IEA HPT Annex 56, Projektnummer 876724

Information model for Heat Pumps

AP3 Daten und Schnittstellen, D3.2

Vienna University of Technology
Informatics, Automation Systems Group

Table of Contents

1 Introduction

1.1 Information Modeling

Computer technologies are used to support and optimize activities throughout a product's lifecycle, with transparent and efficient information exchange between systems being the most critical issue, and formal and unambiguous information modeling languages the major enabler of consistent large-scale, complex, networked computer environments [1]. An information model represents a given domain or application from a specific viewpoint, using concepts as entities, attributes and relationships, as well further concepts derived from them. The concept of a viewpoint is exceptionally important, as it reflect the specificity of a certain model and helps to define and limit its scope. A further important aspect of information modeling is the choice of modeling technologies, as it has a direct impact on the complexity of the resulting model, and consequently its implementability and usability.

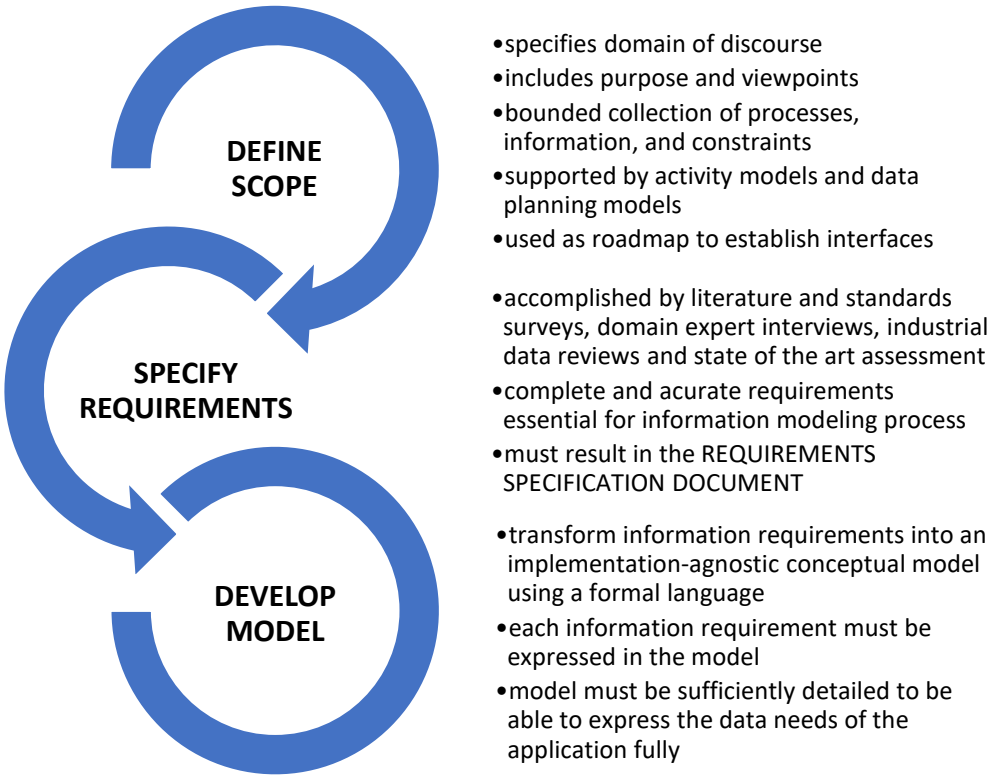

Figure 1: Information Model Development Process. Adapted from [1].

The most relevant modeling methodologies are the entity-relationship (ER), the functional and the object-oriented (O-O) approach [1], each with their own advantages and disadvantages, depending on the particular modeling subject and viewpoint. The ER approach is the generic modeling technique, most prominently used in design of databases, and is also the basis for the functional and O-O approach. The ER approach is best utilized in applications with high level and detailed static data requirements [1]. Where data changes dynamically and functions are more complex than data, the functional approach may be more appropriate, using objects and functions over objects as basis and data-flow diagrams to depict the transformation of data as it passes through the system [1]. Finally, the O-O approach introduces a critical paradigm shift, focusing primarily on defining objects of the domain and adding functions later in the design process [1].

The process of information model development is a three-step iterative process, consisting of scope definition, requirements specification, and model development, as shown in Figure 1, each of them of equal importance in creating a coherent and valid model.

