2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

INTEGRATION OF PARAMETRIC PRODUCT CATALOGUE REQUIREMENTS IN IFC

Noemi Kremer[1], Malte Junge[1], Erich Domme[1], and Jakob Beetz[1]
[1]RWTH Aachen University, Germany

Abstract

An international, standardised product catalogue format representing parametric and variable product data, simplifying their exchange and integration, is to be developed. We map product data requirements in IFC files by leveraging lessons from existing national standards to enhance their parametric modelling capabilities. Using a case study, we evaluate the implementation of property value dependencies and valid combinations using a product catalogue example. Important is the extension of data linking and value calculation in IFC, which integrates control structures. Of the three proposed implementation approaches, we favour the code-based solution to support the integration of parametrics in IFC for different applications.

Introduction

In the Architecture, Engineering, Construction, and Operations (AECO) industry, the creation, distribution, and processing of flexible and parametric Mechanical, Electrical, and Plumbing (MEP) product data are labour-intensive and time-consuming tasks. Since no uniformly standardised catalogue file format currently exists for exchanging and comparing product data, information gaps and data loss impact the exchange between different process participants (Kremer et al., 2023).

For MEP product customers (e.g. architects, project engineers, and facility managers), searching and processing product information for building projects is time-consuming manual work, since the number of possible property value combinations yields thousands of producible product variants, offered by different manufacturers. Additionally, product information is provided in various data formats that are accessible on diverse platforms (e.g. product data portals, manufacturer websites, etc.). Thus, customers rely on established manufacturers and workflows, utilising product data in a workflow-supporting format (e.g. Revit Family) or PDF-based product catalogue and data sheet files (personal communication, expert interviews, February - March 2025).

To ensure open data exchange of building model information, the exchange format Industry Foundation Classes (IFC) were developed and standardised in ISO-16739 (2024). The description of semantic information on building-related objects (e.g. products, spaces) in a machine-readable format is realised as Product Data Templates (PDT) defined in ISO-23387 (2020). MEP products are characterised by dependent parametric properties whose values are calculated based on other static property inputs, selection hierarchies and parametrics. None of the existing international and open-source formats for building information adequately address the structural needs and requirements for mapping MEP product data catalogues, as they focus on entire buildings and construction sites (Wagner et al., 2022).

However, in Germany, a national standard exists that was developed by the engineering and research association *Verein Deutscher Ingenieure* (VDI) for mapping Heating, Ventilation and Air Conditioning (HVAC) product data. VDI 3805 standardises an open-source product catalogue file format for exchanging software-independent and machine-readable manufacturer catalogues (VDI-3805-1, 2022). The format addresses the extensive requirements to implement dependent, flexible and parametric product properties. However, the restriction to HVAC products in the German AECO industry and thus the narrowed scope of the existing software limits the possibilities of application and integration into the Building Information Modelling (BIM) planning process.

To improve product data preparation, provision, search, configuration, and processing, the ISO 16757 working group is developing a multi-part standard that builds on existing and widely used standards and formats (ISO-16757, 2019). ISO 16757-1 outlines the differing property categories (selection-, catalogue-, and product-specific) for product configuration. ISO 16757-2 introduces the product geometry, model representation (both 3D and symbolic) and additional spatial information, such as installation and revision spaces. While ISO 16757 Parts 1 and 2, are published, Parts 4 and 5 are currently under development and revision. Through close dialogue with the ISO 16757 working group and a shared research project called *Parametric BIM for the sustainable energy transition - International market launch of EN ISO 16757*, we gained access to the current state of development of these unreleased parts. In Part 4, the characteristics and requirements of a Data Dictionary (DD) structure are presented, while in Part 5, the catalogue exchange format, based on IFC, is introduced.

Since IFC is a machine-readable and widely adopted format integrated into building design exchange processes, for which software applications already exist, adopting IFC as a product data catalogue enables the expansion of existing infrastructure and simplifies data exchange and integration in planning and execution processes.

Compared to the VDI 3805 standard, which includes schema components for mapping hierarchies, dependencies, and parametrics, the IFC format lacks these structural properties. Therefore, in this work, we disassemble and evaluate the content and structure of the VDI 3805 schema and openly accessible manufacturer catalogues. We are focussing this analysis on particularly diverse, dependent, and highly redundant product-specific properties. As an exemplary MEP product class for our application case, we have chosen heat pumps, due to their technical complexity and wide range of interdependent parameters. To apply the collected results to IFC, the question arises of how to implement the linkage of these dependent properties to other product data, such as static selection parameters and dynamic calculation formulas. We therefore develop and evaluate three options for integrating a parametric approach into the IFC structure by storing calculations and dependencies.

