence of process steps, grouping, symbols for converging and diverging branches, and identifiers for nodes. The SFILES representation corresponds to a graph where the nodes are process steps, and the arcs are process connections.

The authors proposed a set of types for process steps, based on the OntoCAPE ontology. Our taxonomy is richer. In the presentation of the DEXPI Process taxonomy in section we have indicated the mapping to SFILES for our types.

The SFILES text representation is compact and cryptic. It is designed for machine use rather than human interaction. It models only the topology of the process, not the properties of process steps.

## 2.4. A systems approach to design

The information model required to support BFDs and PFDs needs to model the process performed in a facility, as distinct from the plant, which is the realization of the process by means of equipment. In designing DEXPI Process, we have striven to build on and integrate existing standards and best practices. The approach builds on the work described in reference (Cameron et al., 2022), which presented an experimental SysML modelling framework, where the process modelling elements provided a foundation for the model.

DEXPI Process is built using systems engineering concepts. Thus, each element in a BFD or PFD is represented by a system block, with defined inputs and outputs. Here we draw upon modelling languages such as IDEFx (IEEE Computer Society 1998) and SysML (Hernandez et al., 2016). We draw on a small sub-set of SysML, namely the concept of blocks with ports.

We draw on another systems engineering perspective, represented by the ISO/IEC 81346 family of standards (IEC 2022; Balslev, 2020; Balslev and Barré, 2022). Here, complex systems are broken down into constituent elements along different *aspect trees*. The standards describes three primary aspects: function, product, and location. The primary focus of ISO/IEC 81346 is providing a reference designation system (RDS), i.e., a coding standard that identifies objects in the system

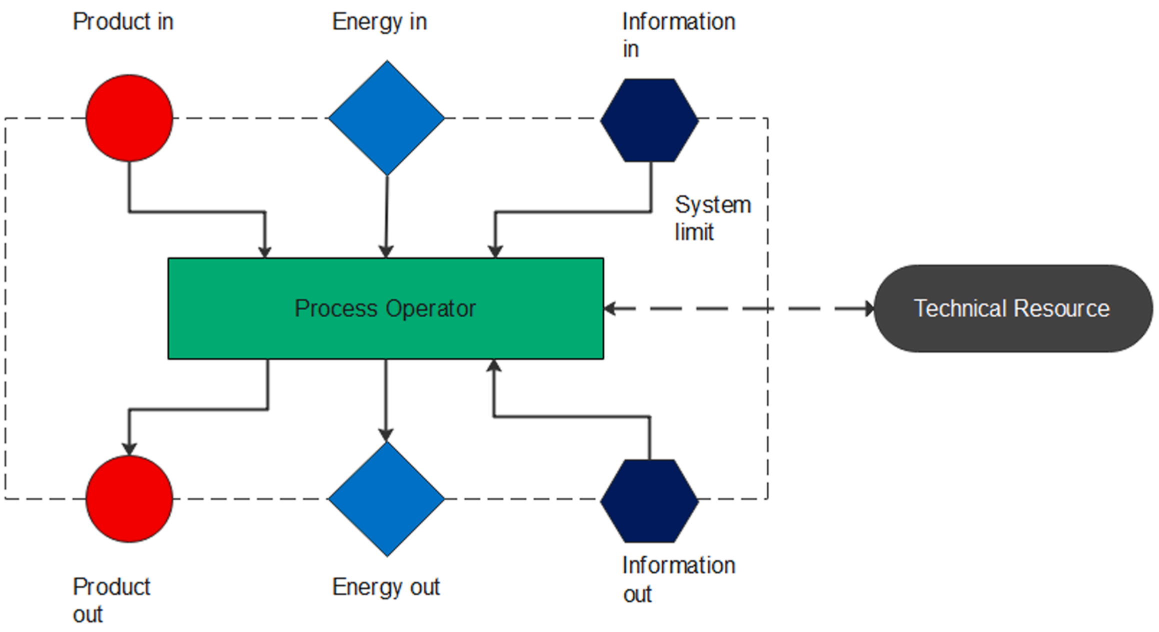


**Fig. 3.** The Process Operator data model in VDI/VDE 3682.

breakdown by type and aspect. ISO/IEC 81346–2 (IEC 2019) provides reference data and a set of RDS codes for component systems. These codes have been adapted for power systems in part 10 of the standard (ISO 2022) and for oil & gas (READI, 2021). Both coding sets differentiate between technical systems, with two-letter RDS codes, defined in the domain-specific standards, and component systems, with three letter codes, and defined in ISO/IEC 81346–2. There is no linkage or inheritance between these codes. The German standard DIN 6779–13 (DIN 2018) provided an RDS for the process industries, which proposed a set of two-letter codes, selected from the component system table in ISO/-IEC 81346–2. By doing this, they provide a set of technical systems that can, if desired, be specialized through inheritance. For example, a pump in DIN 6779–13 will have a `-GP` reference designation. This would apply irrespective of type of pump. However, we can choose to use the component system reference `-GPB`, if the pump is a centrifugal pump, and `-GPA`, if it was a reciprocating pump. Similarly, we can use an `=GP` reference to identify a DEXPI Process `Pumping` block or an `=GPB` reference to identify a `Pumping` block with its `Method` property set to `CentrifugalMotion`.

### 2.5. *Aspect models*

The key insight in the development of DEXPI Process is that a PFD and a PID describe different *aspects* of the processing system. The idea of aspect systems is a foundation for ISO/IEC 81346 and is also the basis of the OntoCAPE knowledge model (Marquardt, 2010) for process systems. An aspect model represents a specific viewpoint – a way of analysing a technical system.

OntoCAPE views a planned or actual facility as an abstract chemical process system. This system can be viewed using several aspects, including requirements, function, realization, and behaviour, as shown in Fig. 4. It differentiates between a **Plant** that represents the physical realization of the system from the **Process** that represents the function of the system.

We follow the OntoCAPE approach by defining a separate package in the DEXPI model, called **Process**, and by using a base class called **ProcessStep** for all classes that represent unit operations or process blocks. Note also that our model contains elements that support both the requirements and behaviour aspects.

Cheng and Ma (Cheng and Ma, 2017) presented the functional

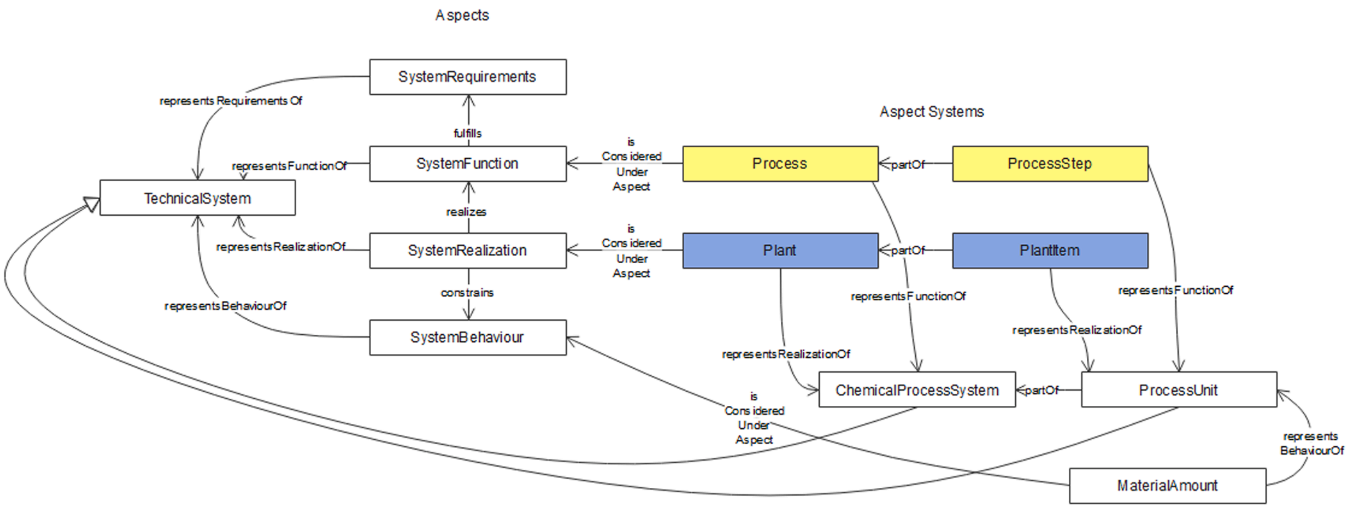


**Fig. 4.** Modelling of aspect systems in OntoCAPE.

feature modelling cube as another way of understanding the interaction between function, structure and behaviour in engineering design. This is adapted to process design in Fig. 5.

The model shown in Fig. 5 complements OntoCAPE. It also makes it clear that we need three sets of semantic reference data to model our facilities. Firstly, we need reference data for the physical objects, the **plant items**, in the plant. This is well-defined in ISO 15926, CFIHOS and DEXPI. Secondly, we need data about **physical quantities and phenomena** that describe processes, plants, and their behaviour. Finally, we need reference data for **process steps**. This has been lacking until now.

### 2.6. *Summary: overview of process and plant modelling approaches*

Table 1 summarizes our overview of the standard approaches to modelling and interoperability of process and plant data.

## 3. The DEXPI process data model

### 3.1. *Overview of the model*

The DEXPI Process data model is shown in Fig. 6. This shows the main classes in the model and how they are related. The section in which the classes are described is indicated on the drawing. The model has been developed using the UML modelling language (Rumbaugh et al., 2004). We did this because the existing DEXPI standard uses UML. However, the reader will see that the model is a graph model that can be expressed as RDL. We have found that UML has been an effective tool for developing and validating the model. However, it is unsuitable as a format for use in interoperability and engineering applications. For this reason, we have provided implementations of the model as an XML schema and an AutomationML library. These are described in more detail in Section 4.

### 3.2. *Process steps and unit operations*

#### 3.2.1. *The processstep class*

The most important class in the model is the **ProcessStep** class. This is the base class for all nodal elements in a BFD or PFD. The term process step is taken from the OntoCAPE semantic model (Marquardt et al., 2010). Here a process step is a system that performs an activity that is part of a process. It focuses on function. The **ProcessStep** is a block data structure that can be implemented as an object-oriented datatype, UML block, SysML block or an AutomationML object.

Each **ProcessStep** can own one or more **Port** objects. A **Port** is a logical point at which the **ProcessStep** exchanges material, energy, and information with another **ProcessStep**. A **Port** has a nominal direction (**Inlet** or **Outlet**). The **Port** class is abstract, so that a port needs to be of a specific type: a **MaterialPort**, an **InformationPort**, an **ElectricalEnergyPort**, a **MechanicalEnergyPort** or a **ThermalEnergyPort**. An outlet port is connected to an inlet port by a **ProcessConnection** object. See Table 2.

An important feature of DEXPI Process is that we can build our process model hierarchically, where a higher-level process step acts a frame for a more detailed model. It is therefore necessary to align a port in the more detailed model with a port in the process step higher in the hierarchy. This is supported by **SubReference** and **SuperReference** properties in each port. This alignment allows tracing of constraints, requirements, and process properties up and down the process hierarchy.

#### 3.2.2. *A taxonomy of process steps*

DEXPI Process defines a hierarchical taxonomy of process steps and unit operations. This is based on existing standards and taxonomies and, we believe, can represent the processes in most process facilities. The taxonomy has up to three layers, with inheritance between the layers. In the top layer, the classes are typed by processing activities: single verbs

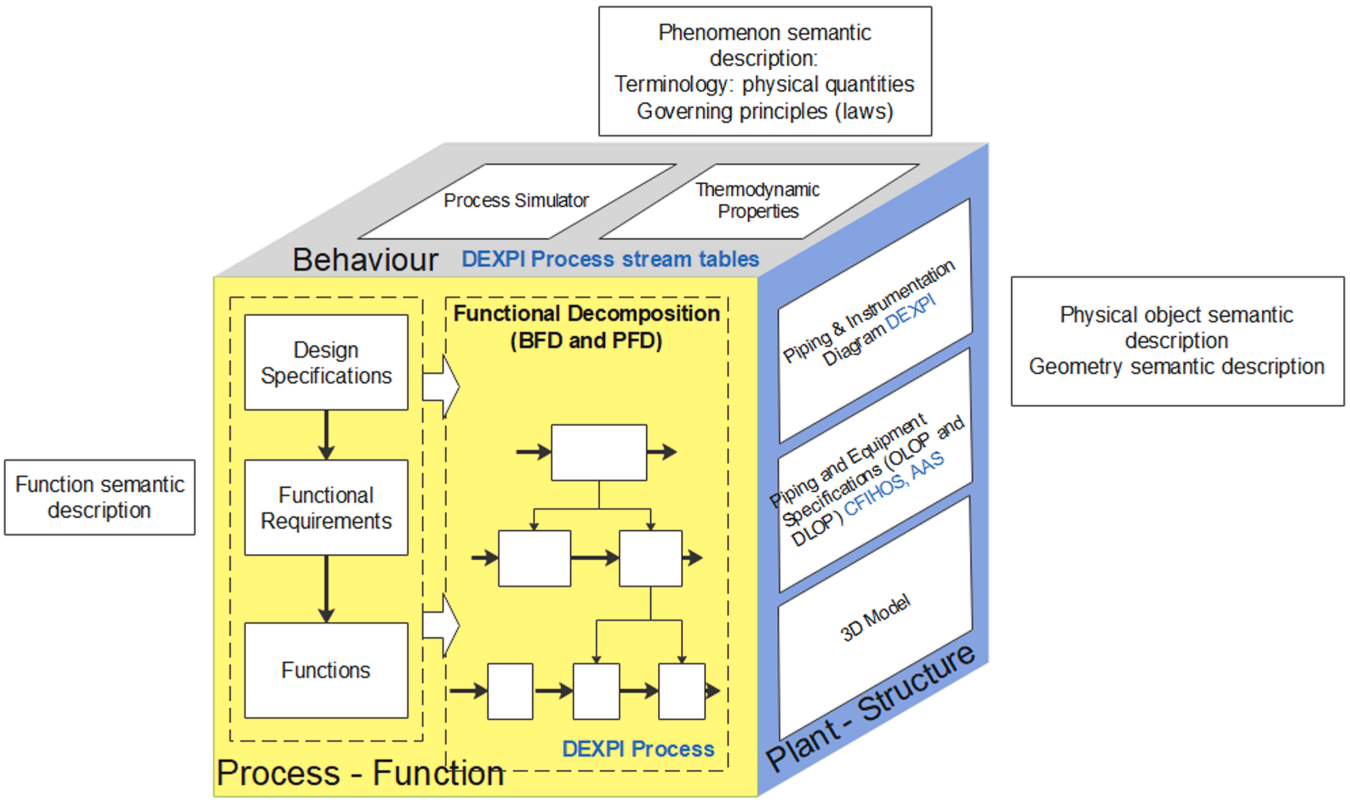

Phenomenon semantic description: Terminology: physical quantities Governing principles (laws)
Process Simulator
Thermodynamic Properties
Behaviour
DEXPI Process stream tables
Design Specifications
Functional Requirements
Functions
Functional Decomposition (BFD and PFD)
DEXPI Process
Process - Function
Piping & Instrumentation Diagram DEXPI
Piping and Equipment Specifications (OLOP and DLOP) CFIHOS, AAS
3D Model
Plant - Structure
Function semantic description
Physical object semantic description Geometry semantic description

**Fig. 5.** The Functional Feature Modelling cube for process systems design, adapted from (Cheng and Ma, 2017). DEXPI Process models the process, the functional decomposition of the system. The PID models the plant, the physical structure of the system.