One of the most important decisions in the process of creating an information model is the choice of the modeling language(s). There exist many different languages for different requirements and purposes, the most prominent formal general-purpose ones being EXPRESS and UML, both providing graphical as well as textual representation. EXPRESS or ISO 10303-11 is part of the Standard for the Exchange of Product model data (STEP), is based on programming languages (Ada, Algol, C, C++, Euler, PASCAL) and the O-O paradigm, and allows unambiguous object definition and specification of data properties, constraints and operations [1]. UML is a widely known and accepted general modeling language based on the O-O paradigm, which organizes models in a number of complementary views representing different aspects of a systems, using diagrams representing common O-O concepts such as classes, objects, messages and relationships between them [1].

The following implementation methods, issues and lessons learned related to information modeling are listed in [20]:

a. Information requirements serve as the foundation of the model. A thorough requirements analysis is a necessity. Literature surveys, standard surveys, domain experts' interviews, industrial data reviews, and state- of-the-art assessments are a source of capturing knowledge. Workshops are a good way to gather requirements.

b. Modeling is an iterative process, as refinements are often necessary. As iteration continues, the information model obtained at the end of each iteration is presented to the user community to obtain further feedback. Based on the feedback, either another iteration starts, or the information model is cast in concrete.

c. It is useful to establish a set of naming conventions for a big and complex model in the beginning of the modeling effort. The naming conventions should be descriptive in nature. Advantages for using naming conventions are consistency, ease of identifying entities, and ease of collaboration.

d. Developing a glossary of terms that are used by the applications is also useful. The purpose of the glossary is to provide a unique definition for each term to eliminate improper use due to conflicting definitions.

e. There are several common problems during the implementation process. If a particular information model serves as the medium for transferring the data, the application system should be brought into some degree of compliance with this information model. Occasionally, there is no complete data mapping between the model and the system. If the data requirements are not complete, further requirements analysis should be conducted. For proprietary data, implementation-specific arrangements should be made.

f. Using different measurement units is another common error in an implementation. Under this situation, the attributes in different units should be included in the information model.

g. Conflicts in precision is another issue. The information model should specify precision for numeric data. If the application system carries a lower precision, the accuracy may be lost.

h. Sometimes the same terms may have different meanings or different terms may have the same meaning. The glossary mentioned in item d) that precisely defines all terms presented with the information model is an effective solution to this problem.

i. Having industry reviews of the information model is critical. It helps to ensure the model's necessity, correctness, and completeness.

1.2 OPC UA for information modeling

The IEC 62541 norm, also known as OPC Unified Architecture (OPC UA) is the evolution of the OPC Specification, introducing the advances from the computer science world (OOP, SOA, Semantic Web, Network Model Databases) to the tested and proven capabilities of its predecessor. OPC UA is an interoperability norm, enabling transparent communication among heterogenous systems, and is considered one of the key technologies within the IIC and Industry 4.0 initiatives.

Communication between distributed systems	*Modeling Data*
• Redundancy, robustness and fault tolerance	• Common model for all OPC data
• Platform-independence	• Object-oriented
• Scalability	• Extensible type system
• High performance	• Meta information
• Internet and firewalls	• Complex data and methods
• Security and access control	• Scalability from simple to complex models
• Interoperability	• Abstract base model
	• Base for other standard data models

Figure 2: Requirements and goals of OPC UA. Source: [2].

OPC UA is meant as much for data modeling as it is a technology for communication between distributed systems, as can be seen from the requirements and goals listed in Figure 2. OPC UA models Clients and Servers as interacting partners, multiple of which can be contained within a system, and allows for the possibility of combining both functionalities into a coherent unit [3].

The core concepts of the OPC UA information model are Nodes and References, which span its address space [3]. Nodes can have different base classes, the most important ones being variables, methods and objects, all of which are a part of the integrated object model [3]. Objects are containers which structure the address space and encompass variables and methods [3]. A Reference is a connection between two nodes and can be accessed only indirectly, by browsing a Node and following References [2]. To expose different semantics on how the Nodes are connected, ReferenceTypes are used, which are defined as nodes (and can thus be accessed by a client) and are organized in a separate hierarchy [2].