Existing formats and previous research

Previous research

Various efforts have been made to improve the parametric characteristics of IFC files through multiple schemas and structural extensions. In addition, integrating MEP product data into building design and construction processes has been the focus of numerous research works.

Ji et al. (2011) address the matter of parametric IFC by presenting in their work an object-oriented data structure extension of *IFC-Bridge*, supporting especially parametric geometry representations. Since in IFC, geometric representation is restricted to explicit extrusion and CSG approaches; they suggest using geometric constraints and mathematical expressions to realise dependencies between geometric entities and dimensions.

The inclusion of product data into building information modelling workflows requires such data to be open, queryable, machine-readable, filterable, flexible, exhaustive, modular, multi-lingual and domain-independent (Wagner et al., 2022). Kebede et al. (2022) address the ability to filter products by mapping heterogeneous, distributed catalogues of building-related products. The implemented process integrates product data into BIM authoring tools using semantic web technology and a visual programming language approach. However, the utilised product data file is prerecorded and ordered. The product selection is performed by the customer, selecting properties hierarchically.

Bac et al. (2021) utilised a hybrid-criteria decision method to comprehensively evaluate Heating, Ventilation, and Air Conditioning (HVAC) systems for an industrial building use case. The defined and categorised twenty-seven criteria are the basis for assessing eleven preselected HVAC systems. While they focused on the best choice determination according to BIM-based simulation and a criteria weighting system, the eleven products' preceding selection and configuration process has not been investigated.

VDI 3805

VDI 3805 provides a format for modelling and exchanging manufacturer product catalogue data in a machine-readable format, facilitating integration into digital planning and simulation processes. The standard is characterised by a modular tree structure comprising multiple parts that address distinct product classes and the grouping of their properties into so-called *record types* (property sets).

VDI 3805 Sheet 1 establishes the fundamental principles and the structural framework of the product data format, delineating methodologies for the uniform structure and exchange of semantic, geometric, and media data (e.g. images, videos, audio files), as well as parametric functions. The hierarchical structure is organised numerically in ascending order. Properties are summarised in categories, representing a product's characteristics (technical and functional properties and geometric data). Properties that are valid for all MEP product classes (e.g. geometric solids) are stored in Sheet 1, while information that pertains only to a particular product class is defined in the respective product data sheets, extending the schema with class-specific properties.

The requirements for heat pumps, the relevant case study product class of this research, are specified in VDI 3808 Sheet 22. VDI-3805-22 (2019) introduces heat pump selection properties specifying the area of application (e.g. heating, domestic hot water heating, etc.), type (e.g. brine-water, water-water, etc.), heat source, design, and drive energy, classifying products based on common functional characteristics. Functions (calculating heating and cooling output) and their parameters are outlined for calculating values of dependent properties. The section on product element-specific properties is the most comprehensive, since element-specific properties further specify product objects through individual technical values. The application of element-specific property sets partly depends on the previously introduced selection properties.

IFC product catalogue

The ISO-16757 working group aims to adopt the VDI 3805 approach by transferring concepts and workflows to IFC, expanding the standard to an international level without inventing a new schema or format.

The actual exchange of manufacturer product data for one product class will be conducted via the IFC Project Library File, representing an electronic, machine-readable product catalogue (Kremer et al., 2023). The IFC Project Library is a Model View Definition (MVD) of the IFC schema standardised in ISO-17549-2 (2021).

Within the MVD, various entities and relations exist to

describe and reference the catalogue information. The `IfcBuildingProxyElement` represents the product class (e.g. heat pump), while all required properties are created using the `IfcPropertySingleValue` entity and grouped by `IfcPropertySets`. Predefined, agreed properties are defined and described in an external Data Dictionary (DD), and all IFC entities refer to the DD using a URL-based `IfcLibraryReference` entity. Manufacturers can add custom properties and values, which are not part of the DD.

