Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

pyDEXPI: A Python framework for piping and instrumentation diagrams (P&IDs) using the DEXPI information model

Dominik P. Goldstein[a], Lukas Schulze Balhorn[a], Achmad Anggawirya Alimin[a], and Artur M. Schweidtmann[a]*

[a] Process Intelligence Research Group, Department of Chemical Engineering, Delft University of Technology, Van der Maasweg 9, Delft 2629 HZ, The Netherlands
* Corresponding Author: a.schweidtmann@tudelft.nl.

ABSTRACT

Developing piping and instrumentation diagrams (P&IDs) is a fundamental task in process engineering. For designing complex installations, such as petroleum plants, multiple departments across several companies are involved in refining and updating these diagrams, creating significant challenges in data exchange between different software platforms from various vendors. The primary challenge in this context is interoperability, which refers to the seamless exchange and interpretation of information to collectively pursue shared objectives. To enhance the P&ID creation process, a unified, machine-readable data format for P&ID data is essential. A promising candidate is the Data Exchange in the Process Industry (DEXPI) standard. We present pyDEXPI, an open-source implementation of the DEXPI format for P&IDs in Python. pyDEXPI makes P&ID data more efficient to handle, more flexible, and more interoperable. We envision that, with further development, pyDEXPI will act as a central scientific computing library for the domain of digital process engineering, facilitating interoperability and the application of data analytics and generative artificial intelligence on P&IDs. We provide the pyDEXPI package with the documentation on GitHub at https://github.com/process-intelligence-research/pyDEXPI.

Keywords: Piping and instrumentation diagram, DEXPI, FAIR data, Data model, Software toolbox, Open-source

1. INTRODUCTION

Piping and instrumentation diagrams (P&IDs) are chemical engineering diagrams that depict the process equipment, piping, pumps, instruments, valves, and further fittings of a chemical plant [1]. They act as a central, interdisciplinary reference document throughout the plant's life cycle including construction, start-up, and operation [2]. For large-scale plants, such as petrochemical processes, P&IDs may span over 1000 pages, containing a vast amount of engineering data.

For such complex projects, many different departments from several companies and parties need to work together to plan, construct, and operate the process plant. The parties involved in this include engineering, procurement, and construction companies, owner-operators, vendors, site services, and authorities. A seamless exchange of P&ID data between contributors is crucial. In reality, however, this seamless exchange is obstructed by the lack of interoperability between different Computer-Aided Engineering (CAE) software systems and other systems used by the participating stakeholders. The software systems use different underlying data representations of the P&IDs, and it is difficult to extract information from these data representations to use in different software.

Missing interoperability of P&ID data representations not only hinders effective engineering workflows but also disrupts the adoption of novel digital process engineering tools, including data analytics or generative artificial intelligence (GenAI) [3]. Recent research works have demonstrated the potential rooted in GenAI to

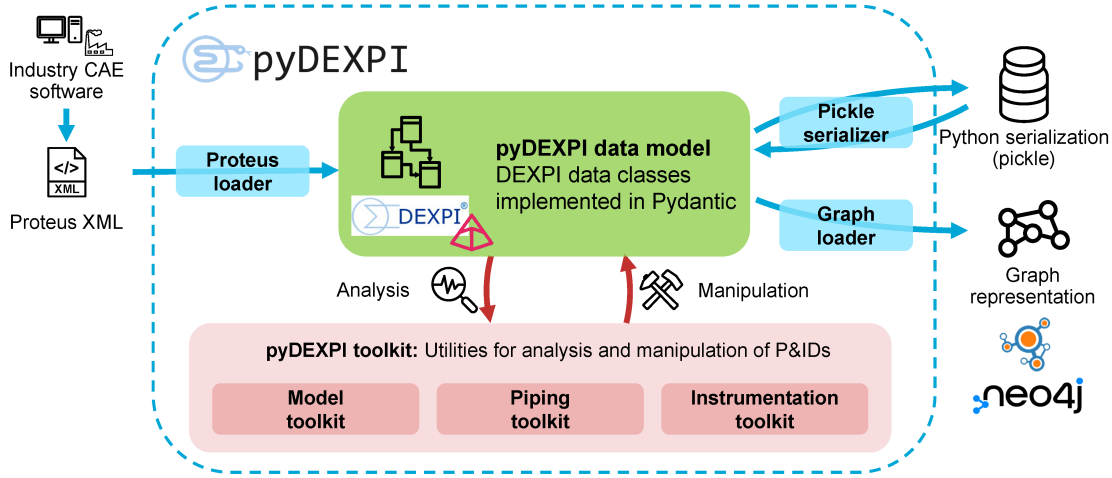

Figure 1: Overview of the pyDEXPI functionality. The Python implementation of the DEXPI data classes lies at the heart of the package (green). The package includes import and export of data (blue), and toolkit functionality for analysis and manipulation (red).

