POLITECNICO DI TORINO
Repository ISTITUZIONALE

A portable application framework for energy management and information systems (EMIS) solutions using Brick semantic schema

Original
A portable application framework for energy management and information systems (EMIS) solutions using Brick semantic schema / Chiosa, R., Piscitelli, M.S., Pritoni, M., Capozzoli, A.. - In: ENERGY AND BUILDINGS. - ISSN 0378-7788. - 323:(2024). [10.1016/j.enbuild.2024.114802]

Availability:
This version is available at: 11583/2992646 since: 2024-09-20T09:28:44Z

Publisher:
Elsevier

Published
DOI:10.1016/j.enbuild.2024.114802

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

30 June 2026

Energy & Buildings 323 (2024) 114802

Contents lists available at ScienceDirect

Energy & Buildings

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

A portable application framework for energy management and information systems (EMIS) solutions using Brick semantic schema

Roberto Chiosa [a], Marco Savino Piscitelli [a,*], Marco Pritoni [b], Alfonso Capozzoli [a]

[a] Department of Energy "Galileo Ferraris", TEBE Research Group, BAEDA Lab, Politecnico di Torino, Corso Duca degli Abruzzi 24, Turin, 10129, Italy
[b] Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, 94720, CA, USA

ARTICLE INFO

Keywords:
Energy management and information systems
Portable application
Brick metadata schema
Anomaly detection
Machine learning

ABSTRACT

This paper introduces a portable framework for developing, scaling and maintaining energy management and information systems (EMIS) applications using an ontology-based approach. Key contributions include an interoperable layer based on Brick schema, the formalization of application constraints pertaining metadata and data requirements, and a field demonstration. The framework allows for querying metadata models, fetching data, preprocessing, and analyzing data, thereby offering a modular and flexible workflow for application development. Its effectiveness is demonstrated through a case study involving the development and implementation of a data-driven anomaly detection tool for the photovoltaic systems installed at the Politecnico di Torino, Italy. During eight months of testing, the framework was used to tackle practical challenges including: (i) developing a machine learning-based anomaly detection pipeline, (ii) replacing data-driven models during operation, (iii) optimizing model deployment and retraining, (iv) handling critical changes in variable naming conventions and sensor availability (v) extending the pipeline from one system to additional ones.

1. Introduction

The digitalization of the building sector, thanks to the widespread adoption of Internet of Things (IoT) technologies, significantly increased the amount of information pertaining actual building operation. In the current paradigm of smart buildings, building owners, energy managers and occupants can leverage more and more sophisticated software solutions, exploiting data analytics, capable to inform and assist them in reducing energy use, enhancing occupant satisfaction, comfort, safety, and facilitating a proactive approach to building operations and maintenance [1–4]. Such solutions belong to the rapidly evolving family of tools that monitor, analyze, and control building energy use and system performance also called Energy Management and Information Systems (EMIS) [3,5]. Such tools are categorized in three main groups: (i) *Energy and Information Systems (EIS)* which focus on meter-level monitored data to perform tasks such as energy consumption forecasting, anomaly detection, advanced benchmarking, load profiling and schedule optimization of building energy systems, (ii) *Fault Detection and Diagnosis (FDD) systems* which are designed to automatically detect unpermitted deviation of a system operation from its usual and acceptable operating range, and finally (iii) *Advanced System Optimization (ASO)* which analyze Building Automation System (BAS) data and modify the control settings for achieving an optimized energy performance of building systems [3,5]. Commercially available EMIS solutions have shown to be effective in reducing energy waste during building operations, with a median savings of 9% reported in Kramer et al. [5]. Meanwhile, a substantial body of academic work has proposed innovative data-driven algorithms that promise even better performance in identifying faults [6] and optimally controlling building systems [7]. Unfortunately, these advanced algorithms have largely not been implemented in today's EMIS tools [3,8,9]. The reasons for this lack of technology transfer are not well understood, but the scientific literature has been primarily focusing on developing applications that enhance the performance, accuracy and robustness of innovative applications while very few studies consider practical implementation, deployment and scaling factors [10,11]. Key areas that need further research include:

- *Perform more tests of the applications in real-world scenarios*. Often, novel algorithms and methods are developed and tested in controlled or constrained environments, raising concerns about their scalability and portability when applied to different system settings [12]. A recent review by Andersen et al. [13], highlighted that only 4% of the studies related to automatic FDD processes for build-

* Corresponding author.
E-mail address: marco.piscitelli@polito.it (M.S. Piscitelli).

https://doi.org/10.1016/j.enbuild.2024.114802
Received 17 July 2024; Received in revised form 21 August 2024; Accepted 9 September 2024
Available online 13 September 2024
0378-7788/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

ing energy systems have been implemented in real-world scenarios. This is largely due to the complexities involved in developing an EMIS application that effectively addresses all the practical problems of a site throughout its life-cycle.

- *Address integration of applications with other systems and software tools*. In field implementations, new EMIS applications must be integrated with existing proprietary communication protocols, legacy building automation systems, IT infrastructure and possibly other EMIS tools [14,15]. Researchers often overlook these aspects, focusing on standalone and monolithic applications rather than integration, which can be a barrier for wider adoption [16].
- *Develop efficient approaches to wrangling data and metadata*. In real buildings, essential operational data may be missing due to inadequate sub-metering, or insufficient data resolution, making implementation unfeasible [17,18]. Additionally, even if data streams are available, they may be unstructured and difficult to interpret, requiring significant effort and expertise to map them to a standard naming convention or metadata schema. [16,19,20]. Identifying efficient methods to address these challenges is essential to facilitate the adoption of these methods by industry.
- *Identify practical methods to deploy Artificial Intelligence (AI) and Machine Learning (ML) pipelines* The majority of the recently proposed applications in the literature rely on Artificial Intelligence (AI) and Machine Learning (ML) techniques able to automatically draw inferences from patterns in the analyzed data [6,21,22]. These techniques require a (i) continuous integration of streaming data, (ii) online model re-training and updating and (iii) dynamic parameter settings. Monolithic ML pipelines, are not well suited to be integrated in field deployments [23] and may require significant effort, expertise and may be difficult to scale [24,25].
- *Evaluate costs and savings*. The upfront and maintenance costs of implementing EMIS applications are frequently neglected in academic research [9]. At the same time insufficient information related to potential savings of these new tools and processes still hinders EMIS market transfer [8]. It is essential to develop cost-effective solutions that are easy to set up and replicate, taking into account the hardware, software, and human resources needed for their implementation. Additionally, potential savings should be evaluated for different common scenarios.

By integrating these considerations into the early stages of design, researchers can develop, share and test EMIS applications that are not only technologically innovative but also practical and ready for being validated in relevant environments supporting their widespread adoption. This holistic approach enhances the potential impact of research by ensuring that the conceived innovative EMIS applications are viable, desirable, and usable in real-world scenarios, harmonizing research output with market expectations. Considering that an EMIS application can be ideally implemented and deployed at any point of the building's life cycle, the key challenge is then to develop modular applications that are easily integrable with existing BAS and monitoring infrastructure, but at the same time are independent from site specific constraint (e.g., naming conventions) while also being maintainable and flexible enough to adapt to different building system configurations [12]. Achieving this balance requires careful consideration of modular design principles, standardization efforts in data handling, and interoperable frameworks. In this context, *modularity* refers to the capability of decomposing an application behind an EMIS solution into smaller, self-contained components that can be easily replaceable or upgradable without disrupting the overall system and each one performing a single specific function. This modular approach facilitates easier maintenance, scalability, and customization, as individual components can be developed, tested, and tailored independently. Additionally, a modular design fosters collaboration among development teams and facilitates the integration of applications into a unified, multilevel EMIS system. This approach enables seamless information exchange between tools and the reuse of existing application modules across different projects, ultimately leading to a more efficient and cost-effective implementation process. Such strategies are essential for ensuring smooth in-field deployment and operation of applications, thereby enhancing their usability and effectiveness across diverse scenarios.

For example, when a new measured variable is available to be included within a FDD workflow, several adjustments may be necessary to ensure effective utilization of the additional information. Firstly, the FDD algorithm may need to be modified to incorporate the new variable into its analytics processes. This might involve updating rules, thresholds, or models to account for the new data input. Furthermore, the integration of the new variable into the overall FDD framework should be carefully managed to maintain the integrity and reliability of the analysis. In a monolithic application, where components are tightly coupled, the inclusion of a new variable could necessitate significant reconfiguration of the entire system, from data collection to analysis implementation. However, in a modular and decoupled approach, where components are independent and interchangeable, integrating new data points can be accomplished more efficiently with minimal disruption to the existing system. This modular approach enables analysts to compose a tailored process that suits the specific requirements of the case study, thereby increasing agility and adaptability to evolving needs.

In this context, the objective of this paper is to contribute in answering the following research questions:

- Can EMIS applications be deconstructed into smaller, self-contained modular components, each dedicated to specific processes and seamlessly replaceable, in order to enable their deployment through a single comprehensive modular framework? For example, using the same framework for both a simple rule based application and a complex online data-driven machine learning pipeline.
- In developing a portable EMIS application, where should developers address the complexity of handling heterogeneous data and metadata? Should this complexity be managed within the application logic, at the interface between the application and metadata, or is it entirely delegated to the building metadata schema modeler?

To this aim, the present work introduces a novel development framework that can help in facing, in the robust and general way as possible, issues arising at the different stages of an EMIS application life-cycle. Together with a theoretical discussion about all the methodological steps and concepts addressed by the proposed framework, a practical case study is also analyzed. Specifically, the introduced framework was applied to develop and implement an ML-based anomaly detection application designed to identify electrical power production anomalies in Photovoltaic System (PV). The PV systems utilized in the field experiment are located in the university campus of Politecnico di Torino, Italy. The application was deployed and tested against a series of tests, such as the substitution of the estimation model and the change of naming conventions, to validate the capabilities of the proposed framework.

The paper is organized as follows: Section 2 provides background related to the EMIS application life-cycle concepts and related issues, Section 3 describes the proposed application framework, Section 4 details its implementation in the real case study. Finally, Section 6 and 7 discuss the results obtained and offer concluding remarks on the work.

2. Related work and contributions

The life-cycle of an EMIS application is a multifaceted process that unfolds in several incremental stages, as shown in Fig. 1. These stages include:

- *Development*: This initial stage defines the methodology and how the algorithms are combined in the analytic pipeline. It is the foundational step where the theoretical aspects of the EMIS application are formulated.

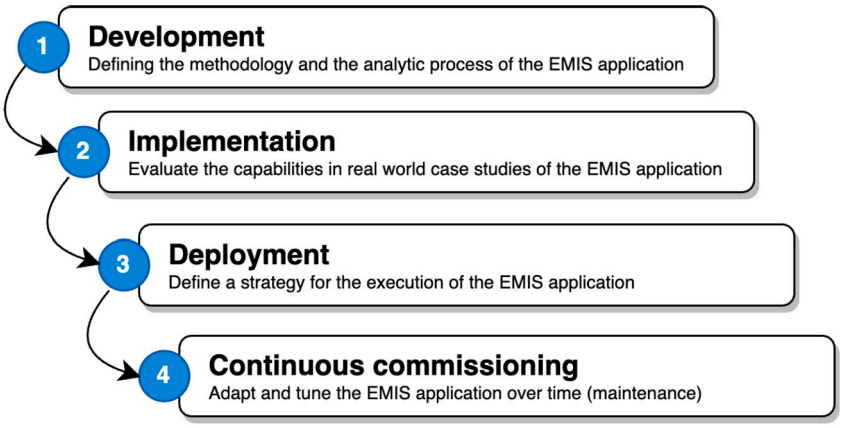

Fig. 1. Life-cycle steps of Energy Management and Information Systems (EMIS) application.