**Table 1**
Summary of process and plant modelling approaches.

| Approach | Modelling focus and assumptions | Evaluation |
|---|---|---|
| ISO15926 | Primary focus is on physical plant items, with process and function tightly coupled to the plant item. | Provides reference data (classes) for CFIHOS and DEXPI. |
| CFIHOS | Current implementation is a data model for specified and supplied properties of a plant item. | Provides a taxonomy of plant items needed to build chemical facilities. |
| DEXPI | Provides a data model for the plant as shown on a PID. | Taxonomy of plant items overlaps with CFIHOS. |
| NAMUR | NE100 and NE150 focus on plant items for automation. NE159 links plant items to process design information. MTP defines modular plant items. | Process steps in DEXPI Process can be realized with MTP modules. |
| VDI/VDE 3682 | Focuses on process, with inputs and outputs of material, information, and energy. | Data model is consistent with DEXPI Process. The standard does not provide a taxonomy of processes |
| SFILES | Focuses on process topology and representation of the process as a graph. | Concise tool for serializing and analysing process structures. |
| ISO/ IEC81346 | Taxonomy and reference designation for aspect modelling of systems. The model has a weak semantic basis but can be aligned with stricter taxonomies such as CFIHOS or DEXPI. | Function and product aspects are used in this work. DEXPI Process is a model in the functional aspect. |

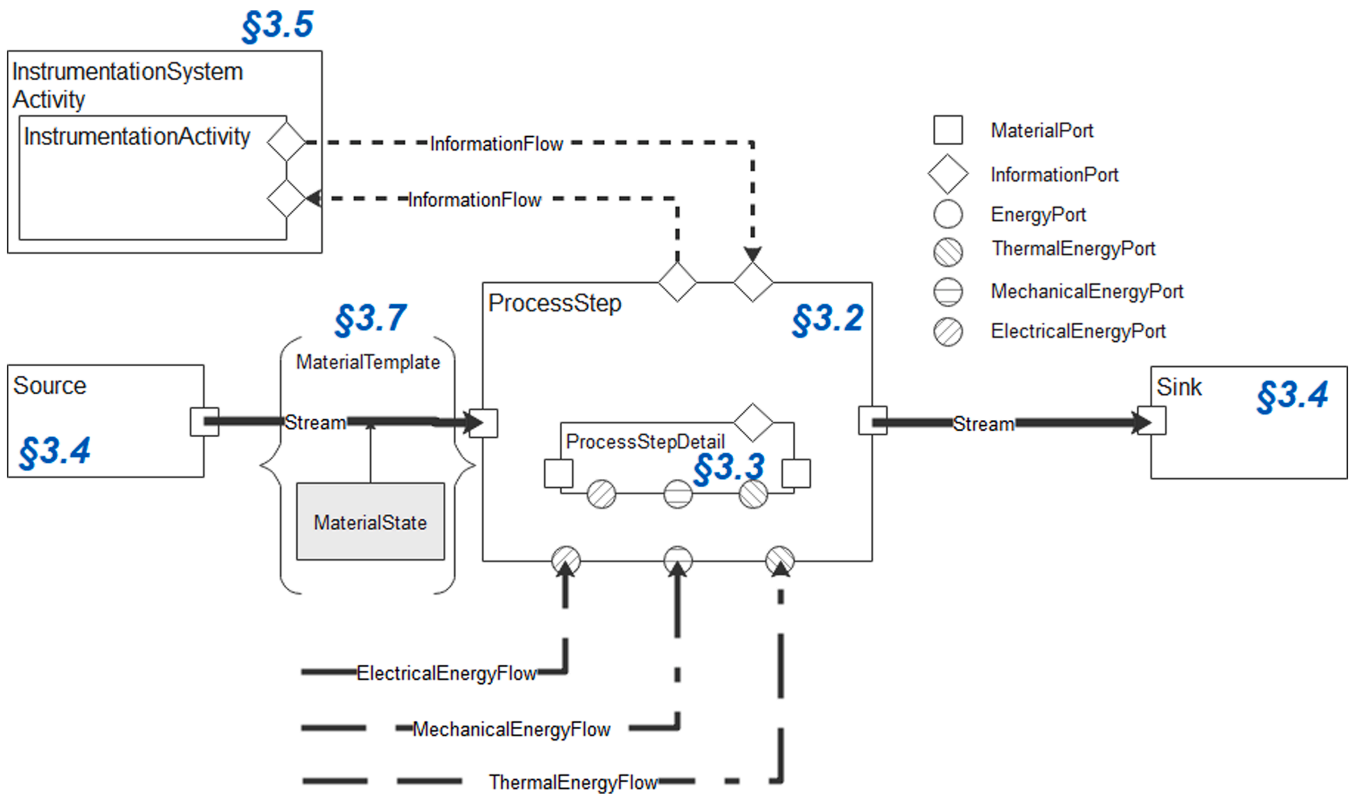


**Fig. 6.** The classes in the DEXPI Process Data Model, showing references to the section where they are described.

**Table 2**
Connections and ports.

| Type of connection | **Port** class | **ProcessConnection** class |
|---|---|---|
| Material | **MaterialPort** | **Stream** |
| Information | **InformationPort** | **InformationFlow** |
| Electrical energy | **ElectricalEnergyPort** | **ElectricalEnergyFlow** |
| Thermal energy | **ThermalEnergyPort** | **ThermalEnergyFlow** |
| Mechanical energy | **MechanicalEnergyPort** | **MechanicalEnergyFlow** |

expressed using the present participle form with an object for the activity. For example: **Separating** and **TransportingFluids**.

At the lower levels, the taxonomy corresponds to the unit operations concept. Here the process step is typed by *how* the activity will be done. Here we specify the physical principle used in the process. Thus, the **Separating** activity can be specialized into a **SeparatingByThermalProcess** activity. The taxonomy here is based on the German DIN-6779 standard (DIN 2018), which, in turn, draws on the taxonomy of processes in the DDR TGL 25 000 Sheet 1 (VVB Chemieanlagen,Leipzig 1974). These middle-level activities can be specialized further according to physical principle. DEXPI Process uses class inheritance to model this hierarchy. Thus, **Distilling** is a specialization of **SeparatingByThermalProcess**, and **VacuumDistilling** is a specialization of **Distilling**.

The following tables list the **ProcessStep** classes defined, with their inheritance relations.

We start with separating processes, shown in Table 3. Here we model with three levels of detail, following the schema in DIN-6779–13 (DIN 2018). The separation processes are classified according to the physical phenomena that are used to achieve separation. At the third level we can see separating processes that correspond to the common separation unit operations.

We define three top-level processes for manipulating the thermal energy content of a material, see Table 4. We can either supply, remove or exchange thermal energy. We can also specify a heat exchange method for any of these blocks. Here we indicate the arrangement of heat transfer surface that will be used to realize the process.

DEXPI Process provides a library of process steps for solids

**Table 3**
Process Steps for Separating.

| Top-level activity | Middle-level activity | Unit operation |
|---|---|---|
| Separating (SFILES sep) | SeparatingByPhaseSeparation | SeparatingByGravity |
| | | SeparatingByCentrifugalForce |
| | | SeparatingByCyclonicMotion (SFILES hcycl) |
| | | SeparatingByGasLiquidSeparation |
| | | SeparatingByScrubbing (SFILES scrub) |
| | | SeparatingByCoalescing |
| | | SeparatingByFlashing (SFILES flash) |
| | SeparatingByThermalProcess | Drying |
| | | Distilling (SFILES dist) |
| | | Evaporating |
| | | StrippingDistilling |
| | | StabilizingDIstilling (SFILES rect) |
| | | VacuumDistilling |
| | SeparatingMechanically | Filtering (SFILES gfilt, lfilt) |
| | | Skimming |
| | | Sieving |
| | SeparatingByElectromagneticForce | SeparatingByElectrostaticForce (SFILES egclean) |
| | | SeparatingByMagneticForce |
| | SeparatingByPhysicalProcess | Absorbing (SFILES abs) |
| | | Adsorbing |
| | | SeparatingByIonExchange |
| | | SeparatingByContact (SFILES extr) |
| | | SeparatingBySurfaceTension |

**Table 4**
Process Steps for working with Thermal Energy.

| Top-level activity | Unit operation |
|---|---|
| ExchangingThermalEnergy (SFILES hex) | Principle given by HeatExchangeMethod attribute in block: Generic, Plate, Spiral or Tubular. |
| RemovingThermalEnergy | Cooling |
| SupplyingThermalEnergy | HeatingInFurnace |
| | Boiling |
| | GeneratingSteam |
| | Flaring |
| | HeatingElectrical |

processing, see Table 5. These are organized under three top-level processes: forming material, increasing particle size, or reducing particle size.

Process steps are defined to store fluids, solids, electrical energy, and thermal energy, see Table 6. Note that we do not differentiate between storing gases and liquids. Both are treated as fluids.

Process steps are defined to supply fluids, solids, electrical energy, and mechanical energy, see Table 7. The **SupplyingMechanicalEnergy** class is parent for all driving processes: motors, engines, and turbines.

Process steps are defined for mixing and splitting flows of material, see Table 8. Two types of process step are defined here. The **SplittingMaterial, SplittingEnergy** and **MixingSimple** steps model simple branching and converging of material and energy flows in a detailed process flow diagram, whereas the other unit operations represent processes that will be realized by specific items of equipment.

The library provides process steps for transporting material and energy, see Table 9. Note that these process steps are used where there is a substantial difference between the state of the material at the inlet and outlet of the system. These blocks are *not* used to connect other process steps. Thus, we would use a **TransportingFluidsInPiping** to model an inlet manifold system or a long pipeline transfer of material between two

**Table 5**
Process Steps for Solids Processing.

| Top-level activity | Unit Operation |
|---|---|
| FormingSolidMaterial | Extruding |
| | Pelletizing |
| IncreasingParticleSize | Agglomerating |
| | Crystallizing |
| | Flocculating |
| ReducingParticleSize | Crushing |
| | Cutting |
| | Grinding |
| | CustomMilling |

**Table 6**
Process Steps for Storing Material and Energy.

| Top-level Activity | Middle-level activity | Unit operation |
|---|---|---|
| Storing (SFILES tank) | StoringFluids | StoringInTank |
| | | StoringInPressureVessel |
| | StoringSolids | StoringInSilo |
| | StoringElectricalEnergy | StoringInBattery |
| | StoringThermalEnergy | |

**Table 7**
Process Steps for Supplying Material and Energy.

| Top-level activity | Unit operation |
|---|---|
| SupplyingFluids | |
| SupplyingSolids | |
| SupplyingElectricalEnergy | GeneratingACPower |
| | GeneratingDCPower |
| | GeneratingInFuelCell |
| | GeneratingCustom |
| SupplyingMechanicalEnergy | DrivingByMotor (principle as attribute) |
| | DrivingByEngine (principle as attribute) |
| | DrivingByTurbine (principle as attribute) (SFILES expand) |

**Table 8**
Process Steps for Mixing and Splitting.

| Top-level activity | Unit operation |
|---|---|
| Splitting (SFILES splt) | SplittingMaterial |
| | SplittingEnergy |
| Mixing (SFILES mix) | Mixing |
| | Kneading |
| | Humidifying |
| | RotaryMixing |
| | StaticMixing |
| | MixingSimple |

**Table 9**
Process Steps for Transporting Material and Energy.

| Top-level activity | Unit operation |
|---|---|
| TransportingFluids (SFILES pipe) | TransportingFluidsInPiping<br>TransportingFluidsInChannel<br>TransportingFluidsInHose |
| TransportingSolids | TransportingSolidsContinuously<br>TransportingSolidsDiscontinuously |
| TransportingElectricalEnergy | |

process steps.

We view **Pumping** and **Compressing** activities as a specialization of a **GeneratingFlow** activity, see Table 10. We note that it is possible to debate whether the function of pump or compressor is to generate flow or increase pressure. We have adopted a pragmatic approach and decided that the primary activity is to generate a flow of material. The principle used for pumping and compressing can be specified using an attribute of the block.

We complete the model with a **ReactingChemicals** class and a **Packaging** class, see Table 11.

### 3.3. Process step details

A PFD can contain elements inside a process step that are important for the process behaviour. For example, a process that performed in a column is denoted by a symbol that shows the arrangement used to bring fluids into contact, either trays or packing. Similarly, a reactor or mixer may require agitation. Here we again see the ambiguity between process and equipment. During process design, we need to indicate that the **Distilling** process uses trays or packing. Similarly, the **ReactingChemicals** process requires a linked **Agitating** process.

DEXPI Process therefore defines **ProcessStepDetail** classes. These are systems that cannot exist independently of a **ProcessStep** but that perform necessary processes in that **ProcessStep**. These classes will usually be used as part of process steps that represent unit operations in Process Flow Diagrams.

Four such classes are defined:

- **Agitating.**
- **ContactingOnTray**. This class is used to represent stages in a column-based separation process. It is functionally important that we can represent and indicate top, bottom, draw-off and feed stages in these processes.
- **ContactingInPacking**. This class is used to represent packed-bed segments in separation and reaction unit operations.
- **SupplyThermalEnergyInBurner**.

### 3.4. Sources, sinks and emitting

**Source** and **Sink** blocks model the flow of material into and out of a specific BFD or PFD. These are used to delimit drawings and plant models. They can also function as off-page connectors between PFDs. These classes correspond to the SFILES **raw** and **prod** nodes.

An **Emitting** class can be used instead of a **Sink** to document the presence of a waste or emission stream. This can simplify identifying and accounting for emissions from a facility.

**Table 10**
Generating Flow: Pumping and Compressing.

| Top-level activity | Unit operation |
|---|---|
| GeneratingFlow | Pumping (SFILES pp)<br>Compressing (SFILES comp, blwr) |

**Table 11**
Other Process Steps.

| Top-level activity | Unit operation |
|---|---|
| ReactingChemicals (SFILES r) | ReactingChemicals. Method parameter defines principle: Tubular, PackedBed, Tank, FluidizedBed, Unspecified |
| Packaging | |

### 3.5. Process instrumentation and control

#### 3.5.1. Modelling process instrumentation and control

ISO 10628–1 says that a PFD can also contain "functional demands for process measuring and control devices at important points" (ISO 2014 §4.3.3). In practice, this means that the PFD will include the supervisory control needed to run the process. For this reason, the PFD also contains "essential valves and their arrangement in the process."

A BFD and PFD are also valuable tools for designing and documenting the safety and segmentation of the facility. For this reason, we also include process steps that implement the safety design according to ISO 10418 (ISO 2019) / API RP14C (API, 2018) or ISO 23251 / API Std 521 (API 2022).

#### 3.5.2. Steering flow: flow control functions

Since the data model is process oriented, we model "valves" in a PFD by their function, using sub-classes of the **SteeringFlow** process step. The classes defined are listed in Table 12.

When showing supervisory control, we use a **RegulatingFlow** or **FeedingMaterial** block as the final control element. The **ShuttingOffFlow, PreventingBackflow, RelievingOverpressure, RelievingVacuum, RelievingVacuumAndOverpressure, BlowingDown, Draining** and **RestrictingFlow** blocks are used if the PFD is to be used to show safety system functions.

#### 3.5.3. Process instrumentation systems and activities

The "functional demands for process measuring and control devices at important points" noted above are modelled by two base classes. An **InstrumentationSystemActivity** is a high-level class that models a complete measuring and control loop function. Each instance of this will contain one or more instances of **InstrumentationActivity** classes that show the functional components of the loop.

We will describe the model using a fragment from the example detailed PFD given in ISO 10628–1 (ISO 2014), shown in Fig. 7.

The discharge from an overhead product pumping system, P1/P2, is split into two streams. A ratio controller is used to implement a reflux ratio on the distilling system. The discharge pressure from the pumping system is monitored.

