Taking advantage of semantic web ontologies and shape constraints for Heating, Cooling and Ventilation Systems

Kücükavci, Ali; Seidenschnur, Mikki; Negendahl, Kristoffer; Hviid, Christian Anker

Published in:
E3S Web of Conferences

Link to article, DOI:
10.1051/e3sconf/202236204002

Publication date:
2022

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

Taking advantage of semantic web ontologies and shape constraints for Heating, Cooling and Ventilation Systems

Ali Kücükavci[1], Mikki Seidenschnur[1,2], Kristoffer Negendahl[1], Christian Anker Hviid[1]
[1]Department of Civil and Mechanical Engineering, Technical University of Denmark, Denmark
[2]Ramboll, Copenhagen, Denmark

Abstract

In recent years semantic web ontologies have improved data interoperability within architecture, engineering, construction, and operation of buildings. One of the persisting issues inhibiting quality assurance is a lack of robust model validation of BIM models used for HVAC flow system simulation and analysis. This article provides a novel approach for automating the BIM validation process using SHACL shapes and FSO/FPO ontologies. Using this approach will ensure that the BIM model contains the required HVAC information for simulating hydraulic systems. The paper presents multiple shapes developed to identify and validate typical HVAC design details in buildings.

Introduction

Knowledge graphs and linked data has proven useful for the representation of Building Information Modeling (BIM) models in the Architecture, Engineering, Construction, and Operation (AECO) industry in the recent years (Rasmussen et al., 2021; Balaji et al., 2016; Pauwels and Terkaj, 2015). Graph representations of BIM models has potential uses in many different aspects of design and engineering. However, one issue in particular ties to the interoperability with the two most common formats for common data exchange, Industry Foundation Classes (IFC) and green building Extensible Markup Language (gbXML). These data formats are not designed to carry over all the data relevant for energy and indoor climate simulations, or Heating, Ventilation, and Cooling (HVAC) simulation data (Redmond et al., 2012). A recent attempt to challenge this issue is Porsani et al. (2021) who shows the difficulty in transforming BIM models to Building Energy Model (BEM) models, where the aspect of transformation between proprietary formats such as Revit (.rvt) to IFC and gbXML inherently generates errors later in the simulation process.

Kukkonen et al. (2022) proposed a semantic web ontology called Flow Systems Ontology (FSO) to describe the composition of flow systems and their energy and mass flow. Kukkonen et al. (2022) provided a comprehensive roadmap for further developments showing that use cases should be developed to further the development of the specific ontology of FSO. Furthermore, it was proposed that an ontology should be developed to include the properties of flow systems (component sizes, flow, material, i.e.). Though FSO proposes a common language to describe flow systems, the data needs to be parsed from existing BIM tools, which means that model validation is paramount to insure data integrity. FPO is a semantic web ontology developed to describe the capacity- and size-related properties of HVAC components (Kücükavci et al., 2022).

Several efforts have sought to create validation tools for the IFC schema based on EXPRESS (Ghannad et al., 2019; Lee et al., 2016, 2021; Bolpagni et al., 2015; Lee et al., 2015). However, IFC is a large super-schema which is interpreted differently by BIM software vendors like Autodesk and Graphisoft, meaning that a building made in Archicad will not be parsed the same as a building made in Revit. Efforts proposes the use of Shapes Contraint Language (SHACL) shapes when using semantic web ontologies (Stolk and McGlinn, 2020; Soman et al., 2020; Soman, 2019; Hamdan and Scherer, 2020). SHACL shapes allow for validation of BIM models represented in a linked data format.

In this article we explore the HVAC system validation process of proprietary BIM models using FSO, FPO and SHACL to ensure that the required information for performing flow simulations are represented.

Methods

A Revit model of a typical office building was created and an airflow calculation was carried out for the HVAC system. The Revit model was transformed from a proprietary file format (.rvt) into a web-based Resource Description Framework (RDF)-triplestore containing a BIM model. Finally, SHACL shapes were used to validate that the model contained the necessary information to find the most critical pressure point in the open- or closed-circuit HVAC system.

