Portable Model Predictive Controller System Design for Demand Side Management using Semantic Web Technologies

Lasitha Chamari (corresponding author), (l.c.rathnayaka.mudiyanselage@tue.nl)
Eindhoven University of Technology, The Netherlands

Shalika Walker, (shalika.walker@kropman.nl)
Kropman B.V., The Netherlands

Lu Wan, (lu.wan@de.bosch.com)
Robert Bosch GmbH, Germany

Ekaterina Petrova , (e.petrova@tue.nl)
Eindhoven University of Technology, The Netherlands

Pieter Pauwels, (p.pauwels@tue.nl)
Eindhoven University of Technology, The Netherlands

Keywords: Brick Ontology, MPC, RDF, Semantic Web, SHACL, SOSA/SSN

Abstract

A smart building relies on heterogeneous information systems, with data originating from different sources and represented in various formats. Semantic Web technologies allow connecting these disparate data sources by standardizing metadata with ontologies and enriching information with logical relationships. The latter enables not only shared understanding and interpretability by machines but also querying and reasoning capabilities. Brick and SOSA/SSN are such widely used ontologies in the smart building domain. These ontologies are currently used to semantically describe physical or virtual assets in a building. Building on these ontologies, we propose an ontology to describe a Model Predictive Controller (MPC) system. This MPC is designed as a Demand Side Management strategy for an office micro-grid system. The proposed ontology complements the use of Semantic Web technologies in the smart building. It allows the designed MPC application to seamlessly modify to a different building for both simulation and experimental purposes by communicating with necessary data sources with using the knowledge embedded in the semantic graphs.

1. Introduction

The complexity and diversity of buildings, including their smart control systems such as Model Predictive Controllers (MPC) and Fault Detection and Diagnostic systems (FDD), challenge the transferability of these systems to other buildings (Balaji et al., 2018; Dibowski et al., 2016). While implementing a simpler system such as a rule-based-controller to another similar building setup may seem straightforward, sophisticated data-driven systems such as MPC are difficult to re-configure to a new building. The reason for this complexity is the intensive prior knowledge that is required about a building and its

systems to i) model the given building and the systems and ii) identify and interpret the data correctly. When the knowledge about the building and its components are not represented formally, it is difficult to modify a control system to a new environment. The development of ontologies in the smart building domain has enabled the standard representation of the knowledge of the physical and virtual assets in buildings. A number of studies (Balaji et al., 2018; Janowicz et al., 2019; Rasmussen et al., 2020; Sagar et al., 2018) have elaborated on the design of such ontologies and the use of Semantic Web technologies in the building (control) domain. Building on these recent developments, it is possible to extend this knowledge representation to improve the portability of control systems (also called application) to new environments. However, when it comes to data-intensive control applications such as a MPC, the literature lacks evidence on how to utilize knowledge representation to make them portable among buildings. Therefore, taking a MPC as a use case, we investigate how to effectively utilize knowledge representation and Semantic Web technologies in sophisticated control applications to improve their semantic interoperability between different buildings, making them highly portable. We aim to do that by reusing existing domain ontologies, creating new ones when necessary, and introducing a software service to execute the tasks required to modify the MPC system to the new building.

The remainder of the paper is structured as follows. Section 2 discusses the relevant literature and the gap that exists when it comes to the data-driven control domain and the transferability of MPC between buildings in particular. Section 3 describes the method of developing new ontologies and their implementation by means of a service-oriented architecture. Section 4 introduces the software architecture designed to manage the developed knowledge-base. In section 5, we demonstrate the proposed knowledge-base and its applicability using the MPC example. Section 6 concludes the paper.