To enable product selection and configuration processes and the determination of valid property value combinations, ISO 16757 proposes the use of `IfcTables` entities for storing and `IfcConstraints` and `IfcAppliedValue` for restricting values. A single-column `IfcTable` is generated in IFC to store multiple valid values of a single property. A multi-column `IfcTable` displays different combinations of property values in a coherent manner. A table column is named after the property it is referencing, while the column fields contain possible property values, and a table row indicates a valid combination of different property values (see Figure 1).

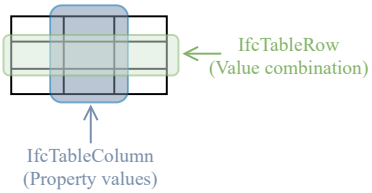

Figure 1: IFC Table structure and information storage

Dynamic properties are represented by an `IfcPropertyDependencyRelationship` entity containing *JavaScript*[1] (Language core standardised as ECMAScript (ECMA 262)) for expressing dynamic structural characteristics (ISO-17549-2, 2021) (see Figure 2). These functions require an `IfcComplexProperty` referencing either as input or output value from the assigned properties. The handling of the additional code section is either realised by containing all code within the IFC file as part of the `IfcPropertyDependencyRelationship` or inside the function header as the declaration is kept within IFC while the function body is stored externally. For upcoming software development, those expected to read this electronic catalogue must be able to execute these code sections and implement a selection and configuration based on IFC constraints expressing logic.

Requirement identification

In this research paper, we investigate the mappability of dependent and linked property values in IFC, by comparing three different methods for establishing flexible information linking and calculation. We identify the required linkages between property values and functions that calculate dependent values by analysing the VDI 3805 structure and focussing on the heat pumps product class as a representative case study. This approach aims to test and evaluate the three identified options using an IFC-based product catalogue file with actual product data, thereby validating their practical execution.

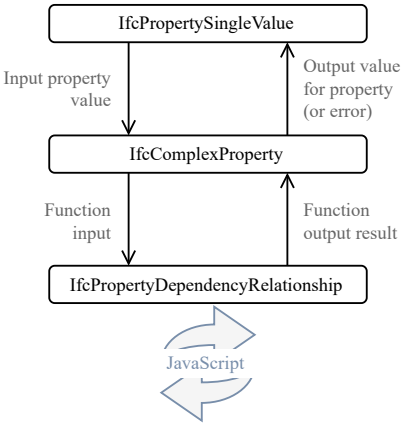

Figure 2: JavaScript in IFC

Initially, we analysed the VDI 3805 file schema in addition to sixty-four product catalogues of various manufacturers made publicly available on the website of the industry association of German heating manufacturers, *Bundesverband der Deutschen Heizungsindustrie* (BDH) [2]. The analysis results outline that heat pumps are one of the most frequently depicted products in all catalogues provided by BDH. Thus, the analysis of VDI 3805 focuses on heat pump product data, particularly considering the catalogue tree structure for the property hierarchy and arrangement, as well as dependent properties and the implementation of their calculation functions.

In VDI 3805 Sheet 22, we examined heat pump-specific properties by compiling statistical data on the distribution of VDI record types. Three of the most frequently occurring record types are product element-specific properties, the geometric description and the functional body definition. In the next step, we analysed the subcategories of product element-specific properties, since their amount and level of detail cause the most data. This is especially true for the subcategories (property sets) *performance data 1* and *performance data 2*, which contain highly dependent operational properties. For example, the heating output depends on the heat pump supply temperature, the temperature of the heat source and the heat pump stage.

According to VDI 3805, performance data 1 sets the static input and limitation parameters (type of output, heat pump supply temperature, maximum and minimum heat source temperature, stage and load) for functions that calculate the heating or cooling output. Output values are stored within performance data 2 (heating or cooling output, heat source temperature, electrical power consumption, COP/ERR). The heat source temperature (performance data 2) is an individual value that lies between a maximum and minimum. While this value is an input parameter, it is not static but changes significantly and thus is stored within