The following sections describe the methods used to;

1. Parse data from a BIM to an RDF-triplestore
2. Validate and enrich the model using SHACL shapes and SPARQL
3. Use typical HVAC system design queries as use case examples

The process starts with a Revit model that contains a part of an HVAC system and the attached spaces. For this article, a parser was created, using C# and the Revit Application Programming Interface (API). First, the script maps the components and spaces, then it builds a .ttl (turtle format) string based on the FSO and FPO ontologies. Once

the parser has looped through all components and spaces, the .ttl string is parsed as a .txt file and sent through a client to the RDF-triplestore. Then, the triplestore stores the data, and then SHACL shapes are used to validate the data integrity (is all parameters filled, etc.). If the data validation fails, it will send an "actions needed" message to the original BIM model and tell it which components need to be fixed to continue. If it succeeds, it will pass to the final stage, which is the pressure and flow calculation of the critical branch. Finally, it displays the results in a table format for the user. The following sections will go through the process shown in Figure 1.

Parsing HVAC data from a BIM model into a Graph model

In the use case a HVAC model is linked with an architectural Revit model. The architectural model describe an office building with four rooms each heated by a radiator and ventilated by one return air terminal and one supply air terminal. The HVAC model involves three systems: heating, cooling, and ventilation seen in Figure 2. The heating system consists of a pump that feeds eight radiators and a heat exchanger. A pump supplies a heat exchanger via the cooling system. The ventilation system consists of a supply fan and exhaust fan connected by air ducts to four air terminals. A detailed description of the systems are illustrated in Figure 2.

The Revit to RDF parser converts the Revit data to RDF to be read into an RDF-based data model (triplestore). The RDF parser plugin written in C# accomplishes this. Revit's API is used to extract data from both the HVAC model and the architect model from the database, as shown in Figure 1. This data is then converted to RDF format expressed in turtle syntax and appended to a StringBuilder in C#.

Listing 1: Code-snippet from the parser showing how a pipe in Revit is converted to RDF using FPO.

```
 //Get all pipes
FilteredElementCollector pipeCollector = new
↪  FilteredElementCollector(doc);
ICollection<Element> pipes =
↪  pipeCollector.OfClass(typeof(Pipe)).ToElements();
List<Pipe> pipeList = new List<Pipe>();
foreach (Pipe component in pipeCollector)
{
    Pipe w = component as Pipe;
    //Type
    string componentID =
    ↪  component.UniqueId.ToString();
    sb.Append($"inst:{componentID} a fpo:Pipe ." +
    ↪  "\n");
}
```

Listing 1 shows a small example of the code from the parser that can be applied to convert Revit data into RDF. The filteredElementCollector class and some other classes from Revit's API are used to retrieve pipes in the model. We extract each pipe's guid number and add it to our StringBuilder object sb.

Validation of model and data integrity

The graph model is validated by parsing the RDF data into a Fuseki database. The server is a SPARQL server that stores knowledge graphs in RDF form. In this case SPARQL is used to enable Create, Read, Update, Delete (CRUD) operations via an endpoint. Queries are used to both enrich or simply request data such as head and flow rates for a specific moving device. Data validation ensure that the necessary data exists and that it has the correct data type, content, and relation to other data in the graph. W3C recommends using SHACL shapes for validating RDF-based data used to describe and constrain RDF graphs. Each shape contains a description of the target it validates. Six SHACL shapes are developed to assure necessary data is present for a query to be performed subsequently to determine head and flow rate:

1. Each supply component must supply fluid to another component of the same system
2. Each return component must return fluid to another component of the same system
3. A component can only supply fluid to one component, except for a tee fitting and heat exchanger
4. A heat exchanger and tee must supply fluid to two components
5. Each supply component must have a parameter of pressure drop. The parameter must have a value and unit and the value must be above 0.0 and have a datatype of double.
6. Each supply component must have a parameter of length. The parameter must have a value and unit and the value must be above 0.0 and have a datatype of double.

In order to validate the data within the triplestore, the database must receive a .ttl file with all six shapes. The SHACL validation engine on the Fuseki server validates our shapes against the RDF graph and returns a validation report. It will then be possible to determine whether our data complies with the rules (shapes) in the validation report. The BIM model is updated if the data set does not comply with the rules. If the dataset matches the graph, we can continue to the next step and query the database. We used Postman, an HTTP client, to send a .ttl file to the Fuseki server and receive a validation report.

