A common data environment for HVAC design and engineering

Seidenschnur, Mikki; Kücükavci, Ali; Fjerbæk, Esben Visby; Smith, Kevin Michael; Pauwels, Pieter; Hviid, Christian Anker

Published in:
Automation in Construction

Link to article, DOI:
10.1016/j.autcon.2022.104500

Publication date:
2022

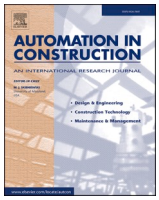
Document Version
Publisher's PDF, also known as Version of record

Automation in Construction 142 (2022) 104500

Contents lists available at ScienceDirect

Automation in Construction

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



A common data environment for HVAC design and engineering

Mikki Seidenschnur [a,c,*], Ali Kücükavci [a], Esben Visby Fjerbæk [a], Kevin Michael Smith [a], Pieter Pauwels [b], Christian Anker Hviid [a]

[a] Department of Civil and Mechanical Engineering, Technical University of Denmark, Copenhagen, Denmark
[b] Department of Civil Engineering, Technical University of Eindhoven, Eindhoven, Netherlands
[c] Ramboll, Copenhagen, Denmark

ARTICLE INFO

Keywords:
Building information modeling
HVAC
Object models
Common data environment
BIM level 3

ABSTRACT

The Architecture, Engineering, and Construction (AEC) industry is transitioning toward using cloud-based Common Data Environments (CDEs) with interlinked BIM models. A CDE that engages all stakeholders of the building's design, construction, and operation phases represents the outset of BIM maturity level 3. This article introduces a CDE called Virtual Commissioning (VC), capable of commissioning an HVAC system before the physical commissioning of the HVAC system. The FSC diagram is introduced, to represent an HVAC BIM model within the VC CDE, and the Revit to FSC exporter, to serialize an HVAC object model from Revit to the FSC diagram. Three microservices were developed to exemplify the ease of developing independently scalable solutions for the VC CDE. Furthermore, the article proves that Modelica simulations can be run, using the microservice architecture of the CDE. To test the robustness of the system architecture for the CDE, two example models were introduced, one simple and one with a high level of complexity. Transferring the example models from Revit to the VC CDE was successful. Finally, in the roadmap for future development, it is proposed that future work should focus on using the CDE for advanced hydraulic simulations, using Modelica and Spawn-of-EnergyPlus.

1. Introduction

Building information modeling (BIM) is the practice of generating and managing well-defined building data [1]. BIM data is typically geometric, spatial, geographic, physical, or quantitative, and it aims to provide a shared repository for stakeholders [2]. BIM can revolutionize the construction sector by streamlining integrated design processes, accurate construction scheduling, and comprehensive error screening [3]. It further can help mitigate climate change and resource depletion by simplifying and enhancing resource- and energy-efficient integrated design processes for new construction [4–6] or renovation [7]. However, most current workflows are manual or semiautomatic [8], often utilizing conventional spreadsheets [9], and most occur too late to impact the design [9] significantly. Most of these workflows do not utilize the full capabilities that BIM can offer if utilized to its full extent.

Succar et al. introduced the BIM maturity levels to describe the BIM-based collaboration between stakeholders [10]. BIM-based collaboration is defined from maturity Level 0 with almost no collaboration to Level 3 with full integration, in which all stakeholders collaborate using a shared model in a cloud-based common data environment (CDE) [10]. CDEs are applications that connect several services. Several CDEs have been developed for the AEC industry [11–13]. With a CDE, it is possible to create bi-directional links between a database model and simulation services. A CDE enables teams always to have the most updated model to run new simulations or make design decisions. Previously developed CDEs for advanced hydraulic simulations use either the format of gbXML (Green Building XML) or IFC (Industry Foundation Classes). Transforming BIM data to BEM tools using gbXML and IFC formats can introduce extreme errors [14], and the process occurs only once due to the need for manual data entry [15]. More importantly, this BIM to BEM process relies on a file-based exchange mostly, which is more common to BIM Level 2. Therefore, to enable a real-time connection with live building data, building models, and simulation data, an approach needs to be taken that is not file-based, but rather web-based including databases and microservices. That may lead to a simulation-enabled CDE.

* Corresponding author at: Department of Civil and Mechanical Engineering, Technical University of Denmark, Copenhagen, Denmark.
E-mail addresses: msei@ramboll.dk (M. Seidenschnur), alikuc@byg.dtu.dk (A. Kücükavci), evifj@byg.dtu.dk (E.V. Fjerbæk), kevs@byg.dtu.dk (K.M. Smith), p.pauwels@tue.nl (P. Pauwels), cah@byg.dtu.dk (C.A. Hviid).

https://doi.org/10.1016/j.autcon.2022.104500
Received 25 March 2022; Received in revised form 16 June 2022; Accepted 24 July 2022
Available online 7 August 2022

1.1. BEM simulation through a micro-service architecture

Tools for simulation of advanced hydraulic simulations have existed for many years, like HVACSIM+ [16], IDA ICE [17], the Modelica Buildings Library [18], and many more. Running advanced hydraulic simulations can be extremely useful to test, whether a building performs adequately compared to the original design. However, most AEC companies today only perform simple calculations based on rule-of-thumb. This means that most systems are designed based on a static calculation that only applies the peak load for the system. This means that the system can be regulated for the peak load, but cannot regulate the flow for the remainder of the time, due to lacking valve/damper authority. The hydraulic systems are rarely simulated at all in the design phase of a building. This is mainly due to the labour intensive manual work of setting up the boundary conditions needed to run an advanced hydraulic simulation. Efforts should be made to be able to automatically convert from BIM to BEM. The suitability of tools for automatic integration of BIM and BEM varies considerably. Modelica is an equation-based object-oriented modeling language that provides a flexible means for constructing BEM while excelling at HVAC systems and controls [19,20]. Creating Modelica simulations today is a time consuming endeavor, due to the complexity of providing the boundary conditions for a complete solution. However, if the BIM model is used to automatically transfer the boundary conditions, this can eliminate a large part of the manual task, as shown by Fjerbæk et al. [21] There are substantial efforts to automate the translation from BIM to Modelica-based BEM [2,22,23]. Kim et al. [22] introduced a library with the name of ModelicaBIM. The idea of the library is to be able to perform Modelica simulations, based on a BIM model. Jeong et al. [23] introduced a tool that could export a building modeled in Revit to perform thermal simulations in Modelica. The article from Andriamamonjy et al. [2] directly translated the geometry, systems, and controls which was encapsulated in an IFC4 file and simulated in a Building Energy Performance Simulation (BEPS) model. There are also efforts to use less well-defined IFC files through enrichment and identification [24] and grey-box modeling [25] to generate Modelica-based BEM.

Even with the recent extension of the HVAC domain (Add2TC1) in the current IFC data model (IFC4), it does not provide the necessary structure and attributes to use third-party simulation tools for HVAC design [26] and therefore requires improvements. Many IFC classes do not map well to the (more detailed) classes needed in a BEM tool (e.g. MechanicalEquipment vs. Air Handling Unit). Therefore, a better object model is needed that includes these specialised HVAC classes and properties. This object model ideally serves as a common data format to enable a CDE to run advanced hydraulic simulations. Furthermore, this object model needs to be web-ready to enable a BIM Level 3 CDE approach (e.g. JSON, RDF), that includes a microservice architecture with horizontal scalability.

Hence, this paper investigates the creation and use of such a common web-ready object model, plus its incorporation in a service-oriented CDE. This paper proposes to create a web application named Virtual Commissioning as a CDE. Virtual Commissioning is envisioned by the authors of this article to generate a virtual environment or CDE, that is capable of commissioning the building services, before, and during operation of the building. We do recognize that traditional commissioning is a quality-focused process that delivers the entire building to the owner, according to the owner's objectives and criteria. In future work, we plan to make the VC platform operational for the full commissioning of the building. In this article, the VC CDE revolves solely on the commissioning of the building services, and their performance. The CDE connects a Revit model with an Application Programming Interface (API) endpoint to a MongoDB database. The database is structured based on the data structure of the FSC object model that is introduced in this paper. The FSC object model is generated from a Revit model using a Extract-Transform-Load (ETL) approach. Finally, the VC platform introduces a microservice architecture that makes it possible to create microservices that can run independently based on the FSC object model in the database. Three microservices are introduced and utilized on two use cases. After testing the VC platform with the creation of microservices, we will test the performance of the FSC exporter tool on a model obtained from a real-world project.

1.2. Aim

The aim of this article is to:

1. Centralize BIM project data so all stakeholders have access to a single source of truth (SSOT) in a web-based CDE based.
2. Create a data structure or object model that can represent a flow system
3. Allow for easy scalability of the CDE using a microservice architecture.

1.3. Outline

Section 2 describes the current state-of-the-art CDEs that raise the BIM maturity to level 3. Section 3 describes in detail the system architecture of the proposed VC platform and the FSC object model (see Fig. 1). Section 4 introduces example models 1 and 2, which we will use to evaluate the performance in Section 5. Section 6 presents the achievements of this paper, together with the limitations. Furthermore, it offers a roadmap for future development. Finally, Section 7 concludes on the contribution of this paper.

2. Background

This section describes the efforts within the development of CDEs for buildings and HVAC systems. This includes the efforts sought to represent HVAC systems with object models. It also describes the state-of-the-art for simulation environments for full building simulation. Finally, the section introduces the state-of-the-art within software development using microservice system architecture.

2.1. Simulation and computation

Andriamamonjy et al. introduced an automated workflow, called IFC2Modelica, for the direct transfer of geometry, system, and control representations encapsulated in an IFC4 file [2]. One issue with this approach is that commercial BIM tools, like Revit, do not serialize all the needed information sufficiently well to carry out the complete data transfer introduced in the IFC2Modelica workflow. After transferring the IFC file from Revit or a similar proprietary BIM tool, the user must manually input the required information to run the simulation in the Modelica environment. Similarly, Jeong et al. created Revit2Modelica to transfer an architectural BIM model into Modelica for an advanced building energy simulation. The Revit2Modelica approach takes a proprietary file format (.rvt) and translates it into the Modelica file format. However, it does not transfer HVAC systems. IFC2Modelica and Revit2Modelica provide a novel approach for simulation of HVAC systems and building energy modeling but based on file-based BIM models. This means that they do not live up to the BIM maturity level 3. A common data environment should be presented to raise the BIM maturity from level 2 to level 3.

2.2. Object models for representation of HVAC models

Efforts have sought to enable a level 3 BIM maturity with automated, flexible data transformation using open standards (e.g., IFC) and semantic web technologies to improve interoperability, data linking, and logical inference [27]. Afsari et al. implemented the IFC schema in a JSON (JavaScript Object Notation) format to facilitate web-based data exchange [28]. Do-Yeop Lee developed a novel framework using BIM

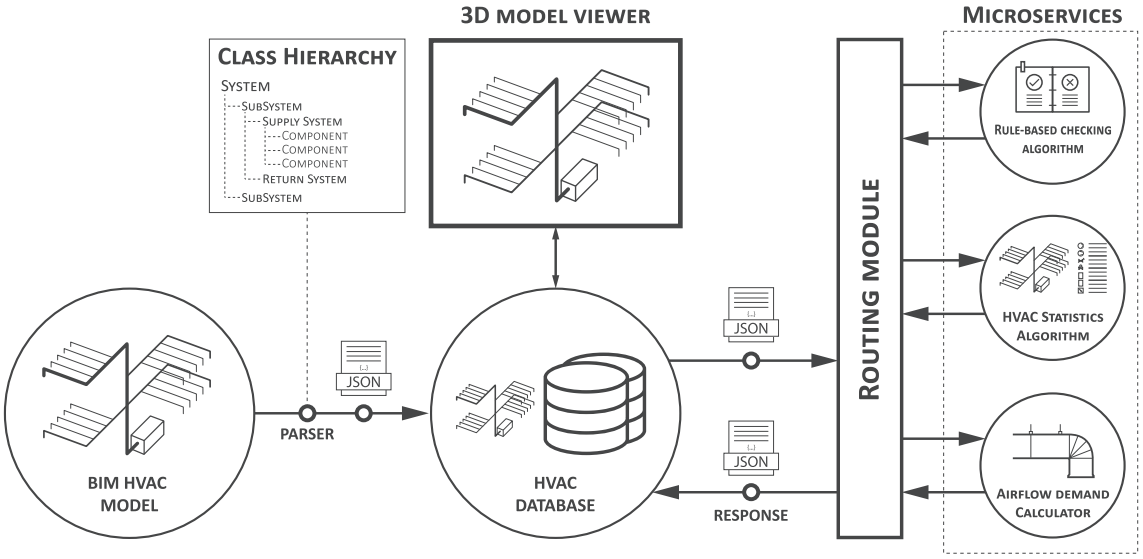


Fig. 1. Proposed System Architecture allows the automated transfer of data from BIM to a database and then follows an automated transfer to a given microservice.

and linked data technologies to share defect data and enhance productivity during construction [29].