[1] https://tc39.es/ecma262/

[2] https://www.bim4hvac.com/en-gb/

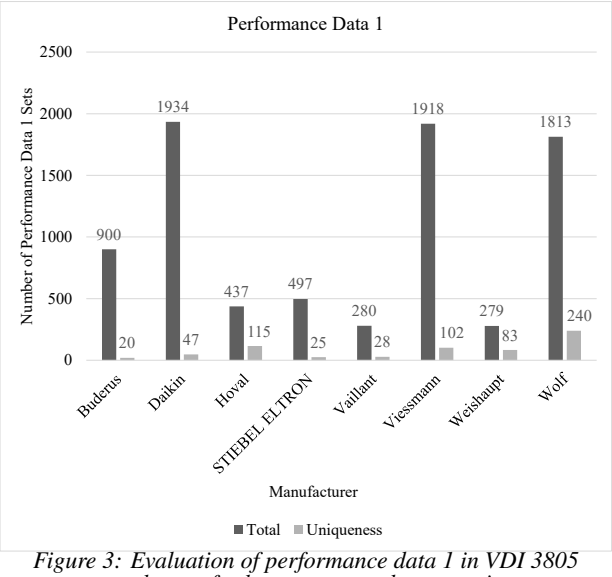

Figure 3: Evaluation of performance data 1 in VDI 3805 catalogues for heat pumps and accessories

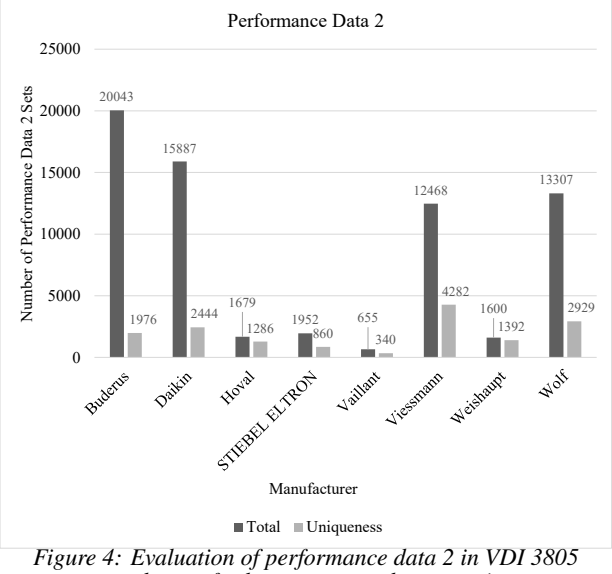

Figure 4: Evaluation of performance data 2 in VDI 3805 catalogues for heat pumps and accessories

the set of performance data 2. However, the manufacturers actually provide a set of general performance data 1 and the calculated results as static performance data 2 sets, without including the calculation functions (headers or bodies).

Since manufacturers list every possible parameter output combination (performance data 2), depending on the alteration of the heat source temperature, in a separate set assigned to its input parameters (performance data 1), they result in being one of the most frequently occurring subcategories, with up to twenty thousand occurrences in a catalogue. A significant result of the analysis is that the hierarchical assignment of record categories and subcategories in the case of performance data 1 and 2 yields apparent redundancies, as illustrated in Figures 3 and 4. While the dark grey bar indicates the total number of sets per manufacturer's catalogue, the number of individual sets, without duplication, is illustrated in light grey.

Due to structural conditions (tree structure), the corresponding performance data 1 (input set) must be assigned to each selection object in a VDI catalogue. These performance data 1 sets are assigned to the results in the performance data 2 set, which results in data redundancies, further increasing the file size. By applying the results drawn from the VDI 3805 evaluation, we identified a need for effective linking, reducing redundant data, and prototypically implementing dependent, computable property values.

For each record type (property set) in VDI3805, a multi-column `IfcTable` is generated in IFC to display different combinations of property values in a coherent manner. Thus, one `IfcTable` combines property values of one property set. To generate a manufacturer-producible product variant, the table rows of different IFC tables must be linked (see Figure 5). Product class-dependent functions that calculate product element-specific property values are a crucial component for the dynamic characteristics of MEP products. Since the manufacturer can individually design functions and valid property value combinations, the question arises of where to store the function body and the links.