Querying head and flow rate

A query is a request for data from a database. Data from the triplestore can be queried using SPARQL. This article shows four queries. To find the head and flow rate of a specific flow moving device, we need to determine whether the flow moving device is part of an open-circuit or closed-circuit and what type of flow medium it is transporting as it impacts how we are going to query the head. In a closed-circuit system, the total head is equal to the dynamic head. When the system is an open-circuit and transports air, the total head is equal to the dynamic head. Lastly, if the system is an open-circuit and transports water, the total head is equal to the dynamic head plus the

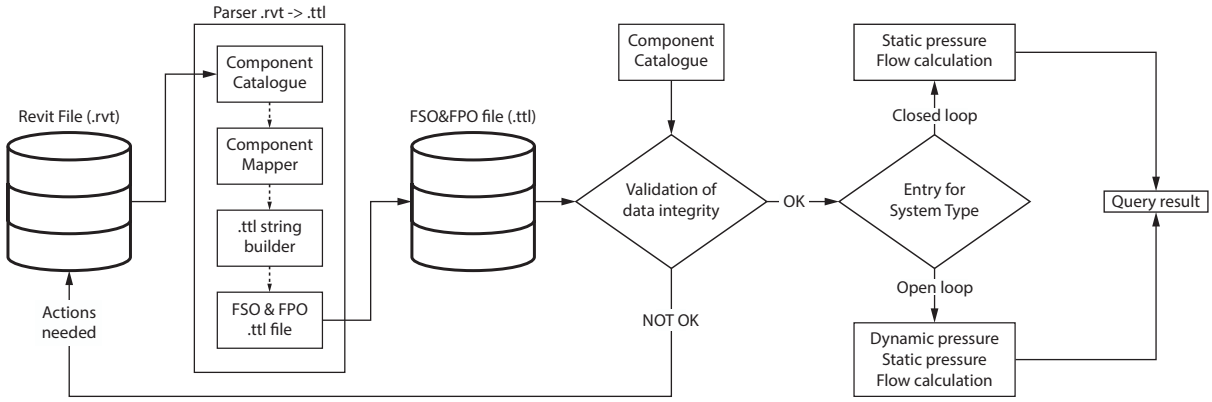

Figure 1: Process diagram for the Revit to FSO and FPO parser. The process consists of a BIM model, a parser, a Fuseki server to store, query and validate RDF models, a set of SHACL shapes and a Simple Protocol and RDF Query Language (SPARQL) select query to find the resulting head and flow rate of a given pump.