- *Implementation*: Consists in the evaluation of the proposed solution through real-world case studies. This phase is crucial for testing the validity and feasibility of implementing the EMIS application in real operational environments.
- *Deployment*: Once the solution has been implemented and evaluated, the focus shifts to deployment. This stage entails the conceptualization and execution of a deployment strategy, ensuring that the EMIS application is effectively integrated into the existing systems and processes.
- *Continuous Commissioning*: The final stage involves ongoing performance monitoring and adaptation to evolving boundary conditions, metering infrastructure, and system changes. Continuous commissioning ensures that the EMIS application remains effective and efficient over time, responding to new challenges and opportunities.

Each stage is critical to the overall successful operation of the EMIS application, ensuring that it not only meets initial requirements but also adapts to changing conditions and continues to provide value throughout its life-cycle [26].

In this section, the related works are structured following the EMIS application life-cycle, highlighting the challenges and constraints encountered at each stage and examining how existing literature addresses them by introducing potential solutions.

2.1. Development

The development of EMIS application in the literature has predominantly focused on demonstrating their effectiveness through simulations or off-line tests. In fact, a comprehensive understanding of EMIS performance under various operating conditions necessitates extensive testing and evaluation on well known datasets and controlled boundary conditions. Usually, EMIS applications are developed upon datasets which provide a well-known ground truth and performance of such tools can be evaluated, alongside a systematic framework for result reproduction and evaluation [27]. This approach is transversal for all the different EMIS applications pertaining EIS, FDD and ASO solutions. As a reference example, simulation models of single duct VAV systems in Energy Plus and Modelica have been used by Chen et al. [28] to assess the short and long-term impacts of faults on building energy consumption. Additionally, a simulation framework was introduced by Chen et al. [29] to facilitate the evaluation of FDD tool effectiveness and the assessment of fault effects in fan coil units. Recent studies have emphasized the provision of detailed and comprehensive datasets to enable researchers to thoroughly test and benchmark algorithms. Work such as ASHRAE research projects RP-1312 [30], and National Institute of Standards and Technology (NIST) 10D243 project [31], represent a pioneer work to provide labeled datasets to develop and evaluate advanced control and FDD technologies. More recently, Granderson et al. [32] released a first of its kind public dataset with ground-truth data on the presence and absence of faults that spans a range of seasons and operational conditions and encompasses multiple building system types. The data were created using both simulation models and experimental facilities providing a common ground for the development and testing of FDD algorithms, thus allowing for performance comparison. Simulated datasets and simulation environments offer a way for testing and fine-tuning not only FDD tools but also ASO solutions. For example, Building Optimization Testing Framework (BOPTEST) provides a simulation environment where developers can assess advanced controls on realistically modeled buildings [33,34]. This test-bed was successfully employed in many research papers for the implementation of new control strategies [35–38], comparison between different controllers [39–41] and assessment and evaluation of control sequences [42,43]. Furthermore, in addition to simulation-based approaches, many EMIS tools have been developed and tested on experimental datasets as well [44–46].

However, one major issue in the development phase is that many EMIS applications are not designed with practical implementation in mind and often fail to account for the complexities and issues that arise during actual building operations [10,47]. When field tests do occur, they are frequently limited in scope, which restricts the understanding of potential implementation issues on a larger scale. Furthermore, the design of many EMIS applications often neglects the practical needs and workflows of building managers and facility operators. This oversight results in systems that are difficult to use and do not integrate seamlessly into daily operations. Additionally, these solutions are not always designed to expand from small pilot projects to full-scale implementation across multiple buildings. Another significant barrier is the monolithic structure of many EMIS applications that makes compatibility difficult and increases integration costs, as any change requires extensive testing to ensure it does not disrupt other system components. To overcome these challenges, adopting modular and micro-services architectures can significantly enhance flexibility and scalability. These architectures make it easier to integrate with existing systems and scale as needed, and they simplify maintenance and updates, reducing downtime and the risk of disruptions.

2.2. Implementation

Implementation studies focused on evaluating the capabilities and real-world application of developed EMIS methods. For example, Gundersen et al. [48] collected more than 415 reproducible online machine learning workflow related to the creation of forecast models for metered building energy usage in the context of ASHRAE Great Energy Predictor [49] competition. Another example is the EMIS implementation enabled by Building Genome Project [50], an open dataset containing 3053 energy meters from 1636 non-residential buildings across North America

and Europe, that allowed developer to test the scaling of developed EIS solutions on real building data [51–53]. A considerable amount of literature has been produced around FDD tools and their application in real case studies, ranging from data-driven [54–56] to rule based methods [57,58]. The typical FDD tools are often integrated as a layer on top of existing BAS, operating in an open-loop manner by providing input to building operators and being adjusted through human intervention. To bridge this implementation gap, some studies have proposed more complex implementations involving two-way communication between the FDD solution and BAS data for read-write operations. For example, Pritoni et al. [59] implemented seven auto-commissioning FDD algorithms in commercial platforms to automatically correct faults and improve the operation of large HVAC system without the intervention of human operators. When it comes to implementation of EMIS application that interact BAS, like FDD auto-correction or ASO [60–62] interoperability issues arise, mainly related to the absence of transparent and standardized naming conventions for data streams and measurement points within proprietary equipment and BAS [63].

Contextualizing data in relation to building operations through semantic schemas has proven to be an effective approach for addressing such implementation challenges [1]. Semantic metadata schemas standardize the meaning of data communicated over a building's extensive network of sensors, abstracting and representing the building and its energy systems using graph structures. Relationships between points and systems are encoded, providing standardized descriptions of the physical, logical, and virtual assets within buildings, along with relationships between these assets [64,63]. A recent review identified more than 40 metadata schemas designed to represent and handle metadata across different building life stages and building services such as HVAC and lighting [65]. Among these metadata schemas Brick Schema is the one gaining significant traction in the smart buildings sector [20,66,63]. Brick is an application oriented metadata schema which incorporated the standard Haystack terms to define entities and relationships [67]. This ontology represents buildings metadata as directed labeled graphs consisting of triples (i.e., subject, edge, and predicate) and adheres to the Resource Description Framework (RDF) data model [68,64]. Subjects and predicates represent node entities (e.g., `Temperature_Sensor`), while edges denote relationships (e.g., `isPartOf`). This schema has been successfully applied in various scenarios, including automatically inferring the nature of unknown BAS data points [69], creating digital representations of AHU control systems [70], implementing rule based FDD [71–73] and enabling advanced demand response strategies [74].

Although addressing and overcoming interoperability issues during implementation is crucial, it is only the first step. When such approaches are applied across heterogeneous case studies, portability issues arise [11]. In the context of EMIS, an application can be defined *portable* if it is able to effectively be implemented on different buildings regardless on the specific data naming convention used, energy systems configurations and so on [24]. Ultimately, portability defines the generalizability of the application across heterogeneous buildings. In this context, some works have attempted to standardize various implementations into a formal framework [66]. In their study, Fierro et al. [75] present an initial attempt to formalize the characteristics of portable applications. They conceptually divide portable applications into five components: (i) qualify, (ii) fetch, (iii) clean, (iv) analyze, and (v) aggregate. These components respectively address metadata requirements, data acquisition, data preprocessing, algorithm execution, and data presentation. The paper illustrates how various EMIS portable applications, ranging from FDD to benchmark applications, can be constructed, tested, and evaluated using Mortar, an open test-bed encompassing over 90 buildings and more than 9.1 billion data points. The primary contribution of this work lies in demonstrating that by formally defining the components of portable applications, it is possible to abstract the application from site-specific implementations. This formalization enables the adaptation of applications to different scopes while maintaining their portability characteristics. More recently, some works have attempted to introduce field-specific portability frameworks, such as those for FDD and ASO. Mavrokapnidis et al. [71] introduced a portable programming model based on Brick ontology for rule-based FDD. This approach specifically examines the separation of FDD application logic (i.e., algorithm development) from configuration with specific data inputs, eliminating the need for rule authors to explicitly query metadata models. This enables the application to self-configure and execute across different building configurations. By executing AHU Performance Assessment Rules (APAR) in two different buildings, they demonstrated its potential to reduce manual and repetitive tasks for rule developers, thus accelerating the widespread adoption and scale of FDD applications. Nehasil et al. [76] combined a rule-based FDD tool, semantic data description and cloud architecture, to create a generizable FDD system that interacts with BAS and SCADA systems. Similarly, de Andrade Pereira et al. [74] presented a real-time implementation of supervisory controls for demand flexibility in buildings. Their approach was able to deploy different strategies in various virtual and real buildings, demonstrating the system's versatility and effectiveness in managing demand flexibility across different environments. All the presented works illustrate some solutions to the challenges encountered when transitioning from development to field case studies. Once these implementation issues are addressed, the deployment process can be carried out.

2.3. Deployment

When deploying EMIS applications, understanding the differences between *static* and *dynamic* deployment strategies is crucial, as each has unique characteristics, advantages, and limitations. *Static* deployment involves implementing and configuring the EMIS applications in a fixed environment with a predefined setup that remains relatively unchanged over time. In this approach, system configuration, including ML models or AI-based control agents and their parameters, is established during the initial setup and remains constant. The system is not designed to adapt dynamically to changing conditions or new data without manual adjustments. For instance, in static deployment, FDD tools are initially configured with a set of predefined rules, algorithms, and parameters tailored to the specific operational conditions of the system [77]. This setup typically involves thorough analysis of operational data to establish a baseline for normal operation. Once the baseline and detection rules are set, the system continuously monitors operations, comparing real-time data against established norms to identify deviations indicative of faults. However, the static nature of this deployment means that the system does not automatically adapt to new conditions or data without manual reconfiguration [78]. Thus, these static techniques do not leverage new data, and updates to FDD algorithms or detection rules require manual intervention by technical staff or service providers. In contrast, *dynamic* deployment of EMIS applications in smart buildings is an ongoing process due to constantly changing operating conditions and varying system characteristics. Over time, the relationship between the input and output of models underlying an EMIS application can change, leading to inaccurate predictions, estimations, or control signals from previously trained models. This issue, known as *concept drift* or *data drift* [79,23], is a primary cause of performance degradation in data-driven models. Concept drift can occur in several forms: *sudden* drift (e.g., sudden machine failures), incremental or *gradual* drift (e.g., changes in occupant behavior), and *recurring* drift (e.g., seasonal meteorological conditions and working patterns).

To counteract such events, EMIS applications often rely on deployment strategies that exploit continual model updates or retraining processes [78]. Two prevalent methods for updating models are *accumulative learning* and *incremental learning* [23]. Accumulative learning involves fine-tuning or retraining a model using both historical and new incoming datasets. An example of this approach is reported in Coraci et al. [80], where an online accumulative retraining strategy is applied to a deep reinforcement learning controller. This technique adapts to new operating conditions but can be computationally expensive. Ide-

ally, the model would learn from new data as they become available, without the need for retraining. This is the approach followed by the incremental learning strategy, which updates the model using only new incoming data. An example of incremental learning is presented by Fekri et al. [78], where an innovative approach for electrical load forecasting is used to update the ML model weights online according to new data and quickly adapt it to new patterns without requiring periodic retraining. These approaches allow learning new patterns without forgetting historical behavior. However, in some cases, accumulative and incremental retraining can lead to poor model accuracy due to changing operating conditions, maintenance, or upgrades. In such scenarios, the training data may become unbalanced, favoring incorrect system operation, necessitating forgetting old behaviors in favor of new ones. This approach is studied in Yang et al. [81], where two adaptive models are proposed: an accumulative training technique and the sliding window training technique. With sliding window training, the size of the dataset used for training remains constant: as new data are added, the oldest data are dropped. Although this approach makes the model "forget" old data, it can be beneficial in situations where old data may be misleading or reflect anomalous or significantly different operating conditions.