Many efforts have specifically concerned building operation, as it is continuous and benefits from a 'live' BIM database. Quinn et al. demonstrated a linked data approach integrating IoT data and BIM [30]. Meanwhile, Tang et al. developed and tested a prototype exchanging building automation system (BAS) data using BACnet and an IFC data model [31]. Similarly, Kim et al. proposed a semantic web-based facilities management approach [32], while Bong et al. developed a BIM-enabled data architecture for fault detection and diagnostics [33]. Furthermore, Mohamed et al. devised an ontology to formalize as-is BIM knowledge for semantic web technologies to improve maintenance [34]. Finally, Balaji et al. created Brick [35] to represent sensors and subsystems, and the relationship between them. However, while the Brick schema is great at representing data points within the building and HVAC system, it does not represent passive components such as pipes and ducts. Therefore, it is not fit to represent an entire flow system and the aspects thereof. Such developments should help exchange BIM data openly and enable web service applications.

2.3. Common data environments

The development of BIMServer.org was an early effort in raising the BIM maturity level from 2 to 3 [?]. The primary purpose of the BIMServer.org project was to provide an IFC database that has features like model checking, versioning, project structures, merging, etc. While BIMserver.org is an open access open-sourced platform, it is based solely on the IFC schema, introducing serious errors and missing data depending on the tool it is generated by [14]. A proprietary file format approach was carried out by the software vendor Autodesk, with the introduction of the cloud platform A360 and the integration of Forge, which implements an API. For the project team, Forge provides an easy way to share and version Revit models in the cloud; it is still based on the proprietary file format from Autodesk Revit. This introduces a limitation in integrating a link with external applications A360 does not support [36]. Cheng et al. made an online CDE that was based on the gbXML schema [12]. Furthermore, they included an energy modeling approach using the open-sourced tool EnergyPlus. While providing an open platform that eliminates the need for file-based sharing of BIM models, the platform only supports gbXML. The efforts mentioned in this subsection could be specified as CDEs, but none of them introduced a CDE capable of storing an HVAC model with the capability of HVAC simulation. The IBPSA project 1 introduced a CDE based upon IFC, CityGML, and Modelica [13]. The project seeks to create an open-source tool that allows next-generation computing for the design and operation phases of buildings and district energy and control systems. The IBPSA project 1 integrates the object-oriented modeling language Modelica into their CDE for HVAC simulations. They use IFC as the file format. While the IFC model represents an open data format, it is also known that most proprietary tools, like Revit, have severe errors in parsing from their native format to IFC [14]. The IPBSA project 1 utilizes a classic monolithic architecture, which makes it difficult to scale the application to a cloud computing setup [37].

2.4. Microservice architecture

With the software engineering domain moving toward cloud computing, microservices are becoming more mainstream [37]. Microservices are deployed, tested, and run independently, making it easy to scale an application, especially in a cloud computing setup [38]. This allows several developers to develop/maintain services while the CDE stays in operation.

2.5. Summary

In summary, CDEs have been introduced in earlier works, like BIMServer.org and Autodesk Forge. However, they are not capable of representing an HVAC object model. Furthermore, the IBPSA project 1 is a CDE that allows for next-generation computing in Modelica, based on the IFC model of an HVAC system. Though the IFC format is considered open, the parsing from proprietary BIM tools like Revit is error-prone, meaning there is a need to introduce an open format to represent flow systems. Furthermore, none of the CDEs above present a way to incorporate a microservice architecture, allowing for horizontal scaling of the web application. While developments of CDEs have concerned building operation, fewer have enabled BEM and dynamic simulations using web-based BIM. Kukkonen et al. devised a semantic web ontology for flow systems in buildings, which aimed to support web-based design and operation [39]. That ontology inspired the development of an FSC object model capable of handling entire flow systems and the component properties for hydraulic simulation. The data format of a flow system is not openly available from tools like Autodesk Revit, so our implementation of the FSC diagram yields a toolchain for enabling web-based services requiring flow specifications from a shared online BIM database. This builds on the developments integrating BIM and BEMs, but it uses a database to increase the BIM maturity level from 2 to 3.

3. System architecture

Based on Section 2, we propose the creation of a VC platform that serves as a CDE connected to a BIM tool that includes HVAC systems. This section considers three core developments for the VC platform: (1) The FSC object model for flow systems, (2) the implementation of the database, and (3) microservices for containerized and decentralized calculation. The source code for the FSC object model and the database is not shared specifically within this report. However, the class hierarchy is shared in Fig. B.1. The source code for microservices is open-source and has been shared in Section 3.3.

In detail, this section describes the system architecture of the VC platform. The system architecture shows a conceptual model that defines the structure and behavior of the platform. The platform allows for the decentralization of applications with the use of microservices. Fig. 1 shows that the system architecture revolves around a web application with a MongoDB database. The platform provides a link between the BIM model in Revit and the database. The BIM model is transferred using the Revit API by mapping and serializing an FSC object model and sending it to the database in the VC platform. Once the data has been transferred to the database, microservices can be utilized for decentralized calculation. The 3D model viewer is depicted in Fig. 3 to show its placement relative to the system architecture. The 3D model viewer will not be discussed further in this article. Section 3.1 introduces the FSC object model used to describe the flow system and its components. Furthermore, a UML class diagram for relating spaces to HVAC components is presented. Section 3.2 showcases the database setup used for the VC platform. Finally, Section 3.3 shows how a microservice architecture is utilized, enabling several microservices to use the database FSC object model for decentralized calculation.

3.1. The FSC object model

This Section introduces the FSC object model. We used a Unified Modeling Language (UML) class diagram to create the FSC object model. FSC describes the composition of the flow system, with the relationship between the flow system and its components. Moreover, it appends the attributes needed to describe the physics of components in a flow system. For instance, a component is defined by its properties and the relation to any connected components and systems. FSC contains three main features that enable the description of the flow system:

1. The HVAC system is divided into subsystems, creating a system topology.
2. Each component of the system is defined with its physical properties.
3. The connectivity of all components are defined in sufficient detail.

Fig. 2 shows a simple UML class diagram, that describes the HVACSystem, SubSystem, Component, and Connector classes. In total, the FSC diagram contains 37 classes and 54 methods. This article will only describe the core classes in the class diagram and not all of the classes and methods in detail. To see a complete UML class diagram, see Section Appendix B. The FSC UML class diagram has been created using the Modelica Buildings library [40] as inspiration, since the purpose of future work is to be able to simulate in Modelica.

3.1.1. Topology of a flow system

Fig. 3 shows an example of a flow system. The HVACSystem can contain the distribution systems for Heating, Cooling, and Ventilation. Fig. 4 shows that within each of those categories, there are always two different SubSystems: a Supply system (solid lines), and a Return system

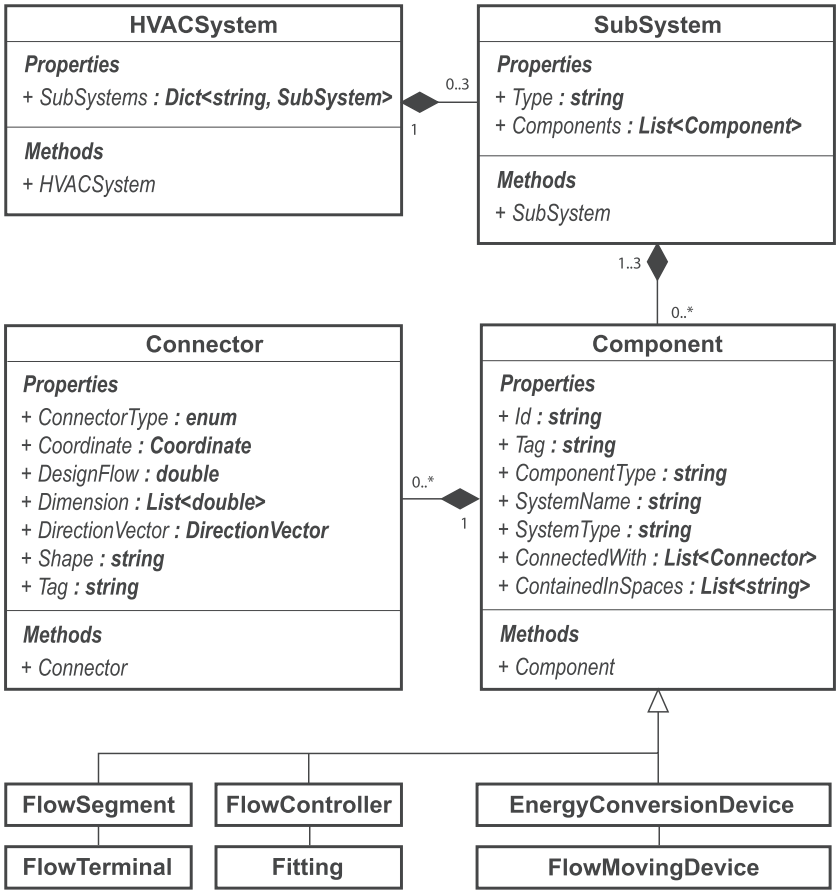


Fig. 2. UML class diagram showing the connection between the four main components of the system. The HVACSystem contains the different SubSystems that is of type SubSystem. In a SubSystem, there is a list of Components.

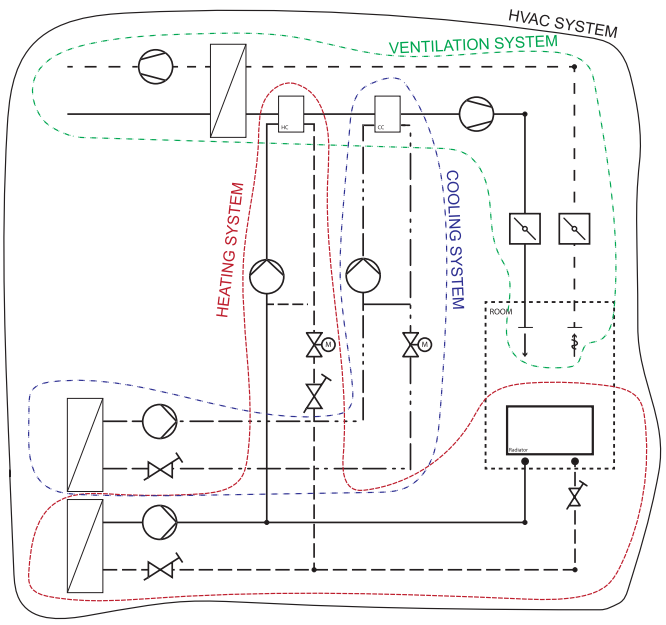


Fig. 3. The overall system topology. An HVAC system can contain a heating, cooling, and ventilation system.