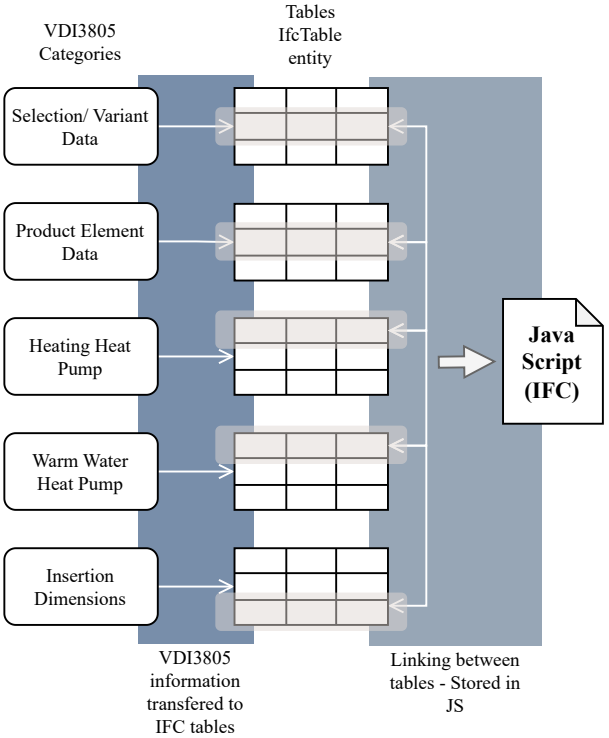

Figure 5: Linking of heat pump table rows

Considering the ISO 16757 working group's suggestions and evaluating the first IFC-based product catalogue implementation approaches, we propose three methods for integrating data linking and dependency calculation, as shown in Table 1. The first option intends to link table rows by referencing their unique keys. The approach includes adding a *key column* and one or multiple columns for storing referenced keys of other table rows. The sec-

ond option considers the manufacturer's current solution of keeping the function declaration out of the catalogue file. The IFC catalogue file serves as data memory, storing static information and references that address external files or databases, where the corresponding functions and links are stored. The third option requires the integration and use of JavaScript in IFC. The second and third options introduce dynamic properties by considering IFC to be a database and introducing an additional layer.

To test and evaluate the property value linkage and dynamic property calculation, part of the data from the catalogue of heat pump manufacturer Viessmann[3] was extracted, parsed and transferred to the IFC catalogue file. According to the preselection, the Viessmann catalogue contains a total of 1628 different heat pump product variants.

Heat pump use-case implementation

We have chosen the heat pump product class as a use-case example, because it is sufficiently complex, and the content preparation for the corresponding part of the ISO 16757 standardisation is under development. Since the preparations have not yet been finalised, we relied on the legacy standard VDI-3805-22 (2019) specifications and categories for implementing and arranging properties. The combination of VDI and IFC content is possible, as there are overlaps in content and structure requirements.

In the following sections, we discuss the use of functions for data linking and dynamic value calculation and demonstrate these concepts using a heat pump example catalogue file (Viessmann). We first simulated the value linking concept intended to be included in ISO 16757, followed by the calculation of dynamic property values using the elaborated implementation approaches shown in Figure 6.

Property value linking

The linking example suggested by the ISO 16757 working group includes the idea of keys (identifiers) for every table row. These table keys are stored in the first column, which is named after the table category (e.g. *geometry* - for the geometry block data), and they are referenced and stored by other tables in an additional table column. A strict hierarchy for linking is required since a table row key can be assigned to multiple other table rows. In addition, varying rows from one table can be assigned to a row of another table, the receiving table to include multiple referencing columns, since an `IfcTable` field cannot store lists. Multiple table columns for one changing field value result in many redundant data records, a problem known from linked tables or database systems.

The second approach we implemented restricts the IFC tables to simple data storage, while the allowed combinations are stored externally. To realise this approach, we introduce a Universally Unique Identifier (UUID) column as a key to identify table rows. Valid product value combinations are stored in a JavaScript Object Notation (JSON)[4] file by mapping the dependent tree structure, referencing the row's UUID.

This approach requires storing the product catalogue as a container file and delivering the necessary information as a package. A modified option is storing the value combinations within the manufacturer's system. When executing the catalogue file, the information is addressed by referencing the manufacturer's Application Programming Interface (API) using a Uniform Resource Locator (URL) stored in an `IfcReference` entity. However, this modified option has only been implemented theoretically and not yet tested.

The third approach involves using JavaScript in IFC to link table rows and determine valid property combinations. The JavaScript code components are stored within an `IfcPropertyDependencyRelationship` entity (see Figure 2), avoiding the obligation to provide additional columns for keys and key references in the tables and, therefore, reducing the total number of table rows. A link was implemented using if-else control structures, *JavaScript* maps and lists. The `IfcPropertySingleValue` entities, *Input* and *Output*, capture the `IfcPropertyDependencyRelationship` requirement of providing a proxy input and output property. Within the JavaScript function, the IFC (internal) identifiers are referenced for linking `IfcTableRow` entities. The implementation of a function for calculation requires a splitting of the underlying performance data structure (VDI3805) to extract the required input parameter and limitations.

