Journal of Building Engineering 86 (2024) 108645

Contents lists available at ScienceDirect

Journal of Building Engineering

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

Full length article

Enabling portable demand flexibility control applications in virtual and real buildings

Flavia de Andrade Pereira [a,b,c,*], Lazlo Paul [c], Marco Pritoni [c], Armando Casillas [c], Anand Prakash [c], Weiping Huang [c], Conor Shaw [a], Susana Martin-Toral [b], Donal Finn [a], James O' Donnell [a]

[a] School of Mechanical & Materials Engineering and UCD Energy Institute, University College Dublin, Dublin, Ireland
[b] CARTIF Technology Centre, Energy Division, Valladolid, Spain
[c] Lawrence Berkeley National Laboratory, Berkeley, United States of America

ARTICLE INFO

Dataset link: https://github.com/LBNL-ETA/DFLEXLIBS

Keywords:
Demand flexibility
Generalizable control
Semantic model
Portability
Simulation and field testing

ABSTRACT

Control applications that facilitate Demand Flexibility (DF) are difficult to deploy at scale in existing buildings. The heterogeneity of systems and non-standard naming conventions for metadata describing data points in building automation systems often lead to ad-hoc and building-specific applications. In recent years, several researchers investigated semantic models to describe the meaning of building data. They suggest that these models can enhance the deployment of building applications, enabling data exchanges among heterogeneous sources and their portability across different buildings. However, the studies in question fail to explore these capabilities in the context of controls. This paper proposes a novel semantics-driven framework for developing and deploying portable DF control applications. The design of the framework leverages an iterative design science research methodology, evolving from evidence gathered through simulation and field demonstrations. The framework aims to decouple control applications from specific buildings and control platforms, enabling these control applications to be configured semi-automatically. This allows application developers and researchers to streamline the onboarding of new applications that could otherwise be time-consuming and resource-intensive. The framework has been validated for its capability to facilitate the deployment of control applications sharing the same codebase across diverse virtual and real buildings. The demonstration successfully tested two controls for load shifting and shedding in four virtual buildings using the Building Optimization Testing Framework (BOPTEST) and in one real building using the control platform VOLTTRON. Insights into the current limitations, benefits, and challenges of generalizable controls and semantic models are derived from the deployment efforts and outcomes to guide future research in this field.

1. Introduction

The International Energy Agency (IEA) estimates the need for a tenfold increase in Demand Flexibility (DF) by 2030 relative to 2020 levels, 50% of which is expected to come from buildings [1]. DF is characterized by active load management across different timescales to support grid demands, particularly during high demand, low supply, or risk of renewable curtailment [2]. DF strategies include load shifting and shedding, which can be triggered by grid signals such as time-varying electricity prices [3].

* Corresponding author at: School of Mechanical & Materials Engineering and UCD Energy Institute, University College Dublin, Dublin, Ireland.
E-mail address: flavia.deandradepereira@ucdconnect.ie (F. de Andrade Pereira).

https://doi.org/10.1016/j.jobe.2024.108645
Received 11 October 2023; Received in revised form 23 January 2024; Accepted 24 January 2024
Available online 30 January 2024

Given the increase of variable renewable energy sources, the more frequent, intense, and longer-lasting extreme weather events, the current energy supply disruptions worldwide, and their impact on grid reliability, DF control applications have gained greater attention [4,5]. This is reflected in international directives and roadmaps, such as in the new reform of the European electricity sector [6] and the US national roadmap for grid-interactive efficient buildings [7].

Despite the potential, DF remains an underemployed resource, mostly applied in large consumers, research pilot projects, or residential buildings through direct load control programs [8–10]. Common challenges for adopting DF applications in buildings have been identified in the literature, for instance, by the IEA EBC Annex 82 working group [11]. Among these challenges, there is a lack of interoperable and portable applications, particularly supervisory controls that can react to grid signals. Supervisory control applications, such as for DF, are rarely deployed at scale due to the heterogeneous nature of the building stock, which leads to variations in the type of systems, data sources, communication protocols, and data formats [12]. While integrating these resources with control platforms can be facilitated through software solutions, configuring control applications predominantly relies on manual processes, which increases overall technology costs. There is a substantial effort to discover and map data points to control platforms due to the non-standard naming conventions used in Building Automation Systems (BAS) for the metadata that describe their data points [13,14]. This process is usually ad-hoc and building-specific, limiting the interoperability and portability of new applications [15,16].

In industry, several Energy Management and Information Systems (EMIS) have gained attention and increased adoption. These platforms are software layers built on top of the BAS, and they use semantic models to support point mapping and configuration of applications [17]. These platforms leverage semantics to add context and meaning to BAS metadata, making data points more discoverable and interpretable [18]. Among EMIS software solutions, Fault Detection and Diagnostics (FDD) tools have the greatest market penetration. These tools use one-way communication (read-only) to ingest BAS data and identify system faults [19]. Although very effective for data analysis, these tools typically lack active control capabilities [20,21]. Whilst recent research has investigated how to unlock their control capabilities, studies have focused on automatic fault correction and improved control sequences for energy efficiency rather than DF applications [22–24].

Research studies have explored the use of semantic models in assisting DF [9,25–32]. However, their focus has been primarily on data integration between heterogeneous resources from the building and grid sides and not on how these models can be used to facilitate the configuration of DF control applications, which may assist with their portability. Additionally, these studies have predominantly relied on semantic models based on custom-built metadata schemas that are neither maintained nor open-source. Moreover, many of these research efforts require storing dynamic data, such as measurement values, within the semantic models to apply semantic rules. This approach is considered inefficient when large time-series data streams are involved [33,34].

Further research has emphasized the promise of semantic models to automate the configuration of applications, facilitating their portability and plug-and-play behavior across various buildings [13,35]. Built on well-established metadata schemas, semantic models can automate the mapping between applications and data points from different buildings instead of having to hard-code them [36,37]. Examples of such efforts include Mortar [38] and Energon [39], which support the portability of analytic applications, as well as the multi-building data management platform for energy analysis proposed in [40]. Nevertheless, the demonstration of automated configuration and portability of control applications that require real-time data reading and writing, such as DF, remains limited. EFOnt has been introduced to foster the DF domain standardization and allow the streamlined configuration of DF control applications [41]. However, it is still in its early phases, with insufficient field demonstrations, expressivity, or alignment with other schemas to adequately represent required BAS points and their interactions with building systems and spaces.

This paper aims to support the development and deployment of portable DF control applications by proposing a semantics-driven framework. The framework addresses existing research gaps, especially the lack of approaches that harness semantic models built upon established metadata schemas to streamline the configuration of DF control applications while not requiring dynamic data to be stored in the models. This aims to facilitate the portability and plug-and-play behavior of DF control applications across various buildings, as has been demonstrated in studies related to analytic applications. To this purpose, this paper intends to answer the following research questions:

1. How can DF control applications be generalizable for different building system configurations and control platforms?
2. How can semantic models of buildings contribute to an automated configuration of generalizable DF control applications?
3. What are the benefits and challenges for the portability of generalizable DF control applications?

The remainder of this paper is organized as follows. Section 2 presents a background and states the contributions of this paper. Section 3 describes the methodology devised for the proposed semantics-driven framework. Section 4 presents two supervisory DF control applications and the virtual and real buildings used to demonstrate their deployment through the framework. Section 5 shows the results of the demonstration. Section 6 discusses the main findings and how they address the research questions. Finally, Section 7 outlines concluding remarks and future research directions.

2. Background and contributions

Portability within the context of buildings refers to the ability of a software application to be applied across different buildings [38]. To enable portability, controls must be developed in a generalizable manner with minimal dependency on individual building details, such as point names or control platforms. This can be supported by the capabilities offered by supervisory controls and semantic models. In this section, we provide the foundational background for these topics that form the basis of this paper's contributions.

2.1. Supervisory controls

Supervisory controls can coordinate the operation of multiple building systems based on high-level objectives, such as maximizing load shedding to reduce consumption during high grid demand [35]. This is generally accomplished by having the supervisory controls derive optimal, high-level setpoints that are then communicated to lower-level (local) controllers, which control equipment through traditional control loops. By allowing these high-level setpoints to adapt to varying building conditions, supervisory controls play a key part in enabling applications to be generalizable across different buildings. One key challenge in achieving this is to create a seamless connection between these local controls, measurement data points, and external data sources with the control platforms hosting supervisory controls [14].

Supervisory controls can be developed, tested, and deployed in virtual and real buildings using different platforms (i.e., simulation, BAS, or EMIS). For instance, the Building Optimization Testing Framework (BOPTEST) exemplifies a simulation platform for testing advanced control applications [42]. BOPTEST integrates realistically modeled buildings and exposes their control interface through an API that resembles that of a BAS. This allows control applications to interact with BOPTEST simulations as if they were controlling a real building [35]. VOLTTRON [43] is an example of an EMIS platform that collects heterogeneous data to perform analytics and execute control strategies, seamlessly integrating with BAS platforms. The mapping of required data points from multiple sources and the portability of supervisory control applications across buildings integrated with these platforms can be facilitated by semantic models.

2.2. Semantic models

Semantic models are data models built on metadata schemas that offer machine-readable, structured, and unambiguous data definitions. These schemas are typically represented as graphs and rely on technologies such as Resource Description Framework (RDF),[1] Web Ontology Language (OWL),[2] and Shapes Constraint Language (SHACL).[3] In the building domain, metadata schemas such as Brick [44], REC [45], SAREF [46], and the upcoming ASHRAE 223P [18,47] can represent buildings, their spatial aspects (e.g., thermal zones), equipment (e.g., chiller), including their components (e.g., fans), measurement and control points (e.g., sensors and setpoints), and the relationship between them.

When we represent the data points of buildings using semantic models, these models can then be employed to configure (instantiate) control applications for each building [35]. This is enabled by an approach known as Ontology-based Data Access (OBDA) that allows applications to use uniform SPARQL[4] queries across different data sources to discover and map required data points, instead of relying on hard-code metadata customized to each case [48]. For instance, a control application can be specified for a generic zone with abstract point names to define its inputs (e.g., related temperature measurements) and outputs (e.g., related temperature setpoints). Then, SPARQL can use common data definitions from the models to assist in mapping the application to specific zones and their respective sensor and control points within a building.

While maintaining time series and real-time data points in their original sources (e.g., BAS platforms or databases), the OBDA approach allows to query the semantic models for their external references (e.g., foreign key, time series identifier, pub-sub topics), as embedded in the models [18]. Interpreting these references can be outsourced to interfaces that connect the applications to the platforms running them, which should determine how to read or write in their data points, such as by using API requests.

The effective use of semantic models to configure control applications relies on the ability of these models to capture their metadata requirements (e.g., needed points and their relationships). Defining metadata requirements based on SHACL constraints has shown the potential to avoid ambiguous interpretation and facilitate reproducible model validation [37]. Based on SHACL, the work from [37] has provided a semantic validation algorithm, referred to as Building Metadata Ontology Interoperability Framework (BuildingMOTIF), which can verify whether a given model meets the metadata required by different applications.

2.3. Contributions of this paper

This paper proposes a semantics-driven framework for enabling portable DF control applications. The framework's main objectives are to create requirements for making supervisory controls more generalizable and to explore ways of using semantic models to facilitate their semi-automatic configuration in various buildings. The framework also employs control platform-oriented interfaces, which link the generalizable control applications to building data points, allowing them to abstract building- and platform-specific aspects. In summary, the contributions of this paper include:

- a method for developing generalizable control applications, identifying their metadata requirements, and validating their suitability in different buildings
- a configuration process that allows DF control applications to be self-configured and customized (when needed) to individual buildings
- a demonstration of the portability of DF control applications across heterogeneous buildings without extensive reprogramming
- an open-source implementation of all of the above[5]

[1] https://www.w3.org/RDF/.
[2] https://www.w3.org/OWL/.
[3] https://www.w3.org/TR/shacl/.
[4] https://www.w3.org/TR/rdf-sparql-query/.
[5] https://github.com/LBNL-ETA/DFLEXLIBS.