**Table 12**
**ProcessStep** classes for steering flow.

| Top-level activity | Unit operation | Realizing piece of equipment |
|---|---|---|
| SteeringFlow (SFILES v) | ShuttingOffFlow | On-off isolation valve |
| | RegulatingFlow | Modulating control valve |
| | PreventingBackflow | Check or non-return valve |
| | RestrictingFlow (SFILES orif) | Flow orifice |
| | FeedingMaterial | Feeding of solid material |
| | RelievingOverpressure | Relief valve with piping |
| | RelievingVacuum | Breather valve with piping |
| | RelievingVacuumAndOverpressure | Bi-directional breather valve with piping |
| | BlowingDown | Blow-down valve with piping |
| | Draining | Drain valve with piping |

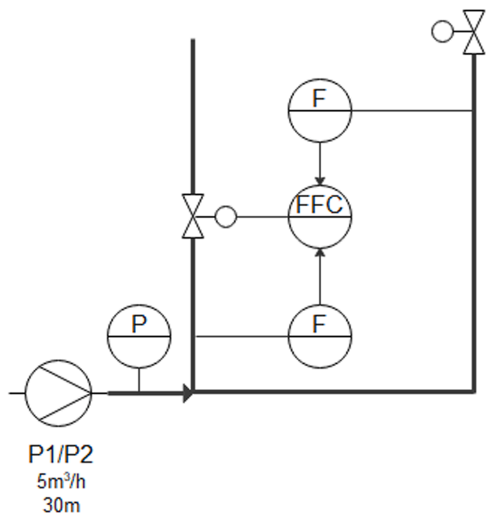


**Fig. 7.** Illustrative fragment from ISO 10628–1.

This structure is modelled using two **InstrumentationSystemActivity** blocks. The first of these represents the pressure monitoring function, while the other models the reflux control function. Each **InstrumentationSystemActivity** block contains one or more **InstrumentationActivity** blocks. The **InstrumentationActivity** classes are listed in Table 13.

The fragment shown above can then be modelled as shown in Fig. 8.

Note that a **MeasuringProcessVariable** block is linked to the process by a reference to a parameter in a **ProcessConnection**, a **ProcessStep** or a **ProcessStepDetail**. Thus, the =BP02 block presents a measurement of the Pressure parameter in the **Stream** between =GP03 and Split3.

### 3.5.6. *Alignment with DEXPI plant instrumentation modelling*

The existing DEXPI standard has a more detailed model to represent instrumentation functions in the PID. Fig. 9 shows how the DEXPI Process functions are realized by instrumentation functions and equipment.

## 3.6. *Parameters and characterizations*

Let us return to the quote from Arthur D. Little (Flavell-White, 2011). "The complexity of chemical engineering results from the variety of conditions as to temperature, pressure etc., under which the unit operations must be carried out in different processes, and from the limitations as to material of construction and design of apparatus imposed by the physical and chemical character of the reacting substances."

The art of chemical process design is taking functional requirements and using them to define a safe and optimal process that can be realized by a safe and operable plant. We document these requirements and verify compliance by making statements about – characterizations of – the process and equipment. For this reason, each DEXPI Process class defines a set of parameters that characterize that **ProcessStep, ProcessStepDetail** or **InstrumentationActivity**. We use inheritance of properties, so that a detailed **ProcessStep** can share properties with a less detailed, higher-level **ProcessStep**, as shown in Fig. 10.

Note that we assume that it is meaningful to specify temperature, pressure, ambient temperature, and ambient pressure for any **ProcessStep**. Here we also specify identification, description and labelling parameters that are common for all blocks. A **SteeringFlow** process step inherits properties from its base class. In addition, we can specify the mass flow and/or volume flow through this process. Further, if we need regulate flow, we will be interested in specifying values of pressure drop, opening time and closing time for the **RegulatingFlow** step.

The data model also allows specifications of properties of **Stream, EnergyFlow** and **InformationFlow** objects at ports. This allows the designer to reference properties and the inlet and outlet of a process step or unit operation.

Note that these parameters are *unqualified*. A DEXPI Process model states that there is pressure associated with every process step. It is up to the designer to supply specifications for, and calculate estimates of, that pressure. The designer will want to state many things about this pressure. We need to set specifications on the upper limit and lower limit design pressure. It may be necessary to specify an upper limit allowable pressure. The designer will also need to specify the expected operating pressure. We can represent each of these specifications by supplying a value of a *qualified parameter.*

A qualified parameter is a value of a parameter that contains information about what the value means.

This is done by supplying qualification information about the value. DEXPI Process provides the option of adding the properties defined in Table 14 to any parameter value.

Note that setting these properties is optional. This means that DEXPI Process can be used as to build a single document where all values supplied are interpreted as being nominal design values.

However, this approach allows the separation of the process model from its characterisations. This is done by building a model that only contains definitions of blocks and their connections. No parameter values are stored in this model. The design system can then maintain multiple data sets that contain qualified parameter values for parameters defined in the data model.

A simple implementation of this can be done using serialized DEXPI Process documents. A master document contains the structure of the process model. Additional documents supply qualified parameters for blocks in the master document. In this way, we can model a PFD with several design cases. The topology of the PFD is in the master and each case is its own document. This is shown in Fig. 11.

## 3.7. *Representation of material properties: streams and stream tables*

The PFD often displays a set of stream tables, which show the flow, state (pressure and temperature), composition and physical properties for important streams. Where there are several phases, the stream tables may also show flows, compositions, and physical properties for each phase. Representation of these properties is essential for process design, as the given properties of the feeds and desired properties of the products are key functional requirements.

Recall the Functional Feature Modelling cube in Fig. 5. The stream tables in a PFD are a snapshot of the *behaviour* of the process at a specific

**Table 13**
**InstrumentationActivity** classes.

| Top-level activity | Activity | Realizing piece of equipment |
|---|---|---|
| InstrumentationActivity | MeasuringProcessVariable | Sensor and Transmitter |
| | CalculatingProcessVariable | Calculating function |
| | ControllingProcessVariable | Controller (SFILES C) |
| | ConveyingSignal | Long or complex signal transmission. InformationFlow is usually used for signals. |
| | TransformingProcessVariable | Sub-class of CalculatingProcessVariable, arbitrary transformation of a process variable |
| | CalculatingSplitRange | Sub-class of CalculatingProcessVariable, split range block. |
| | CalculatingRatio | Sub-class of CalculatingProcessVariable, ratio block. |

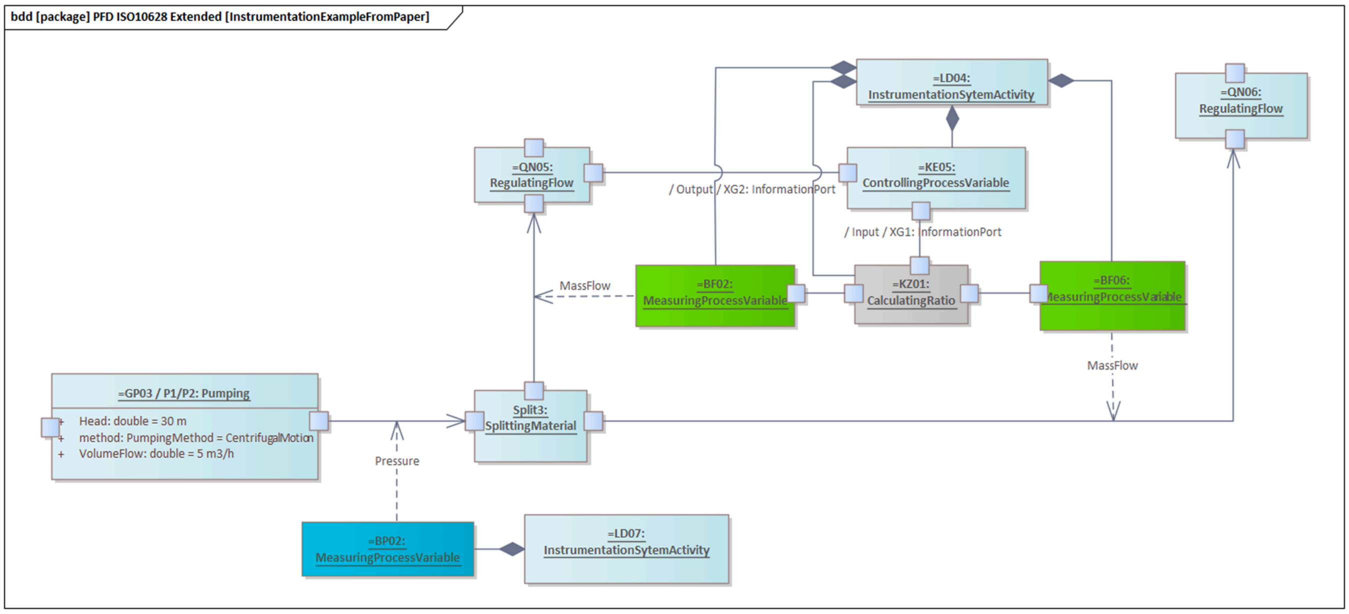

bdd [package] PFD ISO10628 Extended [InstrumentationExampleFromPaper]
=LD04: InstrumentationSytemActivity
=QN06: RegulatingFlow
=QN05: RegulatingFlow
=KE05: ControllingProcessVariable
/ Output / XG2: InformationPort
/ Input / XG1: InformationPort
=BF02: MeasuringProcessVariable
=KZ01: CalculatingRatio
=BF06: MeasuringProcessVariable
MassFlow
MassFlow
=GP03 / P1/P2: Pumping
Head: double = 30 m
method: PumpingMethod = CentrifugalMotion
VolumeFlow: double = 5 m3/h
Split3: SplittingMaterial
Pressure
=BP02: MeasuringProcessVariable
=LD07: InstrumentationSytemActivity

**Fig. 8.** Modelling the illustrative PFD fragment with instrumentation.

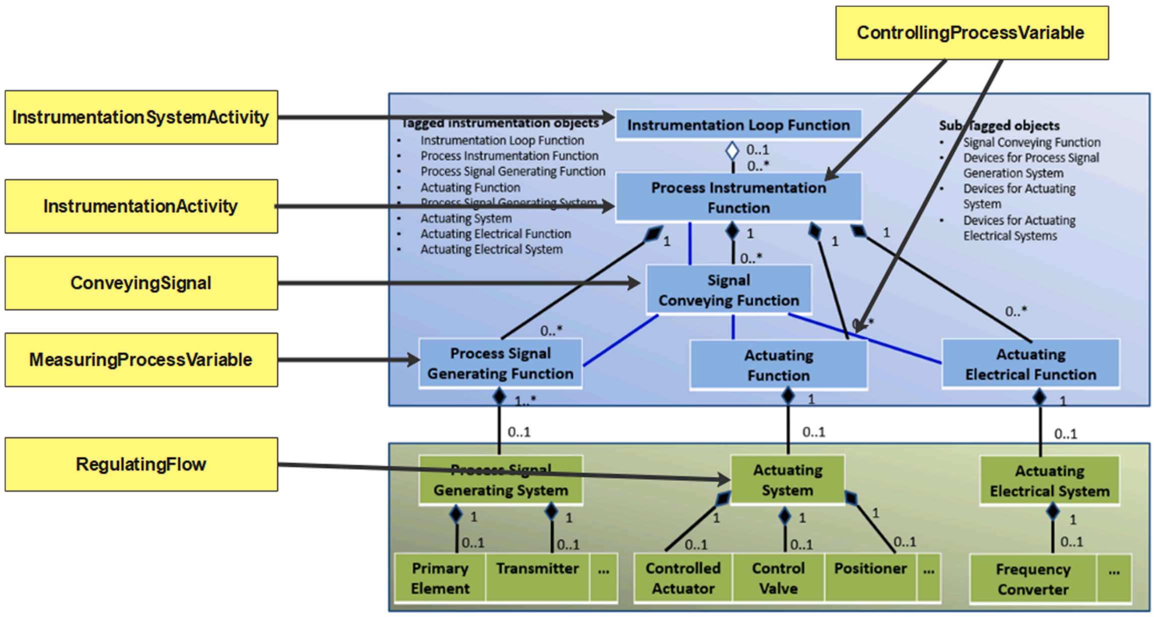

ControllingProcessVariable
InstrumentationSystemActivity
InstrumentationActivity
ConveyingSignal
MeasuringProcessVariable
RegulatingFlow
Tagged instrumentation objects
Instrumentation Loop Function
Process Instrumentation Function
Process Signal Generating Function
Actuating Function
Process Signal Generating System
Actuating System
Actuating Electrical Function
Actuating Electrical System
Sub-Tagged objects
Signal Conveying Function
Devices for Process Signal Generation System
Devices for Actuating System
Devices for Actuating Electrical Systems
Instrumentation Loop Function
Process Instrumentation Function
Signal Conveying Function
Process Signal Generating Function
Actuating Function
Actuating Electrical Function
Process Signal Generating System
Actuating System
Actuating Electrical System
Primary Element
Transmitter
Controlled Actuator
Control Valve
Positioner
Frequency Converter
...
0..1
0..*
1
1..*

**Fig. 9.** Relationships between DEXPI+ (Process) and DEXPI (Plant) instrumentation classes.

set of conditions. This behaviour is calculated using process simulators that build upon software that calculates the physical and thermodynamic properties of the materials to be processed.

During the design process we see an iterative process where specifications from a DEXPI Process model are used to define simulation cases that are run on a process simulator. The results of the simulation are then extracted to produce the stream tables shown on the PFD. This work is usually done using spreadsheets and a proprietary interface to the simulation tool (Fricke and Schöneberger, 2015; Fontalvo, 2014; Ponce-Ortega and Hernández-Pérez, 2019; Romatier et al., 2015). This is shown in Fig. 12.

DEXPI Process defines a data model for representing the properties of process streams. We have based this model on the CAPE-OPEN standards (CoLAN Consortium 2011). In doing so, we hope to simplify interchange of data to and from simulators. The model is shown in Fig. 13.

This builds on the idea of a **MaterialTemplate**. An object of this type is defined for every main type of process material in the facility. There will only be a few templates in any project. For example, an oil & gas facility project will define material templates for the process fluids and for each utility fluid.

The **MaterialTemplate** defines the data structure for each **Stream** object and the **MaterialPort** objects it connects. It defines a list of chemical components in the stream. These can be either a **PureMaterialComponent** object, which have a well-defined chemical composition, or a **CustomMaterialComponent**, such as the project-specific pseudo-components used to characterize heavy hydrocarbons. The **MaterialTemplate** also defines the number of phases to be shown in the stream tables and provides labels for each of the phases.

The **MaterialTemplate** provides all the static information needed to build the row headings in the stream table.