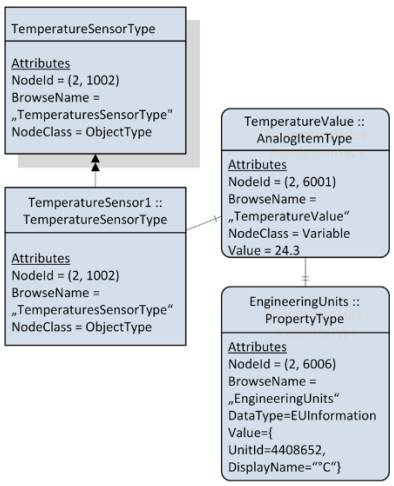

Figure 3: Nodes, References and Attributes - main abstraction units of OPC UA.

All Nodes inherit from the abstract Base NodeClass. The ReferenceType, ObjectType, VariableType and DataType NodeClasses can be used to model types, whereas the Object, Variable and Method NodeClasses are representations of instances, i.e. actual data. With the View NodeClass it is possible to model different aspects of the system under consideration

and represent the same entities with a different set of characteristics, e.g. for different domains. Figure 4 gives an overview of the main concepts of OPC UA as a modeling language.

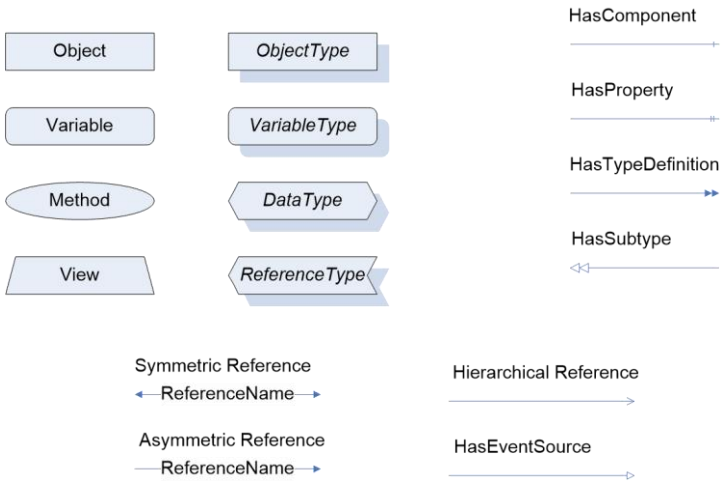

Figure 4: Elements of the OPC UA modeling language - Nodes and References.

As OPC UA is applicable to systems of different scale and capabilities, from small embedded devices, where small amounts of data have to be transferred in short time intervals, to enterprise systems where efficient handling of structured data is more important, it specifies abstract OPC UA Services as interfaces between clients and servers, which can be mapped to different transport mechanisms for different requirements [2]. In practice, OPC UA consists of implementable specifications, communication stacks, SDKs (in multiple programming languages) and higher-level third-party toolkits [3].

2 A Heat Pump Thing - Information Model for IIoT

Different small heat pump circuits as represented in [4] were chosen as the source material for the specification of the information model. As there are many different technologies for engineering data representation, the focus is put on providing structure for specification (and representation) of operational data, i.e. data critical during execution time, as described in [5]. This is a pivotal, yet, paradoxically, not very well researched aspect of automation systems, or Things in general.

For the definition of the information model in accordance with the systems engineering paradigm, an iterative process, as described in Chapter 1.1 was used, consisting of steps:

- User requirements specification
- System (functional) requirements specification
- System design specification
- Implementation
- Test/Validation/Usage

The following Sections give an overview of the steps of information model specification.

2.1 User Requirements specification

Ru1	Is able to represent different configurations of heat-pump installations
Ru2	Is useful to professionals for specification of instances of heat-pump installations
Ru3	Is useful to non-professionals for demonstration purposes of practical IIoT/Industrie 4.0 concepts

Table 1: User requirements.

2.2 System Requirements specification

Rs1	Must be in accordance with the IIoT/Industrie 4.0 paradigm
Rs2	Must be based on, i.e. implemented with, standardized technologies
Rs3	Must be extendable
Rs4	Must be transformable into other common industrial plant representation technologies
Rs5	Must be simple to use, graphical interface

Table 2: System requirements

2.3 System design specification

#	Description	REQ.
Ds1	Implement a type system meta-model based on the specification of common configurations of heat-pump installations in [4].	Ru1 Rs1 Rs3
Ds2	Provide a graphical user-interface for instantiation of concrete heat-pump plants from predefined types (Ru1, Ru2, Ru3, Rs3)	Ru1 Ru2 Ru3 Rs3
Ds3	Provide a graphical user-interface for type extensions (Ru1, Ru2, Ru3, Rs3)	Ru1 Ru2 Ru3 Rs3
Ds4	Provide a possibility for transformation into a standardized technological representation (Ru2, Ru3, Rs1, Rs2, Rs4)	Ru2 Ru3 Rs1 Rs2 Rs4
Ds5	Describe a clear path toward mapping between the created representations and diverse industrial standards for systems representations	Ru1 Ru2 Rs4

Table 3: System Design

Based on the system design specification and elaborations in Chapter , there are two commonly available implementation technologies which should be considered, the AUTOMATION-ML and OPC UA.