assist, automate, and improve the engineering process [4-9]. However, missing interoperability causes a lack of machine-readable data and hinders the integration of solutions into existing systems. To enable seamless exchange of P&ID data and facilitate integration of novel digital process engineering tools, a unified, machine-readable data format for P&ID data is necessary.

An emerging exchange format for P&ID data is the Data Exchange in the Process Industry (DEXPI) data format [10]. It defines a data model for P&IDs intended to facilitate the handover of data across CAE software. The DEXPI format was developed by the DEXPI initiative, which is supported by several industrial software vendors and chemical engineering companies. However, there is no open-source implementation of the DEXPI class framework. This makes it difficult to use this framework for interoperable data exchange in practice, and the application in digital process engineering.

To address the need for an open-source implementation of the industry-compatible DEXPI framework, we have created pyDEXPI, which implements the DEXPI information model for Python (Figure 1). P&ID data can be imported to pyDEXPI from DEXPI-conform Proteus XML files, saved in Python pickle files, and exported to NetworkX graphs. Within pyDEXPI, a model toolkit allows the analysis and manipulation of P&ID data, e.g., by retrieving certain objects with attributes of interest or reconnecting piping connections. With pyDEXPI, we aim to provide an interoperable and accessible informatics package for P&ID data.

2. DEXPI: DATA EXCHANGE FOR THE PROCESS INDUSTRY

The DEXPI initiative has defined the DEXPI information model with support from industrial software vendors, chemical companies, and academic partners. The objective of the information model is the definition of a system-independent, uniform data model for P&ID data that can be exported and imported by participating CAE software vendors [10]. The established version of DEXPI is Version 1.3, although the newest Version 1.4 was recently published in December 2024. The DEXPI data model defines a set of data classes and their relationships. DEXPI P&IDs are exchanged in the Proteus XML schema [11], therefore, the DEXPI specification also outlines how the data model is mapped into this XML schema.

The DEXPI 1.3 specification defines 473 data classes, data types, and enumerations organized into 11 packages. The packages are *DexpiModel*, *MetaData*, *PlantStructure*, *Equipment*, *Piping*, *Instrumentation*, *Customization*, *Enumerations*, *PhysicalQuantities*, *Graphics*, and *DataTypes*. The classes have five properties that define the class relationships and the stored information. First, a DEXPI class can have a set of supertypes, from which it inherits all properties. Second, a class can have subtypes, to which it passes on its properties. This is the inverse of the supertype relationship. Third, a class has a set of compositional attributes, which are lower-level DEXPI classes that the class conceptually consists of. Fourth, a class has reference attributes, which can be all other references to DEXPI classes that do not constitute a composition, such as the target of a pipe. Finally, a DEXPI class can have a set of data attributes, which store the class information in the form of basic data types or enumerations. The data types and enumerations are also defined within the DEXPI framework. For more details on