Calculating property values

The calculation of heating output was used to examine the implementation of dynamic property values. While the input and output parameters are specified by VDI 3805, the specific calculations are not determined. The manufacturer can determine these individually if the pertinent industry standards do not define calculations. In both cases, the implementation of the function bodies and code structure may differ significantly between manufacturers.

The calculation of dynamic values was implemented similarly to the linking, by creating IFC tables containing valid value combinations within each row and linking these rows within an `IfcPropertyDependencyRelationship` using JavaScript, except for two changes. The function (heat output calculation) was altered and the storage strategy of performance data 2 of VDI 3805 Sheet 22 was changed, by separating the combination of properties, extracting the heat source temperature property values as input parameters and the remaining property values as dynamically calculable. The heat source temperature property values are stored within a one-column table, while all remaining property values of performance data 2 (property) sets are not tabulated.

The function *heating output* is implemented accordingly

[3] https://www.bim4hvac.com/de-de/Produktgruppen?ParticipantId=5

[4] https://json.org/json-de.html

Table 1: Possibilities to manage property value across multiple categories linkages

Index	Method	Storage location
1	IFC table entity: Valid table row combinations stored in an IFC table	IFC file
2	External file (Database, file format): Valid table row combinations stored in additional external repository	External (Manufacturer system; Container file)
3	JavaScript Code: Valid table row combinations dynamically queried using JS implementation	JavaScript in IFC file

within an `IfcPropertyDependencyRelationship` entity. The input parameters *heat pump supply temperature*, *heat source temperature*, *heat pump stage* and *load* and output parameters *power factor heating output*, *heating output*, and *electric power consumption* are created as `IfcPropertySingleValue` and defined as `IfcComplexProperty`. All input values are known and static; all output parameters are unknown values and dynamically calculated.

Results

The presented approach analysed the existing product catalogue file structure of VDI 3805 and tested its applicability for IFC parametric design. Analysing the VDI 3805 catalogue files has shown that the format specifications are not strictly adhered to by the manufacturers when creating the files, and corresponding structural differences can be found in the catalogue data. These deviations were taken into account when analysing the results.

The IFC testing was conducted using an existing heat pump VDI 3805 catalogue file. The implementation focused on linking valid property value combinations, thus integrating control structures and functions for parametrically dependent property values.

Three possible linking strategies were presented, implemented and evaluated. The first approach (Table 1, Index 1), based on `IfcTable` entities, uses additional integrated key columns referenced by other tables. Handling multiple table row assignments was a noticeable issue since IFC tables do not include lists or set data as field input. Using the example of heat pumps, several table rows with value combinations of performance data 2 were assigned to a table row with values of performance data 1. The requirement to store multiple combinations instead of 1-to-1 relationships creates additional columns referencing other rows of the same table or additional rows, as shown in Figure 6. In each case, redundant data records are produced, as only some values in a new column or row are changed.

The second approach (Table 1, Index 2) references external sources to provide links. The external source is either a supplied file containing the links or calculations, which makes the catalogue file a container, or stored within the manufacturer's systems, addressable as an API. A disadvantage here is an additional file to be supplied, whose structure and handling must be determined in addition to the IFC file. The advantage of this implementation is that manufacturers can keep calculations in their system and update and/or archive them at their discretion.

The third approach (Table 1, Index 3) enables the integration of functions and links within a JavaScript file extension in IFC itself. A major adoption is the integration of `IfcTableRow` entities as parameters directly in the *JavaScript* expression (function) in addition to the `IfcProperty` attributes required by an `IfcPropertyDependencyRelationship`. Since we assume all property values to be stored within multi-column tables, a table row contains and provides essential calculation inputs. However, the *Expression* attribute is declared to be an `IfcText` and thus, not further specified or restricted.

The second and third implementation approaches both reduce redundant data. However, it occurred to us that, when implementing dependent property values, it is essential to prevent deadlocks; the input parameters of a function must either be static properties or a fixed hierarchy of the function arrangement must be established.

Conclusion

This paper demonstrated three different approaches for improving parametric and dependent property mapping in IFC, one using additional key columns in IFC tables, another leveraging external references from manufacturers, and a third integrating JavaScript-based functions. Based on examining an existing VDI 3805 MEP product catalogue file, the resulting implementation strategies were applied to the heat pump use case, identifying both promising potential and unresolved issues.