AutomationML is able to merge different types of technical representations on the infrastructural level in one place, creating and maintaining complex relations between them. In the context of usability for professionals (Ru2), as well as its extensibility and transformability into different industrial representations (Rs3, Rs4) using AutomationML would certainly provide the more flexible solution. However, it also introduces a level of complexity which would impede the usability of the created information model for non-professionals (Ru1), as well as its simplicity of use (Rs5).

OPC-UA is a technology with a dual purpose – it is a communications standard which provides interoperability between different runtime components on the one hand, but also a powerful standardized information modeling mechanism on the other hand, making possible to reflect the structure of any domain representation. An added benefit is that OPC UA is already integrated with most known standards (including AutomationML). Furthermore, the OPC UA Foundation provides a simple to use graphical interface for the specification of information models as extensions of its base concepts. These can further be used for seamless generation of OPC-UA servers, or transformed to other representations, e.g. using the AutomationML companion specification.

The technology of choice for the purposes of this project is OPC UA. As elaborated above, it is better suited for the specified requirements, in particular in connection with the usability for non-professionals and demonstrational purposes in general.

2.4 Implementation

The basis for the system implementation are the standardized circuits for small heat pumps as described in [4]. Basic OPC UA concepts are extended to model the devices and their typical groupings.

2.4.1 System under consideration

Figure 5 shows two of the six defined types of circuits for small heat pump plants. The simplest circuit on the left contains a heat heat pump with only the main heat consumer section, entailing a single pump and a return temperature measurement probe.

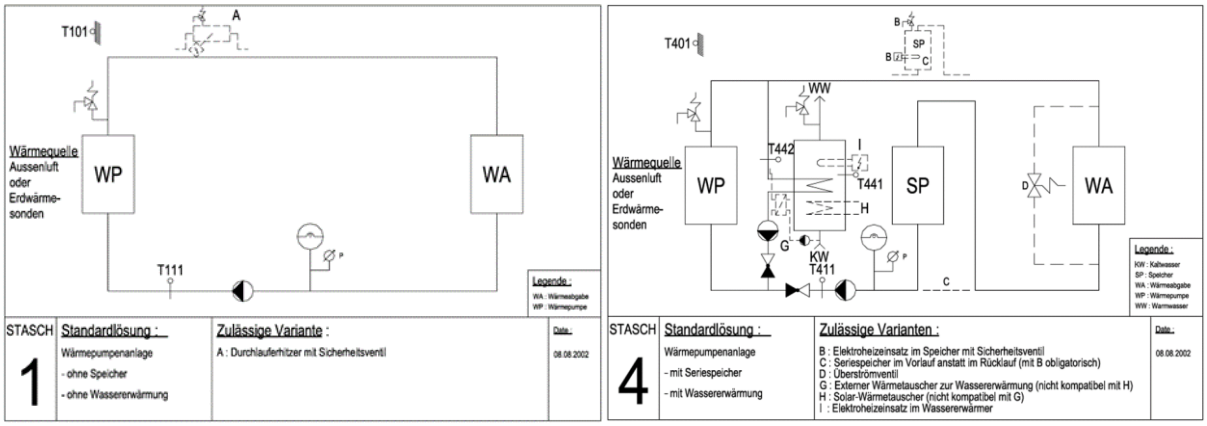

Figure 5: A simple and a more complex standard heat pump circuit (STAndard SCHaltung) STASCH1 and STASCH4.

On the right hand side a more complex circuit is shown. In addition to the main heating section, which now contains an optional heat storage and a flow regulation valve, a warm water heating section is introduced. It has several different possible device configurations, as well as a dedicated pump and a valve.