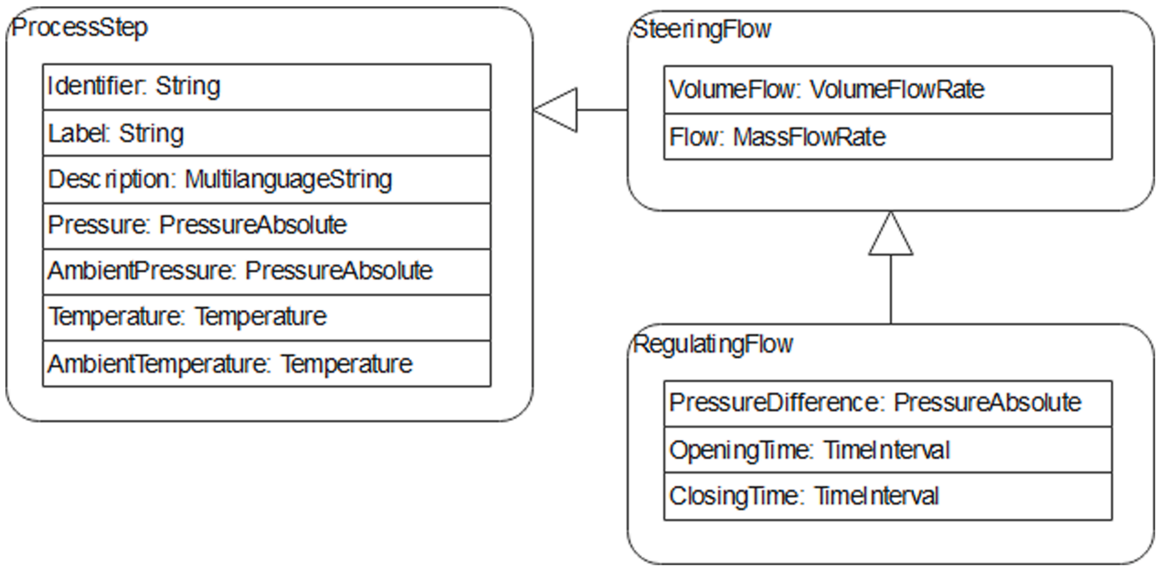

ProcessStep
Identifier: String
Label: String
Description: MultilanguageString
Pressure: PressureAbsolute
AmbientPressure: PressureAbsolute
Temperature: Temperature
AmbientTemperature: Temperature
SteeringFlow
VolumeFlow: VolumeFlowRate
Flow: MassFlowRate
RegulatingFlow
PressureDifference: PressureAbsolute
OpeningTime: TimeInterval
ClosingTime: TimeInterval

**Fig. 10.** Inheritance of parameters between ProcessStep classes.

**Table 14**
Properties for qualified physical quantities.

| Property | Type | Description |
|---|---|---|
| Case | String | An identifier to the **Case** object that relates to this value |
| Description | MultiLanguageString | A human-readable description of the specification or value. It can be in several languages |
| Label | String | A display label for the specification |
| Mode | QuantityMode | The mode of the value: Allowable, Design, Expected, Incidental, Operating or Test |
| Provenance | QuantityProvenance | The provenance of the value: Calculated, Estimated, Observed, Set or Specified |
| ProvenanceURI | AnyURI | A link to further information about the provenance of the value |
| Range | QuantityRange | The range of the value: Actual, Average, LowerLimit, Nominal, Normal or UpperLimit |
| ReferenceDataURI | AnyURI | A link to semantic reference data that defines this specification. |
| SourceURI | AnyURI | A link to information about the source of this value. |

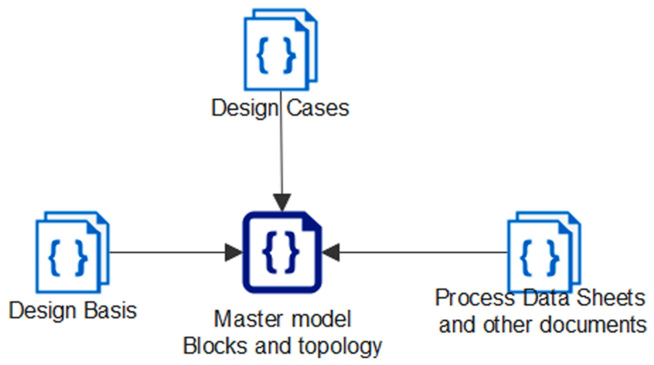


**Fig. 11.** An example of separation of model from characterizations using multiple DEXPI Process documents.

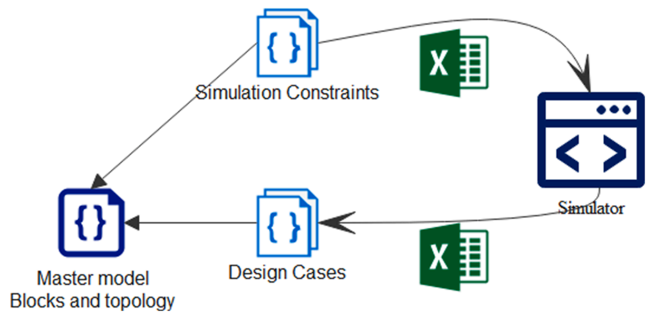


**Fig. 12.** Workflow for preparation of stream tables for design cases. Microsoft Excel is presented as an example of the most common interface between engineering data systems and simulations.

Each column in the stream table is built up of a **Stream** object. This can either be a simple object, characterized by mass and volume flow, temperature, and pressure, or can be expanded through a **MaterialState** object to provide composition and physical property data for the total stream and each phase.

Recall the workflow shown in Fig. 12. The data model allows parameters and stream tables to be associated with a **Case** label. This allows simulation constraints and design case results to be linked to each other in DEXPI Process documents.

### 3.8. *Metadata*

For documents, we use the metadata classes defined in the existing DEXPI standard. This allows us to locate the DEXPI Process model into an enterprise information structure. However, our process structure is distinct from the plant structure, as shown in Fig. 14.

A model representing BFD or PFD would thus have the following metadata.

- Location in the functional hierarchy: names and codes for the current process step and its parents in the hierarchy.
- Reference to the plant hierarchy. The process step represented will usually be realized by a plant section, area, system, or train.
- Project information.

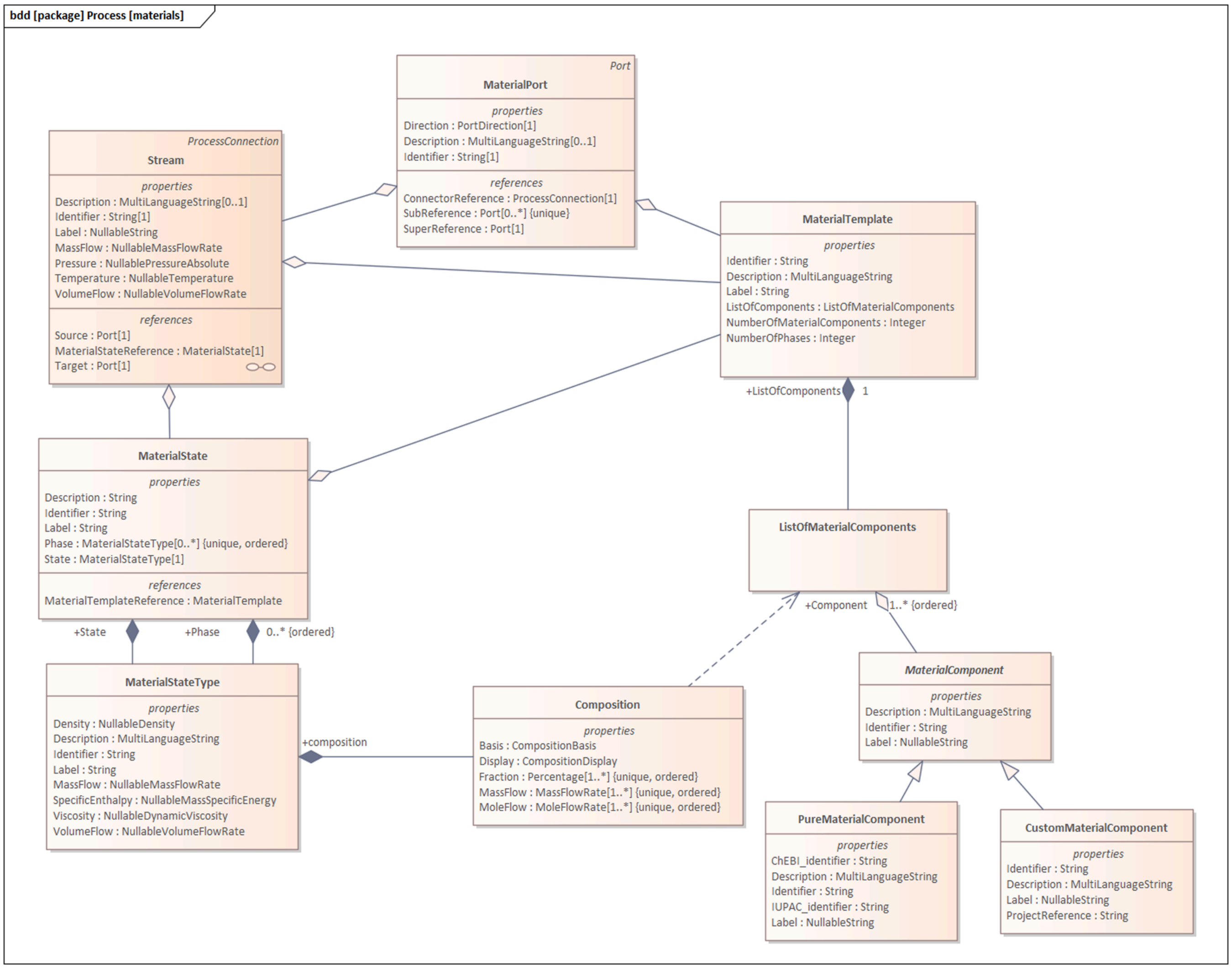

bdd [package] Process [materials]
Port
MaterialPort
properties
Direction : PortDirection[1]
Description : MultiLanguageString[0..1]
Identifier : String[1]
references
ConnectorReference : ProcessConnection[1]
SubReference : Port[0..*] {unique}
SuperReference : Port[1]
ProcessConnection
Stream
properties
Description : MultiLanguageString[0..1]
Identifier : String[1]
Label : NullableString
MassFlow : NullableMassFlowRate
Pressure : NullablePressureAbsolute
Temperature : NullableTemperature
VolumeFlow : NullableVolumeFlowRate
references
Source : Port[1]
MaterialStateReference : MaterialState[1]
Target : Port[1]
MaterialTemplate
properties
Identifier : String
Description : MultiLanguageString
Label : String
ListOfComponents : ListOfMaterialComponents
NumberOfMaterialComponents : Integer
NumberOfPhases : Integer
+ListOfComponents
1
MaterialState
properties
Description : String
Identifier : String
Label : String
Phase : MaterialStateType[0..*] {unique, ordered}
State : MaterialStateType[1]
references
MaterialTemplateReference : MaterialTemplate
ListOfMaterialComponents
+Component
1..* {ordered}
+State
+Phase
0..* {ordered}
MaterialStateType
properties
Density : NullableDensity
Description : MultiLanguageString
Identifier : String
Label : String
MassFlow : NullableMassFlowRate
SpecificEnthalpy : NullableMassSpecificEnergy
Viscosity : NullableDynamicViscosity
VolumeFlow : NullableVolumeFlowRate
+composition
Composition
properties
Basis : CompositionBasis
Display : CompositionDisplay
Fraction : Percentage[1..*] {unique, ordered}
MassFlow : MassFlowRate[1..*] {unique, ordered}
MoleFlow : MoleFlowRate[1..*] {unique, ordered}
MaterialComponent
properties
Description : MultiLanguageString
Identifier : String
Label : NullableString
PureMaterialComponent
properties
ChEBI_identifier : String
Description : MultiLanguageString
Identifier : String
IUPAC_identifier : String
Label : NullableString
CustomMaterialComponent
properties
Identifier : String
Description : MultiLanguageString
Label : NullableString
ProjectReference : String

**Fig. 13.** DEXPI Process classes used to model material properties and stream tables.

- Metadata about the document – i.e., a snapshot of the BFD or PFD information in the model: Revision number, approval information (date, type, and name) and confidentiality.

## 4. Representations of DEXPI process models

### 4.1. UML representation

DEXPI Process is presented as a UML model that is consistent with the model used to represent DEXPI. DEXPI Process is implemented as a new UML package, called **Process**. We must also define the additional physical quantities required to model process parameters and the properties of material streams. These are added to the **PhysicalQuantities** package. The additional quantities required are density, dynamic viscosity, electric conductivity, electric current, electric resistance, energy, heat capacity, heat transfer resistance, kinematic viscosity, magnetic field intensity, magnetic flux density, mass concentration, mass specific energy, mass specific heat capacity, mole concentration, mole flow rate, mole specific energy, moment of force, particle size, pH, surface tension, thermal conductivity, time interval and velocity.

A process model can be built by creating a class or object diagram that contains instances of the **ProcessStep** and **ProcessStepDetail** classes. **Port** objects are placed on these instances. These then provide anchors for connecting the different process steps together.

Stream tables can be built by creating instances of the **MaterialTemplate** class and instances of **Stream** and other **ProcessConnection** classes where it is desired to display information about material and energy flows.

The UML representation has been validated using the example BFD and PFD diagrams presented in ISO 10628–1 (ISO 2014). For reasons of space, we show in Fig. 16 how the model represents the BFD given in Figure A.2 of this standard (Fig. 15).

UML models for this and other examples are provided in the supplementary material.

### 4.2. Experimental XML representation

The UML model is an effective way of designing and documenting ths structure of a data model. However, it is unsuitable for practical implementation in engineering workflows. For this reason, we built an XML schema representation of the model and tested how it worked in representing the information on a set of realistic BFDs and PFDs.

We were able to implement all features of the data model using XML schema representation. The schema was then able to verify the correctness of syntax for XML documents that modelled the example diagrams.

The schema and data files are available in the supplementary material for this paper. The reader should note that this representation is experimental. It has no status as a standard.