Thus, the primary goal of model updates is to balance acquiring new knowledge with retaining prior knowledge. The effectiveness of a model update or replacement is highly influenced by the application framework and how the application is deployed on the field. In this sense, a modular approach allows developers to build EMIS applications in discrete, self-contained units, each addressing specific functions. This architecture facilitates easier updates and maintenance, such as replacing individual components (e.g., prediction models, control agents) without disrupting the entire system. This is particularly important for model retraining or replacement, as it minimizes downtime and reduces the risk of performance degradation. By isolating changes to specific modules, the overall stability and reliability of the EMIS application are maintained, ensuring continuous and efficient operation.

2.4. Continuous commissioning

Continuous commissioning of an EMIS application is an ongoing process that ensures the deployed application operates optimally throughout its life-cycle [82]. This stage involves regular monitoring, updating, and tuning of the application to adapt to changing conditions and incorporate possible new advancements. However, several potential problems can arise during continuous commissioning if certain aspects are not properly addressed.

One significant issue, that often characterize the building and its energy systems, pertains the inconsistency in naming conventions of variables over time. If the naming convention of a variable is changed without proper documentation and communication, it can lead to confusion and errors in data ingestion and interpretation. For instance, if a temperature sensor variable is renamed from `TempSensor1` to `TS1` without updating all references across the application, the system may fail to recognize and process this data correctly. This inconsistency, even though it may seem trivial, can disrupt the analytic pipeline affecting the performance and reliability of the EMIS application. In such a context, without proper metadata that describes the meaning and context of data points, integrating and interpreting new data becomes challenging. Thus, semantic enrichment is essential for ensuring that the EMIS application is able to properly interpret the data in a consistent way. For example, if a new sensor is added to the monitoring infrastructure but it is not integrated in the existing data model, the EMIS application might not effectively use this information in an automatic way. This can prevent the system from leveraging new capabilities and limit its overall functionality. By relying on a properly defined semantic data model, it is possible to query metadata, enabling the retrieval of formalized points and relationships across various data sources without being bound to site-specific variable naming conventions. This decouples the application's internal naming convention from the site-specific naming convention, facilitating more flexible and interoperable data querying [74]. Although having a metadata model allows developer to separate the application logic from the site specific naming convention, its proper definition still requires a deep understanding of building subsystems and a direct knowledge of concept relations and taxonomy of the metadata schema [2]. Despite different methodologies have been introduced to consistently build semantic models from predefined templates [63,83], the creation of such models remains challenging [71]. Given the absence of guidelines in building metadata schema, the same building or systems can be modeled in different ways according to the modeler experience and knowledge. In the scenario in which alternative models of the same building may exist, it is difficult to write an application that is flexible enough to adapt to a set of possible different graph structures. As a result configuring a portable application is not a trivial task and its interface with metadata schema must be flexible enough to adapt to different or changing configuration of the same system in the metadata schema. Thus, development complexity shifts from handling the heterogeneity of data within the application logic to handling the application configuration against different building models. Eventually, during the continuous commissioning phase, effortless changes to pieces of the EMIS application are essential for maintaining it as much as possible robust and adaptable. Even in this case, the absence of modular application frameworks make these changes time consuming and error-prone de-facto requiring extensive re-coding and testing.

2.5. Contributions of the paper

As emerged from the literature review, despite in the last years the focus of the scientific community has been on the definition of novel and advanced analytic processes to be embedded into EMIS solutions, new aspects related to the portability, interoperability, and maintainability of such solutions are becoming as well relevant. Although implementation issues require site-specific answers, providing an all-in-one solution may limit the application to a specific technology or use case. To develop truly portable and widely applicable applications, it is necessary to abstract from site-specific constraints and build a larger framework that can be adapted on a case-by-case basis [24]. This involves creating frameworks that address each step of the application development process with portability in mind, but without being overly specific. In this context, the present paper introduces a novel and robust framework to develop, deploy and maintain portable applications in buildings. The contributions that it brings can be summarized as follows:

- Introduction of an interoperable and portable framework to build EMIS applications that allows to decouple the handling data and metadata from the application logic, leveraging Brick metadata schema;
- Demonstrate how real-life implementation issues also pertaining the deployment strategies of EMIS applications, can be faced while keeping each step of the data analytics pipeline modular and independent;
- Creation of an open-source implementation of the framework through a Python package.[1]

3. Methodology

The proposed methodology is intended to provide a modular and standard framework for researchers and practitioners to speed up the implementation of EMIS applications in real world case studies. In this context, an *application* is intended as a piece of software that is able to query a metadata model of a building in order to configure its operation, retrieve data, verify constraints and execute analytics processes

[1] Code and Python Package available in the following GitHub repository https://github.com/baeda-polito/portable-app-framework.

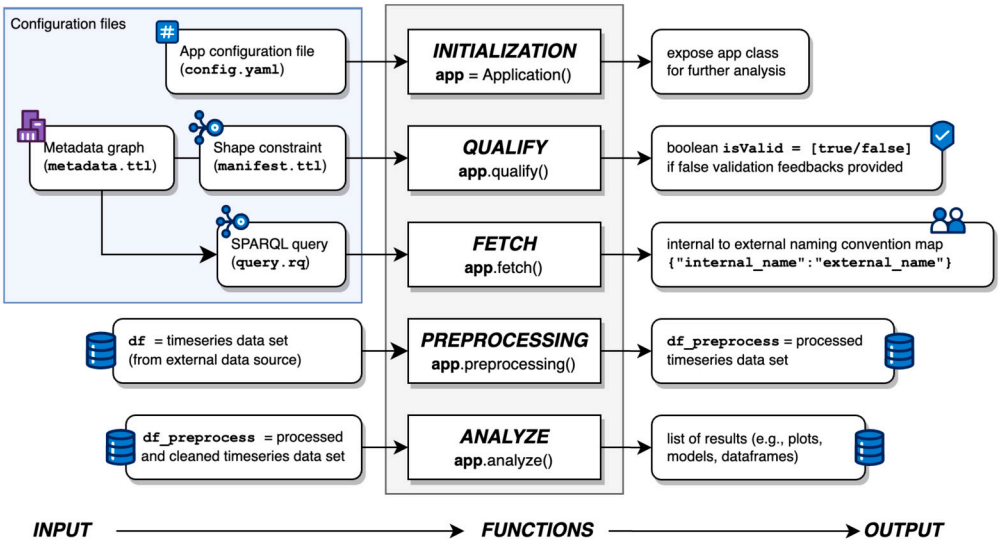

Fig. 2. Overview of the portable application framework in terms of functions exposed, input and output.

[83]. The proposed framework inherits the main concepts of portable application from Fierro et al. [75] and, as shown in Fig. 2, consists of a Python module exposing a set of functions that provide a code-level interface between the EMIS logic implementation and the management of data and metadata from the field. The available functions are the following:

- *Initialization*: function that gathers the necessary information and translate it into a set of formal structured files;
- *Qualify*: function that assess the minimum requirements for the application to run and provides feedback if data and metadata constraints are not satisfied;
- *Fetch*: function that performs query on the building metadata schema and retrieves the required variable names;
- *Preprocessing*: function that performs application specific data preprocessing steps (e.g., data normalization, data transformation, data cleaning) before executing the analyze step;
- *Analyze*: function that contains the application core logic, runs the algorithm, model or analysis on the provided data sets

These functions aim to create a middle-ware that separates the complex development of EMIS logic - which demands significant expertise in the energy and building field - from the essential but time-consuming task of data integration and contextualization. As a result, this framework allows the developers to structure an application which is modular and maintainable along each step of the EMIS application life-cycle. In the following paragraphs an overview of the application functions is presented and an example of a typical execution flow is provided.

3.1. Initialization of the application

The initialization of the application can be divided into two main steps: (i) collecting required information, and (ii) organize it into structured files. The information needed for structuring the application varies, but the basic requirement is knowledge of the case study's monitoring infrastructure and data acquisition strategy. For example, when developing an FDD application, it is essential to understand the HVAC configuration, available sensors, and how to access sensor data. Additionally, it is necessary to define the requirements in terms of installed sensors, data availability, and parameters for the application to run. For instance, an FDD algorithm might need a mixed air temperature sensor measurement with 5 min aggregation to operate a rule-based algorithm which requires a threshold to be set (e.g., 2 °C temperature error tolerance). Such quantitative information must be translated into machine-readable files and organized into a formal structure that facilitates straightforward implementation of the application. In the proposed framework, information related to the building monitoring infrastructure is translated into a Brick schema contained in the `metadata.ttl` file. Then, the application's metadata requirements are translated into specific formats that allow formal definition of constraints and queries: `manifest.ttl` and `query.rq` file respectively. After defining data and metadata requirements, it is possible to proceed with parameter specification, which will be encoded in a configuration file called `config.yaml`. An example of these files is shown in Fig. 3. Structuring the application using such approach is fundamental to guarantee modularity and standardization for developers, thus interoperability and usability for the user. In fact, the introduction of a configuration file is a powerful tool that allows the application developer to change critical parameters with minimal effort. Once the algorithm is built and tested, parameter tuning or sensitivity analysis can be performed by simply modifying this file.

3.2. Identification of application data requirements: qualify

Given that every EMIS application has different requirements in terms of data points and relationships between points, it is necessary to define if the available data and metadata are sufficient for the application itself to run. This concept is known in the literature as *semantic sufficiency* and means that a data model is considered complete only if it includes enough metadata to support the development of a specific application [83]. Referring to Fig. 2, this process of minimum requirement identification is performed though the definition of a *qualify* function which accepts as input the building semantic model and the required application metadata, expressed as application constraints, and provides as output a feedback if the constrains are respected or not. The constraints are contained in the so called *manifest* which captures the description of the application requirements to be included in the analysis. The manifest file (i.e., `manifest.ttl`) defines the requirements through Shapes Constraint Language (SHACL) shapes, a W3C standard language for validating RDF graphs against a set of conditions expressed in the form of an RDF graph called *shapes graphs* [84]. SHACL shape graphs can also be viewed as a description of the data graphs that do satisfy certain conditions and enable the automatic check of the required variables in a fully automatic way.