2.4.2 OPC UA Information (Type-) Model

The *BaseObjectType* has been extended to define different types of devices which can be found in the standardized circuits. These include types for different variations of valves, pumps, probes, heaters, and heat storages, as shown in Figure 6. The *Alarms*, *States*, *ConfigParam*, *States* and *InterfaceSpecification* folders allow for the specification of functional characteristics of a (to be implemented) runtime objects of a particular device, i.e. its *operational model*.

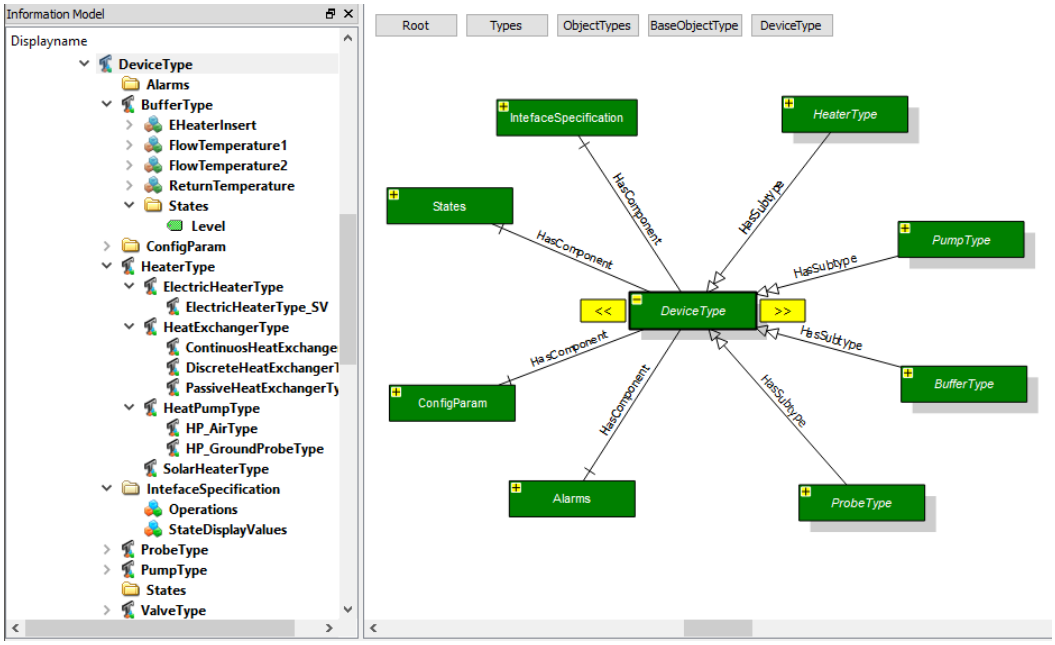

Figure 6: Type model for devices (DeviceType).

A heat pump circuit can essentially be represented as a source (i.e. the heat pump) segment with flow and return connectors to the consumption side, where different heating sections can

be connected in parallel to each other. Figure 7 shows the *HeatingGroupType*, representing different possible configurations of heating sections. The *ConsumerType* subtype represents a simple consumer section with a passive heat exchanger and optional flow and return temperature measurement probes.