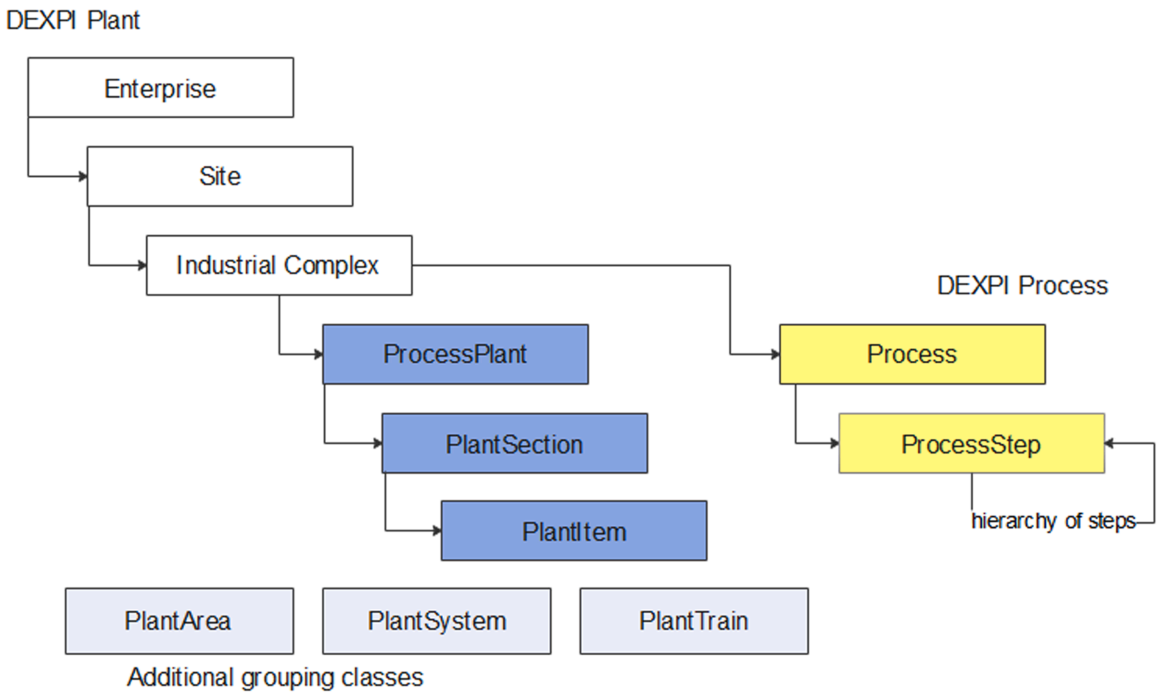


**Fig. 14.** Positioning a facility hierarchy showing both plant hierarchy and process hierarchy.

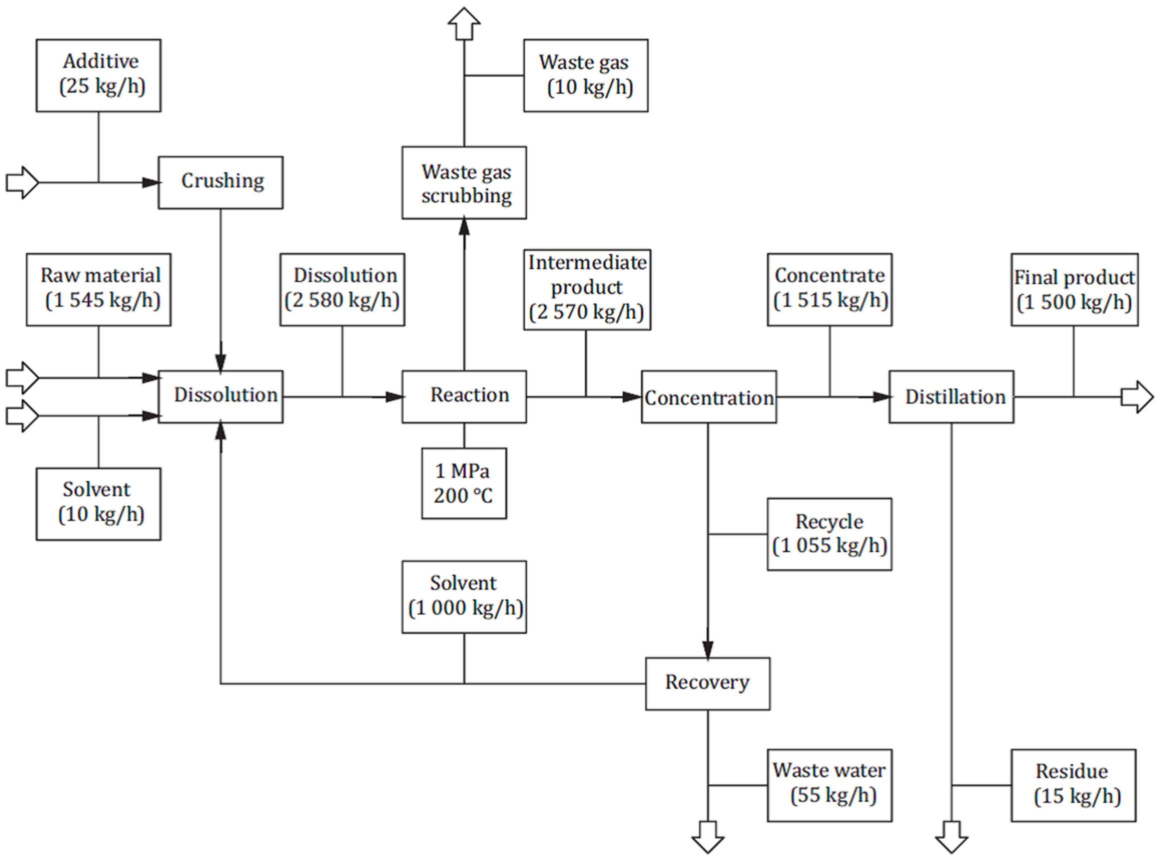


**Fig. 15.** Simple block diagram example, Figure A.4 in ISO 10628–1.

## *4.3. AutomationML representation*

AutomationML is a promising standard for information modelling and document serialization for automation systems (Berardinelli et al., 2016). It has been successfully used to model control-system configurations to support automatic transfer of designs from engineering to implementation (Ingebrigtsen and Drath, 2021). It provides an XML-based, object-orientated framework for information modelling. It

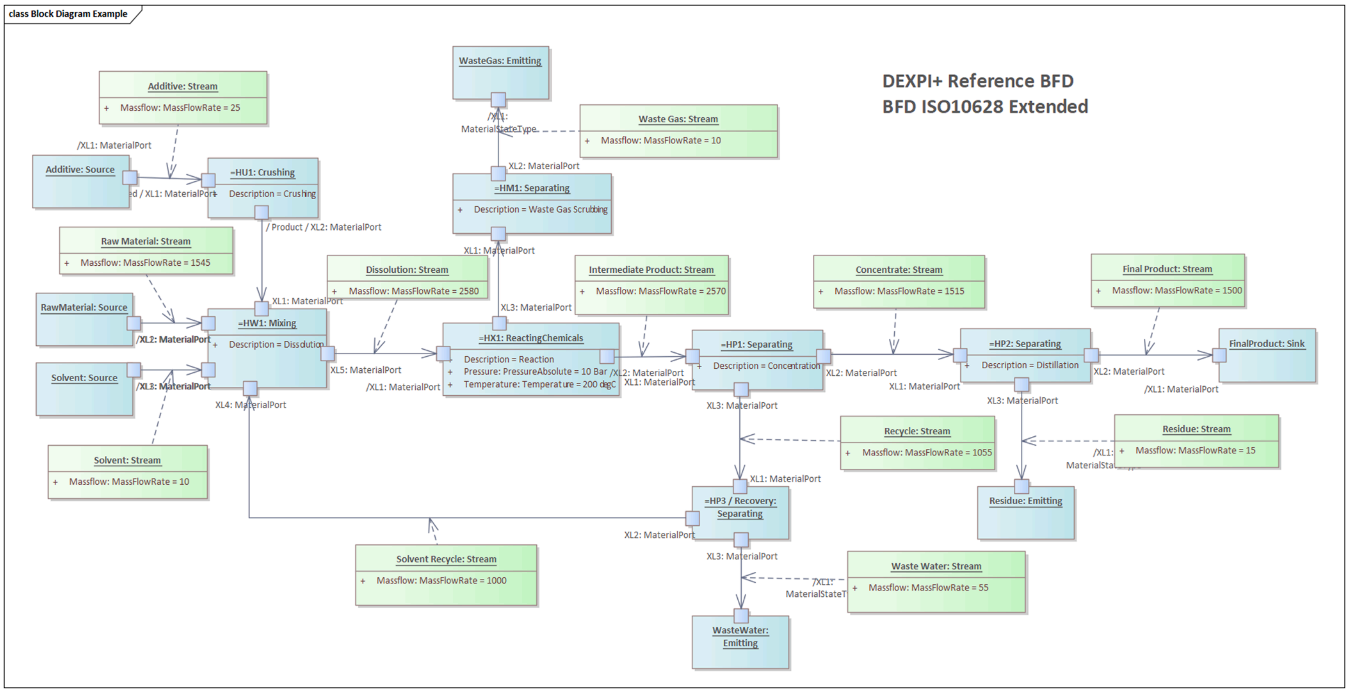
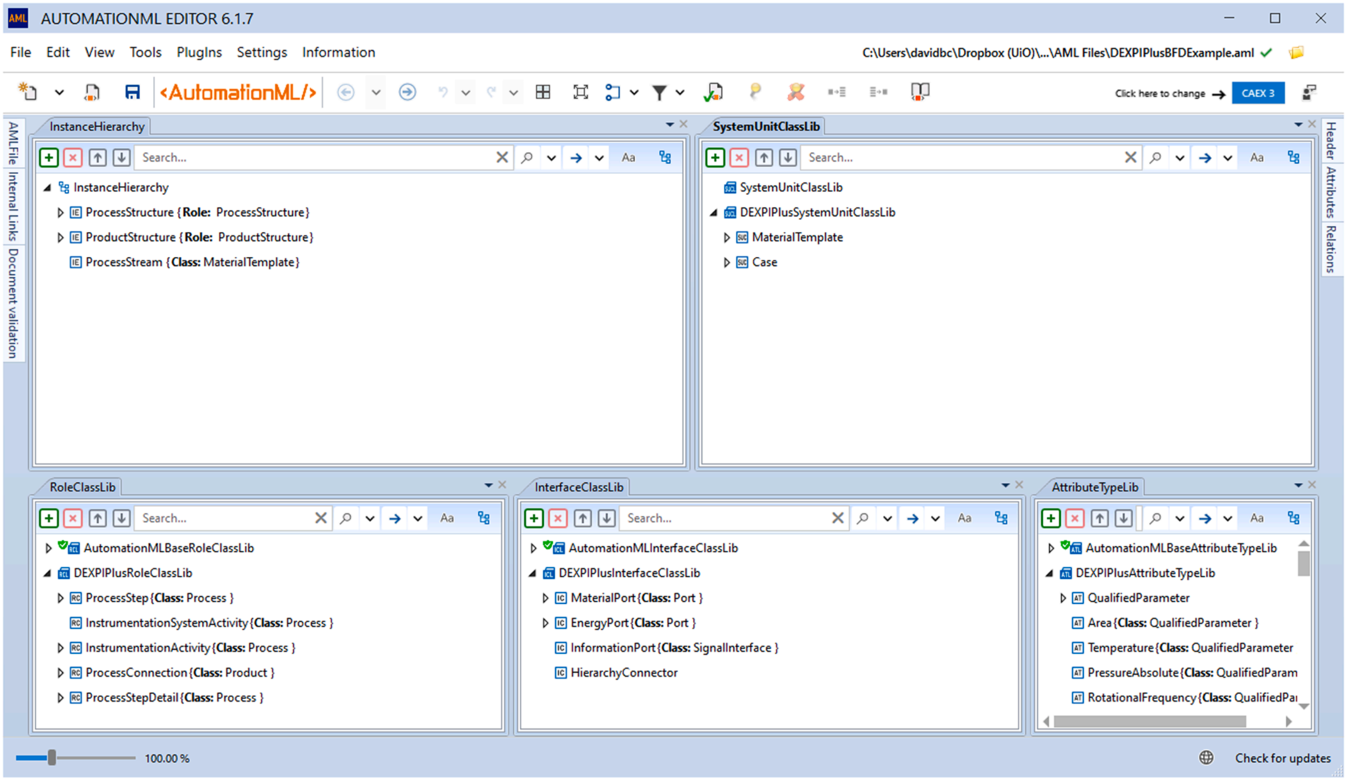


**Fig. 16.** A UML representation of the example BFD.


AUTOMATIONML EDITOR 6.1.7
File Edit View Tools PlugIns Settings Information
C:\Users\davidbc\Dropbox (UiO)\...\AML Files\DEXPIPlusBFDExample.aml
<AutomationML/>
Click here to change
CAEX 3
AMLFile Internal Links Document validation
Header Attributes Relations
Search...
100.00 %
Check for updates

InstanceHierarchy
- InstanceHierarchy
  - ProcessStructure {Role: ProcessStructure}
  - ProductStructure {Role: ProductStructure}
  - ProcessStream {Class: MaterialTemplate}

SystemUnitClassLib
- SystemUnitClassLib
- DEXPIPlusSystemUnitClassLib
  - MaterialTemplate
  - Case

RoleClassLib
- AutomationMLBaseRoleClassLib
- DEXPIPlusRoleClassLib
  - ProcessStep {Class: Process }
  - InstrumentationSystemActivity {Class: Process }
  - InstrumentationActivity {Class: Process }
  - ProcessConnection {Class: Product }
  - ProcessStepDetail {Class: Process }

InterfaceClassLib
- AutomationMLInterfaceClassLib
- DEXPIPlusInterfaceClassLib
  - MaterialPort {Class: Port }
  - EnergyPort {Class: Port }
  - InformationPort {Class: SignalInterface }
  - HierarchyConnector

AttributeTypeLib
- AutomationMLBaseAttributeTypeLib
- DEXPIPlusAttributeTypeLib
  - QualifiedParameter
  - Area {Class: QualifiedParameter }
  - Temperature {Class: QualifiedParameter
  - PressureAbsolute {Class: QualifiedParam
  - RotationalFrequency {Class: QualifiedPa

**Fig. 17.** The AutomationML editor showing parts of the model for the simple ISO 10628 BFD example.

was therefore of interest to see whether DEXPI Process could be implemented using AutomationML.

The model for the simple block flow diagram example from ISO 10628 is shown in the AutomationML Editor tool in Fig. 17.

We found that it was straightforward to implement our model as an AutomationML model. In fact, we found that it was beneficial to base DEXPI Process classes on AutomationML built-in classes (Drath, 2021).

We used the Product-Process-Resources (PPR) model from AutomationML. This required some translation. An AutomationML ***Process*** maps straightforwardly to a **ProcessStep**. In AutomationML a ***Process*** does an operation on a ***Product*** and is realized by a ***Resource***. This means that an AutomationML ***Product*** corresponds to a

**ProcessConnection,** and a ***Resource*** corresponds to a DEXPI plant item.

Thus, we derive the **ProcessStep, InstrumentationSystemActivity, InstrumentationActivity** and **ProcessStepDetail** classes from the AutomationML ***Process*** class. This allows us to use inheritance to build a set of AutomationML role classes, organized as a role class library.

A DEXPI Process **Port** can be derived from AutomationML built-in interface classes. Material and energy ports are derived from the AutomationML ***Port*** interface class, while the **InformationPort** is derived from the ***SignalInterface*** class. These classes are declared in an interface class library.

All property types are declared in an attribute type library. It proved to be straightforward to implement the qualified property approach described above.

Finally, the **MaterialTemplate** and **Case** classes are implemented in a system unit class.

The actual process model is built in the **InstanceHierarchy** window. This is equivalent to the UML object diagram. All process steps are defined in the **ProcessStructure** branch of the tree, and all process connections are defined in the **ProductStructure** branch. A fragment of the instance hierarchy is shown in Fig. 18.

We connect ports together and link them to process connections using the AutomationML **PPRConnector** class. **SubReferences** and **SuperReferences** are implemented as instances of a **HierarchyConnector** interface defined in our interface class library.

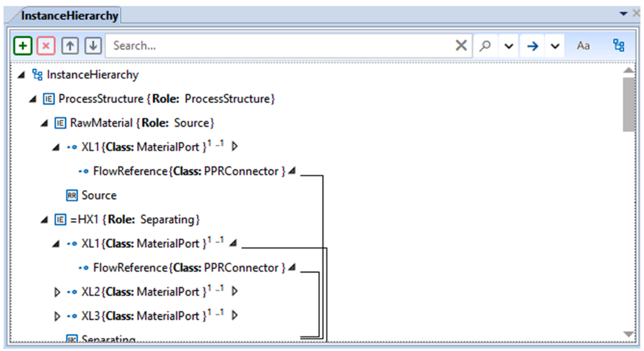
In summary, we found that it was straightforward to implement our model in AutomationML. We were able to leverage built-in classes in AutomationML and use them in meaningful ways to organize the process model. This is promising, as it means that we can use existing AutomationML APIs and tools to build DEXPI Process models in AutomationML. However, we found that the AutomationML editor was not a suitable tool for building large models.

The AutomationML files for the example are supplied in the supplementary material.

### 4.4. *Knowledge graph and text representations*

A DEXPI Process model can be straightforwardly represented as a knowledge graph in RDF format where the nodes are classes in DEXPI process and the arcs model relationships between the classes. In this work, we are collaborating with the READI Information Modelling Framework (IMF) (Fjøsna and Waaler, 2021; Waaler, 2022), which is developing an RDF language for aspect system modelling. DEXPI Process provides a set of types and parameters that cover the functional aspect of chemical process design. Fig. 19 shows a fragment of a DEXPI Process model expressed using the IMF RDF.

It is also simple to map the structure of a DEXPI Process to the SFILES text representation of process flowsheets (Vogel et al., 2022; Mann et al., 2023). The process steps are represented by strings, while the connections map to the topology symbols in SFILES, as described in Section 2.3.6.



```
InstanceHierarchy
  ProcessStructure {Role: ProcessStructure}
    RawMaterial {Role: Source}
      XL1{Class: MaterialPort }1 -1
        FlowReference{Class: PPRConnector }
      Source
    =HX1 {Role: Separating}
      XL1{Class: MaterialPort }1 -1
        FlowReference{Class: PPRConnector }
      XL2{Class: MaterialPort }1 -1
      XL3{Class: MaterialPort }1 -1
      Separating
```

**Fig. 18.** An extract of the instance hierarchy for our example BFD.

We believe that the standard types of nodes and process connections proposed here, implemented as graphs or SFILES strings, will advance research on exploitation of process knowledge graphs, such as that described in (Vogel et al., 2023), where PFDs, represented as knowledge graphs, are used to train generative algorithms for automated process design.

## 5. Demonstration on the Tennessee Eastman Process

### 5.1. *Motivation for the demonstration*

The DEXPI Process model was developed and validated in two engineering projects, where an engineering company used the standard to produce process models that were then successfully transferred to the operating company for use in their modelling tools. The engineering company used DEXPI Process objects implemented as SysML to model all the PFDs – the entire process design - for an offshore oil platform. This was then transferred as RDF to the operator, who was able to reconstruct the model in their modelling tool. Our findings were that DEXPI Process provided all the resources needed to represent the plant PFDs. We found that SysML is an effective tool for building the model structure. However, it is not suitable for use by process engineers, as our modelling uses only a fraction of the capabilities of UML and SysML. We also saw that a graphical model editor is also unsuitable for accessing and manipulating data such as stream tables. We address these issues further in Section 6.2.3.