2. Related Works

Over the past decade, numerous ontologies have emerged within the building domain, covering a wide range of areas, such as; Industry Foundation Classes (IFC) [1] vocabulary for modeling the design and engineering aspects of buildings, Building Topology Ontology (BOT) (Rasmussen et al., 2020) for modeling building topology, Real Estate Core (REC) (Hammar et al., 2019) for asset management, etc. More recent developments include the ontologies in the building automation and control systems domain such as the Brick ontology (Balaji et al., 2018) and Haystack tagging system[2]. The Semantic Sensor Network ontologies (SOSA/SSN) (Janowicz et al., 2019) have formed the core of modelling the IoT and sensor network aspects of smart buildings. Building upon SSN, the Semantic Smart Sensor Network (S3N) (Sagar et al., 2018) expands the capabilities of describing smart sensors and their computational profile. With the introduction of the W3C Thing Description framework[3] for IoT, the Thing Description ontology (TD)[4] is introduced for modeling properties, actions and events of physical or virtual devices (Things). These ontologies adhere to the Resource Description Framework (RDF) data model [5]. RDF provides a way to describe resources and their relationships in a machine-readable format, consisting of subject-predicate-object triples, forming a graph-based knowledge representation. For example, RDF enables to represent the information that "There is a CO_2 *Sensor* located in a *Space* identified by *IFC GUID*

[1] https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/

[2] https://project-haystack.org/

[3] https://www.w3.org/TR/wot-thing-description11/

[4] https://www.w3.org/2019/wot/td

[5] https://www.w3.org/TR/rdf-primer/

"0KLkXPBfvES9D1y7EjijkE" on the 9th *Floor* of the Atlas *Building* on TU Eindhoven campus *Site*"(Chamari et al., 2022).
The use of the above-mentioned ontologies has been extensively demonstrated in various applications such as knowledge-based fault detection in heating, ventilation and air-conditioning (HVAC) systems (Delgoshaei et al., 2017), portable data-driven applications (Balaji et al., 2018), Building Information Model (BIM) and sensor data integration (Chamari et al., 2022), management of on-device IoT applications (Ren et al., 2022), etc. Besides devices and systems, recent literature also shows attempts to model software artifacts using ontologies. Dibowski et al., 2016 use an ontology to specify requirements and configurations of Fault Detection and Diagnosis (FDD) algorithms in a machine-interpretable manner. This ontology-based specification, along with an FDD ontology and the BIM model is used to identify a list of FDD algorithms that can run in a given building. Similarly, Ren et al., 2022 propose a semantic schema for describing the metadata of a Neural Network (NN) and an event processing rule as an example for discovering a list of edge algorithms that can run on an IoT device in a decentralized IoT network. The metadata description includes hardware requirements such as memory, and NN weights. These examples highlight the growing trend of leveraging semantic descriptions to enhance the interoperability, reusability, and machine interpretability of software artifacts in various domains.

3. Method

In our previous study on reference architecture for data-driven buildings (Chamari et al., 2023), we introduced a service-oriented architecture for a smart building. The idea of the architecture is to implement data-driven applications as reusable services. In this study, the MPC system follows this service-oriented pattern. The MPC system is designed for Demand Side Management (DSM) in an office building. The office building's micro-grid consists of four Electric Vehicle (EV) charging poles, each 11 kW (3 phase/16 A), a solar photovoltaic array (16.9 kW), the building load (on average 12 kW) including plug loads, an Air Handling Unit (AHU), a chiller, and the connection to the grid. The objective of the MPC system is to flatten the load incurred on the building by EV charging. Here, the MPC system is a collection of the following services; 1) the MPC algorithm, 2) a building load forecasting algorithm, 2) an EV user schedule handler, 3) a PV forecast handler, and 4) an electricity price forecast handler. All these components are designed as smaller reusable **services**. In addition to the MPC system's objective to flatten the building electrical load, the aim is to develop the MPC system as an portable system using formal knowledge representation.
Fig. 1 describes the workflow of the proposed method. First, the building and its systems such as its equipment, data points, time series references, and any other external references need to be described in a knowledge graph (hereafter called **building graph** in Fig. 1), using the ontologies in TBox. Then, the metadata of the data-driven application (hereafter called an **artifact**) that the above services embed need to be described formally. To achieve the above, first, ontologies are designed for artifacts by extending well-established ontologies as shown in TBox. The metadata of these artifacts are defined using the above ontologies, leading to an **artifact graph** describing the concepts such as inputs and outputs, and their references to a building's systems by linking the concepts with the building graph. This artifact graph is used to generate the shapes (hereafter called **shape graphs**), representing the data requirements of an artifact formally. These shape constraints are based on the Shape Constraint Language (SHACL)[6],