In Fig. 3, an example of a *qualify* function execution against two distinct metadata schema is depicted: the first schema (highlighted in yellow) includes a single mixed air temperature sensor (`MAT1`), whereas the second schema (shown in blue) comprises an outdoor air temperature

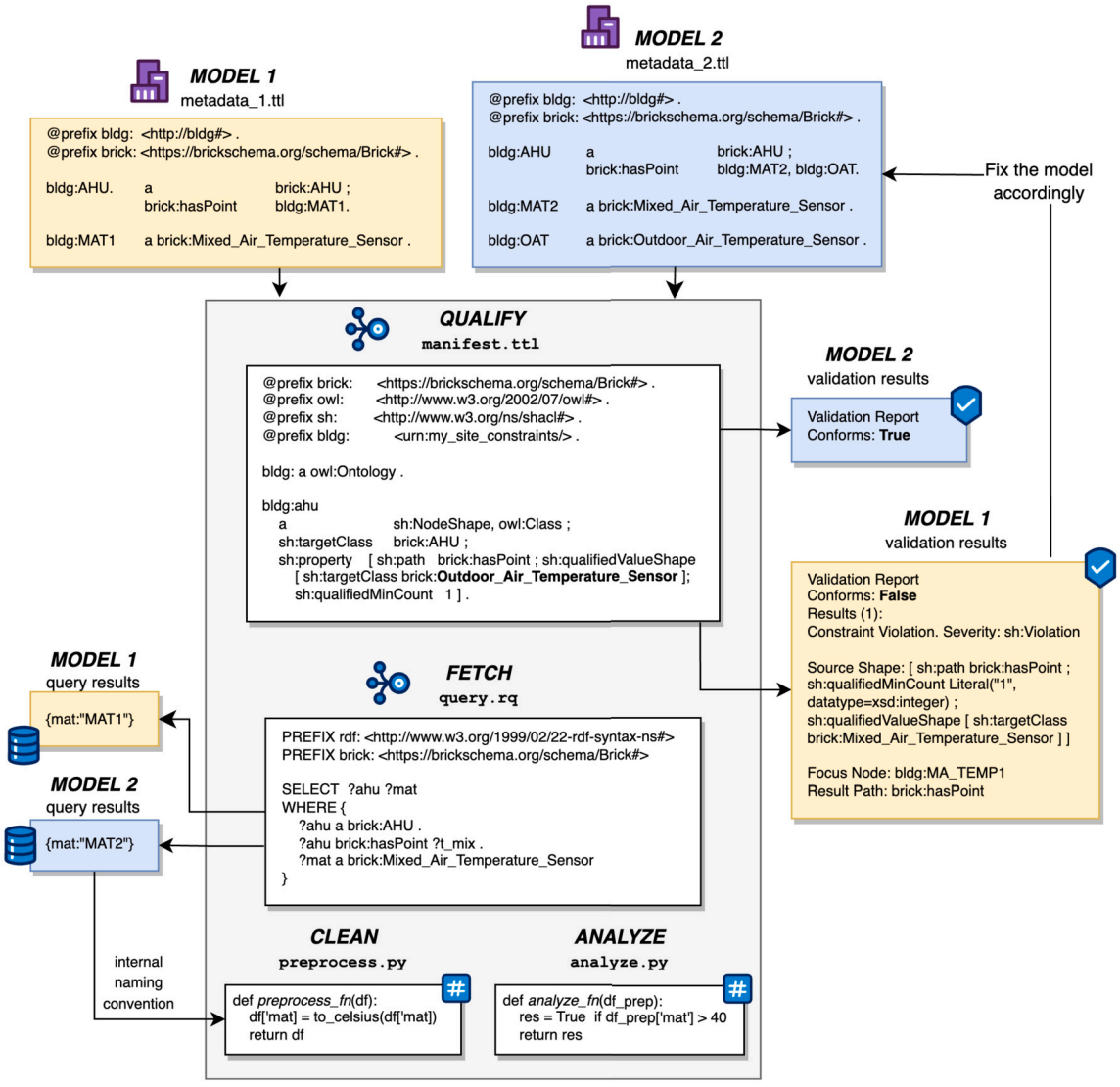

Fig. 3. Example of *qualify* and *fetch* functions execution on two different metadata models. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

sensor (`OAT`) and a mixed air temperature sensor (`MAT2`). By executing the *qualify* function using a manifest which requires the presence of an outdoor air temperature, for the second model the result is `TRUE` (the required sensor is present), conversely, in the first model the result is `FALSE` (the required sensor is not present). In the latter scenario, the function offers, through the use of a validation procedure, a log that assists the developer in potentially adjusting the metadata schema to align with the requirements, if feasible. This validation procedure is enabled by the integration of Building Metadata OnTology Interoperability Framework (BuildingMOTIF) which is a tool that helps storing, managing, and verifying collections of templates, shapes, and metadata models [83]. The integration of this tool enables the model validation and incremental model creation. By iteratively running the *qualify* function on a metadata schema that does not pass the validation and changing the constraint, it is possible to reach the proper configuration of the metadata schema required for the application to run. This procedure is extremely helpful in cases in which the building metadata schema is not available or is not properly configured, and may help the analyst to incrementally build it. From the opposite perspective, this method may help the analyst to understand if an application can be executed or not given a structured metadata model of an existing building or energy system and eventually stimulate an application review in order to make it more flexible against different possible configurations. As a result, it is possible to state that the *qualify* function allows both *app-to-model* and *model-to-app* development.

3.3. Identification of required variables and self configuration: fetch

The advantage of employing an ontology-based framework lies in the ability to identify the required data points for a given application through a graph query. By querying the point classes (e.g., `brick:Temperature_Sensor`) instead of querying a variable name it is possible to create applications that are independent from specific naming conventions. The fundamental concept of the *fetch* function is then to create a mapping between the “internal” variable names used in application development and the “external” variable names used in a specific building or energy system (e.g., data point names from BAS or databases). By using the “internal” naming convention throughout the application development, it is possible to easily decouple the logic from the “external” naming conventions, allowing the creation of portable applications that are independent of specific use cases. For instance, as shown in Fig. 3 in the *fetch* function box, a SPARQL query retrieves the name of the mixed air temperature from two building metadata schemas (i.e., data models), which will be internally referred as `mat`. In the first data model the mixed air temperature name is `MAT1`, while in the second

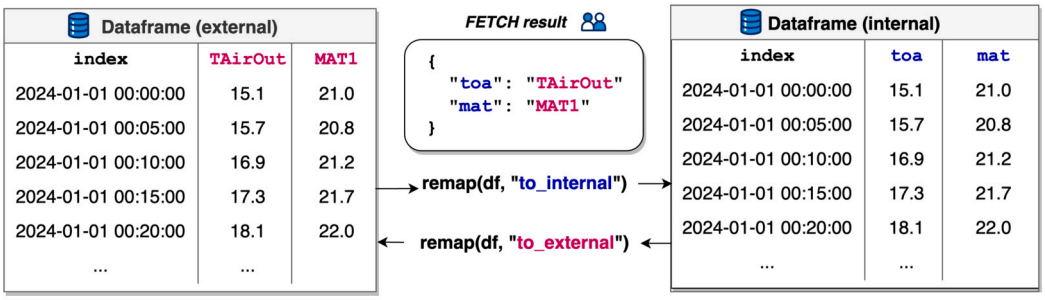

Fig. 4. Example showing how the results of the *fetch* function (i.e., dictionary containing key-value pairs) can be used to change the variables names of a data-frame from the external naming convention (red text) to the internal naming convention (blue text).

data model is `MAT2`. By doing so, the "internal" variable name `mat` will be associated to an "external" variable name through a simple key-value mapping. Understanding the mapping between the external and internal variable name is essential for retrieving data (e.g., from a proprietary database) and consistently refer to that variable within the application code-base. In order to facilitate the application developer to switch between internal and external naming convention, a helper function was implemented to convert the column names of a given data-frame from the internal naming convention to the external and vice-versa; this concept is better explained in Fig. 4.

Differently from other frameworks [75,74], the output of the *fetch* function is solely the mapping dictionary described above, leaving the retrieval of data to the developer. Given the variety of databases, communication protocols, and site-specific architectures, it was decided to let the user handle data retrieval. This approach has several advantages. First, the framework is not dependent on platform-specific data connections, which can be limiting for some users and may require different implementations depending on the use case. This avoids the maintenance of numerous connectors within the package, which can become quickly outdated or poorly tested, discouraging users from adopting the framework due to the need for custom implementations. Secondly, most of the time, such connectors are already available from the application developer's perspective and can be easily implemented. Finally, this choice was made to avoid narrowing the framework's scope of applicability to a specific use case, allowing for greater flexibility to be adopted in various scenarios (e.g., considering both ASO and EIS applications).

3.4. Application logic development: pre-process and analyze

The development of an EMIS application logic is by its nature various and difficult to standardize. However there are always two macro steps to perform: preprocessing and algorithm implementation. In general, building operational data preprocessing consists of five major tasks, i.e., data cleaning, reduction, scaling, transformation and partitioning [85]. The aim of this phase is to arrange the original data into suitable formats for various data analytics algorithms. To this purpose the preprocessing phase is implemented in the *preprocess* function which accepts as input the raw data-frame and return the data-frame in the required format for the analytics phase. The algorithm implementation is performed by using the *analyze* function. This function takes a pre-processed data-frame as input and produces an object containing any possible kind of outputs (e.g., plot, data-frame, boolean, string etc). Within this function, analysts have the freedom to employ any methodology or algorithm they choose, provided that the output results are encapsulated within an output object. An example of application logic development is shown in Fig. 3 where the *preprocessing* function performs a conversion of the mixed air temperature into Celsius degree and the *analyze* function implements a simple threshold rule. The aim of this example is to show that if the analyst uses the internal naming convention of the variables, as described in the *fetch* function, the *preprocessing* and *analyze* functions are completely independent from the specific naming convention on which the application is executed and the analyst can develop the application logic regardless to naming constraint or site specific conventions.

3.5. A reference execution flow of the framework

To demonstrate the implementation of the conceived portable framework, a typical implementation is reported in this section. As outlined in the pseudo-code (see Algorithm 1), the initialization involves a guided process conducted via a Command Line Interface (CLI), during which the necessary files are either created or collected. These files include: (i) a configuration file containing application details, description, and parameters, (ii) a SPARQL query containing required variables and internal naming conventions, (iii) a manifest file containing SHACL shape constraints, (iv) a metadata model for the building or system under investigation, and (v) a data source containing time-series data. While the first three files are automatically generated during the application initialization through the CLI, the metadata schema and time-series data source must be provided by the user in advance.

After the initialization the application execution phase is carried out which involves executing the functions to identify metadata requirements, to perform queries on the metadata schema, query data streams, to process data and to execute the algorithm itself. Ideally, all the discussed functions should be executed sequentially, with the output of one function serving as the input for the next. Initially, the *qualify* function assesses data requirements; upon successful execution, the *fetch* function returns a set of variable names. Subsequently, the corresponding time-series data can be extracted from the data source by the analyst, according to the data source specific modes (e.g., API, database query or simply csv file upload). Once the data are fetched and transformed into data frames following the application internal naming convention, the *preprocessing* and *analyze* functions can be executed.

While the execution phase may be presented as a series of connected steps, each function can actually be implemented independently, offering flexibility for different deployment options for each application. For instance, the flow described earlier could be viewed as a unified application flow from the data source to the final result. However, in certain cases, such as FDD isolation trees, the overall EMIS application can be seen as a sequence of rules (or applications), with each implementing a specific check. Here, the deployment option might resemble a chained execution of multiple portable applications. Conversely, for control applications, it might be advantageous to execute the *qualify* and *fetch* functions just once during the initial run, and subsequently execute the *preprocess* and *analyze* function repeatedly in a loop at each control loop iteration. These examples illustrate just a few deployment strategies for EMIS applications that the introduced framework can accommodate, but the potential applications are diverse and the proposed framework is able to handle potentially all the possible configurations.

4. Cases study

The framework was tested on a real-world case study through the development of an online anomaly detection application implemented

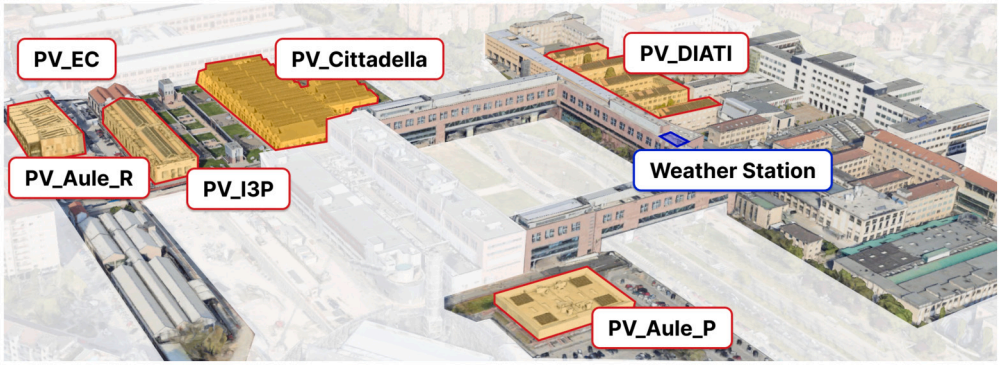

Fig. 5. Aerial representation of the PV plant installed at the Politecnico di Torino (PoliTo) campus. The PV arrays are highlighted in yellow while the weather station, which supports the acquisition of weather-related measurements, is highlighted in blue.