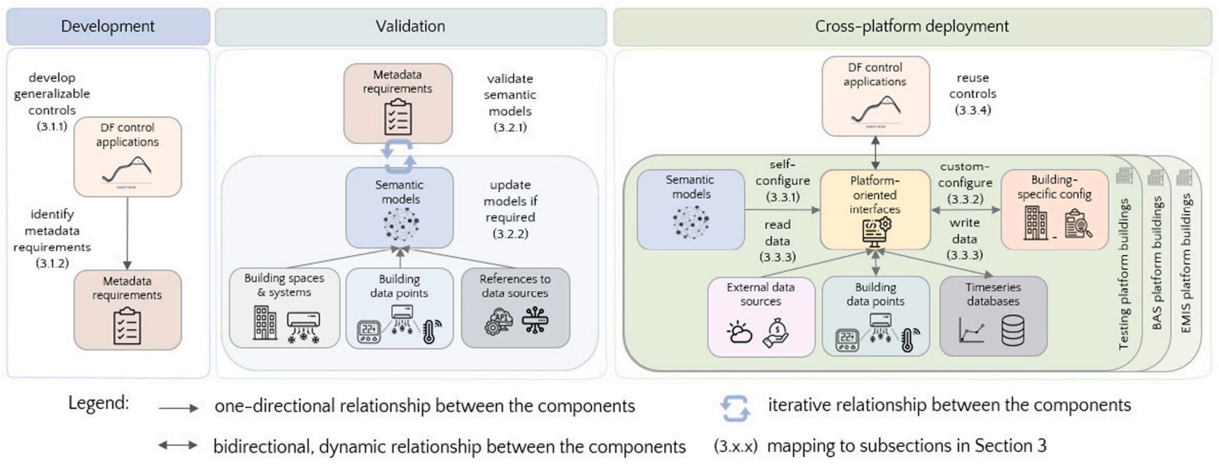

Development
Validation
Cross-platform deployment
develop generalizable controls (3.1.1)
DF control applications
identify metadata requirements (3.1.2)
Metadata requirements
validate semantic models (3.2.1)
Semantic models
update models if required (3.2.2)
Building spaces & systems
Building data points
References to data sources
reuse controls (3.3.4)
self-configure (3.3.1)
read data (3.3.3)
Platform-oriented interfaces
custom-configure (3.3.2)
write data (3.3.3)
Building-specific config
External data sources
Timeseries databases
Testing platform buildings
BAS platform buildings
EMIS platform buildings
Legend:
one-directional relationship between the components
iterative relationship between the components
bidirectional, dynamic relationship between the components
(3.x.x) mapping to subsections in Section 3

Fig. 1. Semantics-driven framework illustrating the main components and phases that guide the development and deployment of DF control applications that are portable across heterogeneous buildings.

3. Methodology

This section outlines the methodology devised for the proposed framework, which facilitates developing and deploying DF applications portable across heterogeneous buildings. The main phases of the framework include development, validation, and cross-platform deployment (Fig. 1), all of which are described in detail in the following sections. The framework serves as a guide for application developers and researchers, offering insights into the development and deployment of portable applications. The design of the framework was based on an iterative design science research approach [49], being greatly influenced by studies on application-driven semantic models [37] and cross-platform frameworks used in mobile app deployment [50,51]. The framework went through several iterations guided by evidence gathered from simulation and field demonstrations prior to the final design.

3.1. Development of control applications

In order for control applications to be portable across heterogeneous buildings, they need to be generalizable. In this section, we propose a method for developing generalizable control applications through a set of defined requirements, and for identifying their metadata needs through SHACL.

3.1.1. Develop generalizable controls

Generalizable controls are referred to in this paper as controls capable of being readily applicable in a wide range of buildings, system types, and control platforms without extensive customization. Being easy to deploy means that the software used to implement these controls should rely on reusable foundational scripts, minimizing the need for labor-intensive reconfiguration efforts. To develop control applications capable of delivering these desired outcomes in our framework, we established the following requirements.

- **Adaptable to changing operating conditions**: for a control to be generalizable to a large set of buildings, it must account for changing operating conditions that buildings might encounter [52]. This is crucial because making incorrect assumptions about current conditions (e.g., a predetermined occupancy schedule) can result in improper control actions' decisions. Supervisory controls can support this requirement by allowing systems to be managed based on diverse monitoring conditions such as occupancy schedules and status, comfort and faulty conditions, HVAC modes, grid signals, and specific constraints, including "hands-off" zones or specific DF settings.
- **Flexible to available data**: to further automate the portability process while minimizing user intervention, control applications need the ability to be configured based on available data [53]. This can be realized by employing methods that enhance the flexibility for querying new data while simultaneously enabling the controls to adapt accordingly.
- **Abstract to specific buildings**: to minimize reconfiguration efforts while making controls portable, these controls must be designed in a way that they are not tied to a specific building's context [35]. For that, applications must exclusively contain control logic, regardless of buildings and their specific control platforms' vendors, communication protocols, point naming practices, and data accessing methods. This can be supported by developing applications based on consistent, abstract, and easily queryable metadata rather than hard-coded point naming. To enhance portability and to support semantic validation processes that can help identify and handle data inconsistencies across buildings, this metadata should be based on well-established schemas and be formalized as requirements in an explicit and machine-readable way.

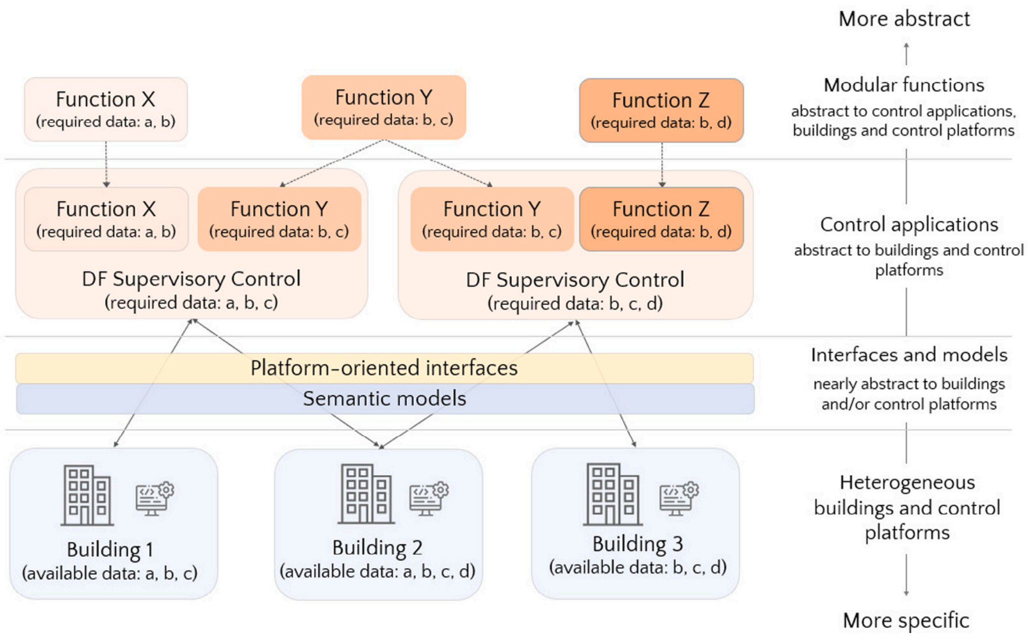

More abstract
Modular functions
abstract to control applications, buildings and control platforms
Control applications
abstract to buildings and control platforms
Interfaces and models
nearly abstract to buildings and/or control platforms
Heterogeneous buildings and control platforms
More specific
Function X (required data: a, b)
Function Y (required data: b, c)
Function Z (required data: b, d)
DF Supervisory Control (required data: a, b, c)
DF Supervisory Control (required data: b, c, d)
Platform-oriented interfaces
Semantic models
Building 1 (available data: a, b, c)
Building 2 (available data: a, b, c, d)
Building 3 (available data: b, c, d)

Fig. 2. Simplified illustration of modular supervisory control applications sharing functions and interacting with buildings via interfaces and semantic models. The spectrum on the right demonstrates the transition from abstract functions to specific buildings and control platforms.

- **Built on modular functions**: to effectively reduce the efforts associated with developing multiple generalizable controls, these controls should be composed of self-contained, modular functions that can be shared across them. Software solutions that are highly modularized can improve adaptability, enhance extensibility, and increase the potential for reuse [54]. To facilitate this, these functions should abstract not only buildings and control platforms' particularities but also the control applications.

Fulfilling these requirements should allow controls to be more easily applied to a diverse set of buildings without requiring extensive customization. Nevertheless, it should be noted that some applications are better suited for specific systems and lack complete generality. For instance, adjusting chiller setpoints strategies in cooling plants only applies to built-up chillers, commonly found in larger commercial buildings. In addition, these requirements are based on the authors' informed judgments, which are drawn from our expertise, a comprehensive review of existing literature, and insights gained while demonstrating this work. While it is crucial to acknowledge their inherent subjectivity and potential uncertainty, they are valuable for driving future development and research in this field. In Fig. 2 and in the following sections, we explore how we meet these requirements within the proposed framework.

Fig. 2 illustrates how our framework allows supervisory controls to share modular functions and be portable across different buildings through interfaces and semantic models. To develop these supervisory DF controls, this paper builds upon the existing work [55] and the ongoing efforts by the authors in providing common sequences of operation for HVAC-based DF applications. The description for two of them implemented and tested in this study is presented in Section 4.

3.1.2. Identify metadata requirements

Data requirements for control applications are often defined in English-language documents, such as ASHRAE Guideline 36, which describes the minimum points needed to implement their control sequences [56]. However, to avoid ambiguity issues and enable semantic-based validation and configuration of applications, data requirements must be described based on their metadata (context) [37]. For example, to adjust deviations from comfort levels while deploying a load shed control, an application must identify not only its inputs (temperature and the range of permitted setpoints values) and outputs (temperature setpoints) but their context as well. This can include relationships with building spaces or systems, and external references to access the points in their respective sources.

Our work proposes formalizing metadata requirements using the SHACL standard based on the Brick schema. The selection of Brick is due to its ability to capture metadata required in various control strategies and analytics [15,57], as well as its open-source nature, widespread use in the research community [58] and support by an industry consortium [18]. Nevertheless, the framework can be extended to other metadata schemas without significant changes. Algorithm 1 gives an example of a formalized definition of metadata requirement using SHACL. It requires a semantic model to have zones (brick:Zone) with at least one (sh:qualifiedMinCount 1) point (brick:hasPoint) of the type "temperature sensor" (brick:Air_Temperature_Sensor) and at least one (sh:minCount 1) external reference relationship (ref:hasExternalReference). Once defined for all control inputs and outputs, these SHACL metadata requirements can be used to validate that a given building, described by a semantic model, can support a specific control application.