[6] https://www.w3.org/TR/shacl/

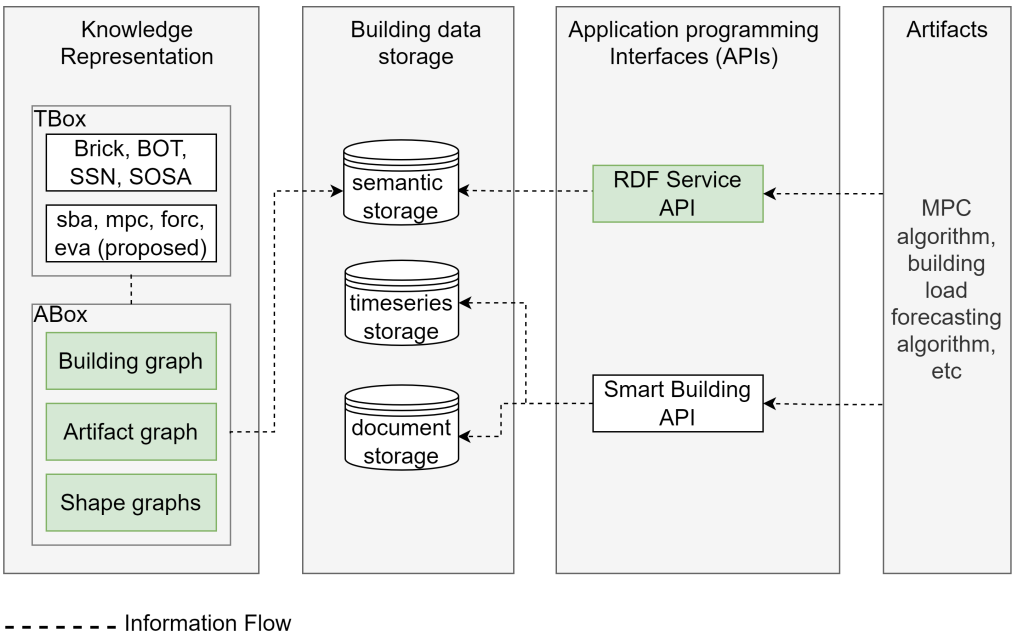

Figure 1: System architecture diagram - RDF Service for data-driven applications

the W3C standard designed to validate RDF data against a set of conditions. They are originally used to ensure that the RDF data adheres to specific structural and integrity rules. However, in this case, we can rely on shapes to check the compatibility of our artifacts with a different building by validating the building graph against the shape graphs of our artifacts.

The portability of MPC system is achieved by comparing the building graph against the shape graphs generated according to the requirements of a given artifact. If they are compatible, then SPARQL Protocol and RDF Query Language (SPARQL) queries specifying constraints and patterns of triples are used to traverse the building graph and locate the necessary data points that match, leading to a successful modification of the MPC system to a different building.

To avoid individual artifacts such as an MPC or FDD system having to implement the above semantics-based logic, all the semantic tasks are dedicated to a new service called the **RDF Service**. This service is responsible for the generation, management and querying the semantic graphs on behalf of artifacts. Therefore, this service is also exposed using a RESTful web interface. The proposed RDF services is implemented as a component in the service-oriented architecture Chamari et al., 2023, enabling other applications such as the MPC system to interact with it for semantic tasks. As shown in Fig. 1, together with the Smart Building API, the RDF service facilitates the artifacts for their data and metadata needs.