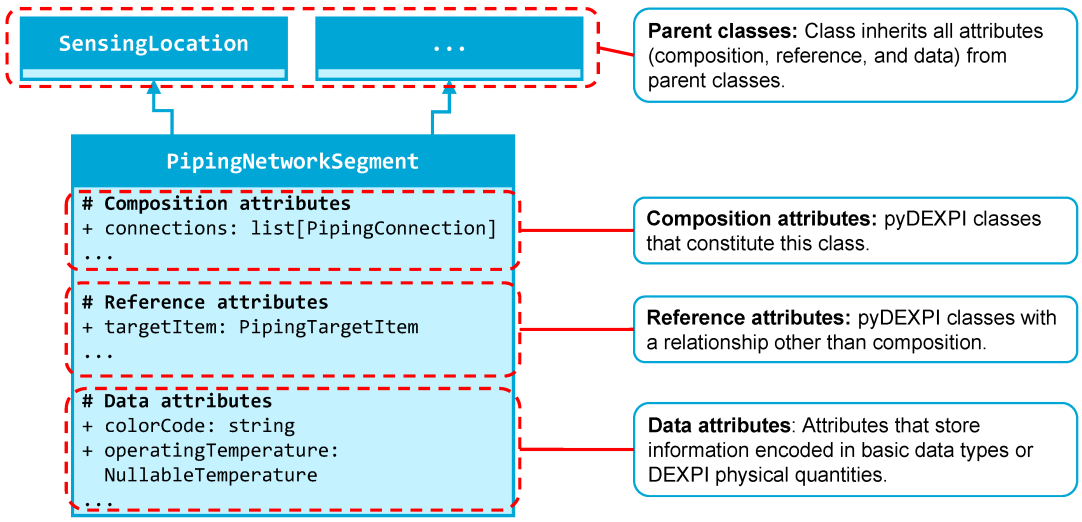

Figure 2: The partial UML diagram of the pyDEXPI class *PipingNetworkSegment* with an example parent class and example attributes.

the DEXPI information model, we refer the reader to the specification of the DEXPI information model [10].

While the DEXPI framework is a promising format for interoperable P&ID data, there was no open-source implementation available. The main, openly available tool for examining data in a DEXPI format is the P&ID Verificator [12]. This application has a comprehensive user interface to visualize a P&ID, investigate the underlying data model, and display verification messages. However, it is not open-source and has no interfaces for other software, which prevents it from being used for further applications beyond DEXPI data verification. Typically, DEXPI data is only exchanged via the Proteus XML schema. However, this is difficult to apply in practice due to the inconsistent design of Proteus XML and DEXPI [10]. Since no prior open-source implementation exists, the value of DEXPI for interoperability and its applicability for GenAI is limited.

3. THE PYDEXPI PYTHON PACKAGE

The pyDEXPI package provides an implementation of the DEXPI class framework in Python. Data can be imported and exported in several formats. To make the data more accessible to users, pyDEXPI also includes functionality to analyze and manipulate P&ID data. An overview of the functionality of pyDEXPI is depicted in Figure 1.

3.1. Implementation of the DEXPI information model as Python data classes

At the heart of pyDEXPI lie the data classes of the DEXPI information model. They are implemented as data classes of the Pydantic Python package [12], which specializes in data modeling and validation. The classes are organized into Python packages corresponding to the DEXPI framework's packages. All pyDEXPI classes inherit from a common base class called *DexpiBaseModel*, which defines some common behavior to use them effectively in Python programming. This includes defining a UUID for each class instance and assigning a new UUID if an object is copied. The data types of DEXPI are also implemented as Pydantic classes but inherit from a different common base class *DexpiDataTypeBaseModel*. This is because data types require different behavior than normal data classes. For example, a data class does not require an ID, but an evaluation of equality. The DEXPI classes representing basic data types, such as strings and integers, are omitted and replaced with the corresponding inbuilt data types of Python. The enumerations of DEXPI are implemented as Python enumerations.

The pyDEXPI classes implement the characteristics of the DEXPI classes with Python programming concepts. The class supertype relationship is implemented using regular class inheritance. Each class has the pyDEXPI classes corresponding to its supertypes as parent classes. This also covers the inverse, subtype relationship of DEXPI. Furthermore, each pyDEXPI class defines a set of attribute fields. The attribute fields have type annotations with the required data type defined in DEXPI, and the type annotations are enforced by Pydantic data validation. To distinguish between composition, reference, and data attributes, the pyDEXPI attribute fields are annotated with an additional indicator using the Pydantic field annotations functionality. The indicator *attribute_category* takes the values *composition*, *reference*, or *data*.

To illustrate the structure of a typical pyDEXPI class, Figure 2 depicts the partial UML diagram of the pyDEXPI

class *PipingNetworkSegment*. A *PipingNetworkSegment* is a child of *SensingLocation*, among others, and inherits all its attributes. A *PipingNetworkSegment* consists, among others, of *PipingConnections* which compose the segment as compositional attributes. In contrast, a *PipingNetworkSegment* references a *PipingTargetItem* as its target. The target item, which may be an equipment nozzle, for example, is in the compositional hierarchy of a different class. Finally, a *PipingNetworkSegment* contains a set of data attributes, which are stored in basic data types or pyDEXPI physical quantities.