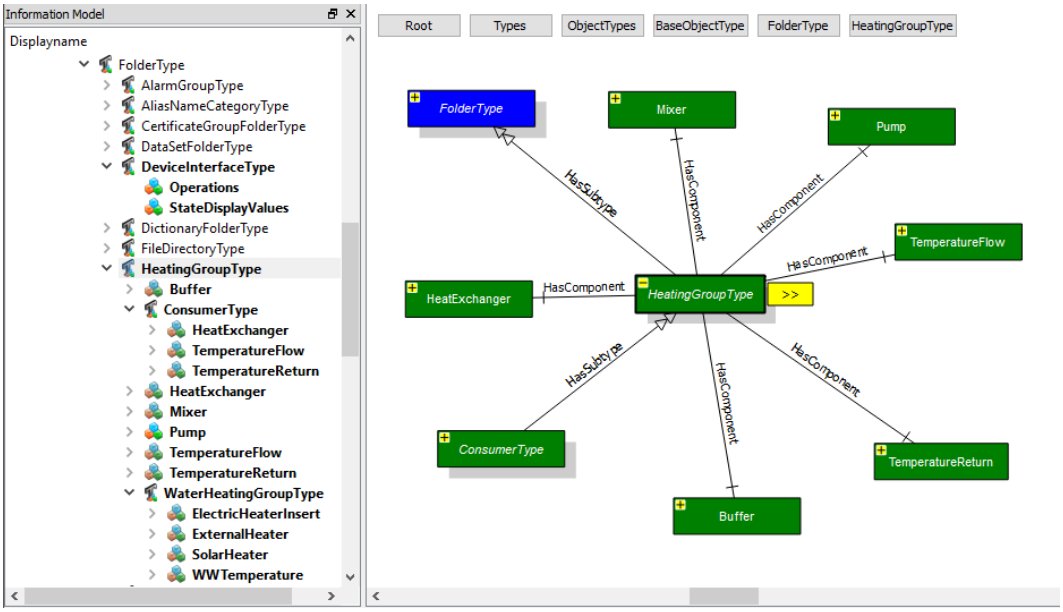

Figure 7: Type model for heating sections (HeatingGroupType).

The *HeatCircuitType*, as described in the previous paragraph is the basis for creating representations for standardized circuits from [4] and shown in Figure 8. Each standardized circuit is a set containing selected elements from the base HeatCircuitType, as represented in Figure 9 for the STASCH4 circuit.

The same as with the defined DeviceTypes, both HeatCircuitType and HeatGroupType may entail the folders for the specification of the operational model, as they also represent potential runtime objects, depending on the specific implementation.

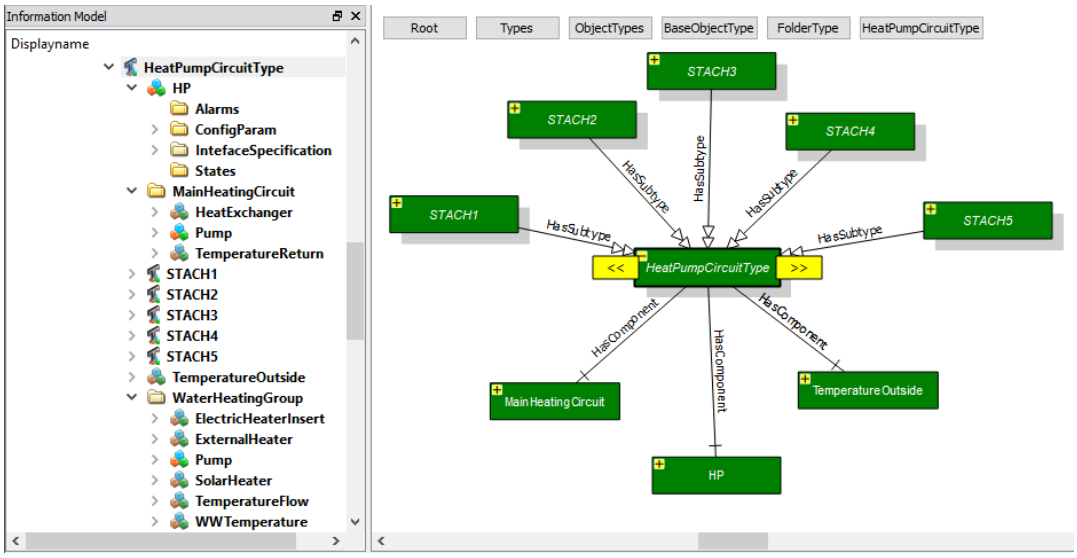

Figure 8: Type model for heat pump circuits (HeatPumpCircuitType).