4. RDF service for portable data-driven applications

This section first explains the design of ontologies for the MPC system using the proposed ontologies in TBox in Fig. 1. Then, the RDF service to handle the semantic graphs for applications is described in detail.

4.1. Semantic descriptions

Semantics of Building, Systems and Sensors

Software artifact interact with the existing building and its systems. A formal definition of the given building, its systems and sensors (building graph) is therefore a prerequisite

to use an artifact described semantically. Here we use the well established BOT ontology to describe the building topology and the Brick ontology to describe the systems and sensors.

Semantics of software artifacts

As previously mentioned, the MPC system is a collection of 1) the MPC algorithm, 2) a building load forecasting service, 3) an EV user schedule handler, 4) a PV forecast handler, and 5) an electricity price forecast handler. In this section, we introduce ontologies to describe the first three and an additional ontology to describe the common concepts; namely, mpc ontology, forc ontology, eva ontology and sba ontology. Common concepts that apply to all artifacts, such as inputs, outputs, KPIs, etc., are described in a main ontology (sba) with the aim of making it extensible for future artifacts. When doing that, the idea is not to start from scratch, but to build on top of a the well known SOSA ontology which already has the concept of a *sosa:Procedure*, defined as *"A workflow, protocol, plan, algorithm, or computational method specifying how to make an Observation, create a Sample, or make a change to the state of the world (via an Actuator)"*. We reuse several other existing ontologies to construct the proposed ontologies for components mentioned. The namespaces and prefixes of reused ontologies and proposed ontologies are shown below. Four new ontologies are proposed as *sba* (smart building algorithm), *eva* (electric-vehicle algorithm), *forc* (forecasting) and *mpc* (MPC algorithm). Since the PV forecast and electricity price forecast refer only to a time series dataset, and do not require an algorithm, they do not need a new ontology. The proposed ontologies and the instantiated semantic graph are illustrated graphically in Fig. 2. For now, these ontologies have not been published yet as they are a part of ongoing testing and validation. The current article contains the results of preliminary local testing.

Listing 1: Existing and new ontologies proposed to describe MPC System

```
# Re-used
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix unit: <http://qudt.org/vocab/unit/> .
@prefix brick: <https://brickschema.org/schema/Brick#> .
@prefix ref: <https://brickschema.org/schema/Brick/ref#> .
@prefix bot: <https://w3id.org/bot#> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .
# Proposed
@prefix sba: <https://example.org/sba-schema#> .
@prefix eva: <https://example.org/sba-ev-schema#> .
@prefix mpc: <https://example.org/sba-mpc-schema#> .
@prefix forc: <https://example.org/sba-forecast-schema#> .
```

The following section discusses the design of the sba ontology, forc ontology and mpc ontology.

Smart Building Ontology (sba): This is the main ontology that describes the common concepts of a data-driven application. All algorithms are described as a subclass of *sba:SmartBuildingAlgorithm*, which is a subclass of *sosa:Procedure*. The algorithm input and algorithm output are sub-classes of *ssn:Input* and *ssn:Output*, respectively. Some of these inputs are directly linked to Brick entities in the building graph (such as building load and weather data), whereas the PV forecast, price forecast, weather forecast, forecasted building load, EV user schedule, ML model and input parameters need to be semantically standardized using the sba ontology. For that purpose, we introduce the classes *sba:Forecast*, *sba:MachineLearningModel*, and *sba:ModelParameters*.