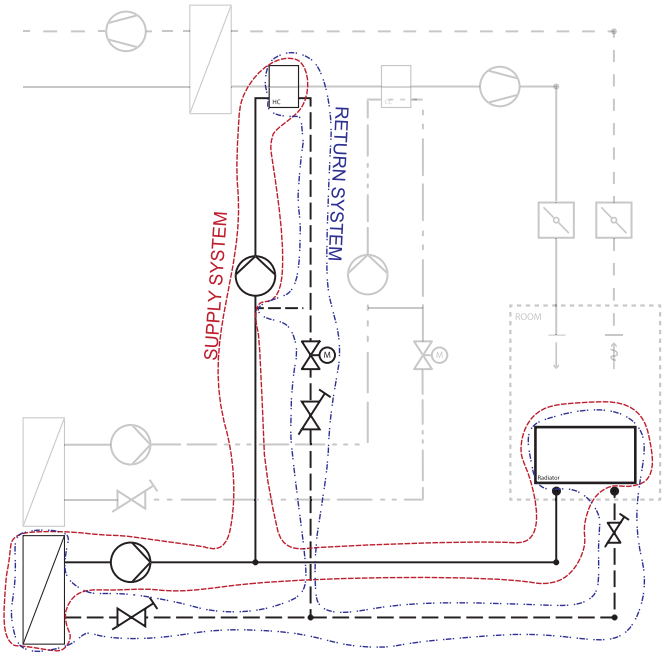


Fig. 4. Every HVAC system contains a supply and return system. As it is illustrated in the figure, the components that bind the supply and return system are both on the supply and return system.

(dashed lines). An end consumer, like a radiator, becomes the interface between the supply and return system. This is reflected by including the same instance of the Radiator in both the supply and return system, with the same Id and Tag. This principle is applied to components that make up the beginning and/or end of a circuit, such as EnergyConversionDevices, and FlowTerminals.

3.1.2. Component properties

In addition to the systems and subsystems, FSC introduces a superclass (Component) that encompasses the properties that exist in all types of components within a system. Fig. 2 shows the properties and methods contained within the Component class. All FSC subclasses contain the following properties: (1) the Id uniquely identifies the component; (2) the tag identifies the component; (3) the classification of the component type; (4) the system name; (5) the system type; (6) a list of connectors (see Section 3.1.3); (7) the spaces that contain the component. Fig. 5 shows a list of all the subclasses of the Component class, which inherit properties from Component. To see the attributes of each component, see Appendix B.

- EnergyConversionDevice is a device that converts energy from one fluid to another; it includes heating coils, heat exchangers, and radiators.
- Fitting typically describes the connection from one Component to another or several other Component. It includes tees, bends, crosses, reductions and caps.
- FlowController describes a component that controls the flow in a flow system. It includes valves and dampers.
- FlowMovingDevice is a component that moves a fluid, which includes pumps and ventilators.
- FlowSegment is a segment that connects any non-FlowSegment component, which includes pipes and ducts.
- FlowTerminal is the terminal unit of any system, which includes ventilation air terminals.

3.1.3. Component connectivity

A logical description of a flow system must include a module to describe the connectivity of components since the purpose of flow systems is to transfer a fluid from one part of the system to another. Fig. 6 shows that the Components contain a logical description of their relations to each other. Pump-1 is supplied with fluid from Pipe-1, and it supplies fluid to Pipe-2. The description of the connection between one Component and another Component is done with the Connector class. In this example, it means that there will be two Connectors for Pump-1. One connector describes its relationship with Pipe-1

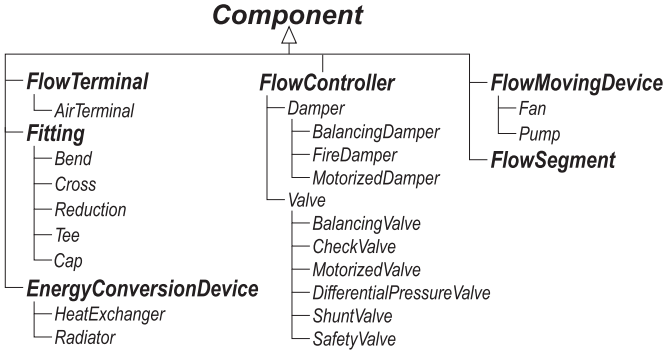


Fig. 5. The tree structure showing all the subclasses to the Component super-class.

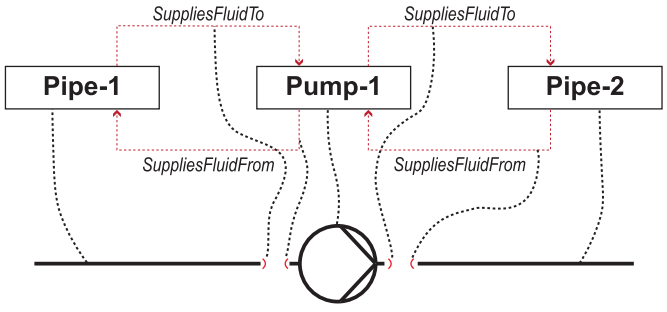


Fig. 6. Example of a topology describing a flow system that consists of a pump connected by a pipe in each connector.

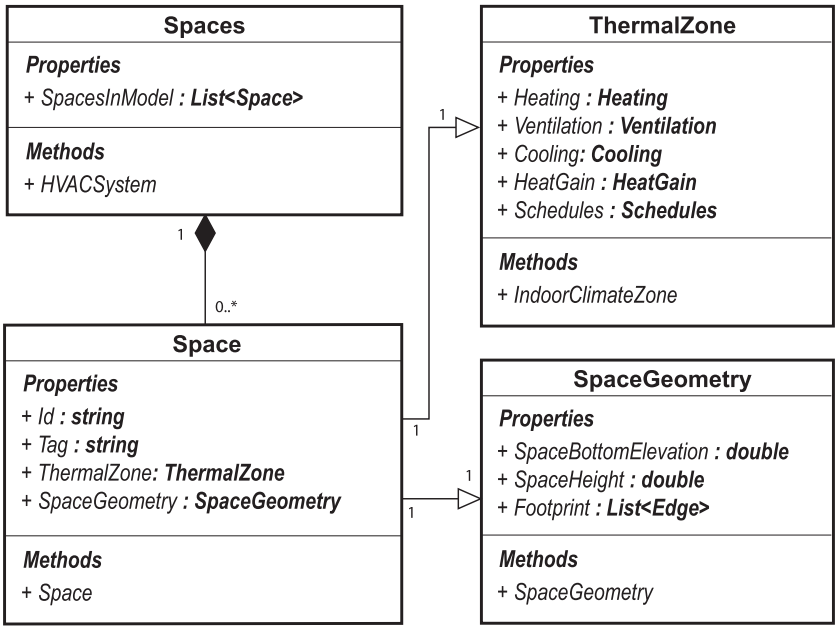


Fig. 7. A part of the UML class diagram for modeling Spaces. For simplicity, the full UML diagram is not shown.

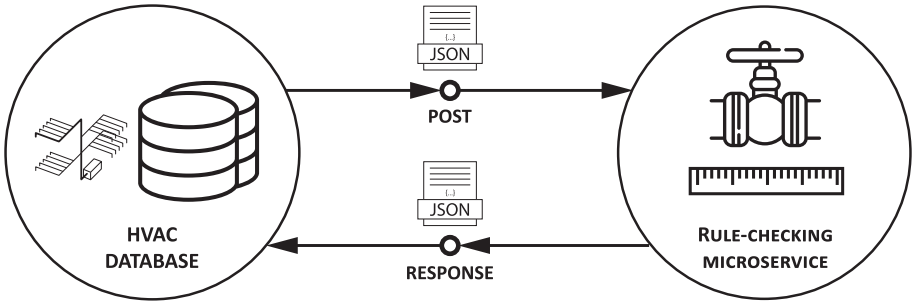


Fig. 8. Illustration of microservice setup. A JSON-file is posted with an HVAC system in the FSC format. Once handled in the microservice, a response JSON is returned.

(SuppliesFluidFrom), and the second Connector describes the relationship with Pipe-2 (SuppliesFluidTo). For consistency and efficiency, every Component describes the relationship to all connected Components, even though the next Component contains a symmetrical connection to the previous Component. "SuppliesFluidTo" describes the forward direction of the flow, "SuppliesFluidFrom" describes from where the flow is coming.

3.1.4. Spaces related to the flow system

In addition to the system representation explained above, we created a class diagram for the properties of spaces to act as boundary conditions for the flow system (see Fig. 7). Such boundary conditions allow for calculating the airflow demand of a ventilation system and sizing the heating system. The ContainedInSpaces property in the Component class is used to describe the relation between spaces and components. The ContainedInSpaces property describes which spaces a component is contained within.

3.1.5. Serialization to JSON data exchange format

Section 3.1 introduced the FSC object model, making it possible to define a flow system and its components, including the spaces of a building. A data exchange mechanism is needed to exchange the FSC object model between platforms. While there are several options to implement such a data exchange (e.g. RDF graphs, XML, CSV, dedicated formats), we chose to focus on a serialization to the JavaScript Object Notation (JSON) format, as nearly all microservices and web service developments use this format. Listing 1 shows a JSON sample for an example model introduced later in this article.

Listing 1: A JSON object example taken from Fig. 6. The listing shows a three component system. Only the most basic attributes from the base class of Component have been included for all the components. The "…" notation indicates that more components are present but not shown.

Listing 2 The "Spaces" attribute is explained in Section 3.1.4. For simplicity, only the most basic attributes from the base class of Space have been included. The "…" notation indicates that more attributes are present but not shown in this example.

3.2. Database implementation

A mongoDB database stores the FSC object model. mongoDB is an object-oriented database (OOD). The objects created with the FSC object model are stored directly into the database. Representing the data in an OOD is so close to the programming objects that the code is simple to implement. In our implementation, the mongoDB database is instantiated with the use of the serialized FSC object model, as seen in Listing 1.

3.3. Microservice implementations

With the object model and database infrastructure in place, the last element in the system architecture comprises of microservices that operate with the data, as shown in Fig. 1. The microservices developed for this article were all created as Python-Flask API endpoints. In our

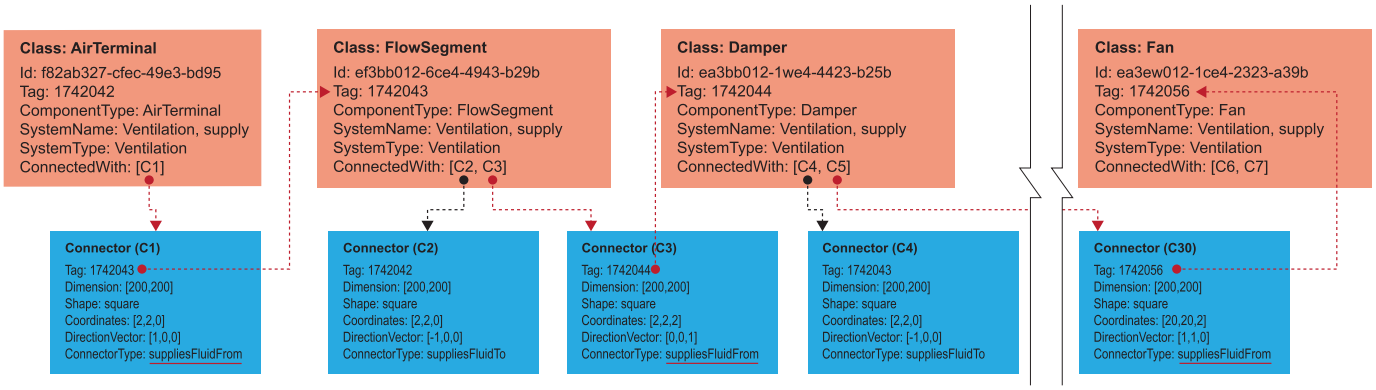


Fig. 9. The Figure shows how to traverse the ventilation system with a recursive function. The recursive function has a stop condition: the ComponentType == "Fan". The red arrows show the path from the air terminal to the fan. In the supply system, the ConnectorType "supplesFluidFrom" is used as a keyword to find the next component that will lead to the supplying fan. When done for the return system, the ConnectorType "suppliesFluidTo" is used as a keyword to find the next return component that will lead to the returning fan. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

case, we implemented three microservices of use in the HVAC engineering domain:

1. Rule-based checking of system integrity
2. Airflow calculation of ventilation system
3. Calculation of HVAC statistics

Overall, these microservices rely on the infrastructure shown in Fig. 8. The figure illustrates how the microservices work using the first micro-service as an example (rule-based checking). The web application backend makes a POST request to the Flask microservice. The body of the request contains a JSON file that represents the entire flow system in a JSON format, as illustrated in Listing 1. The microservice then checks if all components are in compliance with the requirements described in Table A.1.1. Once it has checked the components, it returns a JSON to the backend, storing it in the mongoDB database.