Algorithm 1: Snipped of a formalized definition of metadata requirement using SHACL constraints. The @prefix notations establish namespaces for referencing external data schemas when applying rules.

```
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix sh: <http://www.w3.org/ns/shacl#>.
@prefix brick: <https://brickschema.org/schema/Brick#>.
@prefix ref: <https://brickschema.org/schema/Brick/ref#>.
@prefix df: <urn:df_constraints/>.
df:zone a sh:NodeShape, owl:Class;
    sh:targetClass brick:Zone;
    sh:property [
        sh:path brick:hasPoint ;
        sh:qualifiedValueShape [
            sh:class brick:Air_Temperature_Sensor ] ;
            sh:qualifiedMinCount 1 ; ] ; .
df:timeseries-identifier a sh:NodeShape;
    sh:targetClass brick:Air_Temperature_Sensor;
    sh:property [
        sh:path ref:hasExternalReference ;
        sh:nodeKind sh:BlankNode ;
        sh:minCount 1 ; ] ; .
```

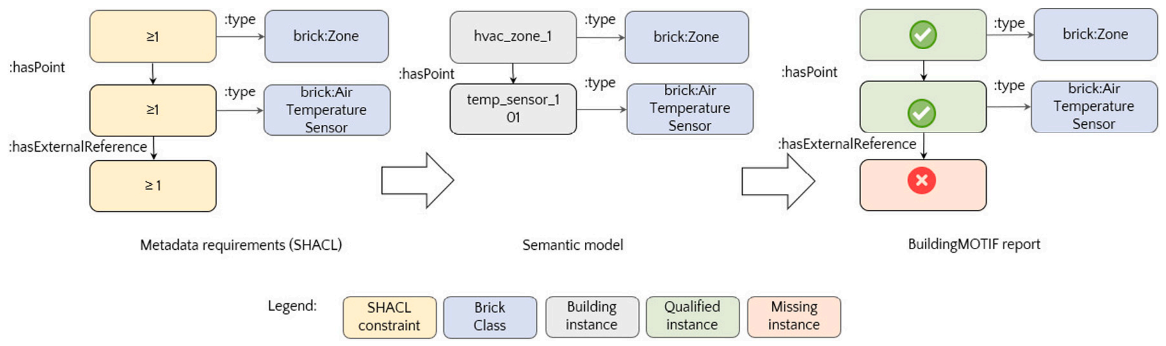


Fig. 3. Example of metadata requirements (SHACL) to validate the semantic model for a building using BuildingMOTIF.

3.2. *Semantic sufficiency validation*

To validate the suitability of a control for a given building based on its semantic model, our framework uses the BuildingMOTIF toolset[6] [37]. In this section, we describe how this toolset is used for validation and how to update the models based on the generated report.

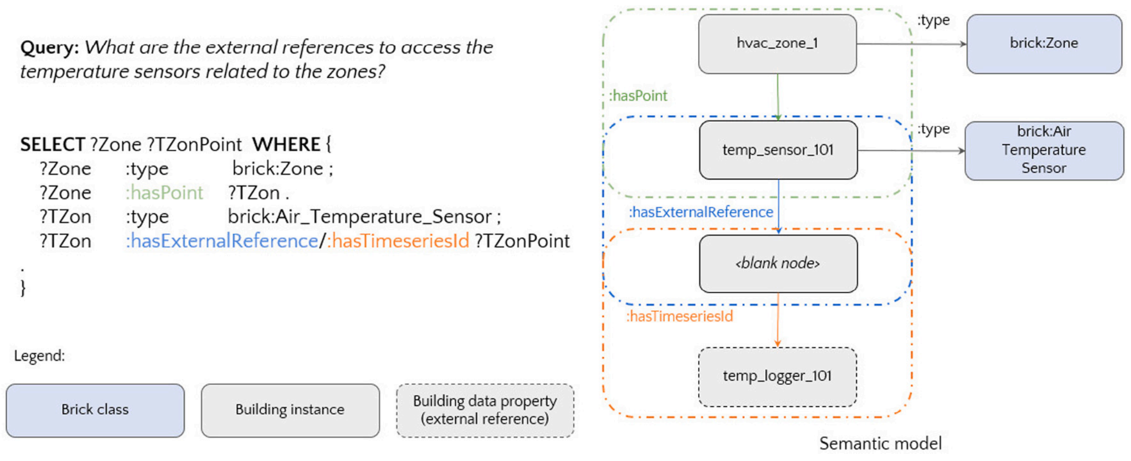
3.2.1. *Validate semantic models*

During the validation, the BuildingMOTIF toolset can help automate the comparison between the control's metadata requirements expressed with SHACL constraints and the metadata available in semantic models of specific buildings. Fig. 3 illustrates an example of metadata requirements (in SHACL) for a given control based on the Algorithm 1. By comparing these requirements with a subset of available metadata from a semantic model, BuildingMOTIF scripts can provide feedback on missing metadata. In this case, it highlights the absence of an external reference connected to the zone temperature sensor, suggesting that the model needs to be updated to run the desired application. This clear and actionable feedback from the validation process helps to expedite error identification and troubleshooting.

3.2.2. *Update models if required*

There are three potential reasons why a semantic model of a building might lack metadata. Firstly, it could be due to the absence of certain elements in the actual building, such as the absence of a return fan in an air handler. Secondly, the element may be present, but it lacks available or reliable metadata about it. For instance, a sensor might be collecting data, but the model lacks

[6] https://nrel.github.io/BuildingMOTIF/README.html ALPHA release.



Query: *What are the external references to access the temperature sensors related to the zones?*

```
SELECT ?Zone ?TZonPoint  WHERE {
   ?Zone      :type          brick:Zone ;
   ?Zone      :hasPoint     ?TZon .
   ?TZon      :type          brick:Air_Temperature_Sensor ;
   ?TZon      :hasExternalReference/:hasTimeseriesId ?TZonPoint
.
}
```

Fig. 4. Snippet of a semantic model of a given building and sample of a query to retrieve the external references from zone temperature sensors needed for the DF control application.
Source: Adapted from [60].

the metadata related to its access through the BAS API or communication protocol. Lastly, the element might exist and have data or metadata, but it has not been included in the semantic model for some reason. In this case, the modeler may have chosen not to incorporate certain types of sensors in the first version of the model. In situations where the second or third scenario applies, the semantic model needs to be updated with the missing metadata, to run the selected application. Ideally, experts responsible for the semantic model of the building should support and execute this update. This way, we can reduce the workload on application developers. The process can be done by explicitly adding those missing metadata in the model or using another capability from the BuildingMOTIF toolset, which can help automate the correction of a model. This correction autogenerates templates based on the missing metadata and adds them as new concepts in the semantic model. Successfully passing the semantic validation for all metadata requirements ensures that a semantic model is comprehensive enough to allow the configuration of required inputs and outputs of a control application.

3.3. *Cross-platform deployment*

Once validated, the deployment of control applications in distinct platforms (e.g., simulation, BAS, and EMIS) relies on their different methods of accessing and interacting with building data points [59]. For instance, while one platform may allow accessing its points through external time series databases, others can offer direct access to publish–subscribe topics or BACnet objects on a network. To ensure that control applications can effectively be configured to read and write data in heterogeneous buildings, it is important to have platform-oriented interfaces. These interfaces can decouple generalizable control applications from specific buildings while enabling any required customization in their configuration and deployment. In this section, we present how our framework allows the applications to be self- and custom-configured for reading and writing data in given buildings and the role of the interfaces in this.

3.3.1. *Self-configure*

To streamline the deployment of control applications within given buildings, our framework leverages their semantic models to self-configure them as much as possible. Enabling self-configuration involves using SPARQL queries to establish mappings between abstract point names defined in the applications and their corresponding physical or virtual counterparts in different buildings.

An example of a SPARQL query used for this purpose is illustrated in Fig. 4. On the left side of the figure is a SPARQL query looking for external references to access zone temperature sensor points for an arbitrary number of zones in a particular control platform. This SPARQL query can be derived from the SHACL shapes, which already define the metadata needed for applications, and be supplemented by concepts that define how the current interfaced platform allows access to its points (i.e., in this example, via a ref:hasTimeseriesId relationship). On the right side of the figure is a snippet of a semantic model instantiating concepts and relationships from Brick to describe a temperature sensor point in a given zone. The answer for the query gives the zone point references to external sources as embedded in the queried model, including “temp_logger_101” for “hvac_zone_1”. Using similar queries to satisfy all metadata requirements of an application allows one to self-configure it for different buildings, reducing the reliance on manual setup.

3.3.2. *Custom-configure*

Although supervisory controls can help generalize applications, data availability and constraints may differ between buildings. For example, knowledge of the current HVAC operating mode (heating or cooling) is required to determine the appropriate setpoint

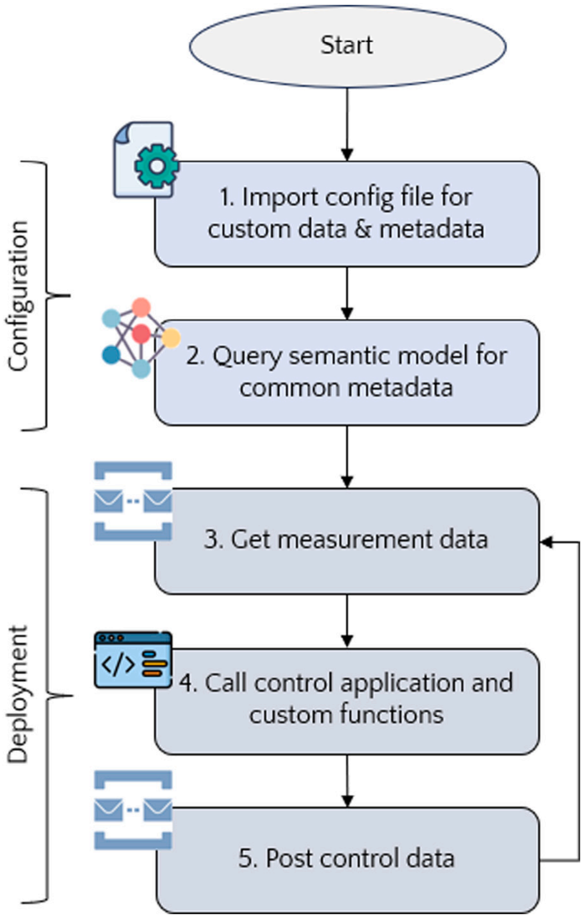

Start
1. Import config file for custom data & metadata
2. Query semantic model for common metadata
3. Get measurement data
4. Call control application and custom functions
5. Post control data
Configuration
Deployment

Fig. 5. 5-step workflow for configuring and deploying generalizable control applications in the interfaces.

adjustments. One building may expose this information as a data point, while another may require this to be inferred. Custom functions to determine control inputs may be needed in these cases. In addition, to accommodate their unique needs, certain buildings may require custom-designed control flows, which may require control outputs to be modified in a given way, such as prioritizing zones or equipment. Moreover, certain applications may require metadata not covered by the chosen metadata schema, not allowing them to be self-configured. Those could involve considerations concerning DF, such as minimum and maximum activation duration for load shedding [61], not often present in building-centric metadata schemas. Our work acknowledges the need for these customizations, allowing tailored functions to be integrated with generalizable control applications through the interfaces. For example, having applications that only rely on the operating mode values, being agnostic on how they are estimated. We also provide a set of templates for configuration files that can incorporate building-specific constraints not covered by the semantic models, while ensuring that they can be managed in a normalized way by the interfaces.

3.3.3. *Read and write data*

To complete the setup for deploying control applications in different buildings and on various control platforms, we must establish their connections. This is achieved through interfaces specifically designed for each control platform, tailored to their methods of accessing and interacting with building data points. Based on these methods, the interfaces can interpret the point's external references originating from the semantic queries and configuration files, enabling the required data by the applications to be read and written accordingly in different buildings.

3.3.4. *Reuse controls*

The configuration and deployment of the control applications through the interfaces are crucial in enabling their reuse in various buildings. Fig. 5 summarizes the sequence of actions the interfaces perform through a 5-step workflow. Step 1, the interfaces import custom data and metadata from configuration files for each building (e.g., getting DF implementation constraints). Step 2, the interfaces query the semantic modes to self-configure the application with common metadata (e.g., related to building data points). Step 3, the interfaces get the available measurement data (in some cases, iterating over each zone). They use the external references

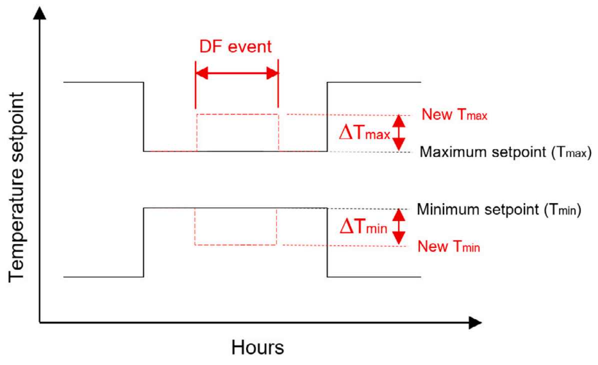


Fig. 6. Relaxing thermal comfort requirements by adjusting permitted setpoint range (minimum and maximum bounds) during a DF shed event. *Source:* Adapted from [55].

obtained by the semantic queries or defined in the configuration files and interact with the data points according to the control platform (e.g., sending RESTful requests). Step 4, the interfaces call the control applications and custom functions scripts, passing the required processed data. Step 5, the interfaces post new control signal data in the control platforms. Then, the interfaces repeat steps 3 to 5 in a loop for the length of the control execution period.