Load Forecasting Algorithm Ontology (forc) : The Load Forecasting Algorithm *forc:LoadForecastModel* is a subclass of *sba:SmartBuildingAlgorithm*. The output of this algorithm is an input to the MPC algorithm and is, therefore, recorded in the same way as for any other time series data. This means that these outputs are recorded in a database dedi-

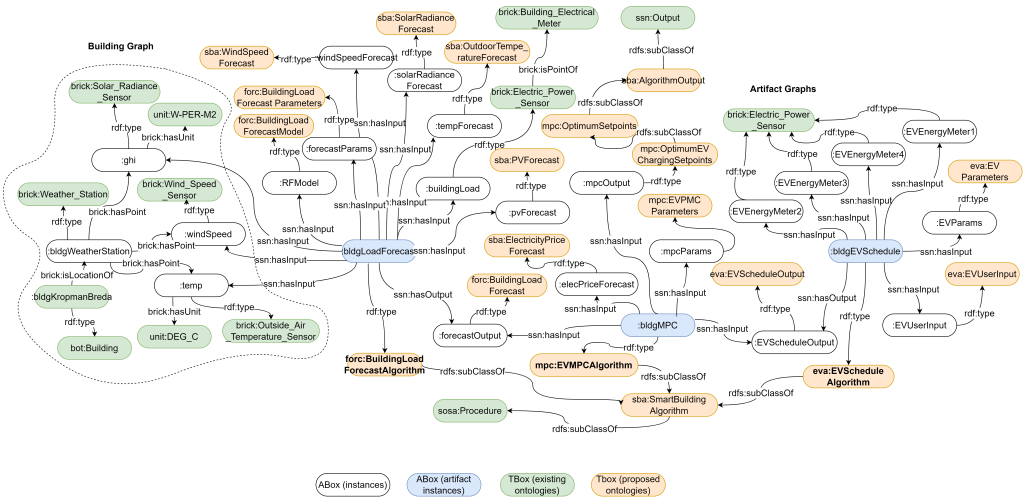

Figure 2: Instantiating the MPC system's graph using existing and proposed ontologies. Full implementation is available via https://github.com/ISBE-TUe/sba-schema

cated to the algorithm with a *ref:TimeseriesReference* according to the Brick ontology. Model Predictive Controller Algorithm Ontology (mpc): The MPC algorithm *mpc:SmartChargingMPC* is a subclass of *sba:SmartBuildingAlgorithm*. The output of MPC is defined as *mpc:OptimumEVChargingSetpoints*, a subclass of *mpc:OptimumSetpoints*, which is a subclass of *sba:AlgorithmOutput*.

4.2. RDF Service

This RDF service is made of four components: 1) semantic graph of building and artifacts 2) SHACL shapes of artifacts, 3) infrastructure to handle graphs, and 4) RDF Service API. The semantics of the building, systems, sensors and artifacts are first described as explained above. Then, based on the semantic graphs of the artifacts, SHACL shapes are generated for each algorithm and each of their inputs for validating the new building's semantic graphs against the required algorithm inputs. This is a one time task and the idea is to generate these shapes along with the semantic graph of an algorithm when it is first designed. Then, the shapes are reused when the algorithm is modified to a new building. Validation with the SHACL shapes gives an idea whether the new building has sufficient data points to execute the given algorithm. The infrastructure, on the other hand, requires a semantic storage, time series storage and a document/object storage to save the semantic graphs and data of the applications. Interactions that are intended with the graphs are made available using a standard API. The API provides the necessary endpoints to interact with the created ontologies and existing building graphs. These endpoints encapsulate the following functionalities:

1. Querying and exploring for building graph, artifact graphs and shape graphs.
2. Read an artifact graph and find its relevant inputs and outputs.
3. Create shape graphs based on an artifact.
4. Check a given building graph against the required inputs of a given artifact using the shape graphs.
5. Generate configuration files to run an artifact tailored to the given building.

   In the final step above, artifacts query the semantic graph of the building, systems and sensors for entities and relationships using SPARQL. For a given artifact, the query

returns the *inputs, their time series reference, and their database reference.* In other words, these constitute the information needed to re-configure the application to the new environment. Once these are known, the initialization is done, and there is no need to execute the configuration query again unless the semantic graphs change. Therefore, once executed, the necessary information is written into a configuration file within the artifact. As such, the algorithm is ready to run in the new building.