3.3.1. *Microservice for rule-based checking of system integrity*

HVAC systems can be complex depending on the size. When handing over a BIM project of the HVAC system, it can be hard to uphold the level of detail, as promised in the Information, Communication, and Technology (ICT) contract of any building design phase. Therefore, it is highly beneficial to have a way to check that the system's integrity holds up. This microservice aims to provide a rule-based checking algorithm with a rule-set to check the FSC object model. The rule-set is shown in Table A.1.1. The source code is made available on GitHub.[1]

The functionality of this microservice was already briefly explained in Fig. 8, as a combination of HTTP POST requests, JSON file exchange, microservice computations, and storing of results. The microservice returns a JSON file to the database that describes whether each component lives up to the rules. The result of each component is returned as a Boolean value. The return values are then stored in the database.

3.3.2. *Microservice for airflow calculation of ventilation system*

With the creation of the object model in the central database, it is possible to traverse through the given HVAC system from one point to another. We created a microservice for airflow calculation of the ventilation system[2] to exemplify that the system can be traversed. The algorithm within the microservice starts by resetting all flows on the ventilation system. After resetting the flows, the algorithm takes the airflow demand available in each space and applies them to the air terminals contained in that space. The property ContainedInSpaces is used to find the connection between each AirTerminal and space. Fig. 9 shows an example of the next step to the algorithm. The algorithm takes the airflow of the terminal and then applies it to the next component's connector.

3.3.3. *Microservice to calculate HVAC statistics*

The primary purpose of the HVAC statistics microservice is to make the VC platform capable of displaying statistics on the HVAC system, including the number of components in the system. Furthermore, it summarizes the total meters of duct/pipe in the model. In summary, the HVAC statistics microservice allows for validation of the FSC object model or even makes it possible to calculate the material usage. Python-Flask was used to create the HVAC statistics calculator with an endpoint that the VC platform can utilize.[3] Listing 3 shows an example response from the microservice.

Listing 3: The listing shows an example of a response JSON from the microservice presented in Fig. 8. Each component in the system is counted, and the length of all FlowSegments are summarized into the given cross-sectional dimension.

4. Example models

This Section introduces two example models for showcasing the VC platform and the FSC Object Model in particular.

4.1. *Example model 1*

4.1.1. *The schematic / principle model*

Fig. 10 shows the first example model created by the authors of this article. The model contains a heating, cooling, and ventilation system, all connected. The heating system starts with a heat exchanger that converts the heat from the primary heating system (not depicted) to the secondary system, then branches out to a mixing loop for a heating coil (HeatingCoil) in the ventilation system and the radiators of each room. Each radiator is adjustable with a balancing valve (BalancingValve). The motorized valve (MotorizedValve) controls the mixing loop of the heating system.

[1] https://github.com/Virtual-Commissioning/VC-HVAC_rule_checking-Service

[2] https://github.com/Virtual-Commissioning/VC-Ventilation_dimensioning-service

[3] https://github.com/Virtual-Commissioning/VC-HVAC_statistics-Service

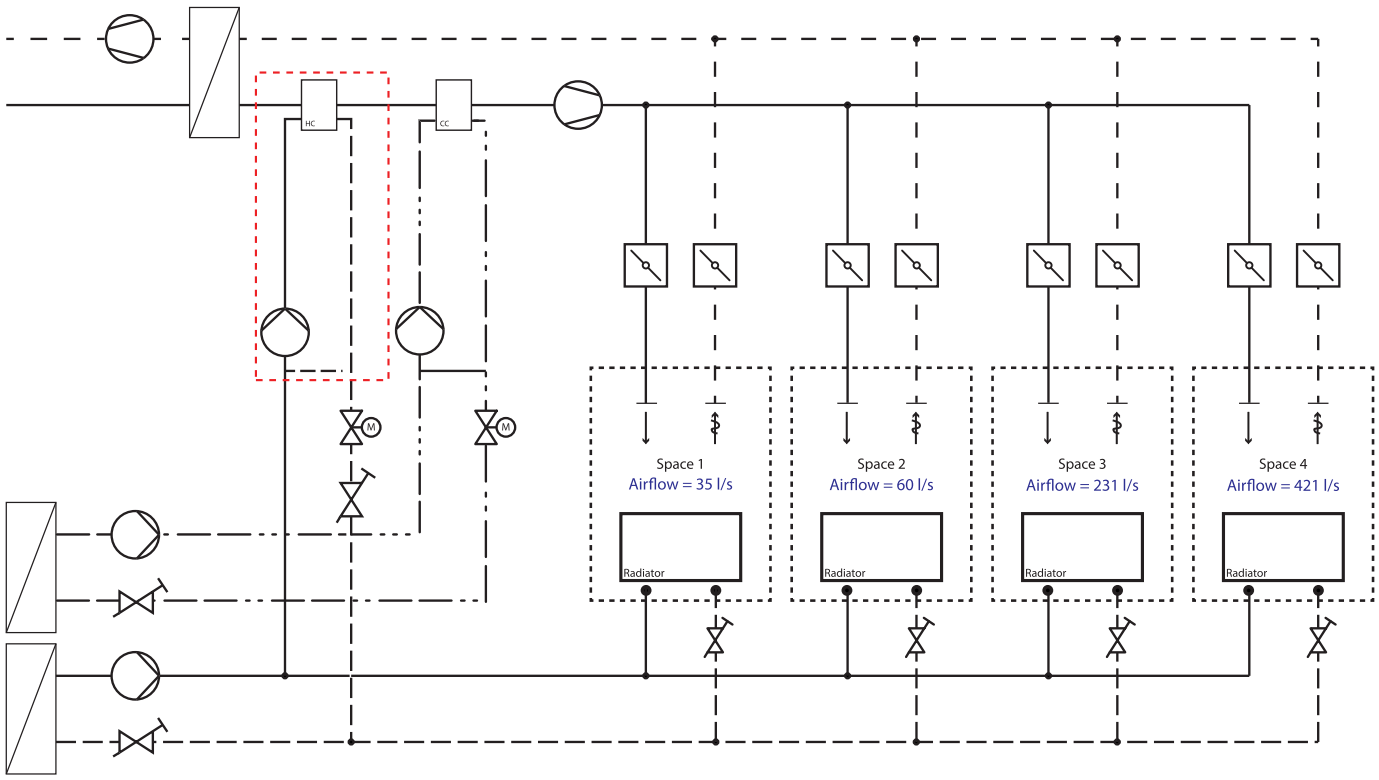


Fig. 10. Example model 1 mechanical schematic. The Figure contains three subsystems: heating, ventilation, and cooling. Furthermore, it includes four spaces that have an airflow. The radiators and the heating coil of the ventilation system provides heating to the room. Similarly, the cooling coil provides cooling by air to the room.

The cooling system is on the secondary side of the heat exchanger (HeatExchanger). The cooling liquid supplies the mixing loop provided by a pump. A mixing loop with another pump may seem excessive in this case, but was included as an example. The shunt is controlled with a motorized valve (MotorizedValve) and a pump (Pump).

The ventilation system contains a ventilation fan that takes the air from the air intake through a heat exchanger (HeatExchanger) and then a heating coil (HeatingCoil) and cooling coil (CoolingCoil) respectively. The air is supplied to space 1, 2, 3, and 4 with the use of air terminals (AirTerminal) controlled by a regulation damper. After supplying the air to the room, the air is extracted through the air terminal (AirTerminal). The air is then exhausted with the ventilation fan (Fan) after it has gone through the heat exchanger, exchanging any excess heat to the supply air.[4]

The example contains spaces, to exemplify the connection between the systems and spaces as seen in Section 3.3.2. All of the spaces are heated by ventilation and radiators, and are cooled by ventilation.

4.1.2. *Instantiating the object model*

This subsection visualizes the instantiated FSC object model for the Revit model shown in Fig. 11. The serialized JSON which represents the FSC object model is provided for the reader.[5] Fig. 10 shows a call-out with a red-dotted line. Fig. 12 visualizes part of the instantiation of the object model within the previously mentioned call-out of a system and then serializes it into JSON. Fig. 12 also shows how each component is instantiated with a relationship to the attached connector. For instance, the Tee (Tag: 1742043) is instantiated with Connector C2 and C3. This means that Connector C2 and C3 are instantiated within the ConnectedWith attribute of the Tee component. Connector C2, displayed in a blue box of Fig. 12, connects component 1742043 with component 1742044. Furthermore, the Connector class contains the physical properties of the connection port that interfaces with the adjacent component. Such physical properties include the dimension, shape, coordinates, and direction vector of the connector. The direction vector of the port will always orient away from the component. Finally, the ConnectorType displays whether another component supplies the connector or if it supplies another component. For instance, C2 suppliesFluidTo component 1742044, and C4 suppliesFluidFrom component 1742043.

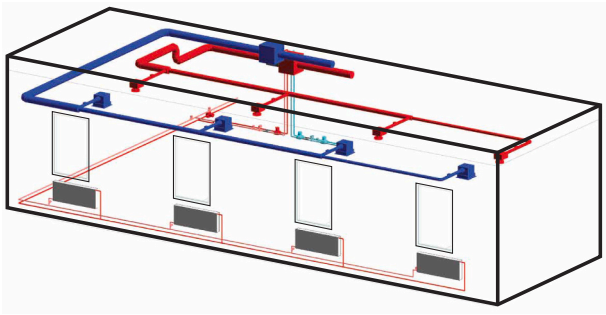


Fig. 11. The example model is shown in Fig. 10, modeled in Revit. The model contains the exact components and rooms shown in Fig. 10, except from the heat exchangers from the primary to the secondary system. The 3D model was modeled in Revit.