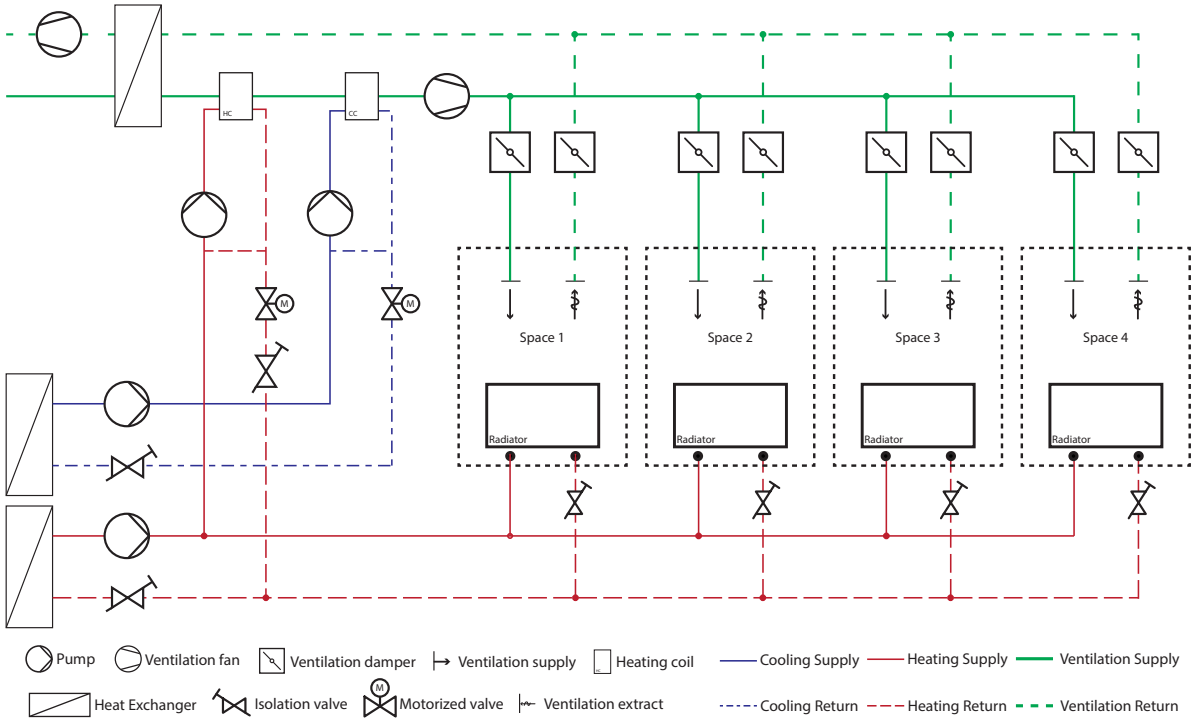

Figure 2: System diagram showing the heating system, cooling system and ventilation system supplying the 4 rooms (spaces) and their components. This schematic was modeled in Revit for the use case of this article.

static head. The first query identifies the type of circuit. A query will be performed according to the circuit type and flow medium type to obtain the head and flow rate of the given flow moving device.

Results

A total of six SHACL shapes and four SPARQL queries have been developed; however, due to article space limitations, only two SHACL shapes and two queries have been described in detail. The descriptions and source code for all SHACL shapes, SPARQL queries and converted triples, has been made available online [1]

To perform a hydraulic calculation to determine the capacity of a flow moving device in an HVAC system two parameters are needed; the total head and flow rate. Based on these, an HVAC designer can select a product. For closed-circuits' water-based systems, the head is the sum of the pressure drop generated by the critical branch. A SPARQL insert query can be used to perform all necessary calculations automatically. This is also possible when the original BIM model lacks critical data containing circuit type information. By using the insert query shown in Listing 2, each system in the triplestore is enriched with circuit type `ex:ClosedCircuit` or `ex:OpenCircuit` based on the medium and consumer components it uses, as well as the supply- and return temperatures.

Listing 2: Sparql update query to determine whether a system is a open-circuit or closed-circuit, and to add that information to that system, expressed in Turtle syntax.

```
INSERT {?system a ?circuit . ?system a ?systemType .}
WHERE {
  ?system fso:hasComponent ?component .
  ?component fso:feedsFluidTo+ ?componentA .
  ?componentA a ?componentAType .
  ?system fso:hasFlow ?flow.
  ?flow fpo:temperature ?temperature .
  ?temperature fpo:value ?temperatureValue .
  BIND (IF((?temperatureValue >= 25 && ?temperatureValue <= 70 && (?componentAType = fpo:SpaceHeater || ?componentAType = fpo:HeatExchanger)), ex:HeatingSystem, IF ( (?temperatureValue >= 5 && ?temperatureValue <= 15 && (?componentAType = fpo:ChilledBeam || ?componentAType = fpo:HeatExchanger)), ex:CoolingSystem, IF ((?temperatureValue >= 16 && ?temperatureValue <= 24 && (?componentAType = fpo:AirTerminal || ?componentAType = fpo:HeatExchanger)), ex:VentilationSystem, "" ))) AS ?systemType )
  FILTER (isIRI(?systemType))
  BIND (IF((?temperatureValue >= 25 && ?temperatureValue <= 70 && (?componentAType = fpo:SpaceHeater || ?componentAType = fpo:HeatExchanger)), ex:ClosedCircuit, IF((?temperatureValue >= 5 && ?temperatureValue <= 15 && (?componentAType = fpo:ChilledBeam || ?componentAType = fpo:HeatExchanger)), ex:ClosedCircuit, IF ((?temperatureValue >= 16 && ?temperatureValue <= 24 && (?componentAType = fpo:AirTerminal || ?componentAType = fpo:HeatExchanger)), ex:OpenCircuit, "" ))) AS ?circuit )
}
```

Every HVAC component in the BIM model must be associated with a parameter `fpo:pressureDrop`. The parameter `fpo:pressureDrop` must also have a value associated with it, and the unit must be consistent across all components. Otherwise, the sum will be incorrect. The SHACL shape shown in Listing 3, validates exactly `fpo:pressureDrop` for all HVAC components in our BIM model. Listing 3 shows how we select our target using a SPARQL select query. The listing includes HVAC components on both a closed circuit's supply and return sides. Since the pump itself for a closed-circuit does not have a pressure drop, we omit this by writing `FILTER NOT EXISTS this is a fpo:Pump`. The rules are assigned to the target with the sh: property. For example, the maximum and minimum of one `fpo:pressureDrop` property is required for the target.