The process designs are confidential, so for this reason we have prepared a demonstration on a realistic, open process model, the Tennessee Eastman process. We provide the source model and serializations of the model as supporting material as a support for further research and development of methods.

### 5.2. *Modelling the block flow diagram*

The Tennessee Eastman Process is an example process, first published in (Downs and Vogel, 1993), that has been widely used as a realistic teaching and research example for process design. It was used in previous work on aspect-oriented SysML modelling of process functions (Cameron et al., 2022).

The process is described by a PFD and a set of stream tables. The identity of the chemical components is obscured by letter identifiers. Sensors and final control elements are shown in the PFD, but the supervisory control structure is omitted deliberately. Fig. 20 shows the PFD as presented in the original paper.

The DEXPI Process model is hierarchical. It starts with a high-level representation of the main process steps, forming a BFD, shown in Fig. 21. Here we use top-level process steps, e.g., **Compressing, Separating** and **ReactingChemicals**. We also create stream objects for each labelled stream in the process and a single **MaterialTemplate** object.

### 5.3. *Detailed process flow diagram*

A detailed process flow diagram is then built for each block in the high-level model. At this level, we build the detailed model inside the frame of the upper-level model. For reasons of space, we will only show the detailed model for the Stripping process, `=KC2, shown in` Fig. 22.

Here we use more detailed unit operations, such as **StrippingDistilling**. We also model the sensors and control elements in the system using **MeasuringProcessVariable** and **RegulatingFlow** blocks. Finally, we use **SubReferences** and **SuperReferences** to align ports. This means that port `XL1` in `=KC2` is aligned with port `XL5` in `=HPD1` and port `XL2` in `=QNA2` is aligned with port `XL3` in `=KC2`.

The full model is supplied as a set of drawings and an XMI model interchange file in the supplementary material.

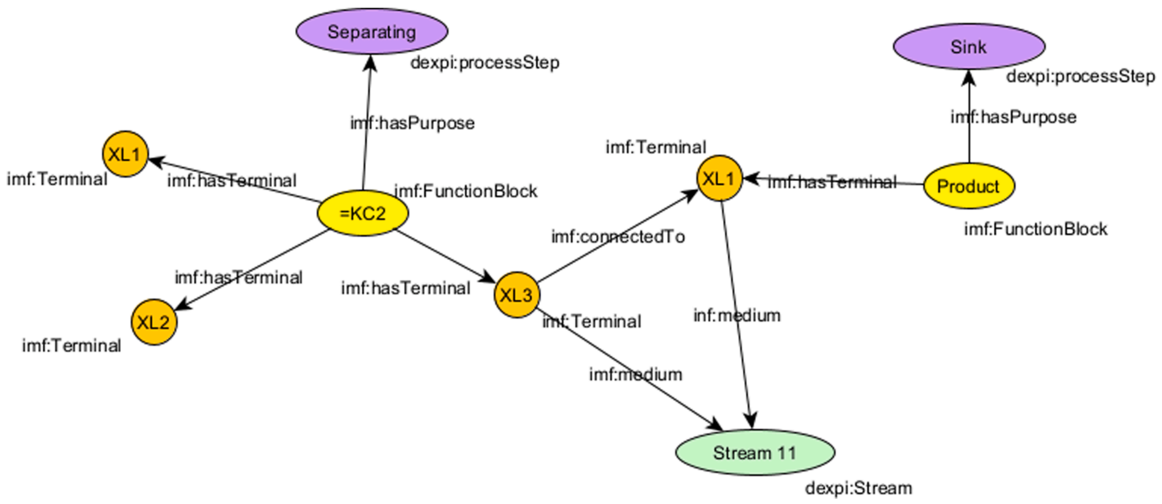


**Fig. 19.** A fragment of a DEXPI Process model expressed as an IMF knowledge graph. Here DEXPI provides the purpose of the IMF blocks and the Stream instances. Relations defined by IMF are used to connect the graph. Note that IMF uses the word terminal instead of port. The terms are synonyms.

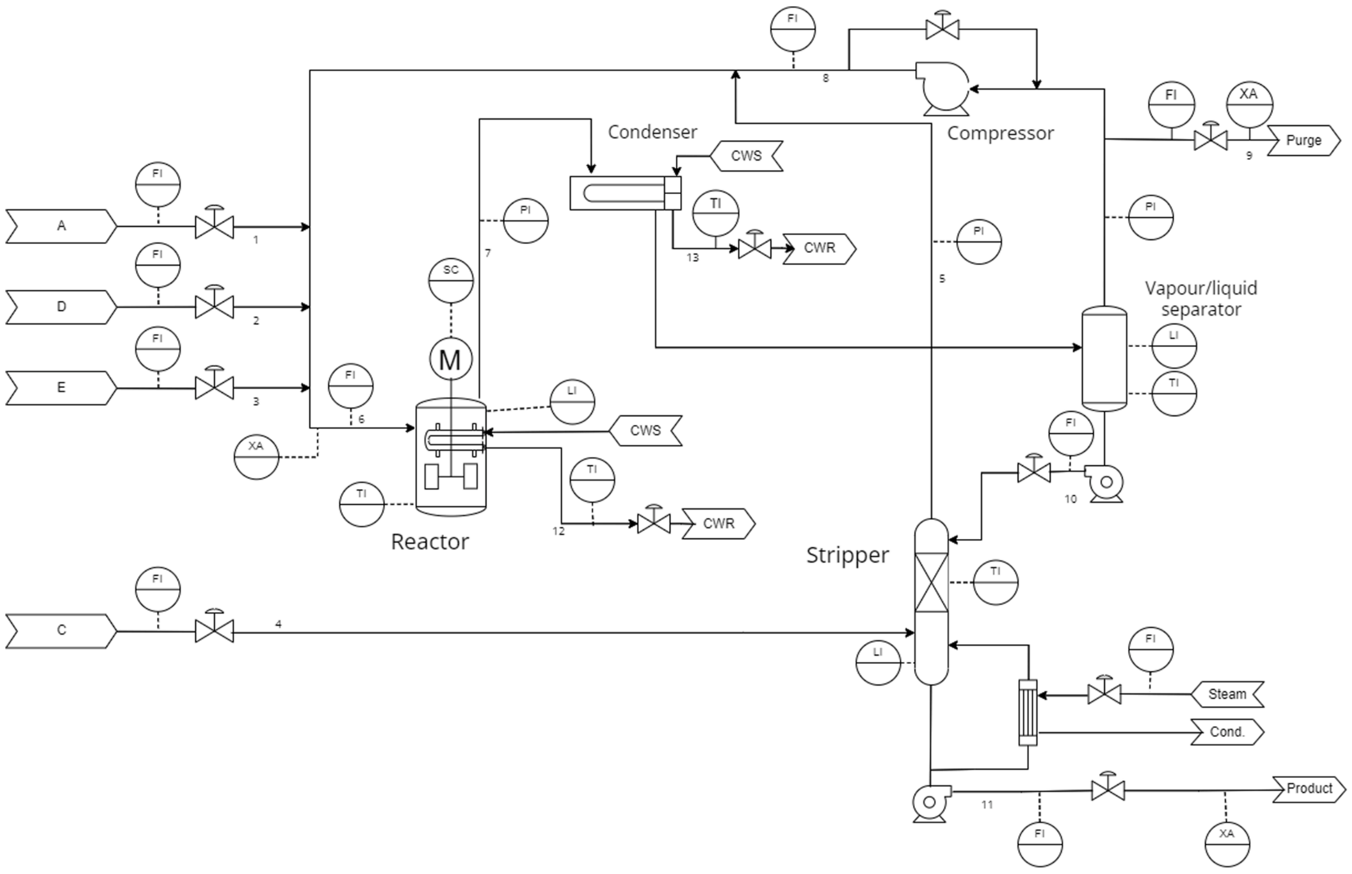


**Fig. 20.** The Tennessee-Eastman process to be modelled, redrawn from diagram in (Downs and Vogel, 1993).

## 6. Conclusion: use of DEXPI Process as a data model

### 6.1. Summary and prospects

This paper has presented the DEXPI Process data model as a proposed standard for modelling process design and its design artefacts: the BFD and PFD. This model is designed to integrate existing standards and, in this way, enable better interdisciplinary collaboration around the engineering and operations of chemical process facilities. The work integrates the following concepts and best practices. We adopt the data model of blocks with ports from SysML (Hernandez et al., 2016), IDEF0 (IEEE Computer Society 1998) and VDI/VDE 3682 (VDI/VDE, 2015b) (VDI/VDE, 2015a). The OntoCAPE ontology (Marquardt, 2010) provided the concept of a **ProcessStep** and formalized aspect modelling. The ENPRO project (Wiedau et al., 2019) provided us with a facility lifecycle that makes a clear distinction between process, plant and asset structure. ISO 15926 provided us with a set of activities that gave us a starting point for defining the types of **ProcessStep**. The CFIHOS

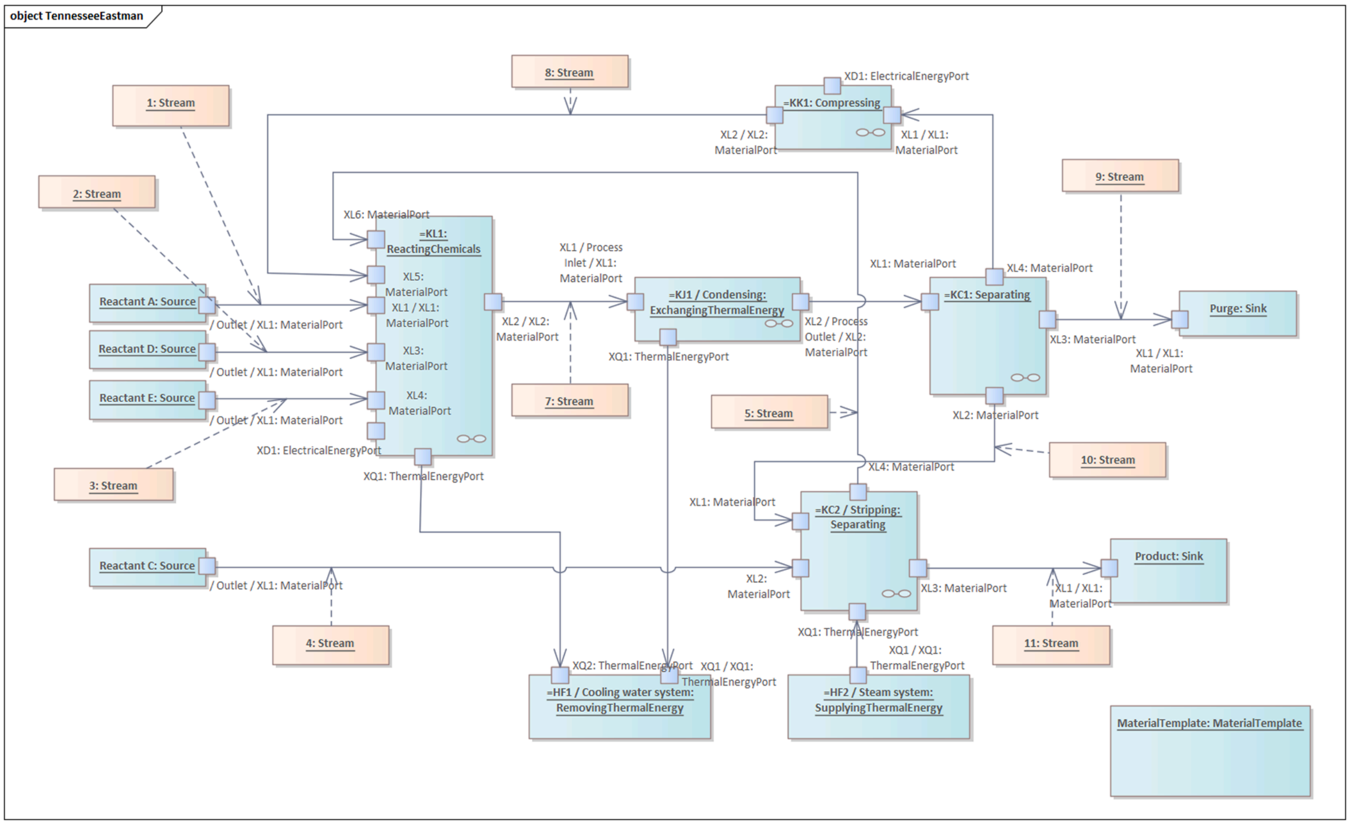


**Fig. 21.** The top-level, BFD representation of the Tennessee Eastman process.

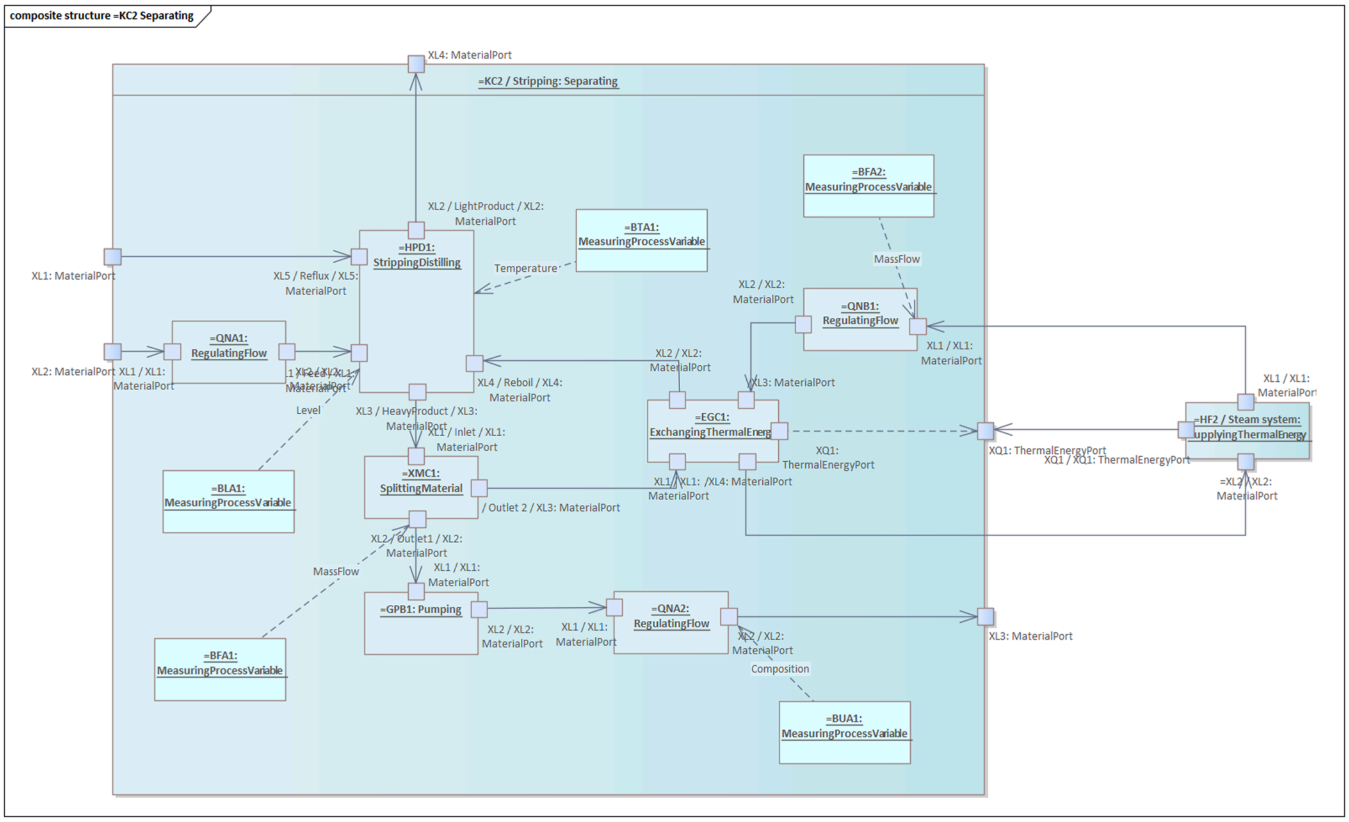

composite structure =KC2 Separating
=KC2 / Stripping: Separating
=HPD1: StrippingDistilling
=BTA1: MeasuringProcessVariable
=BFA2: MeasuringProcessVariable
=QNB1: RegulatingFlow
=QNA1: RegulatingFlow
=EGC1: ExchangingThermalEnergy
=XMC1: SplittingMaterial
=BLA1: MeasuringProcessVariable
=GPB1: Pumping
=QNA2: RegulatingFlow
=BFA1: MeasuringProcessVariable
=BUA1: MeasuringProcessVariable
=HF2 / Steam system: SupplyingThermalEnergy
Temperature
MassFlow
Level
Composition

**Fig. 22.** The lower-level model for the stripping process.

standard (CFIHOS 2023) provides us with PROCESS ACTIVITY and PROCESS STREAM classes that fill the same roles as the **ProcessStep** and **ProcessConnection** classes.