[4] https://github.com/Virtual-Commissioning/VC-HVAC_rule_checking-Service/blob/main/app/ressources/example_model_1.json

[5] https://github.com/Virtual-Commissioning/VC-HVAC_rule_checking-Service/blob/main/app/ressources/example_model_1.json

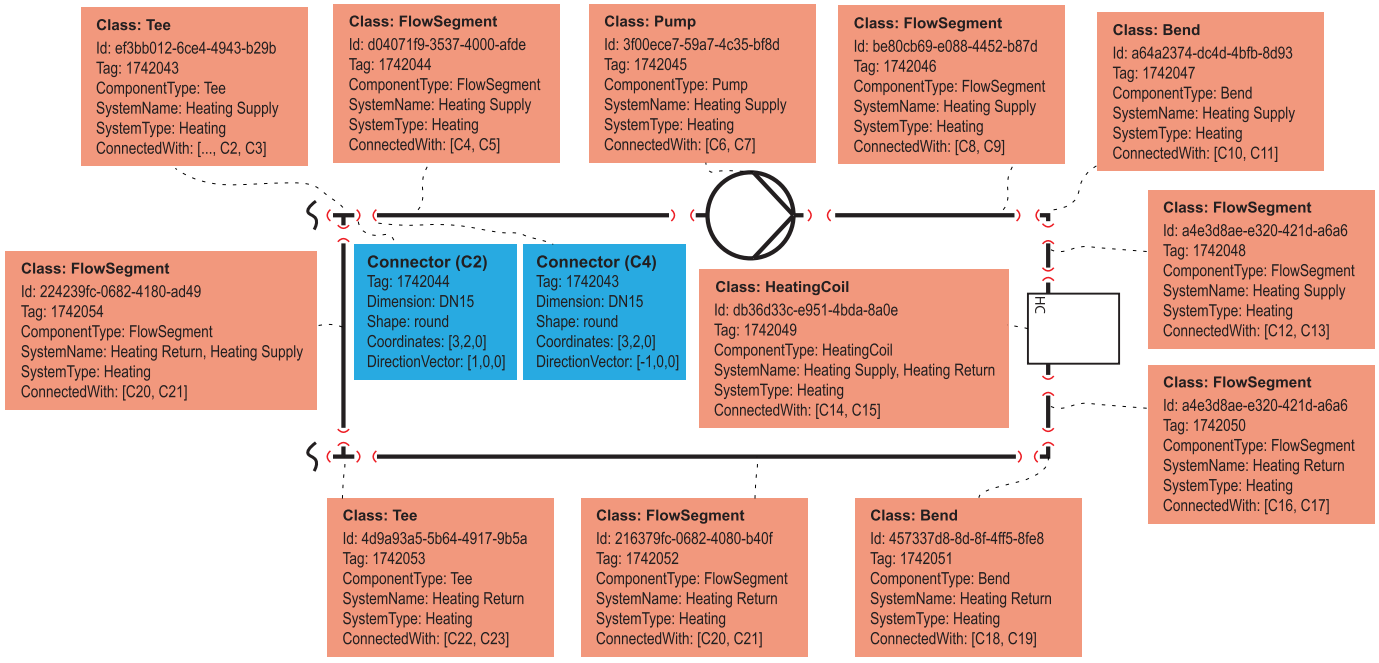


Fig. 12. The figure shows a schematic a small flow system. The red callout from Fig. 10 makes up the example seen in this figure. For simplicity, only the connectors (in the blue boxes) C2 and C4 are shown. Orange boxes make out a component, and blue boxes show the connectors related to a given component. The figure does not contain all properties for all components. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

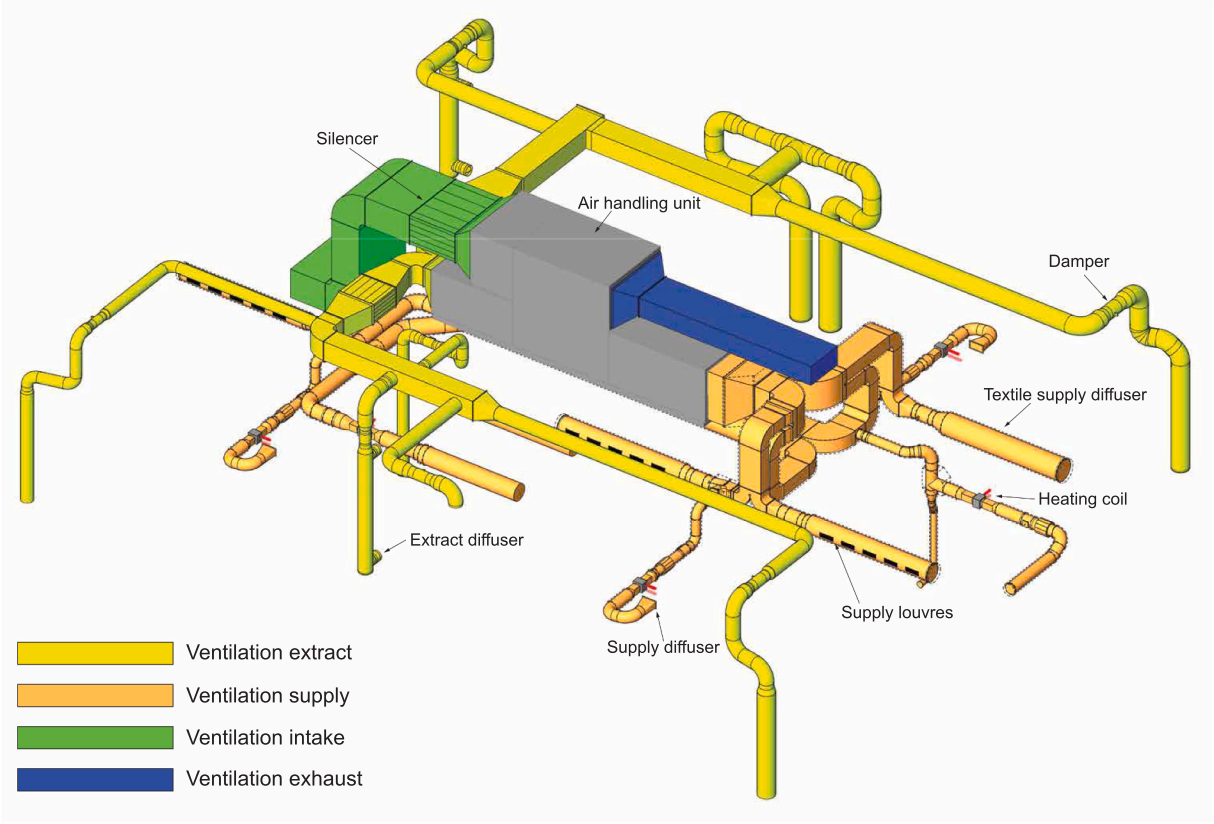


Fig. 13. Example model 2. Revit example model obtained from TU Eindhoven. The model shows a ventilation system with subsystems such as supply, extract, exhaust, and intake air.

4.2. Example model 2

4.2.1. The schematic/principle model

Fig. 13 illustrates the second example model. The Eindhoven University of Technology provided a real-case example model. The model contains a complex ventilation system with extract, supply, exhaust, and intake systems. Furthermore, it has heating coils attached to the supply-side of the ventilation system. The heating system is simplified and contains only the heating coil and the connected pipes. In general, this example aims to show the performance of the VC platform on an "imperfect" model.

The air handling unit (AHU) provides air to the ventilation system. The AHU was modeled as a box with four connectors in this example. This differs from the example shown in Section 4.1 by not specifying the components inside the AHU. A typical AHU consists of fans for supplying and extracting air, heating and cooling coils, silencers, and filters, i.e. it illustrates a "real world problem" in which not all modeling standards are the same - some designers would model all the components within the AHU while some designers (depending on company standards, and the design phase) would model the AHU as a box. The supply and extract system was modeled with variable air volume (VAV) dampers. This means that the ventilation system can vary the airflow in specific rooms. This example model does not contain any spaces, as these are contained in the architectural BIM model.

4.2.2. Instantiating the object model

We used the FSC exporter to generate the FSC object model, based on the Revit model, seen in Fig. 13. Next, the FSC object model was serialized into JSON and imported to the VC platform.

5. Results

This Section first displays the robustness of the format by visually explicating an example of the format. Following, it shows the platform's scalability with the use cases of a rule-based checker of the FSC object model, a BIM to airflow calculator, and an HVAC statistics tool.

5.1. Example model 1

5.1.1. Rule checking

The rule-based checking algorithm was used to see whether example model 1 (Section 4.1) lives up to the rule-set presented in the rule-based checking algorithm (Section 3.3.1). Table 1 shows that 219 out of 225 components lived up to the rules presented in Table A.1.1. The six components that did not live up to the rule-based check were the "open ended" components placed at the beginning and end of each system, including two from the ventilation system, two from the heating system, and two from the cooling system, which was expected.

5.1.2. HVAC statistics

Table 2 shows the result of running the HVAC statistics microservice from Section 3.3.3 on example model 1. The only component type counted differently by the microservice is the heat exchanger. Since the heat exchanger should always be connected with two systems - in this case the ventilation and the cooling system, and the ventilation and heating system, this is accepted. Even though every heat exchanger is represented twice in the format, it is also annotated with the same tag. Therefore, it is still possible to distinguish whether it appears twice.

Table 1
The table shows the result of running the rule-based checking algorithm on example model 1. The table shows that the FSC object model contains 225 components.

Components checked	True	False
225	219	6

Table 2
This table illustrates the total amount of components reported in Revit and after the transfer to the VC platform.

Components	Amount Revit	Amount VC
AirTerminal	8	8
MotorizedDamper	8	8
Bend	30	30
Reduction	40	40
Tee	14	14
BalancingValve	8	8
MotorizedValve	2	2
HeatExchanger	2	4
Fan	2	2
Pump	4	4
ShuntValve	2	2
PressureSensor	2	2
TemperatureSensor	2	2
Radiator	4	4
FlowSegment	96	96
Total components	223	225

Table 3
The table illustrates the length of the FlowSegments transferred from Revit to the VC platform, before and after the transfer. Each duct or pipe type is reported with its deviation from the Revit model to the VC platform. All measurements are in millimeters.

Duct Size	Length Revit	Length VC	Dev %
Round ducts			
Ø80	14,490	14,490	0
Ø125	15,430	15,430	0
Ø200	20,260	20,260	0
Pipes			
Ø15	24,259	24,280	<0.01
Ø18	7572	7570	<0.01
Ø22	39,613	39,620	<0.01

Table 3 shows that the length of the components was counted to a precision better than 0.01%. The precision is arguably caused by rounding off values in the microservice or Revit, which is normal and acceptable when dealing with geometry in different systems. Since the discrepancy is so small, it is considered insignificant.

5.1.3. Airflow calculation

Fig. 14 shows the airflow calculation microservice from Section 3.3.2 applied to example model 1, to visualize the functionality and result of the tool. If Space-1 has an airflow demand of 35 l/s, the airflow demand is applied to the air terminals that are contained in Space-1. Then it is added to the existing airflow on the ventilation duct up to the ventilation fan. The script was tested by applying it to the example model described in Fig. 4.1.1. By using the microservice to analyze the ventilation system, it was found that the total airflow needed to run the system was 747 l/s

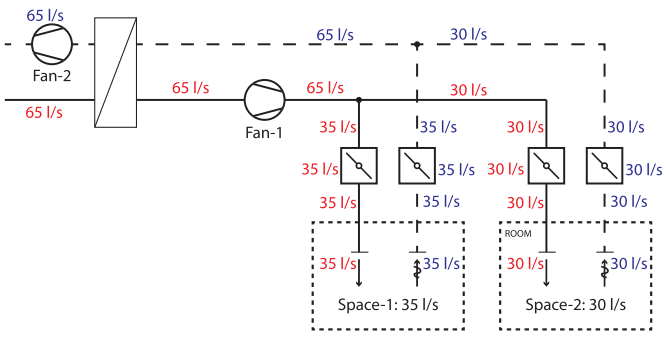


Fig. 14. The ventilation system is traversed and airflows are summed along the supply and return paths.

for the supply and return ventilation fan. The airflow calculation is one of the first steps in choosing a fan that can meet the airflow demand of the system.