Algorithm 1: Pseudo-code of a typical application execution flow by applying the proposed framework: from initialization to logic execution.

```
init Application initialization
    Collect data and metadata (user);
    data ← tabular like data;
    metadata.ttl ← Brick metadata model;
    Create Application files (CLI);
    config.yml ← contains details and parameters;
    query.rq ← contains SPARQL query of required points;
    manifest.ttl ← contains SHACL shape;
qualify Assess minimum requirements
    validate metadata.ttl against manifest.ttl requirements;
    validate data against config.yml requirements;
    while valid is False do
        Fix metadata.ttl according to suggestions
    end
fetch Collect time-series data
    Query points names from metadata.ttl as defined in query.rq;
    Get points from data (e.g., API query);
preprocessing Pre-process time-series data
    Run preprocessing function on data (e.g., fill missing values)
analyze Implement application logic
    Run analyze function on data;
```

on photovoltaic systems. Specifically, the case study pertains the photovoltaic energy production of six Photovoltaic System (PV) systems installed at the Politecnico di Torino (PoliTo) campus, an Italian university located in the Piemonte region. Over the years, the campus has experienced various expansions, including the conversion of an industrial hub into classrooms and the addition of new buildings for research centers and laboratories. To meet the increasing electricity demand, the campus has been equipped with multiple distributed PV shown in Fig. 5, with each system installed on a different building. The arrays are made of about 3000 PV modules and more than 40 inverters. By 2023, the total installed capacity nearly reached 1 MW with an annual average production of approximately 1.3 GWh.

The installation of these PV systems took place at various times, with each system varying in peak power, array configuration, and associated monitoring infrastructure as summarized in Table 1. The `PV_Cittadella` array, with a peak power output of 604 kW_p, was installed in two phases during 2015 and 2016. It includes 1849 modules rated at 327 W each and 27 inverters. The overall electrical production is monitored by an electrical power sensor installed on the AC side of the main switch. Additionally, the array is equipped with sensors for outdoor air temperature and total global irradiance. The `PV_DIATI` photovoltaic array, installed in 2019, has a total capacity of approximately 183 kW_p. It comprises 487 modules rated at 360 W each and is equipped with 8

Table 1
Description of the installed photovoltaic systems at PoliTo with specification of the rated peak power and associated monitored quantities infrastructure.

Name	Peak power	Monitored quantities
PV_Cittadella	604 kW_p	• Outdoor air temperature • Solar irradiance • Electrical power
PV_DIATI	183 kW_p	• Electrical power
PV_Aule_P	50 kW_p	• Electrical power
PV_Aule_R	47 kW_p	• Electrical power
PV_EC	47 kW_p	• Electrical power
PV_I3P (East + West pitch)	31 kW_p	• Outdoor air temperature • Solar irradiance • Electrical power

inverters. An electrical power sensor monitors the total electrical production on the AC side of the main switch. The `PV_Aule_P` array, installed in 2018, has a total capacity of approximately 50 kW_p and consists of 144 modules rated at 345 W, along with 2 inverters. This array is equipped with an electrical power sensor on the AC side of the switchboard to measure total production. The `PV_Aule_R` plant is the newest installed system (2021) and has a total capacity of approximately 47 kW_p. It includes 117 modules rated at 400 W each and 3 inverters. An electrical power sensor on the AC side monitors the total electrical output. The `PV_EC` array, installed in 2016, has a total capacity of approximately 47 kW_p and consists of 154 modules. Differently from the other PV systems, which are primarily roof-mounted, the modules in this array are integrated into different parts of the building: 80 modules on the external facade of the stairwells, 34 modules on the roof of the central building, and 40 modules on its transparent facade. The system is equipped with 4 inverters, each of them has its own electrical power meter. The total electricity production is calculated as a virtual sensor by aggregating the output of each inverter. Finally, the `PV_I3P` plant, installed in 2009, consists of 140 structural glass modules with a glass-glass design and a 15 mm Argon chamber, providing a power output of approximately 31 kW_p. This array includes 2 inverters, each with a capacity of 15.9 kW. It is also equipped with an electrical power meter on the AC side, a solar irradiance sensor, and an outdoor air temperature sensor.

To capture the diversity of systems and monitoring infrastructure configurations, the entire PV plant on the campus has been described in a metadata model using Brick schema (v1.4), and its visual graph representation is shown in Fig. 6. Each PV system, represented as a `brick:PV_Generation_System` equipment, was modeled by following a top down approach: starting from the definition of its geographical

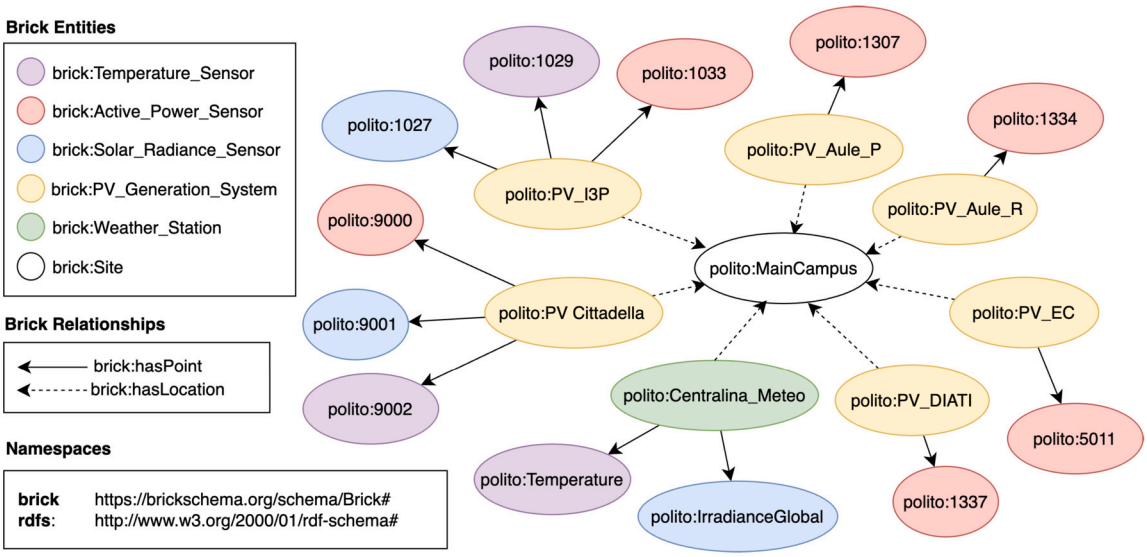

Fig. 6. Graphical representation of the metadata model (i.e., Brick schema) of the monitoring infrastructure of PoliTo campus. The picture shows a subset of the overall graph and represents the PV plant, the weather station and the associated sensors. The colored nodes represent Brick entities while the arrows represent relationships between the nodes. Each color and line type has a different meaning, as explained in the legend.

coordinates (i.e., latitude and longitude) and relative location in the campus to the characterization of the specific monitored data-points, modeled as entities belonging to the `brick:Sensor` class. This semantic representation standardizes the logical and physical relationships in a machine-readable format, enabling the automatic querying of necessary metadata and data points for analysis. Complementing the monitoring infrastructure, the campus is also equipped with a centralized weather station that includes sensors for outdoor air temperature, solar irradiance (global, horizontal, and diffused), outdoor air relative humidity, and wind speed.

Regarding data acquisition and storage, all the sensors are physically connected to data loggers that sample and process raw data before storing the measurements in a Timescale time-series database [86]. In the analyzed case study, raw data is typically sampled at 1 min intervals; however, the sampling frequency may differ depending on the sensor type and manufacturer. For instance, the installed outdoor air temperature sensors record data with a timestamp of 15 min. Additionally, due to communication and processing delays, the measurement timestamps may not always align precisely [87]. To maintain a consistent time interval for analysis and ensure the statistical reliability of the data, the Timescale database's ability to query and aggregate data into specific time buckets was used. This approach enabled querying and aggregating data, using the mean value as the aggregation function, to ensure all measurements were effectively aligned with the common minimum available aggregation interval, which in the specific case is 15 min.

Finally, a front-end tool was made available for real-time monitoring and visualization of collected data. The entire IT infrastructure, along with the capability to deploy custom analytic applications (e.g., anomaly detection tool for PV systems), is supported by an internal open-source cloud, based on Kubernetes open-source software [88].

4.1. Application of the portable framework in the case study

To investigate how the portable application framework could be employed to deploy and adapt an anomaly detection application for the case study described above, we defined and created five tests, between March 2023 to June 2024. These tests, represent realistic scenarios that might occur during the life-cycle of an application and are illustrated in Fig. 7. These tests, along with the objective in demonstrating the framework, are summarized as follows:

1. **Development of the ML pipeline**. After the complete setup of all the hardware and software services required for real-time monitoring (e.g., installation of data-loggers and creation of the monitoring database) an ML pipeline was defined to perform the anomaly detection task on one PV system of the campus. The process involved the training of an Multi Layer Perceptron neural network that estimates the PV production. The model was then statically deployed (i.e., without retraining) and the residuals between the actual and predicted electrical energy production were used as a proxy for detecting anomalous production patterns. The objective of this test was to demonstrate how the framework can be used to develop and deploy ML pipelines.
2. **Substitution of the estimation model**: After the deployment of the first model (i.e., the MLP neural network) a second model was developed (i.e., an LSTM neural network) considering the power production of the same PV system as a target. Since it is likely that new algorithms and models will emerge in the future, this test aimed at demonstrating how the framework enabled a straightforward substitution of the model during the online execution of the application without compromising it. Together with the estimation model also the *pre-process* functions were updated accordingly.
3. **Optimization of the deployment strategy**. Once the LSTM model was statically deployed (substituting the MLP neural network) the deployment strategy of the model was optimized. The goal here was to demonstrate that the framework made it possible to periodically modify the deployment strategy of the application introducing a scheduled bi-weekly retraining of the LSTM model by simply changing the application configuration file.
4. **Change in naming convention and data availability**. This test was artificially inducted during the execution of the application in order to test the robustness of the application to changes in input data. Specifically, the naming convention of the variables was suddenly modified and one sensor required for the execution of the model, was disconnected, simulating an unexpected hardware failure or maintenance event.
5. **Extension of the application pipeline to other systems**. The last test pertained the transferring of the application pipeline from one PV system to all the other systems installed in the campus. The goal was to demonstrate how the framework enables portability to other similar, but not identical systems and to test its robustness to dif-

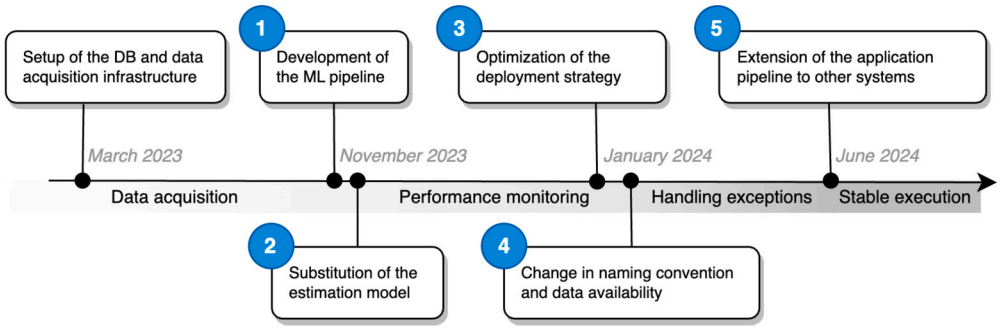

Fig. 7. Timeline of implementation and subsequent tests performed on the EMIS application for PV anomaly detection and diagnosis leveraging the portable application framework.