3.2. Importing and exporting data with pyDEXPI

While the pyDEXPI class model established the structural foundation of the package, pyDEPXI also includes functional components to enable practical application for data import, export, and manipulation. These are essential for the effective utilization of pyDEXPI in data science applications. Most importantly, this includes importing and exporting of P&ID data in different formats.

pyDEXPI includes several possibilities for importing, exporting, loading, and saving data. The most notable import functionality is the import of P&ID data from DEXPI-conform Proteus XML files. This interface enables importing data extracts from DEXPI-compliant CAE tools, which unlocks access to a vast repository of P&ID data. Further, pyDEXPI supports saving and loading data quickly and easily in a Python pickle format.

Although pyDEXPI proposes a unified framework for the handling of P&ID data, downstream data analytics tasks require P&ID data in different formats and granularities to fit the needs of the method applied [3]. pyDEXPI supports exporting data as a graph, in the form of a NetworkX graph object. A P&ID is expressed as a graph intuitively since P&IDs depict the piping and signal connections between process equipment. Furthermore, a graph representation facilitates the application of novel graph machine-learning techniques for advanced P&ID tasks [3]. While the graph export is a very powerful export format, we envision a plethora of additional export formats, such as sequences like the SFILES 2.0 format [14], tables, etc. to accommodate more data representation requirements.

3.3. The pyDEXPI toolkit: Analyzing and manipulating P&ID data

Next to importing and exporting P&ID data, pyDEXPI offers some functionality to analyze and manipulate the data. The DEXPI framework of over 400 classes is extensive and includes complex class relationships and implicit rules. Therefore, it is difficult to find and retrieve data of interest from a model. The pyDEXPI toolkit offers methods to maneuver the class framework, e.g., by retrieving all objects in a model of a certain type or with certain attribute values. Further, the toolkits help assist in the modification of pyDEXPI P&IDs by automatically handling the design rules of the DEXPI framework.

The functions are organized in modules with respect to their scope. Model-level functions are grouped into the *model_toolkit*, functions about piping are in the *piping_toolkit*, and functions for instrumentation are found in the *instrumentation_toolkit*.

4. OUTLOOK: PYDEXPI IN PRACTICE

We envision pyDEXPI as a central computing library for P&IDs, accommodating expert usability, system interoperability, and data analytics and AI applications. Our vision is illustrated in Figure 3.

pyDEXPI already offers several utilities for P&ID data interoperability. On the one hand, integration with existing systems is available via the import of DEXPI-conform Proteus XML files. On the other, the pyDEXPI toolkit acts as a programming interface that makes P&ID data more accessible to users. We anticipate an expansion of the current functionality, e.g., by exporting data back into the interoperable Proteus XML format or by including a graphical interface for direct user interaction with P&ID data. Furthermore, we expect an expansion of pyDEXPI's capabilities for importing and exporting P&ID data in formats suitable for data analytics and AI tasks.

In some of our current work, we already demonstrate the value of pyDEXPI for data analytics and generative AI projects [3]. We envision that computer vision algorithms [15] with digitize image-/paper-based P&IDs and represent them as pyDEXPI P&IDs in the future. This makes them available to further, downstream GenAI applications. Schulze Balhorn et al. [16] propose autocorrection of P&ID based on a graph representation that is generated automatically from pyDEXPI. The algorithm detects errors in P&IDs through rule- and AI-based methods and replaces these with a corrected version. Alimin et al. [17] use P&ID graphs from pyDEXPI as inputs to large language models (LLM). This approach enables the LLM to retrieve information from the P&ID during text generation. Engineers interacting with such LLMs can easily access P&ID information through conversational queries, eliminating the need for manual lookup. This can assist with many time-intensive engineering tasks, such as hazard and operability studies (HAZOPs). In a notable contrast to previous work on AI on P&IDs, these case studies can leverage the interoperability of the pyDEXPI framework to interact with other tools and services, and potentially be integrated as plugins or services in different, existing CAE software with limited effort. They will also profit directly from future improvements to the framework.