5.2. Example model 2

This Section shows a use case test of the HVAC exporter on the example model introduced in Section 4.2. A BIM model has been obtained from the industry to test the performance of the Revit to the VC platform with the use of the FSC exporter. The transfer will be quantified with the HVAC statistics tool (Section 3.3.3) and the HVAC rule checker tool (3.3.1). The purpose is to test the created FSC object model exporter tool with a BIM model that has not been built by the authors of this report. Once the BIM model has been transferred from Revit into the VC platform, the microservices for rule-based checking (Section 3.3.1) and HVAC statistics (Section 3.3.3) will be used to quantify how well the FSC object model based on the BIM model has been transferred into the VC platform.

5.2.1. Rule checking

Table 4 shows the result of the rule-based checking algorithm run on the transfer of the use case from Fig. 13. The table shows that 158 components out of 493 lived up to the rule-check proposed in Table A.1.1. That means that the majority of elements did not pass this check. This will be documented further below in this article, yet the main reason is that the system in the particular building is not as complete and correct as expected by the rules developed in this microservice.

5.2.2. HVAC statistics

Table 5 shows the number of components reported in Revit and the number of components reported after the transfer from Revit to the VC platform with the use of the FSC exporter. The total amount of components for Revit and the VC platform was 487 and 493, respectively. This is explained in the way that FSC divides the flow system. The full system contains both a ventilation and heating system in the model (The heating system only consists of a few pipes connected to the heat exchangers). In the example model, the ventilation and heating systems are connected by heat exchangers. Fig. 3 shows an example of this, near the heat exchangers. This behavior intends to provide an internal connection for each of the systems.

Table 6 shows the length of the FlowSegments reported in Revit and the length of the FlowSegments reported after the transfer from Revit to the VC platform with the use of the FSC exporter. In Table 6 it is reported that not every duct or pipe has the same length after it has been transferred to the VC platform. This behavior is caused by an incorrect Revit model, that does not contain “correct connectivity”. This problem is caused by the microservice not being able to handle specific cases, like it is seen in Fig. 15 where several AirTerminals are placed on the duct. This behavior has not been accounted for in the microservice. It also explains why the Ø630 ducts are not counted a single time in the HVAC statistics microservice. Listing 4 shows the JSON structure in the FSC format behind the object shown in Fig. 15. The Figure shows that five air terminals have been placed directly on the ventilation duct, which is usually not expected. This means that the FlowSegment has a total of 7 connectors.

Listing 4 The listing shows the component mapped from Fig. 15. Not all the connectors are shown, to improve readability of the JSON.

Table 4
The table shows the result of running the rule-based checking algorithm on Fig. 13. The table shows that the FSC object model for example model 2 contains 493 components.

Components checked	True	False
493	158	335

Table 5
This table illustrates the total amount of components reported in Revit and after the transfer of the FSC object model to the VC platform.

Components	Amount Revit	Amount VC
AirTerminal	33	33
Cap	14	14
Bend	69	69
Reduction	125	125
Tee	13	13
BalancingDamper	29	29
HeatExchanger	6	12
FlowSegment	198	198
Total components	487	493

Table 6
This table illustrates the length of the FlowSegments transferred from Revit to the VC platform, before and after. Each duct or pipe type are reported with their deviation from the Revit model to the VC platform. All measurements are in millimeters.

Duct Size	Length Revit	Length VC	Dev %
Round ducts			
Ø160	2862	340	88
Ø250	7250	540	93
Ø315	24,644	21,680	12
Ø355	3624	3624	0
Ø400	31,443	21,120	33
Ø450	12,653	6430	49
Ø500	33,286	24,980	25
Ø600	8152	8152	0
Ø630	21,916	0	100
Square ducts			
1200 × 600	5600	0	100
200 × 200	3402	3402	0
2100 × 900	4535	3390	25
2178 × 1538	500	500	0
350 × 350	83	0	100
400 × 400	33	0	100
600 × 600	5214	340	93
700 × 350	1868	0	100
700 × 400	2118	0	100
700 × 600	5238	620	88
800 × 400	18,142	0	100
850 × 600	3714	2840	24
Pipes			
Ø32	2016	2016	0

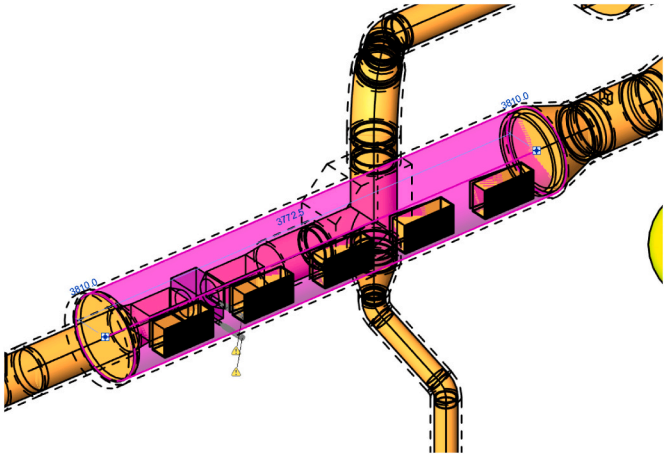


Fig. 15. The ventilation duct in this figure has seven connectors. It has the beginning and the end of the flow segment, but it also has five connectors from the 5 air terminals that have been placed directly on the ventilation duct.

6. Discussion and future work

The VC platform is a common data environment used for HVAC projects during the building's design, construction, and operation phases. This paper describes the process required to develop a CDE and proposes a roadmap for further research and development. The requirements for the VC CDE were that it should expose the proprietary format of a Revit HVAC system in a commonly available platform. Using Python-Flask microservices, we illustrated how the VC platform allows for continuous integration and continuous deployment of functionalities on the platform. The VC platform can scale in any direction with the integration of microservices. Furthermore, it allows for the extension of the FSC object model since the system architecture is based on a microservice architecture. For instance, if the exporter from Revit to the FSC object isn't working, that can be tested and patched independently of touching the code for any other microservices.

6.1. Achievements

We created a CDE with a microservice architecture and showed that the CDE is easily scalable by implementing simple microservices such as the airflow calculator (see Section 3.3.2). A rule-based checker was also implemented to check if the data transfer was successful from Revit to the FSC object model in the VC platform. Furthermore, a microservice was implemented to provide statistics of the data transfer carried out. It offers the user insights into the flow system, such as the length of different ventilation ducts or pipes. The FSC diagram and the FSC exporter allow for the serialization of the FSC object model. A mongoDB OOD stores the FSC object model. The FSC object model creates a connected network of ducts and pipes, making it possible to represent the nature of a flow system. The FSC exporter allows the user to create an FSC object model serialized in JSON to work within the VC platform - or even in external platforms. We used the FSC exporter on an externally developed Revit model and were able to transfer the full FSC object model to the web application to run microservices on it. We proved that the FSC object model is able to link the demand of a space with the airflow system. This can be used to create dimensioning tools for ventilation systems or heating- and cooling systems. In order to enable future use of the VC platform, the authors of this article will continuously maintain and update the platform with new features.

6.2. Limitations of the study

We created BIM model with LOD350 to evaluate the VC platform and the FSC object model, called example model 1. Therefore, the model used to evaluate the VC and FSC was “created to succeed”. For instance, all components must have precisely the right amount of connectors. There is a risk of designing the VC platform for “the perfect scenario” where all data is available in the BIM model. It is often a challenge in the AEC industry that BIM models do not contain all the data necessary for a complete model, like component connectivity, component naming, system naming, etc. To handle problems like this, the AEC industry in Denmark has introduced ICT agreements on building projects. It is a contract that obligates the company to deliver building documentation of a certain standard. However, while LOD descriptions are relatively detailed, consulting engineering companies in the HVAC branch still struggle to provide models that live up to the LOD350 standard. We built the VC platform to handle the delivery issues in the HVAC branch by integrating a microservice architecture. For instance, if the HVAC system in the database has an error, it can be fixed later in the VC platform using the microservices. One of the microservices is the rule-based checking algorithm shown in Section 3.3.1. The rule-based checker will let the user know that some information is missing. For instance, such missing information could be that some components are connected incorrectly. Then, it is possible to specify the actual connectivity directly in the VC platform.

The VC platform was tested on a model developed externally to emulate the situation of an imperfect BIM model, as seen in Section 4.2. In Section 5.2 the HVAC Statistics microservice in Section 3.3.3 and the rule-based checking algorithm in Section 3.3.1 was run on example model 2. The results showed that all components were transferred successfully to the VC platform, see Table 5. However, the performance test with the rule-based checking algorithm showed that 158 of 493 components lived up to the rules established in Table A.1.1. The results of this performance test were caused by, for instance, the modeling practice shown in Fig. 15. The figure highlights that the FSC exporter does not always map the components correctly. This behavior is expected for any Revit model that has not been modeled with the intent to transfer it to the database. The user can use the rule-based checking microservice to find which components do not live up to this algorithm. The 3D-viewer of the VC platform visualizes the results and alerts the user which components do not follow the rule-set. Thereby, the user can solve the issues manually.

The test case presented in Section 4 showed the application of the FSC object model together with the VC platform. As part of the platform's deployment, more extensive testing should be carried out on Revit models to ensure the robustness of the FSC exporter and platform.

There are issues that the VC platform cannot solve. For example, the FSC exporter is highly dependent on the template used within the Revit model. If a flow segment like a pipe is modeled as the “Mechanical Equipment” category instead of a “Pipe” category in Revit, then the pipe will be mapped into the wrong category or not mapped at all. Such a fault will cause an exception within the FSC exporter, meaning that an error will cause the exporter to abort the operation and, therefore, fail to export the FSC object model from Revit to the database.

The FSC exporter was designed to link the Revit model and the database. The FSC exporter demonstrates that it is possible to extract information from a BIM tool like Revit to work within a database. For the purpose of this article, Revit families from the Rambøll library was utilized and modified to be able to represent all the information needed to export it into the FSC object model. The work presented in this article does not exclude exporters from other file formats to be made. Such integration could include the open file format IFC. The VC platform is not dependent on the file coming from Revit or any other proprietary format, as long as it follows the FSC diagram.

It will require detailed HVAC models for advanced hydraulic simulations in Modelica. This presents a problem for the AEC industry with the current status of BIM modeling. The AEC industry needs to model buildings more realistically and contain information in the BIM model in an early design phase to provide a correct design based on actual performance rather than rule-of-thumb. However, we believe that with the younger generation coming into the AEC industry, the market is ripe for the digital transformation it will take. Performing these simulations in the early design phase will be more time spent on design than fixing issues during the physical commissioning or operation phase. In the company supporting this article, the method will be employed to do just that - to save time in the long run.