A recurring problems is the complexity of the data structures to be mapped. The introduction of functions illustrates how the complexity of dependencies on the side of the required IFC entities can be reduced by extending the static IFC file format towards a dynamic approach.

Another potential issue to be addressed is the increasing number of table rows as the property value combination rises, when a few properties exhibit noticeable changes in value. Therefore, it is essential to examine and determine which property values are combined in a multi-column table and which are stored in a single-column table

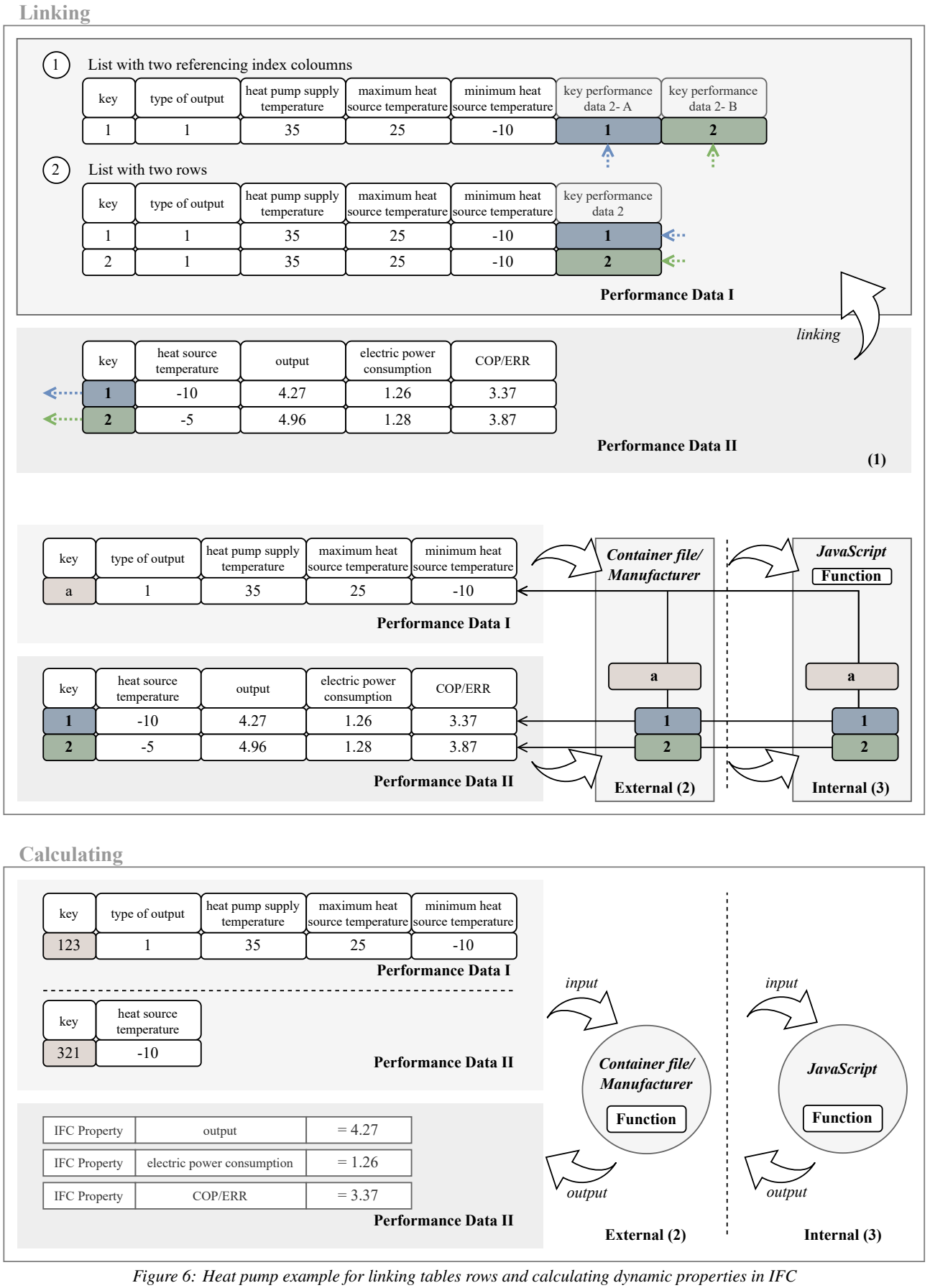

Figure 6: Heat pump example for linking tables rows and calculating dynamic properties in IFC

to achieve data reduction in cases where values frequently change.