Listing 3: Shacl shape of each component must have a parameter pressure drop, value and unit. Expressed in Turtle syntax.

```
ex:Shape-1 a sh:NodeShape ;
sh:nodeKind sh:IRI ;
sh:target [
    a sh:SPARQLTarget ;
    sh:prefixes (fpo: fso: ex:) ;
    sh:select """PREFIX fso: <https://w3id.org/fso#> PREFIX fpo: <https://w3id.org/fpo#> prefix ex:<http://example.org/> SELECT ?this WHERE {?system a ex:ClosedCircuit .?system fso:hasComponent ?this .filter not exists {values ?type {fpo:Pump fpo:Fan} ?this a type} .} """ ;
    ] ;
sh:property [
    sh:path fpo:pressureDrop ;
    sh:minCount 1;
    sh:maxCount 1;
    ];
sh:property [
    sh:path (fpo:pressureDrop fpo:value) ;
    sh:minCount 1;
    sh:maxCount 1;
    sh:minInclusive 0.001;
    sh:dataType xsd:double ;
    ];
sh:property [
    sh:path (fpo:pressureDrop fpo:unit) ;
    sh:minCount 1;
    sh:maxCount 1;
    sh:dataType xsd:string ;
    sh:hasValue "Pascal"^^xsd:string ;
    ].
```

[1] https://github.com/alikucukavci/IBPSA-SPARQL-QUERIES-AND-SHACL-SHAPES

In the first iteration of Listing 3, eight components violated the rules as they were missing the parameters `fpo:pressureDrop`, `fpo:value` and `fpo:unit`. We fixed these components in the HVAC BIM model, and the process from BIM to validation was repeated. After the second iteration, the validation report was conformant since all components met the conditions in Listing 3.

To verify the existence of a parameter, the composition of HVAC systems and components can also be validated. The tee and heat exchanger, for example, feed fluid to at least two other components. In Listing 2, the composition of the tees and heat exchangers for the RDF model is validated, and the validation report was conformant already in the first iteration.

Listing 4: Shacl shape of a heatexchanger- and tee component must supply fluid to two components expressed in Turtle syntax.

```
ex:Shape-2 a sh:NodeShape ;
sh:nodeKind sh:IRI ;
sh:target [
        a sh:SPARQLTarget ;
        sh:prefixes (fpo: fso:) ;
        sh:select """
        PREFIX fso: <https://w3id.org/fso#>
        PREFIX fpo: <https://w3id.org/fpo#>
        SELECT ?this WHERE {
        ?system fso:hasComponent ?this .
        FILTER EXISTS {
        VALUE ?type {
        fpo:Tee fpo:HeatExchanger
        } ?this a ?type .}} """ ;
  ] ;
   sh:property [
        sh:path fso:feedsFluidTo ;
        sh:minCount 2;
    ].
```

Using a SPARQL query, listing 4 calculates the head and flow rate of a given pump. This pump is part of a close circuit, so we ignore the static head and only calculate the dynamic head. We must first calculate the total pressure loss of the critical path before we can calculate the dynamic head. The parameter pressure drop is summarized for each path from the given pump to a terminal to determine the critical path. The next step is to filter the path with the largest pressure loss. Due to this being a closed system, the terminals are either `fpo:SpaceHeater` or `fpo:HeatExchanger`. The total pressure loss for the critical path is converted from Pascal to meters. Finally, the flow rate is summarized for the terminals that the given flow moving device supplies in the same system. For the given flow moving device `inst:98172f87-b31e-4363-a01f-2f3f2d13a48f-00131613`, the SPARQL query returns the id number of the critical consumer component, a head of 1.71 meters of head and a flow rate of 0.034 L/s.