By employing supervisory, generalizable control applications in step 4, the interfaces can decouple them from specific building data points and the communication methods of given control platforms. This allows for streamlining the portability of DF control applications across diverse buildings, helping to minimize reprogramming efforts in the process.

4. DF controls development and deployment across simulation and field testbeds

To demonstrate the portability of the proposed framework, we created two control applications and tested them in five virtual and real facilities. These buildings consist of one residential and four commercial buildings, spanning various locations, floor areas, HVAC systems, baseline controls, boundary and operating conditions, and control platforms (BOPTEST and VOLTTRON). In this section, we describe both applications and the simulation and field testing performed.

4.1. Control applications development

The two proposed applications perform zone temperature adjustments. The first application focuses on load shedding, while the second addresses load shifting. Load shedding entails reducing building demand for a short period of time during shed events, which typically coincident with electric power system peaks, such as extremely hot summer afternoons [62]. Load shifting is the ability to change the timing of electric demand, often moving consumption from peak periods to off-peak times [62]. Shift is frequently implemented as a combination of a "take strategy" that increases energy consumption (e.g., pre-cooling in the morning) and a "shed strategy" (e.g., relaxing setpoints during the hot summer afternoons) [63], which reduces energy use, compared to a baseline. Shed and take conditions can be communicated as events with a clear start and end or via prices, which need to be interpreted by a logic and mapped to event conditions (e.g., price above $1/kWh is considered a shed event). In this paper, we will use prices to communicate grid needs.

4.1.1. Shed control strategy

Shed control strategies can reduce demand by allowing the temperature to "float" to a more relaxed setpoint, which delays the operation of HVAC system [55], as shown in Fig. 6. Its effectiveness depends on thermal comfort boundaries, as well as internal and external heat gains [64].

Algorithm 2 presents a pseudo-code representation of the proposed control application. This application was designed in a modular fashion, where self-contained functions were pieced together according to the chosen strategy. The application begins with a function that relaxes the comfort range for DF event periods with customized offset values. Then, it includes a function that assesses the thermal comfort of different zones. If the zone temperature is within the expanded temperature band, it is considered eligible for DF control. If the zones are eligible, another function evaluates the current grid signal and assesses if a load shed event has started (i.e., if the price is above a threshold estimated as the third quartile of the input price distribution). The shed function computes new setpoints according to the current HVAC operation mode and the zone setpoints. If no shed event is detected or the zones are not eligible for DF controls, the application releases the control, which means it incrementally returns the setpoints to their baseline values.

Algorithm 2: Pseudo-code for the zone temperature adjustment application performing load shedding.

```
1  relaxComfortRange;
2  if ComfortCheck then
3      if ShedEvent then
4          if HeatingMode then
5              $newSetpoint \leftarrow computeHeatSetpoint$
6          else
7              $newSetpoint \leftarrow computeCoolSetpoint$
8      else
9          releaseDFControl
10 else
11     releaseDFControl
```

4.1.2. Shift control strategy

Pre-cooling or pre-heating a building to shift energy is more effective in high-mass buildings, where thermal inertia allows to slow down the temperature (rise or dip) when the setpoints are relaxed [65]. The magnitude of the energy shifted is also a function of outside temperature, outside air flow rate, internal heat gain, and solar heat gains [63].

Algorithm 3 outlines a pseudo-code example of the proposed application that adjusts zone temperatures while performing load shifting (i.e., take and shed). Built upon Algorithm 2, this control logic includes a new take event function. This function evaluates the potential for load increase based on a combination of future price signals, occupancy patterns, and the current zone temperatures. The concept of "future" can be customized by setting a specific time horizon, indicating when the application can begin to check for appropriate conditions (e.g., 3 h before high-price periods). If, at this future time, the price surpasses a certain threshold, and the zone is expected to be occupied, while the current temperature remains within the comfort range, the function will recognize the need to start pre-heating or pre-cooling the building. Then, the application computes a new setpoint based on the current season and zone temperature setpoints.

Algorithm 3: Pseudo-code the zone temperature adjustment application performing load shifting and shedding.

```
1  relaxComfortRange;
2  if ComfortCheck then
3      if TakeEvent then
4          if HeatingSeason then
5              $newSetpoint \leftarrow computeHeatSetpoint$
6          else
7              $newSetpoint \leftarrow computeCoolSetpoint$
8      else if ShedEvent then
9          if HeatingMode then
10             $newSetpoint \leftarrow computeHeatSetpoint$
11         else
12             $newSetpoint \leftarrow computeCoolSetpoint$
13     else
14         releaseDFControl
15 else
16     releaseDFControl
```

4.1.3. Metadata requirements

The metadata requirements for both control strategies include temperature sensors, temperature setpoints, minimum and maximum temperature setpoints, and occupancy sensors related to zones, as well as price signals, DF constraints, and external references to access all of them. For those metadata requirements that have corresponding concepts in the Brick schema, they were translated to SHACL, as described in Section 3.1.2, and are available in our online repository.[7] The use of these requirements for semantic validation against the tested buildings is further detailed in Section 4.3. For those metadata requirements that are not captured by the Brick schema, they were not accounted for in the semantic validation analysis but were included in the templates

[7] https://github.com/LBNL-ETA/DFLEXLIBS.

Table 1
Overview of the tested virtual buildings.

Building (label)	Location	Area	Zones	Type	HVAC system
Bestest air (B1)	Denver, USA	48 m^2	Single	Small commercial	Idealized four-pipe fan with heating coil served by a gas boiler and cooling coil served by a chiller
Bestest hydronic heat pump (B2)	Brussels, Belgium	190 m^2	Single	Residential	Air-to-water heat pump
Singl-zone commercial hydronic (B3)	Copenhagen, Denmark	8500 m^2	Single	Small commercial	Air handling unit with heat recovery and radiant heating
Multi-zone office simple air (B4)	Chicago, USA	1660 m^2	Multiple	Large commercial	Single-duct VAV system with heating coil served by a heat pump and cooling coil served by a chiller

Table 2
Overview of the tested real building.

Building (label)	Location	Area	Zones	Type	HVAC System
Office building (B5)	New York, USA	351 m^2	Single	Small commercial	Dual-fuel heating system of heat pumps and gas-furnace

provided for the custom configuration files. Among them were the references needed to access grid signals from external data sources and constraints related to DF implementation.

4.2. Simulation and field testbed

In our simulation testing, we used four virtual buildings, realistically modeled and validated in BOPTEST [42]. Table 1 presents an overview of the locations, areas, zones, types, and HVAC systems of the tested buildings. More detailed information about these test cases can be found on the official BOPTEST page.[8]

In our field testing, we used VOLTTRON middleware, adding semantic models to its software architecture to facilitate the portable deployment of applications [60]. The testing was performed in a small office building in the US. Table 2 summarizes the location, area, zone, type, and HVAC systems of the building. More detailed information about the characteristics of this building and its baseline control can be found on [60].

4.3. Semantic validation

To validate the tested buildings against the metadata requirements from our developed controls, we created an interface to connect their semantic models and the SHACL constraints into BuildingMOTIF. For the virtual buildings, we generated semantic models for B1, B2, and B3 based on the foundation established in a previous study by the authors for modeling B4 [12]. These models are expected to not only contribute to this study but also advance the ongoing efforts within the BOPTEST community towards an enhanced semantic understanding of its test cases [66]. For the real building, we generated its semantic model using the model creation feature of BuildingMOTIF [37].

Following the example from Section 3.2, Fig. 7 illustrates the validation results from the BuildingMOTIF script applied in each building according to the controls' metadata requirements listed in Section 4.1.3.

To account for the variability of the setpoint definitions in the different buildings, the SHACL constraints were set to accept either a single temperature setpoint or both heating and cooling setpoints. It also accounted for model structure variations, accepting either a direct relationship between zone and point or through equipment. An occupancy sensor point was unavailable for all the buildings, so the interfaces were customized to obtain data from their occupancy schedules instead. Moreover, the semantic model from the real building did not include accessible BAS data points for minimum and maximum temperature setpoints, so fixed values were added for them in the interfaces. With these additional customizations, all tested buildings were suitable for the application. Section 6 provides a more in-depth discussion of these customizations.

4.4. Cross-platform deployment

To allow the proposed controls to interact with the buildings, we created control platform-oriented interfaces that could import the instantiated configuration files, query the semantic models, and call the required custom functions of each building. To self-configure the controls, SPARQL queries similar to the example in Section 3.3.1 were applied for all metadata requirements of the applications. To custom-configure the controls as described in Section 3.3.2, a few customizations were required.

[8] https://ibpsa.github.io/project1-boptest/testcases/index.html.

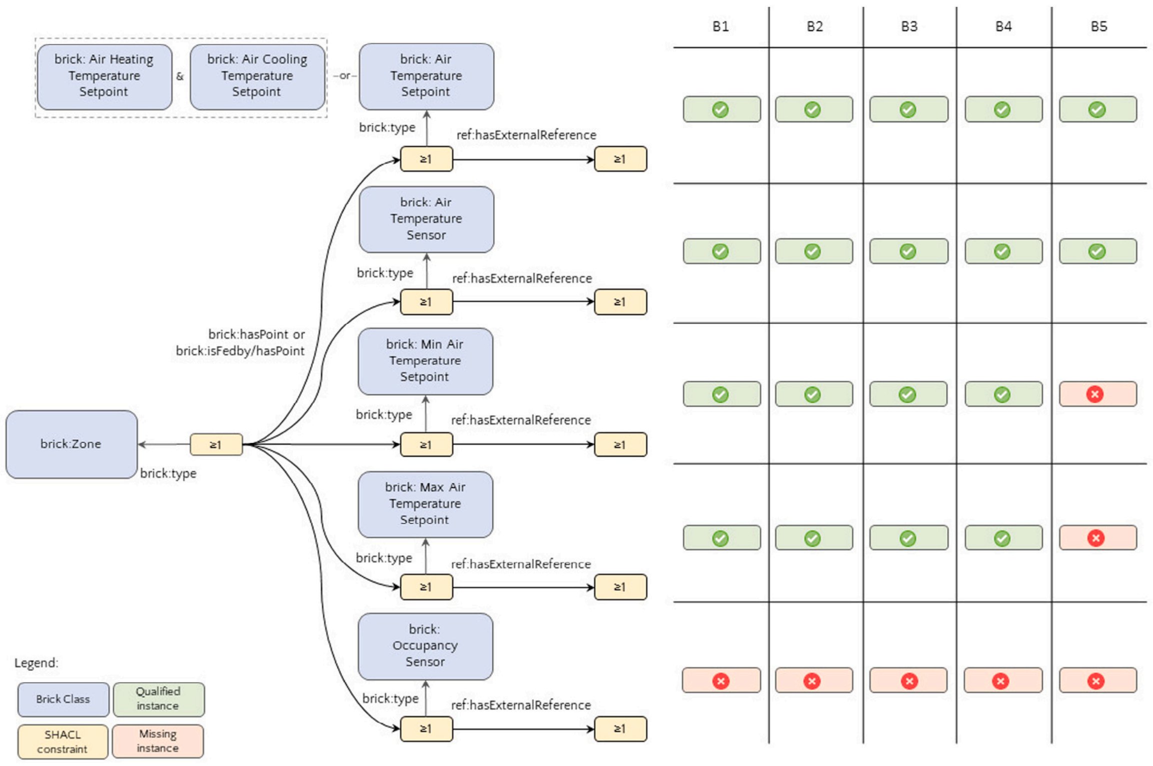

brick: Air Heating Temperature Setpoint
&
brick: Air Cooling Temperature Setpoint
–or–
brick: Air Temperature Setpoint
brick: Air Temperature Sensor
brick: Min Air Temperature Setpoint
brick: Max Air Temperature Setpoint
brick: Occupancy Sensor
brick:Zone
brick:type
brick:hasPoint or brick:isFedby/hasPoint
ref:hasExternalReference
≥1
B1
B2
B3
B4
B5
Legend:
Brick Class
Qualified instance
SHACL constraint
Missing instance

Fig. 7. Simplified demonstration of the semantic validation for the metadata requirements of the shift and shed control application versus the metadata available within the semantic models of each tested building.