This development has drawn extensively on ISO/IEC 81346. Our aspect-orientated approach is motivated by this standard and we have aligned types defined in ISO/IEC 81346–2 (IEC 2019) and the RDS for oil & gas (READI, 2021) with the DEXPI Process types. We have adopted the taxonomy of processes presented in DIN6779–13 (DIN 2018) directly in the standard.

We believe that DEXPI Process fills a critical gap in interoperability standards and can be a mechanism for improving the efficiency of process engineering. It is, however, a work in progress. We therefore conclude the paper with description of the work needed to further develop and exploit the potential of the data model.

## 6.2. Prospects for further work

### 6.2.1. Graphical presentation

The current version of the standard only models the non-graphical design information shown on the BFD or PFD. The graphical representation of the diagram remains to be defined. A clear separation should be kept between the data layer (as modelled here) and the presentation layer. Integration between the two layers may be done through:

- Defining linkages between symbol files and process step classes.
- Defining the position on the symbol for each port.
- Defining the location of labels and other metadata.
- Defining mappings between connection lines and types of process connection objects.
- Defining templates for display of stream tables.

The existing DEXPI standard provides resources for these tasks.

### 6.2.2. Serialization

The data model presented here is independent of its serialization format. However, it is necessary to provide effective formats for exchanging and storing DEXPI Process models.

This paper has presented three serialization formats: a UML format, an experimental XML format and an AutomationML format. The UML format is proprietary but can be exported as an XMI file. This, however, is a poor format for data exchange, as the engineering data is lost in a large quantity of presentation data related to the UML modelling tool.

The experimental XML format verified that the data model could be described and verified using an XML schema. This format is concise and only includes the engineering data. The schema provides a basis for simple integration into XML-aware tools. More concise serialization formats, such as JSON, could also be used.

Finally, we have verified that the AutomationML language can be used to represent DEXPI Process models. This solution was elegant, as we were able to use existing semantics in AutomationML to organize and structure the data model.

The DEXPI Process data model generates a graph data structure. This means that graph formats such as RDF (Cyganiak et al., 2014) can be used to serialize the structural aspects of the model: plant items, ports, and connections.

We are less convinced about using RDF (Cyganiak et al., 2014) to serialize the parameters and stream tables in the model. Here a better approach may be to use formats where nodes and arcs in a graph structure can be allocated parameters, such as JSON-LD (W3C 2020) or Microsoft's Digital Twins Definition Language (Microsoft 2022). We also believe that serialization as Extended SFILES strings will be useful as an enabler for process synthesis activities (Mann et al., 2023).

### 6.2.3. Tools

Efficient use of the data model requires suitable tools. The prototyping and development work done here has used the Enterprise Architect (https://sparxsystems.com) UML and SysML modelling tool, an XML editor (AltovaXML Spy, https://www.altova.com/xmlspy-xml-editor) and the AutomationML Editor (https://www.automationml.org/download-archive/).

These tools were suitable for the development tasks but are not suitable for use in process engineering contexts. UML editors are proprietary and often implement subtly different versions of UML and SysML. This makes transfer of models between tools using XMI files difficult. In addition, the user interface for these tools is powerful, but complex. DEXPI Process uses only a few features of the UML language. However, it is difficult to isolate this functionality. This problem can be alleviated by providing a tailored interface to the tool. For example, the Enterprise Architect tool supports custom symbol libraries and Model Driven Generation (MDG) framework for this purpose.

Practical use of DEXPI Profess models will require the following tools for:

- Graphical construction and maintenance of the structural part of DEXPI Process models. This can be done at present in Enterprise Architect or the AutomationML Editor, but this is not acceptable for process engineers. This functionality can be incorporated into existing process CAD tools, where our type definitions and syntax are associated with symbols and connections in the BFD and PFD drawing functionality.
- Viewing and editing parameters in the model. We have seen in Section 3.6 that a structural model will be associate with many sets of parameters and stream properties. Our experience is that the graphical editors used to build the structural model are inefficient for entering and editing this type of data. Here we need tools with a spreadsheet-like user interface that allow creation and inspection of substantial amounts of tabular data. This functionality will require integration with proprietary and corporate engineering databases.
- Exposing DEXPI Process data models to other applications. An application programming interface (API) that exposes components of a model would be a useful tool for developing analytical applications and integrating the models into engineering workflows.

### 6.2.4. Integration of workflows with DEXPI PID models

Our ambition is that DEXPI Process models of the process can be linked with and provide a context for DEXPI models of the plant. A challenge here is that DEXPI Process uses a different approach to modelling topology to the existing DEXPI model. DEXPI Process uses ports to manage connections, whereas DEXPI uses a **PipingConnection** object to link together **PipingNode** objects. These **PipingNode** objects are owned by a **Nozzle** on equipment (apparatus and machines) or a **PipingComponent** object.

The DEXPI Process data model has been developed in the context of an oil & gas project where we wanted to document a top-down functional design and document it at each level with a BFD or PFD and sets of associated parameters. As the process design advances, engineers will begin to consider how the process is to be realized. This can be done through a transformation of aspect, where we build an information model for the plant using DEXPI classes and do this within a conceptual frame with the same extent as the corresponding **ProcessStep** in the process model. We can then align **Port** objects on the **ProcessStep** with **Node** objects owned by off-page connectors in the plant model.

## 6.3. Alignment with other standards

### 6.3.1. CFIHOS

In Section 2.3.1 we presented how the CFIHOS data model supports the modelling of process steps and process connections. This was done by defining **PROCESS ACTIVITY** and **PROCESS STREAM** objects and establishing relationships between these objects, **TAG** objects and their parameters. This was shown in Fig. 2.

Alignment of DEXPI Process with CFIHOS is straight-forward. Each

DEXPI Process **ProcessStep** maps to a **PROCESS ACTIVITY** object and each **ProcessConnection** maps to a **PROCESS STREAM**. CFIHOS lacks the concept of ports, using a list of input and output streams for each **PROCESS ACTIVITY**.

The DEXPI Process model has defined a set of reference data for **ProcessStep** classes and their parameters. Implementation of this reference data in CFIHOS would support interoperability and fill a current gap in that standard.

#### *6.3.2. ISO 15926*

The DEXPI+ working group has developed a set of types for processing activities that can be incorporated into new revisions of ISO 15926–4. This offers opportunities for using semantic models expressed as ontologies, to model relationships between DEXPI Process types. These semantic models capture the meaning of the data model and thereby support reasoning and checking of consistency (Cameron et al., 2022). We are aware of ongoing activities in this area. It may be possible to reflect the model using the Templates and OWL implementation defined in parts 7, 8 and 11 in the standard. It remains to be seen, however, whether the ontology-based OWL and RDF modelling in ISO 15926 is adequate to model DEXPI Process data in a scalable and user-centred way.

#### *6.3.3. NAMUR standards*

Using DEXPI Process data models can simplify the use of both the NE159 standard for using process data in automation equipment design (NAMUR 2018) and the MTP framework. It is straightforward to write a mapping between NE159 and DEXPI Process data related to a **MeasuringProcessVariable** or **RegulatingFlow** block so that the relevant process data is exported in the NE159 standard.

Regarding the MTP, the process modular breakdown given in this standard fits well with the top-down process breakdown modelled in DEXPI Process. This will allow the transformation of models into definitions of MTP modules and their interfaces.

#### *6.3.4. Asset administration shell*

Finally, the AAS offers a further way of serializing our models. We propose further prototyping to define the best way of doing this. An advantage of adopting AAS is that we can use the open-source tools and APIs that have been developed to work with AAS datasets. The authors of this paper have also been involved in the definition of an AAS template for DEXPI Plant datasets (IDTA, 2023). This template supports two use cases (Grüner and Otten, 2023). Firstly, a DEXPI file can be embedded in an AAS, so that consistent metadata is shared between parties in a data handover. The second use case involves using the AAS to expose identifiers to the objects in the DEXPI model in the shell. This allows other AAS objects to refer to plant items in the DEXPI model. It is a natural further step to apply this approach to DEXPI Process models.

### *6.4. DEXPI Process as the process engineering reference for an integrated, aspect-based engineering information system*

DEXPI Process is a standard for chemical process engineering. However, its structure is amenable to any aspect-based engineering model. This means that similar standards can be developed in related disciplines, such as electrical or construction to support top-down functional modelling. Here we can draw on different parts of ISO/IEC 81346. For example, ISO/IEC 81346 part 10 (ISO 2022) provides a breakdown for electrical systems and utilities.

Here we seen an opportunity for building interdisciplinary models that explicitly represent and track the interfaces between disciplines. This can give benefits in standard chemical facility projects, but we see that the on-going twin transition to digital and renewable energy requires exactly this type of interdisciplinarity. The next generation of process facilities will require ever-tighter integration of information from the process, electrical and construction domain. In this case DEXPI Process provides the reference data for the chemical engineering domain. Similar work is needed to represent other domains. The IMF project, described by Fjøsna and Waaler (Fjøsna and Waaler, 2021) is working to demonstrate this type of framework.

### *6.5. Exploiting the process graph for automation of design and process analyses*

Chemical processes are well suited to the application of graph theory methods for design and analysis (Martinez-Hernandez, 2023). This is due to the graph structure of the process and plant. As noted by Preisig (Preisig, 2009), graph theory was the key to designing the algorithms that converge process simulators. Representation of a process design by a graph allows application of graph algorithms to analyse the structure and contents of the process design. Thus, Rantala et al. (Rantala et al., 2019) use algorithms to find similarities between different designs of pulp and paper plants and Sierla et al. (Sierla et al., 2020; Sierla et al., 2021) use a graphical representation of the PID to partially configure a simulation-based digital twin.

DEXPI Process provides a standard representation of processes that can be used to support automated process synthesis. Mencarelli et al. (Mencarelli et al., 2020) review a variety of methods of representing and optimizing process structure. We believe that all of these can be aligned with this data model. Garg et al. (Garg et al., 2020) present process synthesis algorithms based on a set of Phenomena Building Blocks (PBB). These PBBs map closely to classes in DEXPI Process. We also see potential for using graph reasoning and graph-based machine learning to analyse DEXPI Process models. Oeing et al. (Oeing et al., 2023) explore the ways in which graph-based machine learning can be used to analyse plant graphs (PIDs) represented as graphs to automate good design practices. We expect that this approach can be extended to process graphs.

We have already noted two examples of use of process graphs for process synthesis. Conversion of a model or a graph or SFILES representation (Mann et al., 2023; Vogel et al., 2023) allows the application of generative artificial intelligence algorithms. We are at present working in applying design rules to the graph model built using DEXPI Process.

Graph based methods are also useful for analysing and designing safety systems. For example, O'Halloran et al. (O'Halloran et al., 2021) use a process graph of pressurised-water nuclear reactor to analyse functional failure propagation in the process.

We see an opportunity here for this work to act as a catalyst to the wider adoption of these methods. We also see an opportunity to revisit earlier research on automated applications such as cause and effect analysis (Thambirajah et al., 2009) and HAZOP (Venkatasubramanian et al., 2000; Hu et al., 2015; Rodríguez and de la Mata, 2012). The impact of these earlier works was limited by a lack of standard data models and the complexity and expense of the commercial tools used. A standard, non-proprietary data model like DEXPI Process can act as a stimulus to translation from research to implementation for these algorithms and methods.

### *6.6. Effective data transfer between engineering databases and simulators*

In developing DEXPI Process we have deliberately drawn inspiration from the data models used in the CAPE-OPEN standards for simulator interoperability (van Baten and Pons, 2014). As noted in Section 3.7, the concept of a **MaterialTemplate** object is derived from the CAPE-OPEN Thermodynamic Interface standard (CoLAN Consortium 2011). The ideas of ports and stream objects are also shared between DEXPI Process and CAPE-OPEN.

This model offers therefore an opportunity to expand CAPE-OPEN to support automated extraction of data from simulation results. A specification has been made for a Flowsheet Monitoring Interface (COLaN consortium 2019). This provides a way for a simulator to report values of material, energy and information streams, flowsheet structure and unit operation parameters. The interface provides only read-only access to the simulator. We believe that linking this interface to a DEXPI Process format can increase adoption in the industry and simplify the retrieval and management of simulation results.

In the longer term, we wish to be able to use a DEXPI Process data set to configure simulators. This is at present beyond the scope of CAPE-OPEN. However, we see possibilities in building bridging applications, or harnesses, between DEXPI Process models and commonly used simulators. This will require mapping of **ProcessStep** types to unit operation types (or clusters thereof) in the simulator. We will also need to map parameters. Such a harness requires an API, offered by the simulator vendor, which supports configuration of model topology and setting of modelling constraints.

## Supporting Information

The following supporting information is provided:

- A UML model defining the DEXPI Process classes and containing example models, including the Tennessee-Eastman model described here.
- An XML schema for the DEXPI Process model and XML example files.
- An AutomationML file that defines the DEXPI Process data model and applies it to an example BFD:

## CRediT authorship contribution statement

**David B. Cameron:** Conceptualization, Methodology, Software, Writing – original draft, Visualization. **Wilhelm Otten:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Heiner Temmen:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Monica Hole:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Gregor Tolksdorf:** Conceptualization, Methodology, Writing – review & editing, Supervision, Validation.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgements

This work was supported by the Research Council of Norway through the SIRIUS Centre for Scalable Data Access in the Oil & Gas Domain and the PeTWIN project. Work has also been funded through the DISC (formerly) NOAKA project collaboration with support from Equinor, AkerBP, Aker Solutions and Aibel. The DEXPI+ working group was supported by the in-kind and cash contributions of the operating companies in the DEXPI project.

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2023.108564.

## References

API, 2022. API Standard 521: Pressure-relieving and Depressuring Systems. API, Washington, DC. Nov.

API, 2018. 'API RP 14C. Analysis, Design, Installation, and Testing of Safety Systems for Offshore Production Facilities. Eight Edition, February 2017. Errata 1, May 2018'. API, May 2018.

Balslev, H., Barré, T., 2022. Why system models need the RDS 81346 reference model. In: Proceedings 32nd Annual INCOSE Symposium, p. 18. Detroit, MIJun.

Balslev, H., 2020. *A Guide to RDS - Reference Designation Systems. TAG Numbers for Systems in Accordance with the ISO/IEC81346 Standard Series.*, 3rd ed. in DS Handbooks. Danish Standards Foundation, Copenhagen.

Berardinelli, L., et al., 2016. Cross-disciplinary engineering with AutomationML and SysML. Autom 64 (4), 253–269. https://doi.org/10.1515/auto-2015-0076. Apr.

Cameron, D.B., Waaler, A., Fjøsna, E., Hole, M., Psarommatis, F., 2022. A semantic systems engineering framework for zero-defect engineering and operations in the continuous process industries. Front. Manuf. Technol. 2, 945717 https://doi.org/10.3389/fmtec.2022.945717. Sep.

Cameron, D.B., et al., 2022. The Digital Design Basis. Demonstrating a framework to reduce costs and improve quality in early-phase design. Digit. Chem. Eng. 2, 100015 https://doi.org/10.1016/j.dche.2022.100015. Mar.