Listing 5: Sparql SELECT query to retrieve the dynamic head and flow rate of a given pump for a heating system, expressed in Turtle syntax.

```
PREFIX fpo: <https://w3id.org/fpo#>
PREFIX fso: <https://w3id.org/fso#>
PREFIX inst: <https://example.com/inst#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?consumer
↪  ((?totalPressure/0.00010199773339984) AS
↪  ?dynamicHead) ?totalFlow
WHERE{
  {SELECT (SUM(?flowRateValue) AS ?totalFlow)
    WHERE{
      ?system fso:hasComponent inst:98172f87-b31e-
      ↪  4363-a01f-2f3f2d13a48f-00131613
      ↪  .
      VALUES ?type {fpo:SpaceHeater
      ↪  fpo:HeatExchanger} ?consumer a ?type .
      ?consumer fpo:flowRate ?flowRate .
      ?flowRate fpo:value ?flowRateValue .
    }
  }
  {
    SELECT ?consumer (?totalComponentPressureDrop +
    ↪  ?consumerPresserDropValue AS ?totalPressure)
      WHERE {
      {
        SELECT ?consumer (SUM(?returnPressureValue)
        ↪  + (SUM(?totalSupplyPressureDrop)/COUNT(
        ↪  ?totalSupplyPressureDrop)) AS
        ↪  ?totalComponentPressureDrop)
        WHERE {
          {
            SELECT ?consumer
            ↪  (SUM(?supplyPressureValue) AS
            ↪  ?totalSupplyPressureDrop)
            {
              ?supplySystem fso:hasComponent
              ↪  inst:98172f87-b31e-4363-a01f-
              ↪  2f3f2d13a48f-00131613
              ↪  .
              ?supplySystem a fso:SupplySystem .
              ?supplySystem fso:hasComponent
              ↪  ?supplySystemComponent .
              ?supplySystemComponent
              ↪  fso:feedsFluidTo+ ?consumer .
              values ?type {fpo:SpaceHeater
              ↪  fpo:HeatExchanger} ?consumer a
              ↪  ?type .
              ?supplySystemComponent
              ↪  fpo:pressureDrop
              ↪  ?supplyPressureDrop .
              ?supplyPressureDrop fpo:value
              ↪  ?supplyPressureValue .
            }
            GROUP BY ?consumer
          }
          {
            ?returnSystem fso:hasComponent ?consumer
            ↪  .
            ?returnSystem a fso:ReturnSystem .
            ?returnSystem fso:hasComponent
            ↪  ?returnSystemComponent .
            ?consumer fso:feedsFluidTo+
            ↪  ?returnSystemComponent .
            ?returnSystemComponent fpo:pressureDrop
            ↪  ?returnPressureDrop .
            ?returnPressureDrop fpo:value
            ↪  ?returnPressureValue .
          }
```

```
        } GROUP BY ?consumer
      }
      ?consumer fpo:pressureDrop
      ↪  ?consumerPressureDrop .
      ?consumerPressureDrop fpo:value
      ↪  ?consumerPresserDropValue .
    }
    ORDER BY DESC((?totalPressure)) LIMIT 1
  }
}
```

Discussion

This article has provided a novel approach to generate FSO triples based on a Revit BIM model. Furthermore, it allows for a systematic and standardized way to perform model validation of an HVAC system with the use of SHACL shapes. Such model validation enables better data interoperability in the future, and therefore eases the designer's burden. The SHACL shapes can be used in future work to ensure that models contain the right BIM information to allow for simulation of the hydraulic systems performed in Modelica. The SHACL shapes created in this article, though advanced, needs to be validated further by industry and academia. Listing 5 is limited to query the head and flow rate of one flow moving device at a time and only for closed-circuits. A more generic query that can calculate head and flow rate for every flow moving device on a construction project, regardless of system and circuit type, will be helpful for the HVAC designer. We created Listing 5 as a starting point, but for future work, it should be modified to be more generic. Furthermore, we created the SHACL shapes specifically for the validation of HVAC systems, but their use is not limited to that. There is a potential to use SHACL shapes for validation of many different sub-disciplines within the AECO industry. The article provides a proof-of-concept for a validation method readily available within the world of semantic web ontologies. To further verify the validity of SHACL shapes as a model validation method, further research should be carried out on the topic. Furthermore, this paper suggested a way to perform dimensioning of pumps, using an RDF-triplestore. In the roadmap for future works, the researchers imagine the work of this paper to pave the way for pump manufacturers to create an RDF-triplestore with all of their product data available. Doing this will close the gap from the manufacturer to the HVAC designers by allowing the HVAC designer to automatically query for product data from the manufacturer, based on the static and dynamic pressure calculations carried out in the RDF-triplestore.