Since the IFC format is complex, storing all possible product value combinations results in large files. Complexity can be reduced by storing only static information content in an IFC table entity, while extracting dynamic properties and connecting relationships and storing them in JavaScript code or external repositories. Future software developments must handle the additional code content, provide export and import options, and consider security restrictions that prevent possible damage from being caused by executing JavaScript code.

This research is focussed on calculable values and links for products' semantic data, realising the basic structure to perform configuration on IFC catalogue files. However, we did not address the individual geometry of products or their contextualisation in a system with other MEP products, accessory components and buildings.

Another essential aspect outside this paper's scope is evaluating industrial feasibility. This includes the associated questions of software requirements, implementation guidelines, and best practices (workflow processes for manufacturers and customers). Significantly, best-practice methods and customer-oriented workflow implementation provide a basis for data management and optimised data structures, which must be clarified.

Despite the remaining issues, the current state illustrated in this paper shows that the suggested solution of the uniform and standardised IFC-based catalogue file can improve product data exchange. MEP products are essential components throughout all phases of building planning, construction, and maintenance. They must be uncomplicated, integrable, and flexible to map their own and the building's life cycle.

The development from a static IFC file, representing a snapshot of a current situation, to a dynamically processable structure, linking information on demand and thus supporting configuration concepts, facilitates workflow applications in all AECO domains.

Comparing the three implementation approaches shows that referencing external sources and integrating JavaScript within IFC both reduce redundant data, accommodate dynamic property updates, and offer greater flexibility than the first approach. We therefore advocate for a binary method: either allow functions and dependencies to be stored within the IFC file via JavaScript or reference them directly from the manufacturer's system to avoid additional loosely transported files. By incorporating product data requirements, the IFC format gains new possibilities for integrative, parametric modelling, enabling software vendors, manufacturers, engineers and designers to map parametric product data in building models effectively.

Acknowledgments

This paper is part of the *Parametric BIM for the heat transition - International market launch of EN ISO 16757* project funded by the innovation program *ZukunftBau*. We want to thank our project partners for their versatile support.

References

Bac, U., Alaloosi, K. A. M. S., and Turhan, C. (2021). A comprehensive evaluation of the most suitable hvac system for an industrial building by using a hybrid building energy simulation and multi-criteria decision making framework. 37:102153.

ISO-16739 (2024). Industry foundation classes (ifc) for data sharing in the construction and facility management industries - part 1: Data schema. Deutsche Norm DIN EN ISO 16739.

ISO-16757 (2019). Data structures for electronic product catalogues for building services. Deutsche Norm DIN EN ISO 16757. *Part 1 and 2 pubslished, Part 4 and 5 under development*.

ISO-17549-2 (2021). Building information modelling – Information structure based on EN ISO 16739-1 to exchange data templates and data sheets for construction objects – Part 2: Configurable construction objects and requirements. ISO Norm DIN EN 17549-2. Draft.

ISO-23387 (2020). Building information modelling (BIM) – Data templates for construction objects used in the life cycle of built assets. ISO Norm DIN EN ISO 23387.

Ji, Y., Beetz, J., Bonsma, P., Bisbet, N., Katz, C., and Borrmann, A. (2011). Integration of parametric geometry into ifc-bridge. In Proc. of the 23th Forum Bauinformatik, Cork, Ireland.

Kebede, R., Moscati, A., Tan, H., and Johansson, P. (2022). Integration of manufacturers' product data in BIM platforms using semantic web technologies. Automation in Construction, 144:104630.

Kremer, N., Göbels, A., and Beetz, J. (2023). Use case based implementation of a standardised exchange process for configurable and parametric product data. In 30th International Workshop on Intelligent Computing in Engineering (EG-ICE 2023), London, UK.

VDI-3805-1 (2022). Product data exchange in the Building Services -Fundamentals. Technical Report VDI 3805-1.

VDI-3805-22 (2019). Product data exchange in the building services - heat pumps. Technical Report VDI 3805-22.

Wagner, A., Sprenger, W., Maurer, C., Kuhn, T. E., and Rüppel, U. (2022). Building product ontology: core ontology for linked building product data. Automation in Construction, 133:103927.