6.3. Roadmap for future development

The VC platform and the FSC object model have been carried out as part of a development to create a CDE for full building simulation using BIM models. This paper introduces the FSC object model, and exemplifies how to use microservices for calculation or even simulations. The VC platform is envisioned as a three-stage development project. The stages are illustrated in Fig. 16. *Stage 1* is the work presented in this paper and is the initial development of the VC platform and the FSC object model with the simple calculation microservices. The microservices seek to prove that adding microservices to the VC platform is possible. The authors of this article recognizes that the microservices are very simple, and constitute services that already exist within BIM programs like Revit. The microservices were added to exemplify that tools

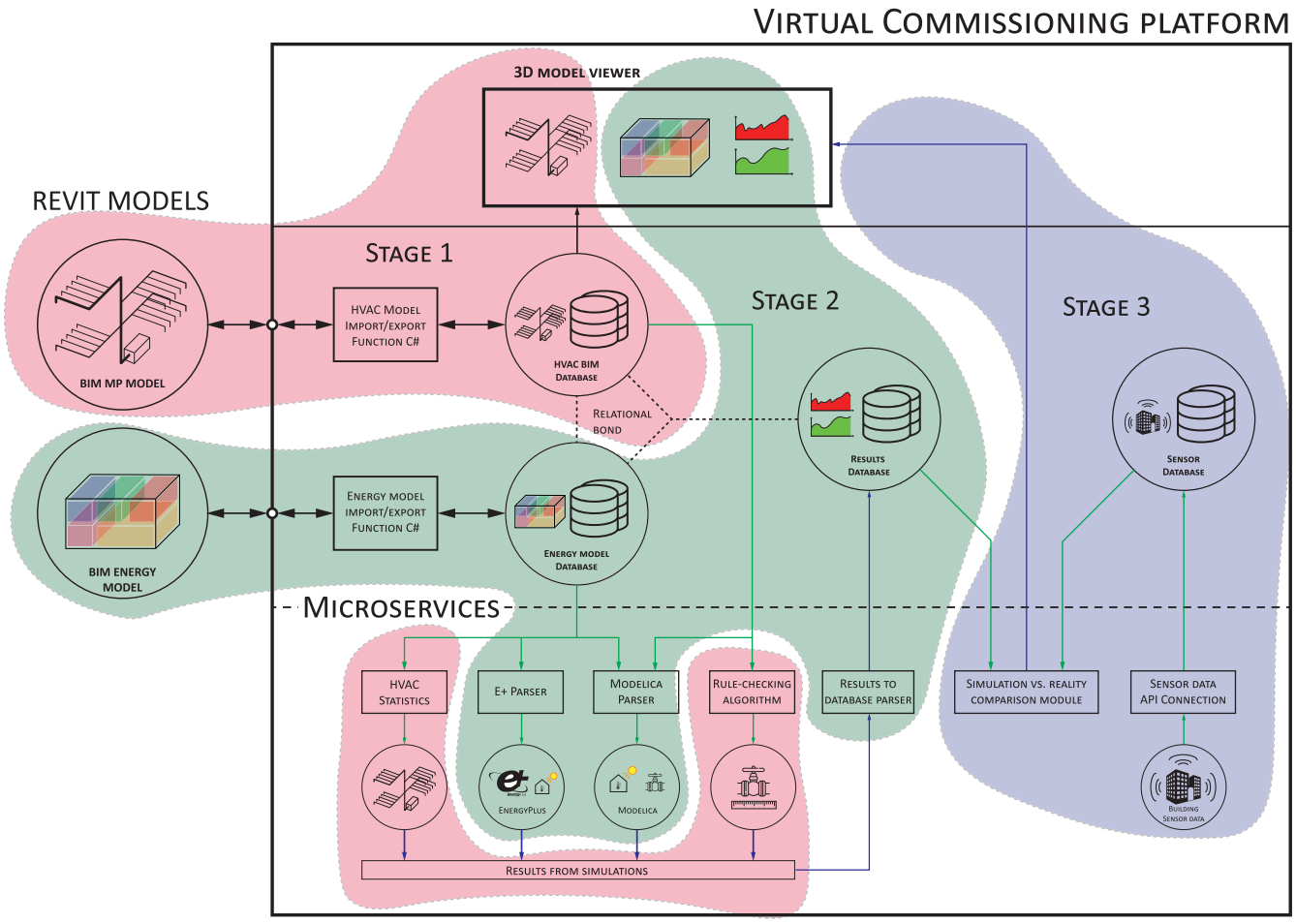


Fig. 16. The Figure shows a roadmap for the future development of the VC platform. The areas marked with red represent the work developed for and presented by this article. The area marked in green represents the next step in developing the VC platform. Finally, the blue area represents the final step, a module that includes sensor data in the VC platform. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

developed within Revit can easily be replicated to a cloud-based CDE based on a non-proprietary format, as opposed to the proprietary format of Revit. The early development of the VC platform and the FSC object model provides a stepping stone for developing a complete CDE for continuous integration of simulation tools through microservices. *Stage 2* includes the development of a space class object model to represent a BIM energy model. A simple space class object model was presented in Section 3.1.4. Fig. 16 shows that the "HVAC BIM database" and the "Energy model database" share a relational bond. Thereby, it will be possible to link these two data formats together. Finally, stage 2 introduces the possibility of running whole building simulations through Modelica and EnergyPlus with the use of Spawn Of EnergyPlus in Modelica [41]. Running simulations on indoor climate and HVAC systems simultaneously might help perform more accurate predictive energy models. The work on stage 2 has already begun, and the authors of this article have proved that the link from CDE to Modelica-based Dymola, is possible [21]. Fjerbæk et al. [21] simulated a small heating system in Modelica and was capable of showing the return temperature for each heating loop. The toolchain created by Fjerbæk et al. [21] made it possible to easily initiate Modelica simulations of a heating system - a process that under normal circumstances would be very time consuming, due to the manual labour of creating a Modelica simulation model. *Stage 3* will include sensor data and connect it to the BIM model with a relational bond. It will be possible to take data from the operating building and do continuous fault detection on it with sensor data. That way, the digital twin in the VC platform can inform the Building Management System (BMS) of the actual building to make certain adjustments. An example of an adjustment could be to run with the objective of minimizing energy costs (as opposed to minimal energy usage).

7. Conclusions

The three aims of this paper were to:

1. Centralize BIM project data so all stakeholders have access to a single source of truth (SSOT) in a CDE based in a web application
2. Create a data structure that can represent a flow system
3. Allow for easy scalability of the web application utilizing the principle of microservice architecture.

This article introduces a CDE called the VC platform. The article exemplifies a paradigm shift from a proprietary file-based BIM model to a web-based database BIM model. The VC platform allows for the development of applications in a fully modularized way through microservices. Microservices make it possible to deploy custom applications that run specific tasks independently. We developed the VC platform to enable advanced simulations of HVAC systems that relate to the actual spaces of the building. Future work has been planned to integrate Modelica and Spawn of EnergyPlus as microservices on the VC platform. An externally provided Revit model was used to test the

performance of the VC platform. The performance test proved that the FSC exporter from Revit to the VC platform worked as intended – it transferred all components to the database in the VC platform. After transferring Revit with the FSC exporter to the VC platform, the performance test revealed that not all components were compliant with the rule-based checking algorithm. However, this is not problematic, as the rule-based checking algorithm intends to highlight errors like this so the user can solve them in the VC Platform frontend.

Declaration of Competing Interest

The authors recognize that there is a potential for conflicts of interest via industry affiliations. Mikki Seidenschnur is working on a doctoral dissertation in Technical University of Denmark, while also working in Ramboll. Ali Kücükavci is working on a doctoral dissertation in Technical University of Denmark, while also working in COWI. Esben Visby Fjerbæk is working as a research assistant at Technical University of Denmark Kevin Michael Smith is working as a researcher at Technical University of Denmark Pieter Pauwels is working as a professor at Technical University of Eindhoven Christian Anker Hviid is working as a associate professor at Technical University of Denmark.

Acknowledgments

Funding: This work was funded by the Ramboll Foundation and the Innovation Fund Denmark (grant 9065-00266A). The use case test model in Section 4.2 was provided by TU Eindhoven. We would like to acknowledge the work of the reviewers of this manuscript for improving the quality of the manuscript with thorough and constructive comments on content and writing.

Appendix A

Table A.1
This table illustrates the rules that exist with all the subtypes of components in the class object model.

Subclass of Component	Rules
FlowSegment	Contains two connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
HeatExchanger	Contains 4 connectors
	Contains two connectors: “suppliesFluidFrom” and two connectors: “suppliesFluidTo”
	Is contained within two different subsystems
Radiator	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
Bend	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
	Has an angle greater than 0
Cross	Contains 4 connectors
	Contains at least one connector of “suppliesFluidFrom” and at least one connector of “suppliesFluidTo”
Reduction	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
Tee	Contains 3 connectors
	Contains at least one connector: “suppliesFluidFrom” and at least one connector: “suppliesFluidTo”
BalancingDamper	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
MotorizedDamper	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
FireDamper	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
BalancingValve	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
CheckValve	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
DifferentialPressureValve	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
MotorizedValve	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
SafetyValve	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
ShuntValve	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
Fan	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
Pump	Contains 2 connectors
	Contains one connector: “suppliesFluidFrom” and one connector: “suppliesFluidTo”
AirTerminal	Contains 1 connector
	Contains connector of “suppliesFluidFrom” if supply system
	Contains connector of “suppliesFluidTo” if return system

Appendix B. UML class diagram FSC

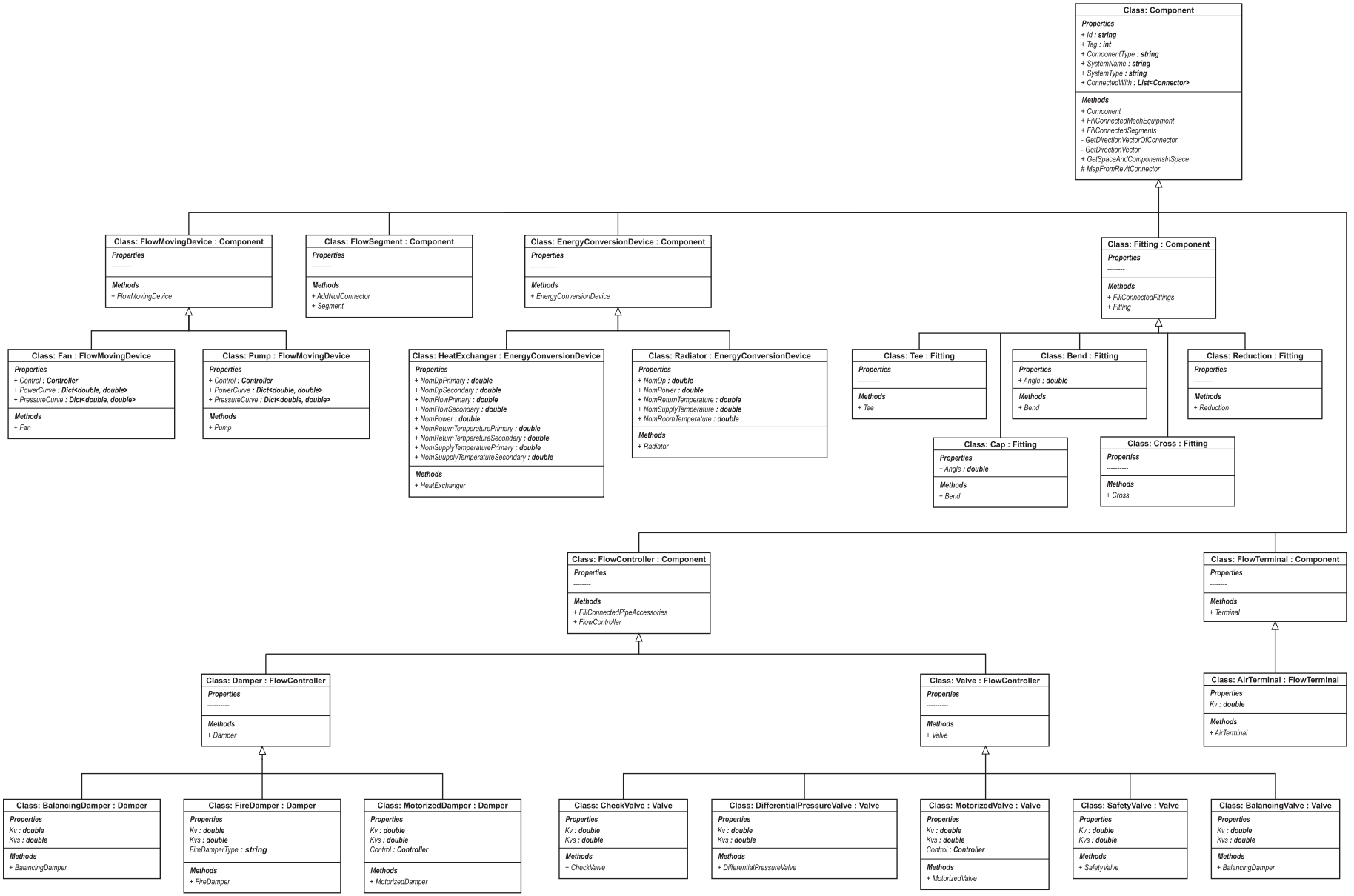


Fig. B.1. Illustration of the full FSC UML diagram. It includes all elements that inherit from component. It extends the diagram displayed in Fig. 2. The methods displayed in the UML are used to create the FSC object model.