In BOPTEST, we had to create custom functions to identify their HVAC operating modes since the buildings had different systems and lower-level logic. In VOLTTRON, we needed to add another logic layer to our generalizable DF application. This combined the setpoint adjustments during shift and shed events with a staging (rotation) logic of the heat pump units to avoid simultaneous demand peaks. The logic considered priority levels assigned to each unit based on factors such as the temperature difference from the setpoint and the duration of their inactive period. In addition, as required, we imposed specific constraints to limit the duration of the shed event to a maximum of two hours.

To read and write data in the buildings, we enabled the interfaces to interact with their control platforms, as outlined in Section 3.3.3. For the virtual buildings, we used the BOPTEST web service[9] to send RESTful requests to GET and POST measurement data and new control signals, functioning similarly to a BAS. For the real building, we used a VOLTTRON agent[10] to receive measurement data and execute control signals from and into the building points.

We simulated both DF control strategies (shed and shift) in the virtual buildings during predefined two-week periods aligned with the peak heating and cooling days of the buildings as set in the BOPTEST testing scenarios. In all these cases, dynamic electricity prices were used to communicate grid needs. We tested the DF shift control strategy in the real building over a four-day period in March 2023, and used the building's real time-of-use price structure to identify the grid needs.

5. Results

This section presents the portability results for applying zone temperature adjustment in the tested buildings. We only show the second control strategy (shift) in detail, but the results from the first control strategy (shed) applied to the virtual buildings can be found in our online repository [11].

We analyze the portability of the shift control application across the five tested buildings by assessing its ability to adjust zone temperature setpoints based on monitored conditions and in response to grid signals, thereby enabling load shifting and shedding. The findings are presented in Fig. 8 by the setpoint changes and Fig. 9 by the load variations across all buildings. In the figures, the shed periods coincide with high energy price signals (yellow shade), while take periods are pre-configured before the shed period

[9] https://ibpsa.github.io/project1-boptest/docs-userguide/api.html.
[10] https://volttron.readthedocs.io/en/main/agent-framework/agents-overview.html.
[11] https://github.com/LBNL-ETA/DFLEXLIBS.

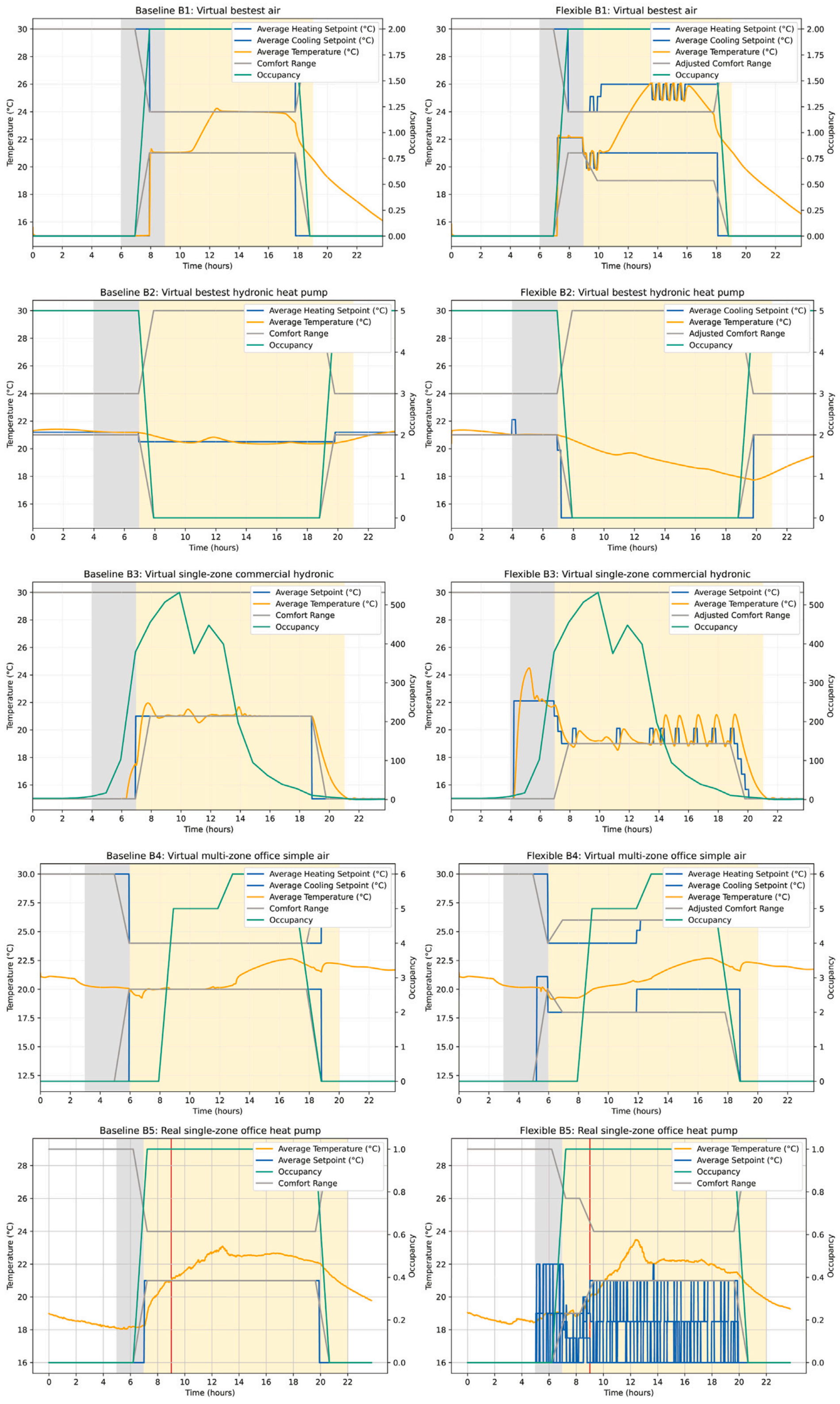


Fig. 8. Adjustment in the zone temperature setpoints and activation of the flexible (shift and shed) control in an average day for the five tested buildings, as per Tables 1 and 2. The gray and yellow shaded areas represent the shift and shed periods, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

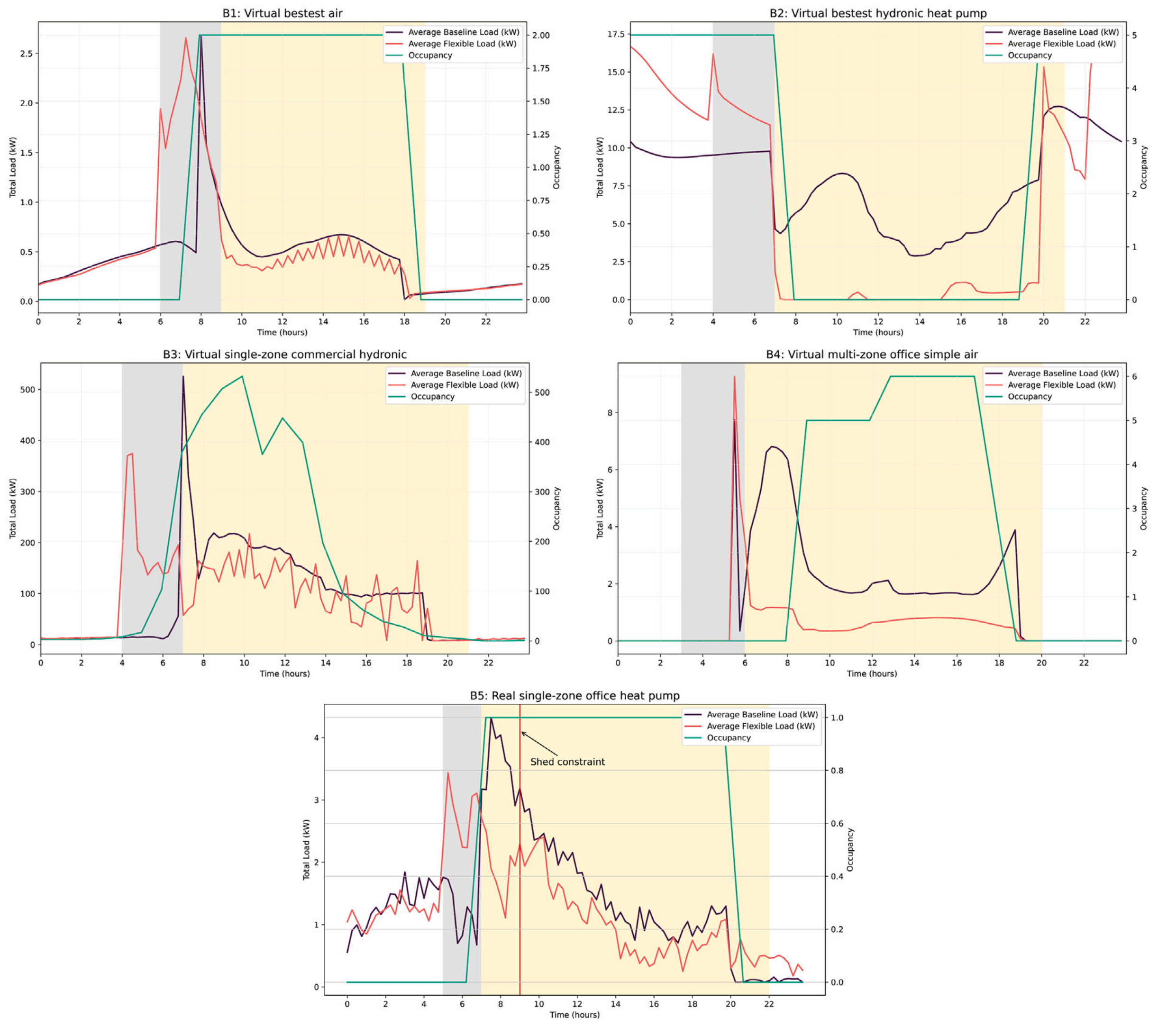


Fig. 9. Daily average power loads for the baseline and the flexible (shift and shed) control in the five buildings, as per Tables 1 and 2. The gray and yellow shaded areas represent the shift and shed periods, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(gray shade). For the virtual buildings, a 3-h period was defined. For the real building, both shift and take periods were limited to 2 h.

Fig. 8 demonstrates the zone temperature setpoint adjustments, including an increase for preheating the building during take and a decrease for reducing the heating loads during shed. The setpoint changes vary in each building based on factors such as occupancy profiles, price schedules, and thermal dynamics. The latter can be seen in B1 and B3, where the flexible setpoints oscillate due to temperature deviations from comfort values, which activate the comfort check function in various instances. This behavior showcases a fast thermal dynamic in the buildings, rapidly dissipating the energy accumulated during the take period, likely due to its low thermal mass. In building B5, we can observe the staging operations of the heat pumps through the variation in their setpoints. Although there is an instance when a manual override is shown, it is shortly counteracted by the control.