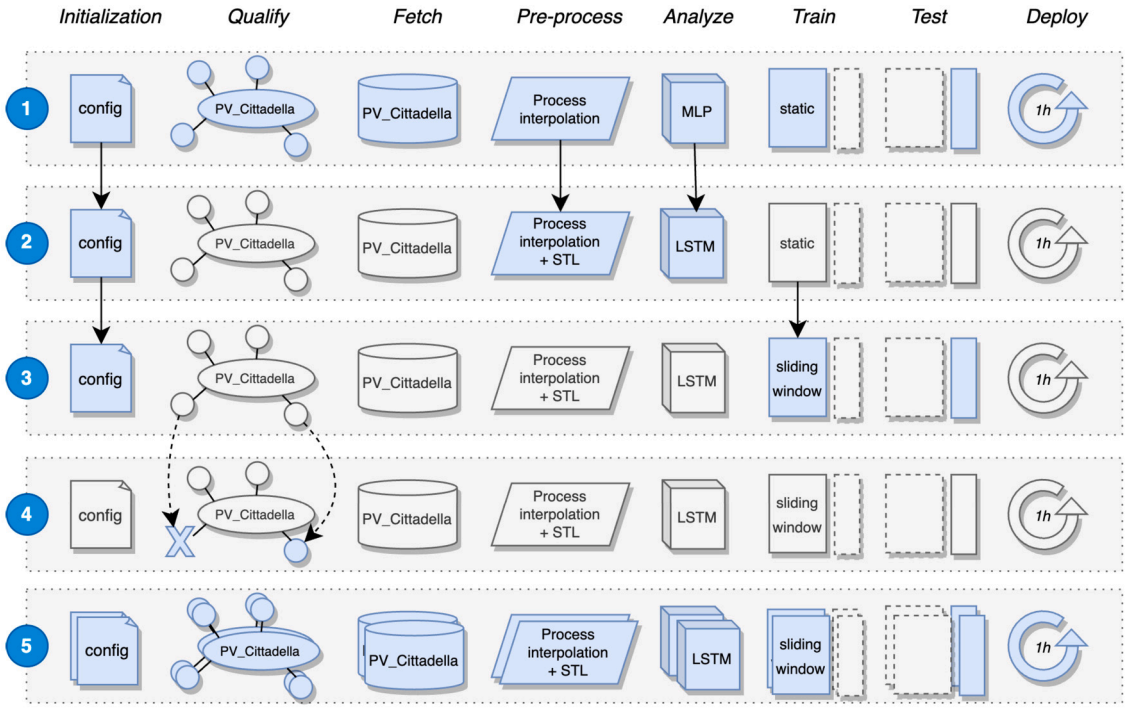

Fig. 8. Illustration of the impact of the five sequential tests represented in Fig. 7 on the components of the framework. Gray elements indicate those that remained unchanged from one test to the next, while blue elements represent those that changed.

ferent system configurations, naming conventions and operational characteristics.

The five tests are reported in a timeline form in Fig. 7 and are also graphically represented in the rows of Fig. 8 where in blue are reported the modules of the procedure affected by an implementation or a change, while in gray the modules that remained unaffected.

5. Results

In the following subsections the implementation of the portable application framework is presented by explaining how the previous described tests have been faced and the associated issues have been solved. In these subsections all the technical aspects related to the setup of the application, the employed models and the obtained results are detailed.

5.1. Development of the ML pipeline

The initial task involved the development of a comprehensive Machine Learning (ML) pipeline employing the introduced portable application framework, see "Test 1" in Fig. 8. This application performs an anomaly detection on the photovoltaic production of a single system by analyzing the residuals between the actual and the estimated power production given certain boundary conditions. The analysis requires the presence of a PV plant (`brick:PV_Generation_System`) with its geographical location, and equipped with an electrical power sensor (represented as `brick:Active_Power_Sensor`), alongside an embedded outdoor air temperature (represented as `brick:Temperature_Sensor`) and solar irradiance sensor (represented as `brick:Solar_Radiance_Sensor`). Such constraints were translated into a proper shape graph in the `manifest.ttl` file of the application. The model-building process started by focusing on the PV system named "Cittadella", where all these metadata requisites were met, ensuring that the *qualify* function would be successfully applied. In the specific case the active power sensor: solar radiance sensor and temperature sensor are identified through a unique ID. Since the sensors were properly structured in the metadata schema, it was possible to identify the required point names through the *fetch* function (`id-9000` `id-9001` and `id-9002`).

Using these IDs, the corresponding time series were retrieved from the time-series database, through an external API. The developed application was based on the training of a simple Multi Layer Perceptron (MLP) fully connected neural network designed for point-to-point esti-

Table 2
Summary of the main hyper-parameters of the MLP model.

Name	Value
Input layer	5 nodes
Hidden layer	256 nodes (Relu activation function)
Output layer	1 nodes
Optimizer	Adam
Learning rate	0.001
Loss function	Mean Square Error (MSE)
Epochs	50

Table 3
Summary of the main hyper-parameters of the LSTM model.

Name	Value
Input layer	5 nodes
Hidden layer	2 layers 16 nodes
Output layer	1 nodes
Optimizer	Adam
Learning rate	0.01
Loss function	Mean Square Error (MSE)
Epochs	100

mation of the electrical power production of the PV system. The input variables used as predictor variables were: (i) the measured outdoor air temperature, (i) the measured solar radiation, (iii) the azimuth, (iv) the zenith and (v) the global horizontal radiation calculated in clear sky conditions based on PV geographical location [89]. The model was trained and tested using data sampled every 15 min from April to November 2023 using 70% of data points for train and 30% for testing (such parameters were set in the application *configuration* file). Missing data and inconsistencies in the training dataset were handled through the *preprocess* function through linear interpolation if the number of consecutive missing values was lower than 4 (i.e., 1 h) otherwise records with missing values were dropped from the training set. After a min-max normalization of the dataset, the neural network model was defined in the *analyze* function of the application. The neural network structure consisted of an input layer of size 5 (as many nodes as the input variables required), an hidden layer with a standard number or 256 neurons and with ReLu activation function and an output layer of dimension 1. The Adam optimizer with a learning rate of 0.001 was used to update the model's parameters. The training process involved feeding the training data through the model, computing the Mean Square Error (MSE) loss, performing back-propagation to calculate gradients, and updating the weights. This process was repeated over 50 epochs. The MLP parameters were defined according to typical settings found in literature [90–92]. Table 2 summarizes the main hyper-parameters and settings for the described model while Table 4 summarizes the training and test performances of the model.

After this train-test process, the model parameters were saved and another application dedicated to the model execution was initialized and statically deployed, without further retraining steps over time. To this purpose, the application was containerized and used for streaming execution, scheduled at the end of every hour of the day when all the input values pertaining to a specific hour have been already measured and stored in the database. Throughout each execution cycle, the *qualify*, *fetch*, data retrieval from the database through APIs, *pre-process* performing incoming data min-max normalization, and *analyze* function performing the estimation were invoked and executed. The estimated power production of the PV system served as a dynamic benchmark value at each time-step under normal operation. Abnormal patterns were identified based on the 3σ rule [93], with potential anomalies flagged when actual power values deviated from the dynamic benchmark over eight consecutive time-steps (i.e., 2 h). This approach allowed for a robust identification of critical events while minimizing false positives caused by model inaccuracies. The sequential execution ensured continuous validation of model constraints, detection of potential changes in those constraints, adherence to the naming convention, and proper preprocessing of input variables for model execution. This demonstrated that a ML pipeline can be robustly encapsulated and deployed using the proposed framework.

5.2. Substitution of the estimation model

As a subsequent step, it was decided to switch the model from Multi Layer Perceptron (MLP) to a sequence to sequence Long Short-Term Memory (LSTM) that in literature was identified to have better performances for time-series forecasting of PV power production [91,92]. The proposed LSTM structure was implemented in the *analyze* function by maintaining the same input variable as the MLP described in the previous paragraph. The proposed LSTM model consists of an input layer of size 5, two hidden layers with 16 neurons each, an output layer of dimension one. The ADAM optimizer was used in the model training for the parameter identification with a learning rate of 0.01. The training process involves feeding the training data through the model, computing the Mean Square Error (MSE) loss, performing back-propagation to calculate gradients, and updating the weights. Specifically, the LSTM model was formalized as a sequence to point model and estimates the incoming PV electrical power production by exploiting the past 48 observations, i.e., 12 hours considering one measurement every 15 min. All the models parameters were reported in the application *configuration* file in order to support the analyst with an easier way to change the parameters if needed. Similarly to the MLP, the LSTM parameters were set according to typical values found in literature [90–92]. A summary of the main parameters and settings is listed in Table 3.

Table 4
Comparison of train, test and validation performances between two neural network models, respectively, Multi Layer Perceptron and Long Short-Term Memory for the "Cittadella" PV system. The training and test period is from 1st April to 1st of November 2023, and the validation period is from 1st of November 2023 to 1st of January 2024.

Model	R^2 train	R^2 test	R^2 validation
Multi Layer Perceptron (MLP)	0.82	0.81	0.53
Long Short-Term Memory (LSTM)	0.98	0.99	0.99

Given that the LSTM relied on a sequence of input values for the estimation task, the *preprocessing* function was enhanced by incorporating a more advanced seasonal trend decomposition process, in addition to the linear interpolation process described above. This was done to reconstruct possible sequences of more than four consecutive missing values [94,95]. The model was trained and tested on the same dataset used for developing the MLP using 70% of records for training and 30% for testing. As a result the LSTM model exhibited higher accuracy values (in terms of R^2 values) in both training and testing, as shown in Table 4, highlighting the improved performance over the MLP model.

After this train-test process, the model parameters were saved and another application dedicated to the model execution was initialized and deployed. Executing the LSTM model also required an update to the *pre-process* function to normalize input values and fill missing values within the input sequence, which could otherwise compromise the model execution over a significant amount of time-steps. Throughout each execution cycle, the following functions were invoked and executed: *qualify*, *fetch*, to retrieve the past 12 h input sequences, *pre-process* to perform min-max normalization and missing data replacement, and *analyze* to perform the estimation. Abnormal production patterns were identified using the procedure explained above. Fig. 9 presents a snapshot of the Campus dashboard, which visualizes the output of the application. The dashboard shows the actual power production for a single PV system (i.e. `PV_Cittadella`) and the estimated power over a six-day period in November 2023, presented both as time series and scatter

Fig. 9. Screenshot of the online Campus dashboard used to visualize the results of the LSTM model during the validation period.

plots. In the scatter plot, blue dots represent residuals that fall within a certain range around zero (i.e., 3 times the standard deviation of the residuals obtained in training), while red dots indicate data points that are out-of-range. The identified red dots correspond to occasional outliers that exceed the thresholds, which in this case are most likely associated with model inaccuracies rather than anomalies in the operation of the PV (as discussed earlier, eight consecutive out-of-range time-steps are required to confirm an anomaly).

To summarize, the model switch from an MLP to an LSTM impacted the *preprocess* and *analyze* functions, while the rest of the procedure remained unchanged, as shown in the "Test 2" in Fig. 8. This demonstrated that the portable application framework supports replacement of the estimation model without disrupting the online execution of the previously deployed EMIS application.

5.3. Optimization of the deployment strategy

During the deployment phase and the execution of the LSTM model online, it was decided to switch from a one-time training approach to a bi-weekly retraining strategy with a sliding window of 1 month, as shown in the "Test 3" reported in Fig. 8. This change was introduced due to the high dependence of the estimation model accuracy on the changing climatic conditions throughout the year, a re-training strategy was implemented to prevent performance degradation. To achieve this objective, the model was re-trained every two weeks using data from the past month as the train-test dataset, ensuring consistency of the trained model with the boundary conditions at the time of estimations. This enhancement was seamlessly implemented without altering the application itself; only the *configuration* file required modifications by setting a moving window period for model training instead of specifying start and end dates statically. This test aimed to demonstrate that, due to a decoupled approach in the design of the application, critical parameters as the retraining period, can be simply modified by updating the configuration file. While this approach was found to be effective for data-driven algorithms, it can also be adapted for other types of models, as it operates independently from the specific algorithm implemented in the *analyze* function.