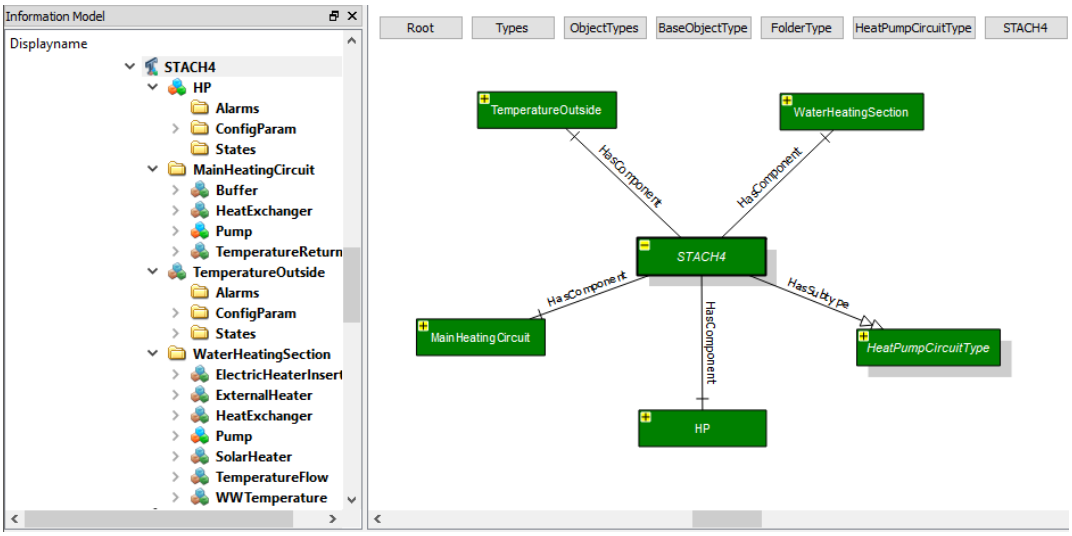

Figure 9: Type model for the STASCH4 circuit.

2.4.3 Using the OPC-UA model – creating heat pump circuit instances

With types for every circuit defined, instantiating and modeling a concrete plant is reduced to selection of predefined possible elements, as shown in 10 and Figure 11 for the process of creation of a STASCH5 circuit instance.

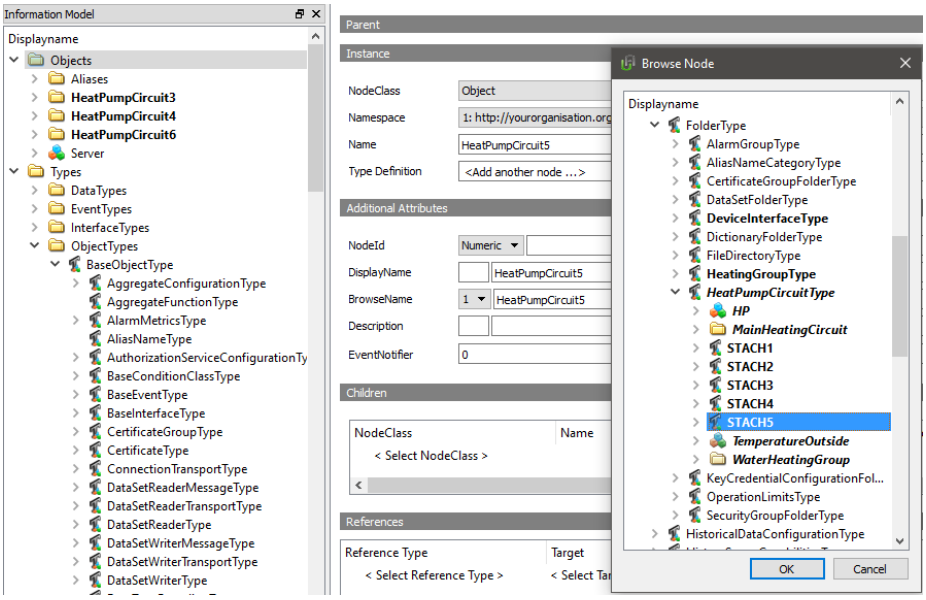

Figure 10: Creating instances of concrete plant from the predefined types for standard circuits.