CFIHOS, 2023. CFIHOS Data Model', IOGP, London, C-DM-001 Version 1.5.1. AprAccessed: Aug. 03, 2023. [Online]. Available. https://www.jip36-cfihos.org/wp-content/uploads/2023/04/C-DM-001-CFIHOS-Data-Model-V1.5.1.pptx.pptx.

Cheng, Z., Ma, Y., 2017. Explicit function-based design modelling methodology with features. J. Eng. Des. 28 (3), 205–231. https://doi.org/10.1080/09544828.2017.1291920. Mar.

CO-LaN consortium, 2019. Flowsheet Monitoring Interface Specification'. CO-LaN. Jul.

CoLAN Consortium, 2011. CAPE-OPEN Thermodynamic and Physical Properties v1.1'. CoLAN Consortium. May 10.

Cyganiak, R., Wood, D., Lanthaler, M. 2014, 'RDF 1.1 Concepts and Abstract Syntax', W3C Recommendation. Accessed: Dec. 02, 2021. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/.

DIN, 2018. DIN 6779-13 Kennzeichnungssystematik für technische Produkte und Technische Produktdokumentation – Teil 13: Chemieanlagen'. DIN Deutsches Institut für Normung e. V., Berlin. Jan.

Downs, J.J., Vogel, E.F., 1993. A plant-wide industrial process control problem. Comput. Chem. Eng. 17 (3), 245–255.

Drath, R., 2021. AutomationML: The Industrial Cookbook. De Gruyter, Berlin. Accessed: Jul. 07, 2023. [Online]. Available. https://www.degruyter.com/document/doi/10.1515/9783110676693/html.

Fjøsna, E., Waaler, A., 2021. READI Information modelling Framework (IMF). Asset Information Modelling Framework', READI Joint Industry Project, Oslo. Mar [Online]. Available. https://readi-jip.org/wp-content/uploads/2021/03/Information-modelling-framework-V1.pdf.

Flavell-White, C., 2011. Dedicated to industrial progress. The Chemical Engineer 841, 54–56.

Fontalvo, J., 2014. Using user models in Matlab® within the Aspen Plus® interface with an Excel® link. Ing. E Investig. 34 (2), 2. https://doi.org/10.15446/ing.investig.v34n2.41621. ArtMay.

Fricke, A., Schöneberger, J., 2015. Industrie 4.0 with MS-Excel? Chem. Eng. Trans. 43, 1303–1308. https://doi.org/10.3303/CET1543218. Jan.

Garg, N., Kontogeorgis, G.M., Gani, R., Woodley, J.M., 2020. A process synthesis-intensification method for generation of novel and intensified solutions. Chem. Eng. Process. - Process Intensif. 156, 108103 https://doi.org/10.1016/j.cep.2020.108103. Oct.

Grüner, S., Otten, W., 2023. DEXPI Submodel For Industry 4.0 Asset Administration Shell. Towards a Digital Thread through Engineering and Operations', Presented At the 24. VDI-Kongress Automation 2023. Baden-BadenJul.

Grüner, S., Hoernicke, M., Stark, K., Schoch, N., Eskandani, N., Pretlove, J., 2023. Towards asset administration shell-based continuous engineering in process industries. Autom 71 (8), 689–708. https://doi.org/10.1515/auto-2023-0012. Aug.

Hernandez, C., Rodriguez, M., Diaz, I., Sanz, R., Kravanja, Z., Bogataj, M., 2016. Model based engineering of process plants using SysML. In: Computer Aided Chemical Engineering, 38. Elsevier, pp. 1281–1286. https://doi.org/10.1016/B978-0-444-63428-3.50218-6, 26 European Symposium on Computer Aided Process Engineering.

Hu, J., Zhang, L., Cai, Z., Wang, Y., 2015. An intelligent fault diagnosis system for process plant using a functional HAZOP and DBN integrated methodology. Eng. Appl. Artif. Intell. 45, 119–135. https://doi.org/10.1016/j.engappai.2015.06.010. Oct.

IDTA, 2023. 'DEXPI Submodel Template For AAS', GitHub. Accessed: Sep. 06, 2023. [Online]. Available: https://github.com/admin-shell-io/submodel-templates/tree/main/development/DEXPI/1/0.

IEC, 2019. IEC81346-2 Industrial systems, Installations and Equipment and Industrial products: Structuring Principles and Reference designations. Part 2. Classification of Objects and Codes For Classes. IEC, Geneva.

IEC, 2022. IEC81346-1 Industrial systems, Installations and Equipment and Industrial Products - Structuring principles and Reference Designations - Part 1: Basic rules. IEC, Geneva.

IEEE Computer Society, 1998. IEEE Standard for Functional Modeling Language - Syntax and Semantics for IDEF0. IEEE, New York, NY. Nov.

Ingebrigtsen, I.P., Drath, R., 2021. 10 AML Domain Model For System Control Diagrams: Automatic code Generation Through Digitization of the IEC PAS 63131', in 10 AML Domain Model For System Control Diagrams: Automatic code Generation Through Digitization of the IEC PAS 63131, De Gruyter Oldenbourg, pp. 165–196. https://doi.org/10.1515/9783110745979-011.

ISO, 2010. ISO15519-1 Specification for Diagrams For Process Industry — Part 1: General rules. ISOGenevaMar. 01.

ISO, 2012. ISO10628-2 Diagrams for the Chemical and Petrochemical Industry Part 2: Graphical symbols. ISO, GenevaDec. 01.

ISO, 2014. ISO10628-1 Diagrams for the Chemical and Petrochemical Industry Part 1: Specification of Diagrams. ISO, GenevaSep. 15.

ISO, 2019. ISO10418 Petroleum and Natural Gas Industries — Offshore production Installations — Process safety Systems. ISO, GenevaMay.

ISO, 2022. ISO-81346-10 Industrial systems, Installations and Equipment and Industrial Products — Structuring principles and Reference Designations — Part 10: Power supply Systems. ISO, GenevaAug.

ISO TC 184, 2019. Automation Systems and integration, and Subcommittee SC 4, Industrial data, 'ISO/TS 15926-4 Industrial automation Systems and Integration - Integration of Life-Cycle Data For Process Plants Including Oil and Gas Production Facilities - Part 4: Initial reference Data. ISO, Geneva, Technical Specification IS0/TS 15926-4:2019(E)Oct.

Kim, B.C., Jeon, Y., Park, S., Teijgeler, H., Leal, D., Mun, D., 2017. Toward standardized exchange of plant 3D CAD models using ISO 15926′. Comput.-Aided Des 83, 80–95. https://doi.org/10.1016/j.cad.2016.10.005. Feb.

Kim, B.C., Kim, B., Park, S., Teijgeler, H., Mun, D., 2020. ISO 15926–based integration of process plant life-cycle information including maintenance activity. Concurr. Eng. 28 (1), 58–71. https://doi.org/10.1177/1063293X19894041. Mar.

Leal, D., 2005. ISO 15926 "Life Cycle Data for Process Plant": an Overview. Oil Gas Sci. Technol. 60 (4), 629–637. https://doi.org/10.2516/ogst:2005045. Jul.

vol. 52 Mann, V., Gani, R., Venkatasubramanian, V., Kokossis, A.C., Georgiadis, M.C., Pistikopoulos, E., 2023. Intelligent Process Flowsheet Synthesis and Design using Extended SFILES Representation. In: Computer Aided Chemical Engineering, 52. Elsevier, pp. 221–226. https://doi.org/10.1016/B978-0-443-15274-0.50036-6. vol. 5233 European Symposium on Computer Aided Process Engineeringvol.

Marquardt, W., Morbach, J., Wiesner, A., Yang, A., 2010. Chemical Process Systems. *OntoCAPE*, in RWTHedition. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 241–321. https://doi.org/10.1007/978-3-642-04655-1_8.

Marquardt, W., 2010. *OntoCAPE: a Re-Usable Ontology For Chemical Process engineering.* in RWTH Edition. Springer, Heidelberg ; New York.

Martinez-Hernandez, E., 2023. Digitalisation of chemical processes as graphs and applications of modular decomposition to process design and analysis. Digit. Chem. Eng. 6, 100075 https://doi.org/10.1016/j.dche.2022.100075. Mar.

Mencarelli, L., Chen, Q., Pagot, A., Grossmann, I.E., 2020. A review on superstructure optimization approaches in process system engineering. Comput. Chem. Eng. 136, 106808 https://doi.org/10.1016/j.compchemeng.2020.106808. May.

Microsoft, 2022. Digital Twins Definition Language'. Microsoft Azure. Jan. 02Accessed: Jan. 04, 2022. [Online]. Available. https://github.com/Azure/opendigitaltwins-dtdl/blob/fba3c79b9c2363b0b44bd5d85cb958371aaa847f/DTDL/v2/dtdlv2.md.

NAMUR, 2014. NE150 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'. NAMUR - Interessengemeinschaft Automatisierungstechnik Der Prozessindustrie e.V. LeverkusenOct. 13.

NAMUR, 2018. NE159 Standardized NAMUR Interface For Data Exchange Between CAE Systems For Process Design and CAE Systems For PCT Hardware Planning'. NAMUR - Interessengemeinschaft Automatisierungstechnik Der Prozessindustrie e.V. LeverkusenFeb. 27.

NAMUR, 2021. NE100 Merkmalleisten und Deren Nutzung in PLT-Engineering-Workflows'. NAMUR - Interessengemeinschaft Automatisierungstechnik Der Prozessindustrie e.V., Leverkusen. Oct. 21.

Nzetchou, S., Durupt, A., Remy, S., Eynard, B., 2019. Review of CAD visualization standards in PLM. In: Fortin, C., Rivest, L., Bernard, A., Bouras, A. (Eds.), Product Lifecycle Management in the Digital Twin Era. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, pp. 34–43. https://doi.org/10.1007/978-3-030-42250-9_4.

O'Halloran, B.M., Papakonstantinou, N., Giammarco, K., Van Bossuyt, D.L., 2021. A graph theory approach to predicting functional failure propagation during conceptual systems design. Syst. Eng. 24 (2), 100–121. https://doi.org/10.1002/sys.21569.

Oeing, J., Brandt, K., Wiedau, M., Tolksdorf, G., Welscher, W., Kockmann, N., 2023. Graph learning in machine-readable plant topology data. Chem. Ing. Tech. 95 (7), 1049–1060. https://doi.org/10.1002/cite.202200223.

Plattform Industrie 4.0, 'Plattform Industrie 4.0 Glossary'. Accessed: Aug. 04, 2023. [Online]. Available: https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/Glossary/glossary.html.

Ponce-Ortega, J.M., Hernández-Pérez, L.G., 2019. Optimization of Process Flowsheets Through Metaheuristic Techniques. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-91722-1.

Preisig, H.A., 2009. A graph-theory-based approach to the analysis of large-scale plants. Comput. Chem. Eng. 33 (3), 598–604. https://doi.org/10.1016/j.compchemeng.2008.10.016. Mar.

Rantala, M., Niemistö, H., Karhela, T., Sierla, S., Vyatkin, V., 2019. Applying graph matching techniques to enhance reuse of plant design information. Comput. Ind. 107, 81–98. https://doi.org/10.1016/j.compind.2019.01.005. May.

READI JIP, 2021. 'Reference Designation System For Oil and Gas – READI'. Accessed: Dec. 21, . [Online]. Available: https://readi-jip.org/reference-designation-system-for-oil-and-gas/.

Rodríguez, M., de la Mata, J.L., 2012. Automating HAZOP studies using D-higraphs. Comput. Chem. Eng. 45, 102–113. https://doi.org/10.1016/j.compchemeng.2012.06.007. Oct.

Romatier, C., Huang, R., Klecka, R., 2015. Utilizing Spreadsheet User Interfaces with Flowsheets of a CPI Simulation System', US 9,053,260 B2. Jun. 09.

Rumbaugh, J., Jacobson, I., Booch, G., 2004. Unified Modeling Language Reference Manual, The (2nd Edition). Pearson Higher Education.

Sierla, S., Sorsamäki, L., Azangoo, M., Villberg, A., Hytönen, E., Vyatkin, V., 2020. Towards semi-automatic generation of a steady state digital twin of a brownfield process plant. Appl. Sci. 10 (19), 19. https://doi.org/10.3390/app10196959. ArtJan.

Sierla, S., et al., 2021. Roadmap to semi-automatic generation of digital twins for brownfield process plants. J. Ind. Inf. Integr., 100282 https://doi.org/10.1016/j.jii.2021.100282. Sep.

Tauchnitz, T., 2022. *MTP Automation of Modular Plants*, 1st Edition. Vulkan Verlag, Essen.

Thambirajah, J., Benabbas, L., Bauer, M., Thornhill, N.F., 2009. Cause-and-effect analysis in chemical processes utilizing XML, plant connectivity and quantitative process history. Comput. Chem. Eng. 33 (2), 503–512. https://doi.org/10.1016/j.compchemeng.2008.10.002. Feb.

van Baten, J., Pons, M., 2014. CAPE-OPEN: interoperability in Industrial Flowsheet Simulation Software. Chem. Ing. Tech. 86 (7), 1052–1064. https://doi.org/10.1002/cite.201400009.

VDI/VDE, 2015a. VDI/VDE 3682 Blatt 1 Formalisierte Prozessbeschreibungen Konzept Und Grafische Darstellung Part 1 Formalised process Descriptions Concept and Graphic Representation. VDI Verein Deutscher Ingenieure, Berlin. May.

VDI/VDE, 2015b. VDI/VDE 3682 Blatt 2 Formalisierte Prozessbeschreibungen Informationsmodell /Part 2 Formalised process Descriptions Information model. VDI Verein Deutscher Ingenieure, Berlin. May.

Venkatasubramanian, V., Zhao, J., Viswanathan, S., 2000. Intelligent systems for HAZOP analysis of complex process plants. Comput. Chem. Eng. 24 (9), 2291–2302. https://doi.org/10.1016/S0098-1354(00)00573-1. Oct.

Vogel, G., Balhorn, L.S., Hirtreiter, E., Schweidtmann, A.M., 2022. SFILES 2.0: an extended text-based flowsheet representation. arXiv. https://doi.org/10.48550/arXiv.2208.00778. Jul. 25.

Vogel, G., Schulze Balhorn, L., Schweidtmann, A.M., 2023. Learning from flowsheets: a generative transformer model for autocompletion of flowsheets. Comput. Chem. Eng. 171, 108162 https://doi.org/10.1016/j.compchemeng.2023.108162. Mar.

VVB Chemieanlagen,Leipzig, 1974. TGL 25 000 Blatt 1 Verfahrenstechnik Grundoperationen Klassifikation. Deutsche Demokratische Republik, Berlin. Mar.

W3C, 2020. JSON-LD 1.1', W3C, W3C Recommendation. JulAccessed: Aug. 07, 2023. [Online]. Available. https://www.w3.org/TR/json-ld11/.

Waaler, A., 2022. IMF Asset Information Modelling Framework', Presented At the Join the READI Revolution. OsloMay 24.

Wagner, C., et al., 2017. The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, Limassol, pp. 1–8. https://doi.org/10.1109/ETFA.2017.8247583. Sep.

Wiedau, M., von Wedel, L., Temmen, H., Welke, R., Papakonstantinou, N., 2019. ENPRO data integration: extending DEXPI towards the asset lifecycle. Chem. Ing. Tech. 91 (3), 240–255. https://doi.org/10.1002/cite.201800112.

Wiedau, M., Tolksdorf, G., Oeing, J., Kockmann, N., 2021. Towards a systematic data harmonization to enable AI application in the process industry. Chem. Ing. Tech. 93 (12), 2105–2115. https://doi.org/10.1002/cite.202100203.

Xiao, J., Anwer, N., Durupt, A., Duigou, J.Le, Eynard, B., 2018. Information exchange standards for design, tolerancing and additive manufacturing: a research review. Int. J. Interact. Des. Manuf. IJIDeM 12 (2), 495–504. https://doi.org/10.1007/s12008-017-0401-4. May.