Fig. 9 compares the baseline and flexible loads in each tested building. It refers to 15-min load profiles averaged throughout the testing periods. The plot shows that buildings B2 and B4 experience increased peak loads during the take period when using the proposed control strategy. This increase can be attributed to their higher setpoints calculated for pre-heating the zones prior to occupancy. This pre-heating aimed to ensure an adequate temperature that allows for an extended shed period. Buildings B1 and B3 do not exhibit the same behavior. This might come from the 15-min resampling interval applied, which might not have captured their highest peak values, or by external factors, including their lower-level control logic, which deserves further investigation for a comprehensive understanding. Nonetheless, this issue did not arise in the real building as its heat pump units continuously rotated their operation, enabling effective peak reduction. The effect of the comfort check function with many oscillations in the flexible load can also be seen in Fig. 9, especially in B1 and B3.

These preliminary findings highlight the portability of generalizable DF control applications across diverse buildings, whether virtual or real. The controls could adapt to varying conditions, occupancy patterns, and grid signals with minimal reprogramming efforts (mostly on creating the platform-oriented interfaces and custom functions). Although the evaluation performance of the application is beyond the scope of this paper, the control implemented in the real building did not cause any occupant complaints (except for the one control override instance during the four days) and was able to reduce electricity costs at the site by 27%, successfully demonstrating the potential of this framework for delivering real DF control applications.

6. Discussion

This paper presents a novel framework that enables developing and deploying DF portable controls across diverse buildings with reduced reprogramming effort. This was made possible by the capabilities of the supervisory, modular controls, semantic models, and interfaces. In the subsequent subsections, we examine the initial research inquiries within the context of the established framework and the obtained results.

6.1. Generalization of DF control applications

Although our work demonstrates the deployment of a generalizable DF application across different building system configurations and control platforms, our analysis indicates that the idea of a comprehensive, portable control application remains limited. Some level of customization has to be considered during the configuration of the applications. In this section, we outline three main customization categories and explore how they are currently being tackled in existing research and how we approach them in our framework.

6.1.1. Baseline control and reliance on lower-level logic

Supervisory controls often rely on lower-level logic that may vary among buildings according to their system configurations, available sensor types, writing capabilities, and approaches to defining operating modes and scheduling [15]. For instance, one building can use compressor speed to identify the operating mode while another can use supply air temperature with a different logic. Understanding the baseline supervisory controls and their underlying logic in different buildings is critical to ensuring the intended outcome of new applications that replace and interact with them.

A software to facilitate the configuration of analytic applications using computational quantities based on SHACL rules and Brick concepts was designed in [53]. These quantities express various methods for obtaining data within a building, including direct sensor observations, indirect computations based on many observations, and default values. This could potentially help with the challenge of obtaining required data from various underlying logic. However, due to the lack of standardization in these logic, with each new building, developers might need to incorporate new computational quantities to express them. To tackle inconsistent control logic across different buildings, the OpenBuildingControl (OBC) initiative aims to establish a foundation for uniform control implementation, developing and encouraging tools, procedures, and standards (specifically ASHRAE 231P) [67]. This effort could serve as the basis for defining reusable computational quantities within the software provided by [53]. The problem is that the OBC initiative is currently in its developmental stages, so combining these works is not yet possible. Moreover, the software by [53] does not clearly identify how the data from different control platforms, or that are not within Brick nor default across buildings (e.g., shed offset value) can be handled for the same application.

In an effort to acknowledge these customization needs while easing the burden on application development, we use control platform-oriented interfaces in our framework. These allow generalizable control applications to be detached not only from site-specific control logic and constraints but also from diverse data accessing methods and the lack of expressivity by the Brick schema. Particularly looking at site-specific control logic, the interfaces allow custom functions to be easily handled and integrated with the generalizable control. For example, one Python-based module has been created to store various logic for determining HVAC operation modes. Developers may readily access this module to add new logic as needed. To select the applicable logic for a building, we can simply change 2 lines of code in the interface. As long as the low-level logic in the buildings can be identified, this approach helps to easily represent them and navigate the different options. This could be further enhanced using the portable computational quantities by [53], which have not been tested for control applications. The other benefits of our interfaces are further described in the next sections.

6.1.2. Platform-specific dependencies

Different control platforms often have distinct methods for accessing and modifying data (e.g., through APIs, publish–subscribe messaging, or given communication protocol methods). Consequently, to enable applications to be portable in buildings with distinct platforms, it is necessary to consider their intricacies and compatibility consideration. Moreover, different platforms may use different programming languages or methodologies, such as visual programming. Portable approaches must be based on a common programming language and methodologies to ensure consistent control across various buildings.

The Mortar platform [38] was developed to create and assess analytic applications that can be portable across different buildings. However, the current portability of this solution is limited to the buildings' data sets stored within the time series and Brick model databases linked to the platform. The investigation for additional databases or establishing connectivity with real-time data sources across different platforms, which is essential for control applications, has not been undertaken yet.

To leverage different data access methods, we implement interfaces that can be tailored to individual control platforms (such as VOLTTRON and BOPTEST). These interfaces, once created, may be easily reused across buildings sharing the same platform and across applications that require the same data points. That was realized in our demonstration when we reused the BOPTEST interface across the four virtual buildings for the two controls by only changing 3 lines of code. To facilitate deployment across various platforms while abstracting their different programming methodologies, our applications were developed in Python. Then, through each platform's data accessing methods, data exchange with the same control can be enabled as needed in different buildings.

6.1.3. Site-specific constraints

Deploying control applications, especially within real buildings, presents various challenges that can lead to errors and incorrect control actions. These can be related to inconsistent units of measurement, real-time data access delays, and missing data due to sensor unavailability, malfunctions, or communication issues. Despite these challenges being common across various buildings, their manifestations can vary and be site-specific. Hence, developing generalizable control functions to respond to them can be difficult. Moreover, in some cases, additional functions on top of DF applications are required in different buildings. For example, in the tested real building, an additional function had to be added to ensure the setpoint changes for DF were applied along with a staging operation of the heat pump units.

Existing controls often hard-code such site-specific constraints in the main control logic, resulting in applications that are not portable without extensive effort. Our framework includes a validation process that can help handle and normalize building inconsistency. This process can ensure that issues, such as those related to measurement units' mismatch and missing data due to sensor unavailability, are promptly reported during the control selection, allowing necessary (and possible) adjustments to be made before its deployment. Our framework may not offer straightforward solutions for challenges such as real-time data access delays or sensor failures due to emerging issues. Nonetheless, by allowing flexible semantic queries, it could be possible to support an adaptive control based on current conditions and alternative available data. In addition, our framework allows additional functions to be added on top of common DF applications in different buildings. They can be imported by the interfaces to use the outputs from the main DF control, handling the control signals as needed prior to sending them to the buildings.

6.2. Automated configuration with semantic models

While our work exploits semantic models and queries to self-configure the applications, we found that well-established building-centric metadata schemas [44–46] often lack essential metadata for running DF control applications. Those include metadata related to weather forecasts, grid signals, and DF settings. Examples of the latter include maximum duration or temperature offsets for relaxing comfort range during shed events, as well as lead time to initiate shift (take) periods. As a result, fully automated control applications are not possible yet.

Existing solutions employ custom-built metadata schemas to accommodate DF-related information such as grid signals [9,26,28,29]. However, these custom schemas are often unmaintained or not open-source, restricting their adaptability and reusability, especially when requiring extensions. The standardized SAREF metadata schema has *price* as one concept, but it falls short in establishing links to external time series data sources where this information is often available or in defining metadata for other grid signals and DF settings.

In our approach, we propose the use of template-based configuration files to systematically include missing metadata needed for DF applications through the interfaces. Although not exhaustive, these templates are meant to contain only information that is not currently represented by well-established metadata schemas. This approach could also be well-suited to facilitate the setup of standard building-to-grid communication mechanisms such as openADR[12] and CTA-2045.[13]

6.3. Benefits and challenges for portability of DF control applications

In order to better describe the benefits of our proposed framework, Fig. 10 compares a traditional workflow for developing and deploying supervisory control applications with ours. A traditional workflow mainly consists of (i) evaluating existing building systems (including their baseline controls) and user requirements, (ii) discovering the buildings' data points (identifying their metadata, communication network and which are possible to be overridden), (iii) developing a building-specific control logic based on (i) and (ii), (iv) mapping the inputs and outputs from the control to these points (usually embedding them in the logic) and connecting with their related communication network, (v) running and fine-tuning the control. In this workflow, each process is often bespoke, resulting in applications hard-coded to particular buildings. This makes it challenging to enable the portability of the same application in different buildings, often requiring one to redo most of the effort. Even when working within the same building, point mapping by one developer or for one application may be inadequate for another; they likely have to repeat them.

Our workflow proposes pre-processes, which may demand a labor-intensive effort but are highly reusable (one-time costs), and main processes, which are the ones repeated in different buildings but should be semi-automated with minimal user intervention

[12] https://www.openadr.org/.

[13] https://standards.cta.tech/apps/group_public/project/details.php?project_id=192.

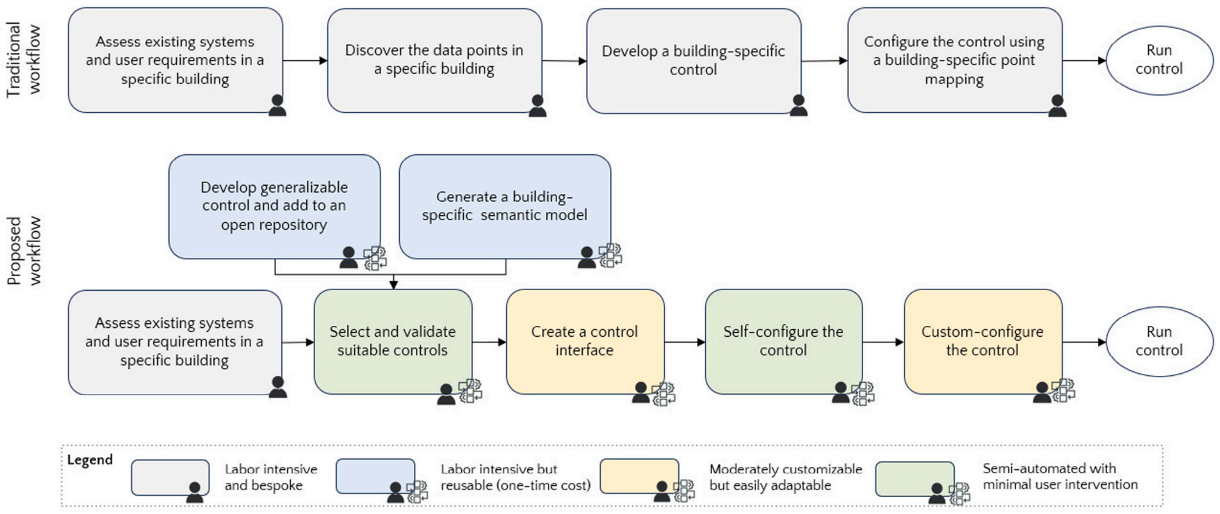

Traditional workflow
Assess existing systems and user requirements in a specific building
Discover the data points in a specific building
Develop a building-specific control
Configure the control using a building-specific point mapping
Run control
Proposed workflow
Develop generalizable control and add to an open repository
Generate a building-specific semantic model
Assess existing systems and user requirements in a specific building
Select and validate suitable controls
Create a control interface
Self-configure the control
Custom-configure the control
Run control
Legend
Labor intensive and bespoke
Labor intensive but reusable (one-time cost)
Moderately customizable but easily adaptable
Semi-automated with minimal user intervention

Fig. 10. Comparison between a traditional and our proposed workflow for developing and deploying supervisory control applications.

and customization. The pre-processes include developing generalizable controls and building specific semantic models. The controls should be developed along with templates for their metadata requirements (in SHACL), SPARQL queries, and configuration files, and they need to adhere to our pre-established requirements, which involve being adaptable, abstract, flexible, and modular in nature. The building-specific semantic models should be based on a well-established, open metadata schema.