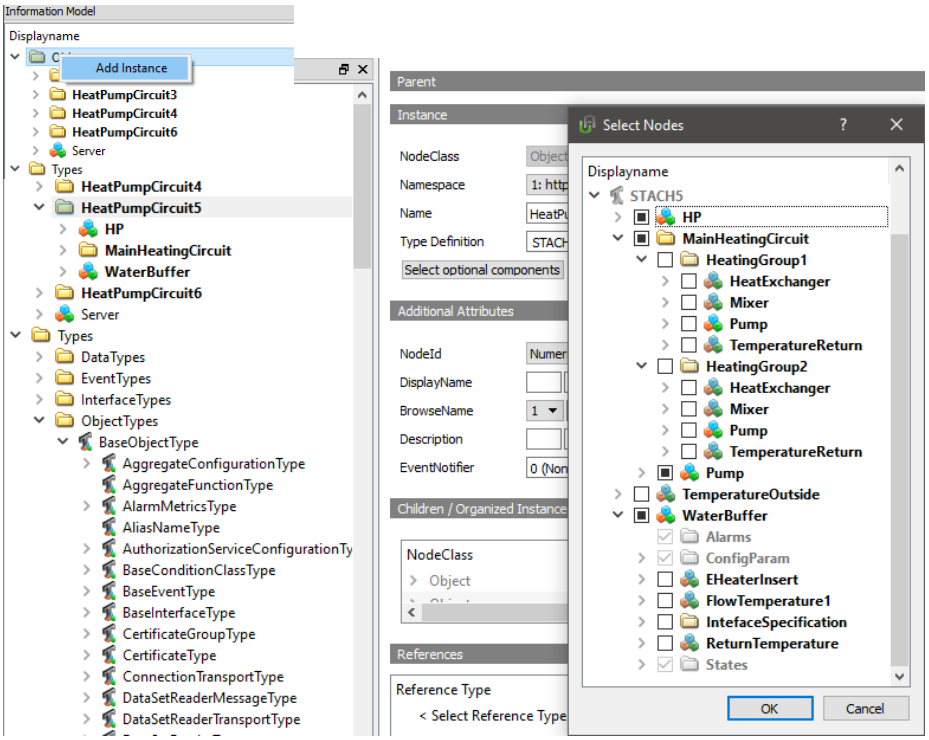

Figure 11: Selecting elements of the created instance to match the specification of the concrete plant.

Figure 12 depicts a created instance for a previously described and discussed STASCH4 circuit. It entails a heatpump, the main consumer (heating) circuit entailing a pump with a flow regulation valve and a heat exchanger (i.e. the consumer itself). Additionally, a water heating section is added, containing an electric and solar heater, as well as a pump and a flow temperature measurement probe.

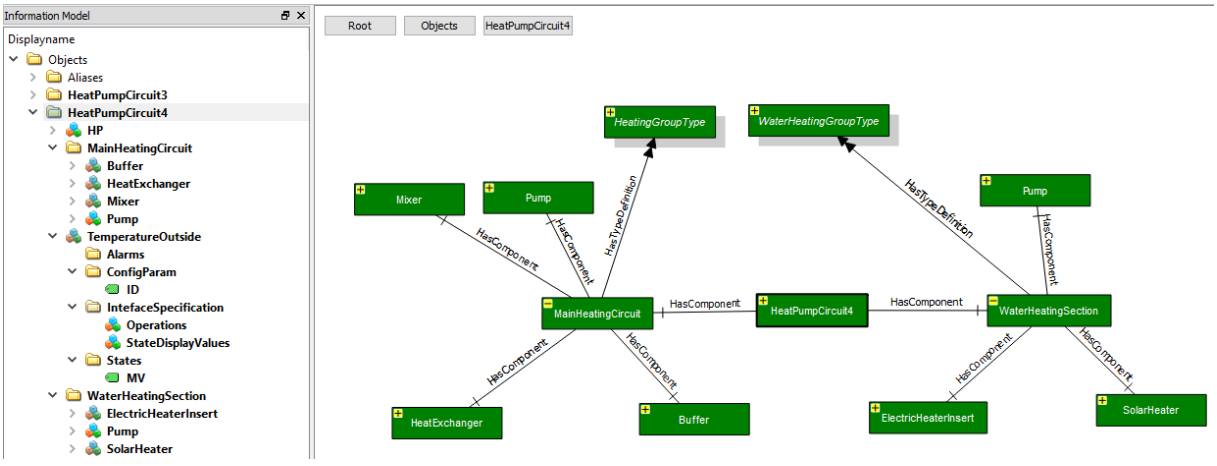

Figure 12: Instantiated and configured STASCH4 circuit.

3 References

[1] Y. T. Lee, „Information Modeling: From Design to Implementation,“ National Institute of Science and Technology (NIST), Gaithersburg, USA, 1999.

[2] W. Mahnke, S.-H. Leitner und M. Damm, OPC Unified Architecture, Springer, 2009.

[3] OPC Foundation, „OPC Unified Architecture Part 1: Overview and Concepts, Release 1.03,“ 2015.

[4] Bundesamt für Energie, „Standardschaltungen für Kleinwärmepumpenanlagen Teil 1: STASCH-Planungshilfen“.

[5] G. Music, B. Heinzl und W. Kastner, „AVA - A component-oriented abstraction layer for virtual plug&produce automation systems engineering,“ *Elsevier Journal of Industrial Information Integration (JII),* 2021.