5.4. Change in naming convention and data availability

During deployment, the solar irradiance sensor ID was artificially modified in the database, simulating an unexpected and undocumented renaming of the variable. The subsequent update of the metadata schema ("Test 4" in Fig. 8) ensured the application continued to run without interruption. This was made possible by the semantic data layer, which decouples the application logic from the handling of the metadata, such as sensor IDs. Additionally, the flexibility of the *fetch* function allowed for a dynamic retrieval of sensor/variable names based on the Brick class and its relationship with other equipment, enabling robust execution in real-world scenarios. During the same test, another event was artificially inducted, by disconnecting the outdoor air temperature sensor from the monitoring infrastructure of the "Cittadella" PV system. In this case the application could not run due to absence of a required variable. However, the campus is equipped with a centralized weather station that can serve as a fallback when local temperature or solar radiation sensors are not working properly. To enable automatic querying of alternative data when the primary data source becomes unavailable, the application *manifest* and metadata model were modified. This update introduced additional queries to check and validate the presence of alternative data points when the primary data point was corrupted, as depicted in Fig. 10 (a). This formal relaxation of the constraints allowed the *qualify* function to provide a valid output and the *fetch* function to retrieve the needed data points, thus making the application execution robust against potential future occurrences of the same issue.

5.5. Extension of the application pipeline to other systems

Finally, the anomaly detection application, which includes both the training and execution pipelines, was extended to all six available PV systems ("Test 5" of Fig. 8). While the process was successful, enabling rapid deployment of an LSTM model for each system, it required some limited updates to the application configurations, queries and manifests. This process is likely to be common when an application, initially developed for specific use cases, is extended to slightly different scenarios. Specifically, some of the PV systems required slightly different preprocessing. For example, power production values for the "I3P" PV system

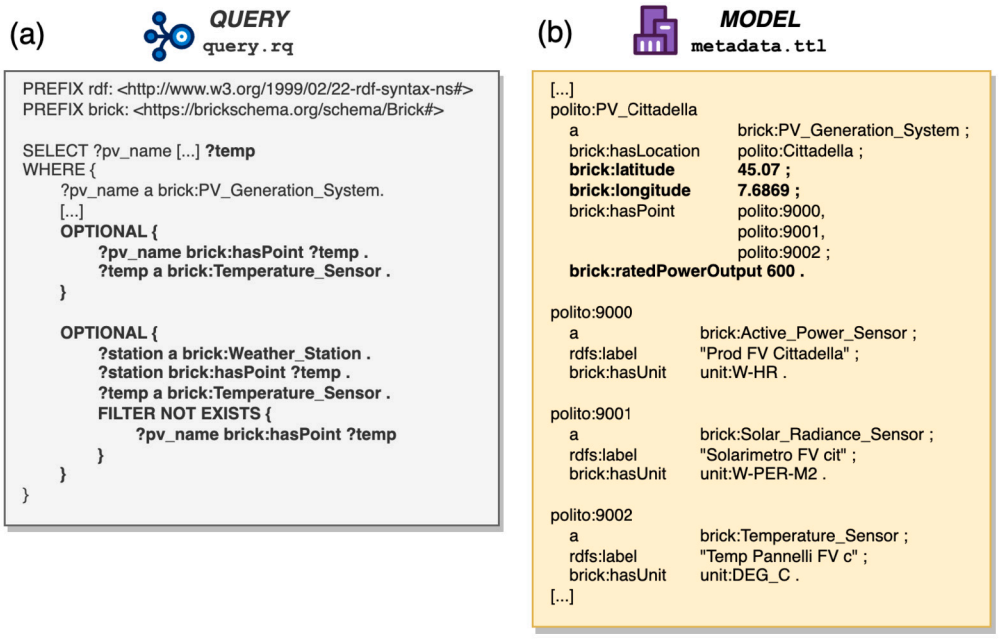

Fig. 10. Figure (a) on the left shows an extract of the SPARQL query, illustrating how the retrieval of the temperature sensor was generalized as described in Section 5.4. The query first searches for a temperature sensor on the PV array, and if no sensor is found, it then retrieves the temperature sensor from the weather station. Figure (b) on the right shows the additional information was added to the metadata model as described in Section 5.5. Geographical location and rated power output enabled a more accurate input variable calculation for the neural network model and enhanced plant-specific preprocessing procedures.

were observed to exceed its rated peak power (e.g., more than 200 kW when the installed power is 31 kW_p). After a thorough investigation, it was determined that this discrepancy resulted from a data monitoring error. To address this issue, all the *preprocessing* functions in the deployed applications were updated to include a check that uses a cutoff threshold to discard measured power values exceeding the rated peak power of the PV system. To automate the entire process, the metadata schema of the Campus PV plant was then updated including a new property for each PV system referred to its peak power. Additionally, more detailed latitude and longitude coordinates were incorporated into the metadata schema to provide accurate values for input variables dependent on the system's location (e.g., azimuth, zenith and global horizontal radiation calculated in clear sky conditions). For reference, an extract of the `metadata.ttl` file is shown in Fig. 10 (b), highlighting in bold the new metadata associated with the PV system "Cittadella".

6. Discussion and lesson learned

The results section highlighted the framework's capability to deploy an EMIS application based on a ML application and adapt it to the five tests introduced in the case study section. The framework allowed to handle these application updates without the need to rewrite the full code base and, in some cases, without interrupting its operation. However, the proposed framework relies on several assumptions regarding the availability, quality and maintenance of the underlying metadata model. Naturally, this raises questions about the limitation of such approach, the open questions and potential areas for future research.

6.1. Is the framework limited to the Brick schema?

The proposed framework exploits the Brick schema to express the semantic or meaning of the data points required by the applications. Brick allows to describe physical, logical and virtual assets in buildings and the relationships between them [96]. It was designed to represent a wide variety of building types and has been deployed in buildings across the globe. Brick is also extensible, enabling the addition of new concepts to its schema. Brick has been extensively used in both academic literature [96] and industry [97,98], and it remains under active development at the time of writing. However, Brick may not cover all the use cases an application developer may need. Recent literature examines the advantages and limitations of Brick in relation to two EMIS use cases, including FDD and optimal supervisory control, compared to other ontologies [65]. Brick does not offer a detailed description of the layout of the equipment components, nor does it include representations of envelope characteristics or architectural elements like room geometry and orientation [65,1]. For this reason, it is plausible that other application developers may prefer using different ontologies that better suit their specific use cases. The field of semantic schemas for buildings is extensive and rapidly evolving [20,65,99], and it is uncertain whether there will be convergence towards an unified, interoperable schema. Fortunately, the framework developed in this work is easily adaptable to different ontologies, as long as they support two core semantic web technologies: RDF [68] and SHACL [84]. The combined use of these two W3C technologies, enables the framework to offer a flexible and standardized approach for defining and validating data [65,96]. Specifically, RDF provides a standard method to represent data as directed, labeled graph, where the edges represent the named link (i.e. relationship) between two resources, represented by the graph nodes. In addition, SHACL defines the expected structure and constraints of the data, independent of the specific ontology used. While the use of additional ontologies is not explored in this paper, future work will focus on modifying the underlying data model by switching and/or combining ontologies, thereby exploring additional limitations and opportunities.

6.2. How is a semantic model of a building created?

A fundamental assumption of the proposed framework is that a complete metadata model of the building is available before deploying an EMIS application. However, only a small number of buildings today have complete semantic models. In industry, these models are often developed when an application is deployed, for example during the installation and configuration of FDD tools [100]. These semantic models are constructed using information from a variety of sources such as building automation data, mechanical drawings and control sequence documentation [20,65,101]. Industry has historically used proprietary

tools to map this information into a semantic model, internal to their products [102]. Recently, the organizations supporting building ontologies have started developing public tools to streamline this process [99]. An example are equipment and component templates that can support the manual process of building a metadata model from scratch [83]. A template is a high level function that allows the model developer to create RDF graphs in a consistent manner. Although this reduces the amount of work required, the developer still needs deep knowledge of ontologies and knowledge graphs. Academia has also proposed the use of artificial intelligence to create Brick models from building automation labels and time-series data [103].

6.3. Is there only one way to create a semantic model?

When crating a semantic model, an important design decision involves determining the level of detail required to describe the building and its systems. For example, a piece of equipment can be modeled with extensive details about its configuration, components, and parts, or it could be modeled with only the information necessary for monitoring. Currently, the approach is left to the modeler ant it can vary significantly depending on the analyst approach and sensitivity. For example, one model developer may model the same `brick:Damper_Position_Command` of the same equipment (e.g., AHU) as a point of the `brick:AHU` equipment, while the other might consider a `brick:Damper_Position_Command` as a point of the `brick:Damper` which is a part of the `brick:AHU` equipment. In addition, different use cases may require models that differ in scope or in granularity. For instance, a building control ASO application may require more detailed description of component relationships (e.g., specifying control hierarchies), while an EIS application might only need a higher-level information to be executed. Consequently, the creation of manifests (i.e., `manifest.ttl`) and corresponding queries (i.e., `query.rq`), which enable the application to verify the availability of required data points and retrieve them from the metadata model, is heavily dependent on the specific details of the model created. Even minor modifications to the relationships, classes, and structure of the metadata model an affect query results or prevent the application from running correctly. These two examples demonstrate that for some applications it is challenging to formalize queries or manifests that are flexible enough to adapt to a number of possible different graph structures describing the same building or system. The lack of guidance on structuring metadata models and determining the necessary level of detail for different use cases remains a significant barrier to develop consistent and interoperable models. As a result, applications must accommodate various representations of the same components, which complicates the development and interpretation of queries and manifests. Although different methodologies and tools have been introduced to build consistent semantic models from templates, the construction of such models and accessing the required data remains challenging [63,83,71]. Standardizing these model would help reduce complexity of the application and ensure more consistent and reliable data handling. Another way to addressing this issue is to shift the focus from creating the metadata model to its querying. Authors in Bennani et al. [2] proposed a method to relax queries, making them adaptable to alternative model formulations. While this approach shows promise, its generalizability has not yet been demonstrated. In summary, there is a trade-off between the effort required to ensure consistency in metadata models and the effort needed to manage the complexity of the EMIS application in handling different model implementations.

6.4. Does the metadata model remain static in time?

During “Test 4”, the metadata model was modified to simulate an abrupt change in naming conventions and data availability, in order to evaluate the impact on the application’s execution. Although successful in demonstrating the framework’s flexibility, the test also highlighted that the model represents a snapshot of the building information at a specific point in time, which is likely to undergo modifications throughout the building’s life cycle [104]. Without proper versioning of the model, changes can compromise the robustness and portability of an application [105]. As a reference example, it is possible to take into consideration an application where a ML model, similar to the ones used in this paper, requires time series data for a specific variable (e.g., outdoor air temperature) over a one-month period. For example, consider an application where a ML model, requires time-series data for a specific sensor (e.g., outdoor air temperature) over a one-month period. If the variable name and corresponding metadata file are updated during this period, it becomes very challenging to ensure the application recognizes the sensor both before and after the update, and consequently retrieves the correct time series data. Proper versioning is essential to maintain a record of changes over time and address this issue [106]. Versioning and continuously updating the model is particularly important for enabling an EMIS system to recognize new applications that can be executed when new data points become available in the metadata model. Future work should improve the framework by periodically scanning the metadata model for new or updated data. The system could then match these data points against the requirements of potential applications, such as the presence of specific sensors needed for advanced FDD analysis. When new data meets the requirements for an application, the system could either prompt the user or automatically configure it. Additionally, this approach could support the maintenance of existing applications throughout the building life cycle, allowing them to dynamically adapt to the evolving monitoring infrastructure and data availability.