We expect many more applications of pyDEXPI-based data representations in the future. For the existing

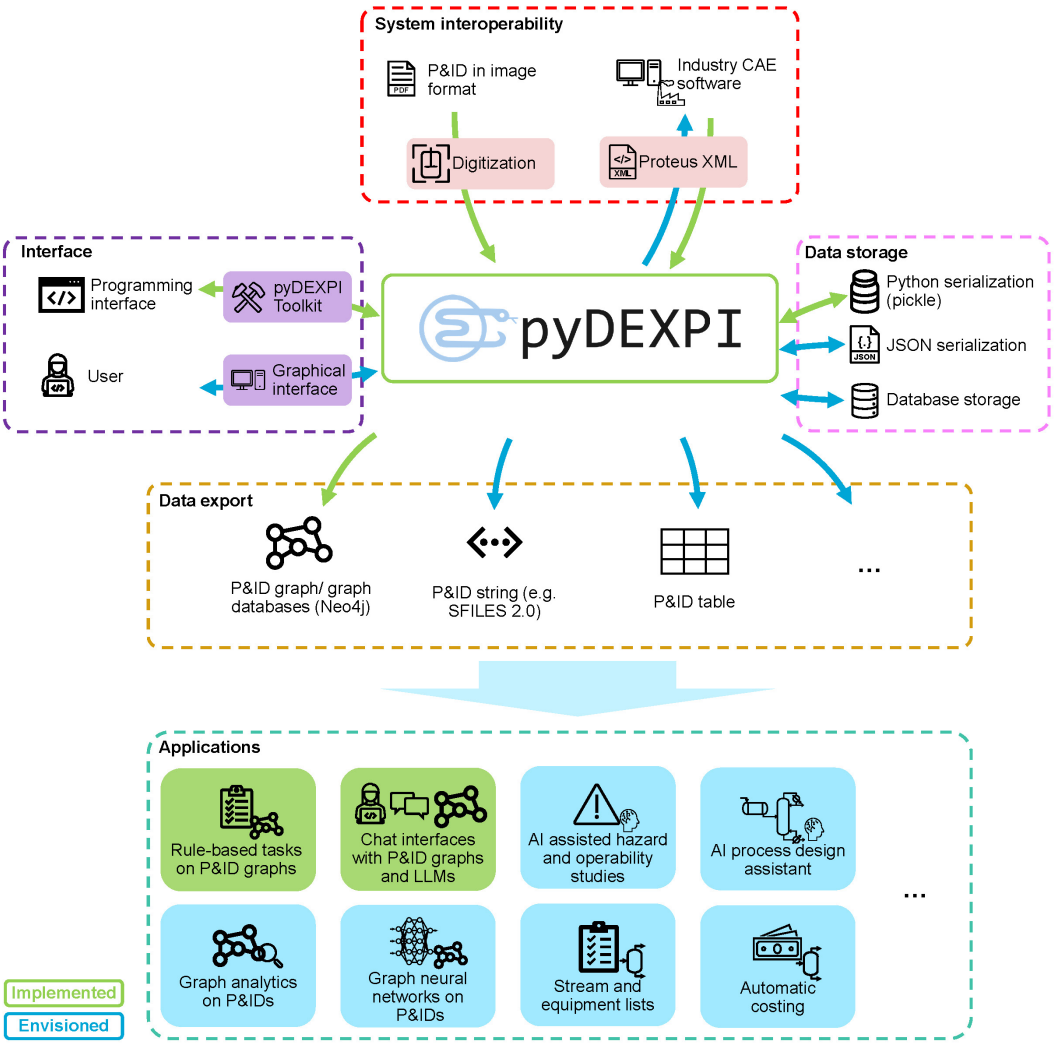

Figure 3: Our vision for pyDEXPI: A powerful P&ID informatics package that enables (1) interoperable data exchange between industry systems and import of digitized P&ID images (2) interfacing with users and other systems for P&ID analysis and manipulation (3) Saving and loading P&ID data in a variety of formats, depending on the requirements (4) Exporting P&ID data to a multitude of machine-readable data formats suitable for downstream data analytics, AI, and GenAI applications.

export of graph representations, future applications can include the use of graph analytics methods or graph neural networks on P&IDs. Additionally, we expect additional data export formats to become available in the future, such as sequences, strings, or tables. This will allow researchers and industry to select the most suitable data representation for their use cases and easily exchange their output with further tools or existing CAE software.

5. CONCLUSION

Addressing the need for interoperability for P&ID data, we present pyDEXPI, an open-source Python implementation of the DEXPI specification. pyDEXPI implements the data classes outlined by the DEXPI information model, which is widely supported by industry and research. pyDEXPI can import DEXPI data from the Proteus XML format, save and load it from a Python pickle file, and export it to a graph format suitable for graph data analytics and graph machine learning tasks. We anticipate a substantial expansion of pyDEXPIs data import, export, analysis, and manipulation capabilities, and envision pyDEXPI to become a central data analytics package for P&ID data. Finally, we foresee that pyDEXPI will be an enabling software for future development and adoption of GenAI technologies for the process industry [3].