Our main processes include selecting and validating the suitability of generalizable controls for given buildings based on their required and available metadata. This step is semi-automated because for validating the suitability, it is also important to identify the data points that can actually be overridden in a building, any alternative to missing points the building may offer, and the user requirements. Once the control is selected, interfaces should be set to handle the required data according to the site-specific platform (communication network). To self-configure the application with the building point's external references, the SPARQL query templates for the chosen control can be adapted as needed. Then, using the template configuration files, the control can be customized by adding specific setting parameter values (e.g., DF constraints) and metadata not covered by the metadata schema. This process also involves connecting the application to site-specific custom functions through the interface. Successfully adhering to this streamlined workflow can simplify control configuration processes, minimize user involvement, and potentially reduce the deployment costs associated with portable control applications.

Despite the potential benefits of portable control applications, deploying them in real buildings presents several challenges. The generalizability of complex control applications, such as those involving distribution and plant-level controls, requires further investigation, especially to understand their required degree of customization. Furthermore, the current absence of semantic models in real buildings, and the inconsistent design choices to represent the same concepts, present additional barriers to achieving control application portability [68]. Other challenges lie in the limited writing capability of supervisory control when applied to overlay software platforms like EMIS into BAS, and in the lack of trust by building owners and operators in third-party applications overriding existing controls [24].

7. Conclusions and future work

There is a critical need for solutions that enable the scalable development and deployment of DF control applications, particularly in the context of achieving global decarbonization goals for buildings. This paper proposes a novel framework that defines the main requirements and processes to allow applications to be decoupled from specific buildings while enabling them to be self-configured with minimal reprogramming efforts. By integrating features from modular supervisory controls, semantic models, and platform-oriented interfaces, the framework can support application developers and researchers in streamlining the onboarding of new applications that could otherwise be time-consuming and resource-intensive.

We demonstrated the portability potential of the framework by deploying applications sharing the same codebase in various virtual and real buildings using BOPTEST and VOLTTRON. The framework successfully reduced reconfiguration efforts, requiring only minimal script modifications in most cases. Such modifications were required to accommodate the current limitations in achieving a comprehensive, portable control application. These limitations were discussed in detail, highlighting how our framework addresses them and providing valuable insights for future research. In addition, although the applications proposed in this paper relate to DF controls, the framework may be widely adapted to other uses, including automated fault correction and advanced analytics.

Future work involves undertaking a more comprehensive validation of the framework in other real-world scenarios and performing an in-depth performance analysis of the controls to ensure their reliability and robustness. To further evaluate our framework, we aim to use software metrics grounded in computational complexity to quantitatively assess its benefits when compared with traditional workflows. Our future efforts also include creating an open-source library of reference DF control applications associated with various building archetypes. By linking the library of controls with our proposed framework, we will provide a comprehensive approach to the design and reuse of portable DF applications.

CRediT authorship contribution statement

Flavia de Andrade Pereira: Writing – original draft, Visualization, Software, Methodology, Data curation, Conceptualization. **Lazlo Paul:** Writing – review & editing, Software, Conceptualization. **Marco Pritoni:** Writing – review & editing, Software, Methodology, Conceptualization, Supervision. **Armando Casillas:** Writing – review & editing, Software, Conceptualization. **Anand Prakash:** Writing – review & editing, Software, Conceptualization. **Weiping Huang:** Writing – review & editing, Software, Conceptualization. **Conor Shaw:** Writing – review & editing. **Susana Martin-Toral:** Writing – review & editing. **Donal Finn:** Writing – review & editing. **James O' Donnell:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data and code for this study are available in our open-access repository via the following link: https://github.com/LBNL-ETA/DFLEXLIBS.

Declaration of generative AI in scientific writing

During the preparation of this work, the authors used AI-powered tools, such as Grammarly and Quillbot, in order to improve readability and language in specific portions of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Acknowledgments

This work was supported by the CBIM-ETN funded by the European Union's Horizon 2020 research and innovation program under the Marie Skodowska-Curie grant agreement No 860555; and by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors would like to thank Jessica Granderson, Dave Blum, and Peter Grant at the Lawrence Berkeley National Laboratory for their support and advice.

References

[1] International Energy Agency, Demand response, 2023, URL https://www.iea.org/energy-system/energy-efficiency-anddemand/demand-response.
[2] J. Liu, R. Yin, L. Yu, M.A. Piette, M. Pritoni, A. Casillas, J. Xie, T. Hong, M. Neukomm, P. Schwartz, Defining and applying an electricity demand flexibility benchmarking metrics framework for grid-interactive efficient commercial buildings, Adv. Appl. Energy 8 (2022) http://dx.doi.org/10.1016/j.adapen.2022.100107.
[3] A.Q. Santos, Energy in Buildings and Communities Programme, Annex 67 Energy Flexible Buildings, Teknologisk Institut, Technology Collaboration Programme, Control Strategies and Algorithms for Obtaining Energy Flexibility in Buildings, Danish Technological Institute, Taastrup, 2019, URL https://www.annex67.org/media/1898/control-strategies.pdf.
[4] H. Li, H. Johra, F. de Andrade Pereira, T. Hong, J. Le Dréau, A. Maturo, M. Wei, Y. Liu, A. Saberi-Derakhtenjani, Z. Nagy, A. Marszal-Pomianowska, D. Finn, S. Miyata, K. Kaspar, K. Nweye, Z. O'Neill, F. Pallonetto, B. Dong, Data-driven key performance indicators and datasets for building energy flexibility: A review and perspectives, Appl. Energy 343 (2023) http://dx.doi.org/10.1016/j.apenergy.2023.121217.
[5] X. Zhou, H. Du, Y. Sun, H. Ren, P. Cui, Z. Ma, A new framework integrating reinforcement learning, a rule-based expert system, and decision tree analysis to improve building energy flexibility, J. Build. Eng. 71 (2023) 106536, http://dx.doi.org/10.1016/j.jobe.2023.106536.
[6] N. Fabra, Reforming European electricity markets: Lessons from the energy crisis, Energy Econ. 126 (2023) 106963, http://dx.doi.org/10.1016/j.eneco.2023.106963.
[7] A. Satchwell, M. Piette, A. Khandekar, J. Granderson, N. Frick, R. Hledik, A. Faruqui, L. Lam, S. Ross, J. Cohen, K. Wang, D. Urigwe, D. Delurey, M. Neukomm, D. Nemtzow, A National Roadmap for Grid-Interactive Efficient Buildings, Tech. Rep., 2021, http://dx.doi.org/10.2172/1784302.
[8] S. Jensen, J. Parker, P. Engelmann, A.J. Marszal, Examples of energy flexibility in buildings, 2019, URL https://annex67.org/media/1921/examples-of-energy-flexibility-in-buildings.pdf.
[9] I. Esnaola-Gonzalez, F. Díez, Integrating building and IoT data in demand response solutions, in: Proceedings of the 7th Linked Data in Architecture and Construction Workshop - LDAC2019, 2019, URL https://ceur-ws.org/Vol-2389/07paper.pdf.
[10] J.A. Rama Curiel, J. Thakur, A novel approach for direct load control of residential air conditioners for demand side management in developing regions, Energy 258 (2022) http://dx.doi.org/10.1016/j.energy.2022.124763.
[11] R. Li, A.J. Satchwell, D. Finn, T.H. Christensen, M. Kummert, J. Le Dréau, R.A. Lopes, H. Madsen, J. Salom, G. Henze, K. Wittchen, Ten questions concerning energy flexibility in buildings, Build. Environ. 223 (2022) 109461, http://dx.doi.org/10.1016/j.buildenv.2022.109461.
[12] G. Fierro, A.K. Prakash, D. Blum, J. Bender, E. Paulson, M. Wetter, Notes paper: enabling building application development with simulated digital twins, in: Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 250–253, http://dx.doi.org/10.1145/3563357.3564060.

[13] G. Fierro, A.K. Prakash, C. Mosiman, M. Pritoni, P. Raftery, M. Wetter, D.E. Culler, Shepherding metadata through the building lifecycle, in: Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 70–79, http://dx.doi.org/10.1145/3408308.3427627.

[14] W. Wang, M.R. Brambley, W. Kim, S. Somasundaram, A.J. Stevens, Automated point mapping for building control systems: Recent advances and future research needs, Autom. Constr. 85 (2018) 107–123, http://dx.doi.org/10.1016/j.autcon.2017.09.013.

[15] M. Pritoni, D. Paine, G. Fierro, C. Mosiman, M. Poplawski, A. Saha, J. Bender, J. Granderson, Metadata schemas and ontologies for building energy applications: A critical review and use case analysis, Energies 14 (7) (2021) http://dx.doi.org/10.3390/en14072024.

[16] P. Delgoshaei, M. Heidarinejad, M.A. Austin, A semantic approach for building system operations: Knowledge representation and reasoning, Sustainability 14 (10) (2022) http://dx.doi.org/10.3390/su14105810.

[17] E. Crowe, H. Kramer, J. Granderson, EMIS applications showcase: Highlighting applications of energy management and information systems (EMIS), 2020, URL https://smartenergyanalytics.org/assets/EMIS%20Showcase.pdf.

[18] G. Fierro, P. Pauwels, Survey of metadata schemas for datadriven smart buildings (Annex 81), 2022, URL https://annex81.iea-ebc.org/Data/publications/IEA%20Annex%2081%20Survey%20of%20Metadata%20Schemas.pdf.

[19] H. Kramer, G. Lin, C. Curtin, E. Crowe, J. Granderson, Proving the business case for building analytics, 2020, http://dx.doi.org/10.20357/B7G022.

[20] M. Pritoni, G. Lin, Y. Chen, J. House, E. Crowe, J. Granderson, Market Barriers and Drivers for the Next Generation Fault Detection and Diagnostic Tools, Tech. Rep., Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2022, http://dx.doi.org/10.20357/B7801T.

[21] G. Lin, M. Pritoni, Y. Chen, J. Granderson, R. Moromisato, S. Kozlen, Can we fix it automatically? Development of fault auto-correction algorithms for HVAC and lighting systems, 2020, URL https://www.osti.gov/biblio/1755353.

[22] G. Lin, M. Pritoni, Y. Chen, J. Granderson, Development and implementation of fault-correction algorithms in fault detection and diagnostics tools, Energies 13 (10) (2020) http://dx.doi.org/10.3390/en13102598.

[23] L. Guanjing, M. Pritoni, Y. Chen, C. Weyandt, R. Vitti, J. Granderson, Fault "auto-correction" for HVAC systems: A preliminary study | building technology and urban systems, 2021, URL https://buildings.lbl.gov/publications/fault-auto-correction-hvac-systems.

[24] M. Pritoni, G. Lin, Y. Chen, R. Vitti, C. Weyandt, J. Granderson, From fault-detection to automated fault correction: A field study, Build. Environ. 214 (2022) http://dx.doi.org/10.1016/j.buildenv.2022.108900.

[25] J.L. Hippolyte, S. Howell, B. Yuce, M. Mourshed, H.A. Sleiman, M. Vinyals, L. Vanhee, Ontology-based demand-side flexibility management in smart grids using a multi-agent system, in: 2016 IEEE International Smart Cities Conference, ISC2, 2016, pp. 1–7, http://dx.doi.org/10.1109/ISC2.2016.7580828.

[26] J. Koh, S. Ray, J. Hodges, Information mediator for demand response in electrical grids and buildings, in: 2017 IEEE 11th International Conference on Semantic Computing, ICSC, 2017, pp. 73–76, http://dx.doi.org/10.1109/ICSC.2017.26.

[27] S. Howell, Y. Rezgui, J.-L. Hippolyte, M. Mourshed, Semantic interoperability for holonic energy optimization of connected smart homes and distributed energy resources, 2016, URL https://www.taylorfrancis.com/chapters/edit/10.1201/9781315386904-39/semantic-interoperability-holonic-energy-optimization-connected-smart-homes-distributed-energy-resources-howell-rezgui-hippolyte-mourshed.

[28] A. Fernandez-Izquierdo, A. Cimmino, C. Patsonakis, A.C. Tsolakis, R. Garcia-Castro, D. Ioannidis, D. Tzovaras, OpenADR ontology: Semantic enrichment of demand response strategies in smart grids, in: 2020 International Conference on Smart Energy Systems and Technologies, SEST, CRC Press, Istanbul, Turkey, 2020, pp. 1–6, http://dx.doi.org/10.1109/SEST48500.2020.9203093.