5. Implementation

Developing the artifact graph and shape graphs

The artifact graph and its shape graphs are only generated once and reused later for other buildings. In our case, the MPC system depends on the building load forecasting algorithm's output, the electric vehicle schedule algorithm's output, the PV forecast, the electricity price forecast, and the input parameter file. A snippet from the artifact graphs for the MPC (mpc:EVMPC), forecasting algorithm (forc:LoadForecastAlgorithm) and the building graph is shown in Listing 2. This graph describes the artifacts using the proposed ontologies for sba, mpc, forc, eva, with their links to Brick entities in the building graph. Parts of the graph are intentionally removed in the Listing 2 and the reader may refer to https://github.com/ISBE-TUe/sba-schema for the full semantic graph.

Listing 2: Model Predictive Controller artifact graph

```
# MPC artifact graph
inst:bldgSmartChargingMPCAlgorithm a mpc:EVMPC ;
    ssn:hasInput
        inst:bldgMPCParameters ,
        inst:bldgLoadForecastAlgorithmOutput ,
        inst:bldgPVForecast ,
        inst:bldgEVSheduleAlgorithmOutput ,
        inst:bldgElectricityPriceForecast ;
    ssn:hasOutput inst:bldgSmartChargingMPCOutput .

inst:bldgMPCParameters a mpc:EVMPCModelParameters ;
rdfs:label "MPCParameters" ;
    ref:hasExternalReference [
    a ref:TimeseriesReference ;
    ref:hasTimeseriesId "c29fb1d7-57b2-487f-9456-5aa3e6900520" ;
    ref:storedAt  inst:appdata ; ] .

inst:bldgSmartChargingMPCOutput a mpc:OptimumEVChargingSetpoints ;
rdfs:label "SmartChargingMPCOutput" ;
    ref:hasExternalReference [
    a ref:TimeseriesReference ;
    ref:hasTimeseriesId "4a911207-12e1-4c7a-88b7-a217de337c8c" ;
    ref:storedAt  inst:appdata ; ] .

# Building load forecast artifact graph
inst:bldgLoadForecastAlgorithm a forc:LoadForecastAlgorithm ;
    ssn:hasInput
        inst:bldgOutdoorTemperatureSensor ,
        inst:bldgSolarRadianceForecast ,
        inst:bldgOutdoorTemperatureForecast  ;
    ssn:hasOutput inst:bldgLoadForecastAlgorithmOutput ; .

inst:bldgLoadForecastAlgorithmOutput a forc:BuildingLoadForecast ;
rdfs:label "LoadForecastAlgorithmOutput" ;
    ref:hasExternalReference [
    a ref:TimeseriesReference ;
    ref:hasTimeseriesId "889cefd5-0c95-4bf4-8462-8b5b9abc2a40" ;
    ref:storedAt  inst:appdata ; ] ; .

# Building graph
inst:bldgWeatherStation a brick:Weather_Station ;
    brick:hasPoint
        inst:bldgOutdoorTemperatureSensor ,
```

```
        inst:bldgSolarRadianceSensor ,
        inst:bldgWindSpeedSensor .

inst:bldgOutdoorTemperatureSensor  a brick:Outside_Air_Temperature_Sensor ;
rdfs:label "OutdoorTemperature" ;
  ref:hasExternalReference [
    a ref:TimeseriesReference ;
    ref:hasTimeseriesId "e264add0-9637-4a4a-ba07-42fdf7d86fec" ;
    ref:storedAt  inst:insiteview ; ] .
```

Once the artifacts are described, their shapes can be generated in order to use them in the implementation process for a different building. For this step, the classes that have a relationship based on *ssn:hasInput* can be programatically extracted from artifact graph using the SPARQL query shown in Listing 3.