References

[1] K.M. Kensek, Handbook of green building design and construction, Build. Informa. Model. (2014) 1–285, https://doi.org/10.4324/9781315797076.

[2] A. Andriamamonjy, D. Saelens, R. Klein, An automated IFC-based workflow for building energy performance simulation with Modelica, Automation in Construction 91 (September 2017), 2018, pp. 166–181, https://doi.org/10.1016/j.autcon.2018.03.019.

[3] B. Hardin, D. McCool, BIM and Construction Management: Proven Tools, Methods, and Workflows, 2nd edition, Sybex/Wiley, 2015.

[4] F.H. Abanda, L. Byers, An investigation of the impact of building orientation on energy consumption in a domestic building using emerging BIM (Building Information Modelling), Energy 97 (2016) 517–527, https://doi.org/10.1016/j.energy.2015.12.135.

[5] T.O. Olawumi, D.W. Chan, Identifying and prioritizing the benefits of integrating BIM and sustainability practices in construction projects: a Delphi survey of international experts, Sustain. Cities Soc. 40 (February) (2018) 16–27, https://doi.org/10.1016/j.scs.2018.03.033.

[6] J.P. Carvalho, L. Bragança, R. Mateus, Optimising building sustainability assessment using BIM, Autom. Constr. 102 (2018) (2019) 170–182, https://doi.org/10.1016/j.autcon.2019.02.021. URL 10.1016/j.autcon.2019.02.021.

[7] R.E. Edwards, E. Lou, A. Bataw, S.N. Kamaruzzaman, C. Johnson, Sustainability-led design: feasibility of incorporating whole-life cycle energy assessment into BIM for refurbishment projects, J. Build. Eng. 24 (February) (2019), 100697, https://doi.org/10.1016/j.jobe.2019.01.027. URL 10.1016/j.jobe.2019.01.027.

[8] K. Safari, H. AzariJafari, Challenges and opportunities for integrating BIM and LCA: methodological choices and framework development, Sustain. Cities Soc. 67 (2020) (2021), 102728, https://doi.org/10.1016/j.scs.2021.102728.

[9] T.P. Obrecht, M. Röck, E. Hoxha, A. Passer, BIM and LCA integration: a systematic literature review, Sustainability (Switzerland) 12 (14) (2020) 1–19, https://doi.org/10.3390/su12145534.

[10] B. Succar, W. Sher, A. Williams, Measuring BIM performance: five metrics, Architect. Eng. Des. Manag. 8 (2) (2012) 120–142, https://doi.org/10.1080/17452007.2012.659506.

[11] J. Beetz, N. Gu, BIMserver.org - an open source IFC model server, in: Proceedings of the CIB W78 2010, 2009, pp. 1–9. URL, https://www.academia.edu/1905765/BIMSERVER_ORG_AN_OPEN_SOURCE_IFC_MODEL_SERVER.

[12] J.C. Cheng, M. Das, A bim-based web service framework for green building energy simulation and code checking, J. Inform. Technol. Construct. 19 (June) (2014) 150–168. URL, http://www.itcon.org/2014/8.

[13] M. Wetter, C.V. Treeck, L. Helsen, A. Maccarini, D. Saelens, D. Robinson, G. Schweiger, IBPSA project 1: BIM / GIS and Modelica framework for building and community energy system design and operation – ongoing developments, lessons learned and challenges, in: IOP Conf. Ser.: Earth Environ. Sci., 2019, https://doi.org/10.1088/1755-1315/323/1/012114.

[14] G.B. Porsani, K.D.V. de Lersundi, A.S.O. Gutiérrez, C.F. Bandera, Interoperability between building information modelling (Bim) and building energy model (bem), Appl. Sci. 11 (5) (2021) 1–20, https://doi.org/10.3390/app11052167.

[15] K.U. Ahn, Y.J. Kim, C.S. Park, I. Kim, K. Lee, BIM interface for full vs. semi-automated building energy simulation, Energy Build. 68 (PART B) (2014) 671–678, https://doi.org/10.1016/j.enbuild.2013.08.063. URL 10.1016/j.enbuild.2013.08.063.

[16] C. Park, HVACSIM+ User's Guide Update, 2008, https://doi.org/10.6028/NIST.IR.7514. URL, http://www.fire.nist.gov/bfrlpubs/build08/PDF/b08030.pdf.

[17] Equa, IDA Indoor Climate and Energy, URL, https://www.equa.se/en/ida-ice.

[18] L. B. N. Laboratory, Modelica Buildings Library, URL, https://simulationresearch.lbl.gov/modelica/, 2022.

[19] M. Wetter, Modelica-based modelling and simulation to support research and development in building energy and control systems, J. Build. Perform. Simul. 2 (2) (2009) 143–161, https://doi.org/10.1080/19401490902818259.

[20] W. Zuo, M. Wetter, W. Tian, D. Li, M. Jin, Q. Chen, Coupling indoor airflow, HVAC, control and building envelope heat transfer in the Modelica Buildings library, J. Build. Perform. Simul. 9 (4) (2016) 366–381, https://doi.org/10.1080/19401493.2015.1062557.

[21] E.V. Fjerbæk, M. Seidenschnur, A. Kücükavci, K.M. Smith, C.A. Hviid, From BIM databases to Modelica - automated simulations of heating systems, in: REHVA 14th HVAC World Congress, 2022, pp. 1–7, https://doi.org/10.34641/clima.2022.365.

[22] J.B. Kim, W. Jeong, M.J. Clayton, J.S. Haberl, W. Yan, Developing a physical BIM library for building thermal energy simulation, Autom. Constr. 50 (C) (2015) 16–28, https://doi.org/10.1016/j.autcon.2014.10.011.

[23] W.S. Jeong, J.B. Kim, M.J. Clayton, J.S. Haberl, W. Yan, A framework to integrate object-oriented physical modelling with building information modelling for building thermal simulation, J. Build. Perform. Simul. 9 (1) (2016) 50–69, https://doi.org/10.1080/19401493.2014.993709.

[24] D. Jansen, E. Fichter, V. Richter, A. Barz, J. Brunkhorst, M. Dahncke, P. Jahangiri, C. Warnecke, P. Mehrfeld, M. Dirk, C.V. Treeck, L. Bruno, R. Otto, M. Technik, C. Kg, BIM2SIM -Development of semi-automated methods for the generation of simulation models using Building Information Modeling BIM2SIM – Development of semi-automated methods for the generation of simulation models using Building Information Modeling (September), 2021, pp. 2–4.

[25] A. Andriamamonjy, R. Klein, D. Saelens, Automated grey box model implementation using BIM and Modelica, Energy Build. 188-189 (2019) 209–225, https://doi.org/10.1016/j.enbuild.2019.01.046. URL doi:10.1016/j.enbuild.2019.01.046.

[26] S. Hauer, A. Bres, R. Parti, M. Monsberger, An approach for the extension of openBIM MEP models with metadata focusing on different use cases, Build. Simul. Conf. Proc. 1 (2019) 182–189, https://doi.org/10.26868/25222708.2019.210932.

[27] P. Pauwels, S. Zhang, Y.C. Lee, Semantic web technologies in AEC industry: a literature overview, Autom. Constr. 73 (2017) 145–165, https://doi.org/10.1016/j.autcon.2016.10.003. URL 10.1016/j.autcon.2016.10.003.

[28] K. Afsari, C.M. Eastman, D. Castro-Lacouture, JavaScript object notation (JSON) data serialization for IFC schema in web-based BIM data exchange, Autom. Constr. 77 (2017) 24–51, https://doi.org/10.1016/j.autcon.2017.01.011. URL 10.1016/j.autcon.2017.01.011.

[29] D.Y. Lee, H.L. Chi, J. Wang, X. Wang, C.S. Park, A linked data system framework for sharing construction defect information using ontologies and BIM environments, Autom. Constr. 68 (2016) 102–113, https://doi.org/10.1016/j.autcon.2016.05.003.

[30] C. Quinn, A.Z. Shabestari, T. Misic, S. Gilani, M. Litoiu, J.J. McArthur, Building automation system - BIM integration using a linked data structure, Autom. Constr. 118 (May) (2020), 103257, https://doi.org/10.1016/j.autcon.2020.103257.

[31] S. Tang, D.R. Shelden, C.M. Eastman, P. Pishdad-Bozorgi, X. Gao, BIM assisted building automation system information exchange using BACnet and IFC, Autom. Constr. 110 (2019) (2020), 103049, https://doi.org/10.1016/j.autcon.2019.103049. URL 10.1016/j.autcon.2019.103049.

[32] K. Kim, H. Kim, W. Kim, C. Kim, J. Kim, J. Yu, Integration of ifc objects and facility management work information using Semantic Web, Autom. Constr. 87 (2017) (2018) 173–187, https://doi.org/10.1016/j.autcon.2017.12.019.

[33] B. Dong, Z. O'Neill, Z. Li, A BIM-enabled information infrastructure for building energy fault detection and diagnostics, Autom. Constr. 44 (2014) 197–211, https://doi.org/10.1016/j.autcon.2014.04.007.

[34] A. Gouda Mohamed, M.R. Abdallah, M. Marzouk, BIM and semantic web-based maintenance information for existing buildings, Autom. Constr. 116 (March) (2020) 103209, https://doi.org/10.1016/j.autcon.2020.103209.

[35] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Berges, D. Culler, R. Gupta, M.B. Kjærgaard, M. Srivastava, K. Whitehouse, Brick: Towards a unified metadata schema for buildings, in: BuildSys '16: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, 2016, pp. 41–50, https://doi.org/10.1145/2993422.2993577.

[36] C. Preidel, A. Borrmann, C. Oberender, M. Tretheway, Seamless integration of common data environment access into BIM authoring applications: the BIM integration framework, eWork and eBusiness in architecture, Eng. Construct. (2016) 119–128. https://doi-org.proxy.findit.cvt.dk/10.1201/9781315386904.

[37] L.D. Lauretis, From monolithic architecture to microservices architecture, IEEE Int. Symp. Software Reliability Eng. Workshops (ISSREW) (2019) 93–96, https://doi.org/10.1109/ISSREW.2019.00050.

[38] J. Thönes, Microservices, IEEE Softw. 32 (1) (2015), https://doi.org/10.1109/MS.2015.11.

[39] V. Kukkonen, A. Kücükavci, M. Seidenschnur, M.H. Rasmussen, K.M. Smith, C. A. Hviid, An ontology to support flow system descriptions from design to operation of buildings, Autom. Construct. 134 (2020) (2022), 104067, https://doi.org/10.1016/j.autcon.2021.104067.

[40] M. Wetter, W. Zuo, T.S. Nouidui, X. Pang, Modelica Buildings library, J. Build. Perform. Simul. 7 (4) (2014) 253–270, https://doi.org/10.1080/19401493.2013.765506.

[41] M. Wetter, T.S. Nouidui, D. Lorenzetti, E.A. Lee, A. Roth, Prototyping the next generation energyplus simulation engine, in: 14th International Conference of IBPSA - Building Simulation 2015, BS 2015, Conference Proceedings, no. April 2016, 2015, pp. 403–410. URL, https://www.semanticscholar.org/paper/PROTOTYPING-THE-NEXT-GENERATION-ENERGYPLUS-ENGINE-Wetter-Nouidui/8faf811f9752b54dd56ab6e60ebf204ff7c74cfc.