[29] G. Santos, Z. Vale, P. Faria, L. Gomes, BRICKS: Building's reasoning for intelligent control knowledge-based system, Sustainable Cities Soc. 52 (2020) http://dx.doi.org/10.1016/j.scs.2019.101832.

[30] A. Cimmino, J. Cano-Benito, A. Fernández-Izquierdo, C. Patsonakis, A.C. Tsolakis, R. García-Castro, D. Ioannidis, D. Tzovaras, A scalable, secure, and semantically interoperable client for cloud-enabled demand response, Future Gener. Comput. Syst. 141 (2023) 54–66, http://dx.doi.org/10.1016/j.future.2022.11.004.

[31] Y. Li, Y. Rezgui, S. Kubicki, An intelligent semantic system for real-time demand response management of a thermal grid, Sustainable Cities Soc. 52 (2020) http://dx.doi.org/10.1016/j.scs.2019.101857.

[32] H. Wicaksono, T. Boroukhian, A. Bashyal, A demand-response system for sustainable manufacturing using linked data and machine learning, in: M. Freitag, H. Kotzab, N. Megow (Eds.), Dynamics in Logistics: Twenty-Five Years of Interdisciplinary Logistics Research in Bremen, Germany, Springer International Publishing, Cham, 2021, pp. 155–181, http://dx.doi.org/10.1007/978-3-030-88662-2_8.

[33] P. Pauwels, A. Costin, M.H. Rasmussen, Knowledge graphs and linked data for the built environment, in: M. Bolpagni, R. Gavina, D. Ribeiro (Eds.), Industry 4.0 for the Built Environment: Methodologies, Technologies and Skills, in: Structural Integrity, Springer International Publishing, Cham, 2022, pp. 157–183, http://dx.doi.org/10.1007/978-3-030-82430-3_7.

[34] S. Hu, J. Wang, C. Hoare, Y. Li, P. Pauwels, J. O'Donnell, Building energy performance assessment using linked data and cross-domain semantic reasoning, Autom. Constr. 124 (2021) http://dx.doi.org/10.1016/j.autcon.2021.103580.

[35] A. Roth, M. Wetter, K. Benne, D. Blum, Y. Chen, G. Fierro, M. Pritoni, A. Saha, D. Vrabie, Towards digital and performance-based supervisory HVAC control delivery, 2022, http://dx.doi.org/10.20357/B70G62.

[36] D. Schachinger, W. Kastner, Context-aware optimization strategies for universal application in smart building energy management, in: 2018 IEEE 16th International Conference on Industrial Informatics, INDIN, 2018, pp. 478–483, http://dx.doi.org/10.1109/INDIN.2018.8472000.

[37] G. Fierro, A. Saha, T. Shapinsky, M. Steen, H. Eslinger, Application-driven creation of building metadata models with semantic sufficiency, in: Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 228–237, http://dx.doi.org/10.1145/3563357.3564083.

[38] G. Fierro, M. Pritoni, M. Abdelbaky, D. Lengyel, J. Leyden, A. Prakash, P. Gupta, P. Raftery, T. Peffer, G. Thomson, D.E. Culler, Mortar: An open testbed for portable building analytics, ACM Trans. Sens. Netw. 16 (1) (2019) http://dx.doi.org/10.1145/3366375.

[39] F. He, Y. Deng, Y. Xu, C. Xu, D. Hong, D. Wang, Energon: A data acquisition system for portable building analytics, in: Proceedings of the Twelfth ACM International Conference on Future Energy Systems, ACM, Virtual Event Italy, 2021, pp. 15–26, http://dx.doi.org/10.1145/3447555.3464850.

[40] J. Li, N. Li, B. Yue, R. Yan, K. Farruh, A. Li, K. Li, Research on the semantic web representation for building operation with variable refrigerant flow systems, J. Build. Eng. 56 (2022) 104792, http://dx.doi.org/10.1016/j.jobe.2022.104792.

[41] H. Li, T. Hong, A semantic ontology for representing and quantifying energy flexibility of buildings, Adv. Appl. Energy 8 (2022) 100113, http://dx.doi.org/10.1016/j.adapen.2022.100113.

[42] D. Blum, J. Arroyo, S. Huang, J. Drgoňa, F. Jorissen, H.T. Walnum, Y. Chen, K. Benne, D. Vrabie, M. Wetter, L. Helsen, Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings, J. Build. Perform. Simul. 14 (5) (2021) 586–610, http://dx.doi.org/10.1080/19401493.2021.1986574.

[43] S. Katipamula, J. Haack, G. Hernandez, B. Akyol, J. Hagerman, VOLTTRON: An open-source software platform of the future, IEEE Electr. Mag. 4 (4) (2016) 15–22, http://dx.doi.org/10.1109/MELE.2016.2614178.

[44] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Bergés, D. Culler, R.K. Gupta, M.B. Kjærgaard, M. Srivastava, K. Whitehouse, Brick : Metadata schema for portable smart building applications, Appl. Energy 226 (2018) 1273–1292, http://dx.doi.org/10.1016/j.apenergy.2018.02.091.

[45] K. Hammar, E.O. Wallin, P. Karlberg, D. Hälleberg, The RealEstateCore ontology, in: The Semantic Web – ISWC 2019, Vol. 11779, 2019, http://dx.doi.org/10.5281/zenodo.2628367.

[46] European Telecommunications Standards Institute, Smart applications reference ontology and oneM2M mapping, 2020, URL https://www.etsi.org/deliver/etsi_ts/103200_103299/103264/03.01.01_60/ts_103264v030101p.pdf. ETSI TS 103 264 V3.1.1 (2020-02).
[47] H. Bergmann, C. Mosiman, A. Saha, S. Haile, W. Livingood, S. Bushby, G. Fierro, J. Bender, M. Poplawski, J. Granderson, M. Pritoni, Semantic interoperability to enable smart, grid-interactive efficient buildings, 2020, p. 18, http://dx.doi.org/10.20357/B7S304.
[48] D. Calvanese, L. Ding, A. Mosca, G. Xiao, Realizing ontology-based reusable interfaces for data access via virtual knowledge graphs, in: CHItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter, ACM, Bolzano Italy, 2021, pp. 1–5, http://dx.doi.org/10.1145/3464385.3464744.
[49] M.K. Sein, O. Henfridsson, S. Purao, M. Rossi, R. Lindgren, Action design research, Manag. Inf. Syst. Res. Cent. 35 (1) (2011) 37–56, http://dx.doi.org/10.2307/23043488.
[50] I. Dalmasso, S.K. Datta, C. Bonnet, N. Nikaein, Survey, comparison and evaluation of cross platform mobile application development tools, ISBN: 978-1-4673-2479-3, 2013, pp. 323–328, http://dx.doi.org/10.1109/IWCMC.2013.6583580.
[51] M. Abdal, T. Ahmed, S. Jan, F. Khan, A. Khattak, A comparative analysis of mobile application development approaches, Proc. Pak. Acad. Sci. 58 (2021) 35–45, http://dx.doi.org/10.53560/PPASA(58-1)717.
[52] K. Nweye, B. Liu, P. Stone, Z. Nagy, Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings, Energy AI 10 (2022) 100202, http://dx.doi.org/10.1016/j.egyai.2022.100202.
[53] D. Mavrokapnidis, G. Fierro, I. Korolija, D. Rovas, A programming model for portable fault detection and diagnosis, in: Proceedings of the 14th ACM International Conference on Future Energy Systems, in: e-Energy '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 127–131, http://dx.doi.org/10.1145/3575813.3595190.
[54] L. Lazzari, K. Farias, Event-driven architecture and REST: An exploratory study on modularity, J. Appl. Res. Technol. (2023) 338–351, URL https://jart.icat.unam.mx/index.php/jart/article/view/1764/995.
[55] N. Motegi, M.A. Piette, D. Watson, S. Kiliccote, P. Xu, Introduction to commercial building control strategies and techniques for demand response, 2007, p. 76, URL https://eta-publications.lbl.gov/sites/default/files/59975.pdf.
[56] ASHRAE guideline 36-2018 - high-performance sequences of operation for HVAC systems, 2021, URL https://www.techstreet.com/ashrae/standards/guideline-36-2018-high-performance-sequences-of-operation-for-hvac-systems?product_id=2016214.
[57] F. De Andrade Pereira, C. Shaw, S. Martín-Toral, J. L. Hernández, R. Sanz Jimeno, D. Finn, J. O'Donnell, Towards Semantic Interoperability for Demand-Side Management: A Review of BIM and BAS Ontologies, in: Computing in Construction, vol. 3, University College Dublin, 2022, http://dx.doi.org/10.35490/EC3.2022.154.
[58] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Bergés, D. Culler, R.K. Gupta, M.B. Kjærgaard, M. Srivastava, K. Whitehouse, Brick : Metadata schema for portable smart building applications, Appl. Energy 226 (2018) 1273–1292, http://dx.doi.org/10.1016/j.apenergy.2018.02.091.
[59] L. Chamari, E. Petrova, P. Pauwels, Extensible Real-Time Data Acquisition and Management for Iot Enabled Smart Buildings, in: Computing in Construction, vol. 4, European Council on Computing in Construction, 2023, http://dx.doi.org/10.35490/EC3.2023.300.
[60] L. Paul, F. De Andrade Pereira, S.w. Ham, M. Pritoni, R. Brown, J.D. Feng, Open building operating system: an open-source grid responsive control platform for buildings, 2023, URL https://buildings.lbl.gov/publications/open-building-operating-system-open.
[61] ISO/TC 205 Building environment design, ISO 16484-3:2005 building automation and control systems (BACS) — Part 3: Functions, 2007, URL https://www.iso.org/obp/ui/en/#iso:std:iso:16484:-3:ed-1:v2:en.
[62] M. Neukomm, V. Nubbe, R. Fares, Grid-interactive efficient buildings technical report series: Overview of research challenges and gaps, 2019, http://dx.doi.org/10.2172/1577966.
[63] J. Liu, R. Yin, L. Yu, M.A. Piette, M. Pritoni, A. Casillas, J. Xie, T. Hong, M. Neukomm, P. Schwartz, Defining and applying an electricity demand flexibility benchmarking metrics framework for grid-interactive efficient commercial buildings, Adv. Appl. Energy 8 (2022) 100107, http://dx.doi.org/10.1016/j.adapen.2022.100107.
[64] Y. Fu, Z. O'Neill, J. Wen, A. Pertzborn, S.T. Bushby, Utilizing commercial heating, ventilating, and air conditioning systems to provide grid services: A review, Appl. Energy 307 (2022) http://dx.doi.org/10.1016/j.apenergy.2021.118133.
[65] Y. Ruan, J. Ma, H. Meng, F. Qian, T. Xu, J. Yao, Potential quantification and impact factors analysis of energy flexibility in residential buildings with preheating control strategies, J. Build. Eng. 78 (2023) 107657, http://dx.doi.org/10.1016/j.jobe.2023.107657.
[66] M. Wetter, J. Hu, A. Prakash, P. Ehrlich, G. Fierro, M. Grahovac, M. Pritoni, L. Rivalin, D. Robin, Modelica-json: Transforming energy models to digitize the control delivery process, 2021, http://dx.doi.org/10.26868/25222708.2021.30141.
[67] M. Wetter, P. Ehrlich, A. Gautier, M. Grahovac, P. Haves, J. Hu, A. Prakash, D. Robin, K. Zhang, OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification, Energy 238 (2022) http://dx.doi.org/10.1016/j.energy.2021.121501.
[68] I.L. Bennani, A.K. Prakash, M. Zafiris, L. Paul, C.D. Roa, P. Raftery, M. Pritoni, G. Fierro, Query relaxation for portable brick-based applications, in: Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, ACM, Coimbra Portugal, 2021, pp. 150–159, http://dx.doi.org/10.1145/3486611.3486671.