6. ACKNOWLEDGEMENTS

We gratefully acknowledge the support provided by Linde GmbH, Linde Engineering and Siemens Aktiengesellschaft (DPG), Fluor© and the Dutch government via the Top Sector Alliance for Knowledge and Innovation (TKI) (CHEMIE.PGT.2023.033) (LSB), as well as the Indonesia Endowment Fund for Education Agency LPDP (AAA).

7. REFERENCES

1. Towler GP, Sinnott RK. Chemical Engineering Design. Elsevier/Butterworth-Heinemann (2008).
2. Toghraei M. Piping and Instrumentation Diagram Development, First Edition. John Wiley & Sons, Inc. (2019).
3. Schweidtmann AM. Generative artificial intelligence in chemical engineering. *Nature Chemical Engineering* 1:193–193 (2024) https://doi.org/10.1038/s44286-024-00041-5
4. Oeing J, Welscher W, Krink N, Jansen L, Henke F, Kockmann N. Using artificial intelligence to support the drawing of piping and instrumentation diagrams using DEXPI standard. *Digital Chemical Engineering* 4:100038 (2022) https://doi.org/10.1016/j.dche.2022.100038
5. Nabil T, Le Moullec Y, Le Coz A. Machine learning based design of a supercritical CO2 concentrating solar power plant. In: Conference Proceedings of the European sCO2 Conference 3rd European Conference on Supercritical CO2 (sCO2) Power Systems 2019: 19th–20th September 2019. 148-157 (2019) https://doi.org/10.17185/DUEPUBLICO/48885
6. Gowaikar S, Iyengar S, Segal S, Kalyanaraman S. An agentic approach to automatic creation of P&ID diagrams from natural language descriptions. *arXiv preprint* (2024) https://doi.org/10.48550/ARXIV.2412.12898
7. Schulze Balhorn L, Caballero M, Schweidtmann AM. Toward autocorrection of chemical process flowsheets using large language models. In 34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering. 3109-3114 (2025)
8. Vogel G, Schulze Balhorn L, Schweidtmann AM. Learning from flowsheets: a generative transformer model for autocompletion of flowsheets. *Computers & Chemical Engineering* 171:108162 (2023) https://doi.org/10.1016/j.compchemeng.2023.108162
9. Hirtreiter E, Schulze Balhorn L, Schweidtmann AM. Toward automatic generation of control structures for process flow diagrams with large language models. *AIChE Journal* 70(1) (2023) https://doi.org/10.1002/aic.18259
10. Theißen M, Wiedau M. DEXPI P&ID Specification. DEXPI Initiative (2021).
11. Proteus XML. Proteus schema for P&ID exchange. https://github.com/ProteusXML/proteusxml
12. pnb plants & bytes GmbH. PID Verificator 1.0.1 for DEXPI 1.3. Software (2025) https://www.plants-and-bytes.de/en/p-id-and-dexpi
13. Colvin S. Pydantic. https://github.com/pydantic/pydantic
14. Vogel G, Hirtreiter E, Schulze Balhorn L, Schweidtmann AM. SFILES 2.0: an extended text-based flowsheet representation. *Optimization and Engineering* 24:2911–2933 (2023) https://doi.org/10.1007/s11081-023-09798-9
15. Theisen MF, Flores KN, Schulze Balhorn L, Schweidtmann AM. Digitization of chemical process flow diagrams using deep convolutional neural networks. *Digital Chemical Engineering* 6:100072 (2023) https://doi.org/10.1016/j.dche.2022.100072
16. Alimin AA, Goldstein DP, Schulze Balhorn L, Schweidtmann AM. ChatP&ID: talking to P&IDs through large language models and knowledge graphs. In: ESCAPE35 - Proceedings of the 35th European Symposium on Computer Aided Process Engineering, Ghent, Belgium. (2025)
17. Schulze Balhorn L, Seijsener N, Dao K, Kim M, Goldstein DP, Driessen GHM, Schweidtmann AM. Rule-based autocorrection of piping and instrumentation diagrams (P&IDs) on graphs. In: ESCAPE35 - Proceedings of the 35th European Symposium on Computer Aided Process Engineering, Ghent, Belgium. (2025)