Listing 3: Querying for an artifact's Inputs. Here {algorithm} is a placeholder that holds the type of algorithm (e.g., forc:LoadForecastAlgorithm, mpc:EVMPC)

```
SELECT ?input ?datapoint ?label ?tsid ?database WHERE {
    ?algorithm a {algorithm} .
    ?algorithm ssn:hasInput ?input .
    OPTIONAL {?input  ref:hasExternalReference ?arr .
              ?arr ref:hasTimeseriesId ?tsid . ?arr ref:storedAt ?db .
              ?db inst:connstring ?database . ?input a ?datapoint .
    OPTIONAL {?input rdfs:label ?label . } } }
```

Modifying the MPC for a different building

Due to the page limit, not all artifacts can be fully explained. Therefore, taking the building load forecast algorithm (forc:LoadForecastAlgorithm) as an example, this section describes the rest of the workflow. For the *forc:LoadForecastAlgorithm* algorithm, inputs extracted are of type *brick:Electric_Power_Sensor, sba:OutdoorTemperatureForecast, sba:SolarRadianceForecast, sba:WindSpeedForecast, sba:BuildingLoadForecastModel, sba:-BuildingLoadForecastModelParameters, brick:Outside_Air_Temperature_Sensor, brick:Wind-_Speed_Sensor, and brick:Solar_Radiance_Sensor.* Then, this list of classes is used to generate the shape graphs corresponding to each of them. In this version of the implementation, we only consider having a *ref:hasTimeseriesId* property as the required constraint in order to map the required data points from a building. The shape graph for each target class is similar to the one listed in Listing 4.

Listing 4: SHACL Shapes generation for each Input. {point} is a placeholder that holds the target class name

```
inst:AlgorithmInputShape
    a sh:NodeShape ;
    sh:targetClass {point} ;
    sh:property [
    sh:path ref:hasExternalReference ; sh:minCount 1 ; sh:node sh:NodeShape ;
    sh:property [ sh:path rdf:type ; sh:hasValue ref:TimeseriesReference ; ] ;
    sh:property [ sh:path ref:hasTimeseriesId ; sh:minCount 1 ; sh:datatype
        xsd:string ; ] ;
    ] .
```

The idea of the proposed approach is to improve the portability of the developed application by leveraging knowledge graphs. A formal knowledge representation makes that application agnostic to a particular building, system or sensors, thereby making it highly portable to a different building. This is done by means of discovering the capabilities of that new building and adjusting the artifact to new data points.

This is achieved by interacting with the RDF Service described in Section 4.2's five steps. An intermediate output when generating the configurations for the building load forecasting artifact in Step 5, is shown in Table 1. Although the full list of data points is

Table 1: Data points mapping of the building load forecasting artifact to the new building graph. (Only five of the data points are shown)

No.	Target Class	label	tsid	database
1	brick:Electric_Power-_Sensor	BuildingLoad	24988dbf-9ca6-40e7-bc24-104deb8e7672	https://ivs.kropman.nl
2	brick:Outside_Air-_Temperature_Sensor	OutdoorTemperature	0de8b6d5-7f06-4b58-8493-924be6c968af	https://ivs.kropman.nl
3	sba:SolarRadianceForecast	SolarRadiance	aef5c0e8-a480-4153-a8b6-e8e07756dae4	https://ivs.kropman.nl
4	brick:Wind_Speed_Sensor	WindSpeed	a6aff25e-943f-4b45-a534-5f40262a879b	https://ivs.kropman.nl

exhaustive, only a few points are shown here. Here, "tsid" refers to the unique time series identifier of a particular data point. "database" provides information about the the URL that needs to be queried to get this data. This table is dynamically filled with the metadata from the given building. Then, Table 1 is used to generate configuration files or environment variable files as necessary within the data-driven application. After all the steps are complete, the MPC system is now modified to the given building.

6. Conclusion

Ontologies in the smart building domain allow describing a building in a machine-readable format and enable programmatic exploration of different features of a building. However, current approaches do not fully describe the data-driven applications (artifacts). A standardized metadata representation of such software artifacts complements the knowledge representation by enabling smart building applications to be configured easily to a different environment. This study proposes an adaptable workflow with the help of building graphs, artifact graphs, shape graphs and an RDF service. The proposed workflow relies on a generic method that can be adapted to data-driven applications in general. It is worth mentioning that the true benefit of the proposed workflow can be harnessed if the buildings share the vocabularies such as the Brick ontology, SSN, and the proposed new ontologies. By definition, an ontology should be an explicit, formal specification of a shared conceptualization (Gruber, 2008) and the agreement upon these ontologies is important to make the proposed workflow adaptable.