Conclusion

It is possible to transform a typical BIM model with limited HVAC data into an RDF-triplestore using FSO and FPO. Beside being able to make useful HVAC queries on original data we demonstrate that HVAC data can be enriched via SPARQL insert queries. We demonstrate that the HVAC data can be validated for consistency and coherence through the use of generic SHACL shapes. Finally, we conclude that SPARQL select queries can be used to compute critical pressure and flow rate for a given flow moving device thus exploiting the inherent advantages of an open source graph database using FSO, FPO, SHACL, and SPARQL as key enablers.

Acknowledgements

Funding: This work was funded by; the Ramboll Foundation; the Innovation Fund Denmark; EU-Interreg ÖKS "Data-driven Energy Management in Public Buildings".

References

Balaji, B., 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, and K. Whitehouse (2016). 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*, pp. 41–50.

Bolpagni, M., A. Luigi, C. Ciribini, and S. M. Ventura (2015). Informative content validation is the key to success in a BIM-based project validation is the key.

Ghannad, P., Y.-c. Lee, J. Dimyadi, and W. Solihin (2019). Automated BIM data validation integrating open-standard schema with visual programming language. *Advanced Engineering Informatics 40*(January), 14–28.

Hamdan, A. H. and R. J. Scherer (2020). Integration of BIM-related bridge information in an ontological knowledgebase. *CEUR Workshop Proceedings 2636*, 77–90.

Kücükavci, A., M. Seidenschnur, and H. C. A. Pauwels, Pieter (2022). Proposing a Semantic Web Ontology to Support Capacity- and Size-Related Property Descriptions of Heating, Ventilation and Air Conditioning Components in The Design Phase of Buildings.

Kukkonen, V., A. Kücükavci, M. Seidenschnur, M. H. Rasmussen, K. M. Smith, and C. A. Hviid (2022). An ontology to support flow system descriptions from design to operation of buildings. *Automation in Construction 134*(November 2020), 104067.

Lee, Y.-c., C. M. Eastman, and J.-k. Lee (2015). Validations for ensuring the interoperability of data exchange of a building information model. *Automation in Construction 58*, 176–195.

Lee, Y.-c., C. M. Eastman, and W. Solihin (2021). Rules and validation processes for interoperable BIM data exchange. *Journal of Computational Design and Engineering 8*(August 2020), 97–114.

Lee, Y.-c., C. M. Eastman, W. Solihin, and R. See (2016). Modularized rule-based validation of a BIM model pertaining to model views. *Automation in Construction 63*, 1–11.

Pauwels, P. and W. Terkaj (2015). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction 63*, 100–133.

Porsani, G. B., K. D. V. de Lersundi, A. S. O. Gutiérrez, and C. F. Bandera (2021). Interoperability between building information modelling (Bim) and building energy model (bem). *Applied Sciences (Switzerland) 11*(5), 1–20.

Rasmussen, M. H., M. Lefrançois, G. F. Schneider, and P. Pauwels (2021). BOT: The building topology ontology of the W3C linked building data group. *Semantic Web 12*(1), 143–161.

Redmond, A., A. Hore, M. Alshawi, and R. West (2012). Exploring how information exchanges can be enhanced through Cloud BIM. *Automation in Construction 24*, 175–183.

Soman, R. K. (2019). Modelling construction scheduling constraints using shapes constraint language (SHACL). *Proceedings of the 2019 European Conference on Computing in Construction 1*(2006), 351–358.

Soman, R. K., M. Molina-Solana, and J. K. Whyte (2020). Linked-Data based Constraint-Checking (LDCC) to support look-ahead planning in construction. *Automation in Construction 120*(August), 103369.

Stolk, S. and K. McGlinn (2020). Validation of IfcOWL datasets using SHACL. *CEUR Workshop Proceedings 2636*, 91–104.