Future work focuses on implementing and validating the proposed RDF service to support the MPC and other data-driven applications. The ontologies will also be extended to integrate KPI ontologies to standardize the performance requirements of algorithms. Future research will also focus on adding inference and reasoning to building graphs to also introduce implicit relationships to the graph, before the execution of the proposed graph validation steps, to enhance querying.

Acknowledgements

This project received funding from the Dutch Ministry of Economic Affairs and Climate Policy and Ministry of the Interior and Kingdom Relations under the MOOI program. The authors would like to thank Jan-Willem Dubbeldam and Petros Zimianitis from team TCC, Kropman B.V.

References

Balaji, B., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., Johansen, A., Koh, J., Ploennigs, J., Agarwal, Y., Bergés, M., Culler, D., Gupta, R. K., Kjærgaard, M. B., Srivastava, M., & Whitehouse, K. (2018). Brick: Metadata schema for portable smart

building applications. *Applied Energy*, *226*(September 2017), 1273–1292. https://doi.org/10.1016/j.apenergy.2018.02.091

Chamari, L., Petrova, E., & Pauwels, P. (2022). A web-based approach to BMS, BIM and IoT integration: a case study. *REHVA 14th HVAC World Congress*, 2628–2635. https://doi.org/https://doi.org/10.34641/clima.2022.228

Chamari, L., Petrova, E., & Pauwels, P. (2023). An End-to-End Implementation of a Service-Oriented Architecture for Data-Driven Smart Buildings. *IEEE Access*, (October), 117261–117281. https://doi.org/10.1109/ACCESS.2023.3325767

Delgoshaei, P., Austin, M. A., & Veronica, D. (2017). Semantic Models and Rule-based Reasoning for Fault Detection and Diagnostics: Applications in Heating, Ventilating and Air Conditioning Systems. *The Twelfth International Conference on Systems*, (100), 48–53. https://pdfs.semanticscholar.org/b6aa/62285513182e28eef0d5977b524b108397cf.pdf

Dibowski, H., Vass, J., Holub, O., & Rojicek, J. (2016). Automatic setup of fault detection algorithms in building and home automation. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, *2016-Novem*. https://doi.org/10.1109/ETFA.2016.7733622

Gruber, T. (2008). A translation approach to portable ontology specifications. http://tomgruber.org.

Hammar, K., Wallin, E. O., Karlberg, P., & Hälleberg, D. (2019). The RealEstateCore Ontology. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11779 LNCS*. https://doi.org/10.1007/978-3-030-30796-7_9

Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., & Lefrançois, M. (2019). SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, *56*, 1–10. https://doi.org/10.1016/j.websem.2018.06.003

Rasmussen, M. H., Lefrançois, M., Schneider, G. F., & Pauwels, P. (2020). BOT: The building topology ontology of the W3C linked building data group (K. Janowicz, Ed.). *Semantic Web*, *12*(1), 143–161. https://doi.org/10.3233/SW-200385

Ren, H., Anicic, D., & Runkler, T. A. (2022). Towards Semantic Management of On-Device Applications in Industrial IoT. *ACM Transactions on Internet Technology*, *22*(4). https://doi.org/10.1145/3510820

Sagar, S., Lefrançois, M., Rebaï, I., Khemaja, M., Garlatti, S., Feki, J., & Médini, L. (2018). Modeling Smart Sensors on top of SOSA/SSN and WoT TD with the Semantic Smart Sensor Network (S3N) modular Ontology. *9th International Semantic Sensor Networks Workshop*, 1–15. https://hal.archives-ouvertes.fr/hal-01885330/