6.5. Which is the role of Generative Artificial Intelligence?

The rapid success of Large Language Models (LLMs), such as ChatGPT [107] and Copilot [108], suggests the potential for using LLMs to create metadata models for buildings. These tools may be able to streamline the creation of metadata models by automating template generation based on the descriptions of building components and equipment, provided by professionals [109]. LLMs, excel at understanding and processing natural language inputs. This capability may enable the translation of natural language description of buildings components, energy system, and monitoring infrastructure into a metadata model for the building. This intuition is supported by recent literature that explores the use of LLMs in crating knowledge graphs [110] or in detecting inconsistencies in manually created models and suggest corrections and improvements. Such approaches are promising not only to automatically create metadata schema but also to support non experts users in the writing of the queries and manifests. Although promising, this approach has not yet been tested in the building domain. Further, concerns about inaccuracy of the output produced by LLMs remain unaddressed in the field of AI and needs more work.

7. Conclusions

This paper, introduces a portable framework for developing, deploying and maintaining EMIS applications using the Brick semantic schema. The framework aims to enhance the portability, interoperability, and maintainability of EMIS applications in buildings. Key contributions include the development of an interoperable software layer leveraging the Brick metadata schema, the formalization of application constraints in terms of metadata and data requirements, and a field demonstration. The framework was also implemented and made publicly available as a Python package, offering functions for querying metadata models, fetching data, preprocessing, and analyzing data. The framework was tested through a case study involving a data-driven, ML-based anomaly detection tool for photovoltaic systems, showcasing its ability to address various practical issues throughout the EMIS application life-cycle. Future

research in this area will focus on examining metadata schema versioning strategies, testing the use of different ontologies and investigating the use of Large Language Models (LLMs) to support model creation. Additionally, the research will explore how the proposed framework contributes to the seamless integration of newly developed applications with existing ones, enabling the creation of more effective multi-purpose EMIS systems.

Acronyms

AHU Air Handling Unit
AI Artificial Intelligence
APAR AHU Performance Assessment Rules
API Application Programming Interface
ASO Advanced System Optimization
BAS Building Automation System
BOPTEST Building Optimization Testing Framework
BuildingMOTIF Building Metadata OnTology Interoperability Framework
CLI Command Line Interface
EIS Energy and Information Systems
EMIS Energy Management and Information Systems
FDD Fault Detection and Diagnosis
HVAC Heating, Ventilation and Air Conditioning
IoT Internet of Things
LLMs Large Language Models
LSTM Long Short-Term Memory
ML Machine Learning
MLP Multi Layer Perceptron
MSE Mean Square Error
PoliTo Politecnico di Torino
PV Photovoltaic System
RDF Resource Description Framework
SHACL Shapes Constraint Language
SPARQL SPARQL Protocol and RDF Query Language

CRediT authorship contribution statement

Roberto Chiosa: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Formal analysis, Conceptualization. **Marco Savino Piscitelli:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Data curation, Conceptualization. **Marco Pritoni:** Writing – review & editing, Validation, Supervision. **Alfonso Capozzoli:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Co-author Marco Savino Piscitelli is a young Editorial board member in the Journal. Other 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

Data will be made available on request.

Acknowledgements

We extend our gratitude to the CALOS administration office at Politecnico di Torino for enabling the implementation process of this research project. Additionally, we acknowledge the Crown Labs infrastructure for providing the necessary cloud resources, and the Living Lab and PROGES for providing technical support. The work of Marco Savino Piscitelli was carried out within the Ministerial Decree no. 1062/2021 and received funding from the FSE REACT-EU - PON Ricerca e Innovazione 2014-2020. Part of this research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy under contract number DE-AC02-05CH11231.

References

[1] G. Fierro, P. Pauwels, Survey of Metadata Schemas for Datadriven Smart Buildings (Annex 81), CSIRO, Australia, 2022.
[2] 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, 2021, pp. 150–159.
[3] H. Kramer, G. Lin, C. Curtin, E. Crowe, J. Granderson, Building analytics and monitoring-based commissioning: industry practice, costs, and savings, Energy Effic. 13 (2020) 537–549.
[4] G. Lin, H. Kramer, J. Granderson, Building fault detection and diagnostics: achieved savings, and methods to evaluate algorithm performance, Build. Environ. 168 (2020) 106505, https://doi.org/10.1016/j.buildenv.2019.106505.
[5] H. Kramer, G. Lin, J. Granderson, C. Curtin, E. Crowe, Synthesis of Year One Outcomes in the Smart Energy Analytics Campaign Building Technology and Urban Systems Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, 2017.
[6] Z. Chen, Z. O'Neill, J. Wen, O. Pradhan, T. Yang, X. Lu, G. Lin, S. Miyata, S. Lee, C. Shen, et al., A review of data-driven fault detection and diagnostics for building hvac systems, Appl. Energy 339 (2023) 121030.
[7] E.T. Maddalena, Y. Lian, C.N. Jones, Data-driven methods for building control—a review and promising future directions, Control Eng. Pract. 95 (2020) 104211.
[8] G. Lin, H. Kramer, V. Nibler, E. Crowe, J. Granderson, Building analytics tool deployment at scale: benefits, costs, and deployment practices, Energies 15 (2022) 4858.
[9] J. Granderson, G. Lin, R. Singla, E. Mayhorn, P. Ehrlich, D. Vrabie, S. Frank, Commercial fault detection and diagnostics tools: what they offer, how they differ, and what's still needed, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2022.
[10] M. Ahern, D.T. O'Sullivan, K. Bruton, Implementation of the idaic framework on an air handling unit to transition to proactive maintenance, Energy Build. 284 (2023) 112872.
[11] K. Xu, Z. Chen, F. Xiao, J. Zhang, H. Zhang, T. Ma, Semantic model-based large-scale deployment of ai-driven building management applications, Autom. Constr. 165 (2024) 105579.
[12] C. Clark, A. Prakash, M. Pritoni, M. Kloss, P. Gupta, B. Nordman, M.A. Piette, M. Kamel, T.S. Semaan, A. Eisele, et al., Harvesting the low-hanging fruit of high energy savings–Virtual Occupancy using Wi-Fi Data, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2023.
[13] K.H. Andersen, S.P. Melgaard, H. Johra, A. Marszal-Pomianowska, R.L. Jensen, P.K. Heiselberg, Barriers and drivers for implementation of automatic fault detection and diagnosis in buildings and hvac systems: an outlook from industry experts, Energy Build. 303 (2024) 113801.
[14] N. Luo, M. Pritoni, T. Hong, An overview of data tools for representing and managing building information and performance data, Renew. Sustain. Energy Rev. 147 (2021) 111224.
[15] M. Pritoni, C. Clark, A. Prakash, M. Kloss, P. Gupta, B. Nordman, M.A. Piette, Harvesting the low-hanging fruit of high energy savings–Virtual Occupancy using Wi-Fi Data, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2020.
[16] A.K. Prakash, M. Pritoni, M. Kloss, M.A. Piette, M. Kamel, D. Hage, Cloud-Control of Legacy Building Automation System: a case study, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2022.
[17] Z.J. Zhai, A. Salazar, Assessing the implications of submetering with energy analytics to building energy savings, Energy Built Environ. 1 (2020) 27–35.
[18] H. Li, T. Hong, S.H. Lee, M. Sofos, System-level key performance indicators for building performance evaluation, Energy Build. 209 (2020) 109703.
[19] J. Kim, K. Trenbath, J. Granderson, Y. Chen, E. Crowe, H. Reeve, S. Newman, P. Ehrlich, Research challenges and directions in hvac fault prevalence, Sci. Technol. Built Environ. 27 (2021) 624–640.
[20] H. Bergmann, C. Mosiman, A. Saha, S. Haile, W. Livingood, S. Bushby, G. Fierro, J. Bender, M. Poplawski, J. Granderson, et al., Semantic interoperability to enable smart, grid-interactive efficient buildings, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2020.
[21] J. Sievers, T. Blank, A systematic literature review on data-driven residential and industrial energy management systems, Energies 16 (2023) 1688.
[22] S. Mischos, E. Dalagdi, D. Vrakas, Intelligent energy management systems: a review, Artif. Intell. Rev. 56 (2023) 11635–11674.
[23] A. Li, C. Zhang, F. Xiao, C. Fan, Y. Deng, D. Wang, Large-scale comparison and demonstration of continual learning for adaptive data-driven building energy prediction, Appl. Energy 347 (2023) 121481.

[24] G. Fierro, M. Pritoni, M. AbdelBaky, P. Raftery, T. Peffer, G. Thomson, D.E. Culler, Mortar: an open testbed for portable building analytics, in: Proceedings of the 5th Conference on Systems for Built Environments, 2018, pp. 172–181.
[25] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, et al., Portable queries using the brick schema for building applications: demo abstract, in: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, 2016, pp. 219–220.
[26] R.O. Yussuf, O.S. Asfour, Applications of artificial intelligence for energy efficiency throughout the building lifecycle: an overview, Energy Build. (2024) 113903.
[27] Y. Himeur, K. Ghanem, A. Alsalemi, F. Bensaali, A. Amira, Anomaly detection of energy consumption in buildings: a review, current trends and new perspectives, arXiv 2020, arXiv preprint, arXiv:2010.04560, 2010.
[28] Y. Chen, S. Huang, D. Vrabie, A simulation based approach to impact assessment of physical faults: large commercial building hvac case study, in: 2018 Building Performance Modeling Conference and SimBuild Co-Organized by ASHRAE and IBPSA-USA, Chicago, IL, USA, 2018, pp. 823–830.
[29] Y. Chen, G. Lin, Z. Chen, J. Wen, J. Granderson, A simulation-based evaluation of fan coil unit fault effects, Energy Build. 263 (2022) 112041.
[30] S. Li, Development and validation of a dynamic air handling unit model, part i, ASHRAE Trans. 116 (2010) 45.
[31] J. Wen, S. Pourarian, X. Yang, X. Li, Tools for evaluating fault detection and diagnostic methods for hvac secondary systems of a net zero building, Sci. Data (2015) 2015.
[32] J. Granderson, G. Lin, A. Harding, P. Im, Y. Chen, Building fault detection data to aid diagnostic algorithm creation and performance testing, Sci. Data 7 (2020) 65.
[33] D. Blum, J. Arroyo, S. Huang, J. Drgoňa, F. Jorissen, H.T. Walnum, Y. Chen, K. Benne, D. Vrabie, M. Wetter, et al., Building optimization testing framework (boptest) for simulation-based benchmarking of control strategies in buildings, J. Build. Perform. Simul. 14 (2021) 586–610.
[34] D. Blum, F. Jorissen, S. Huang, Y. Chen, J. Arroyo, K. Benne, Y. Li, V. Gavan, L. Rivalin, L. Helsen, et al., Prototyping the BOPTEST framework for simulation-based testing of advanced control strategies in buildings, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2019.
[35] L. Maier, J. Brillert, E. Zanetti, D. Müller, Approximating model predictive control strategies for heat pump systems applied to the building optimization testing framework (boptest), J. Build. Perform. Simul. 17 (2024) 338–360.
[36] T. Marzullo, S. Dey, N. Long, J. Leiva Vilaplana, G. Henze, A high-fidelity building performance simulation test bed for the development and evaluation of advanced controls, J. Build. Perform. Simul. 15 (2022) 379–397.
[37] J. Arroyo, C. Manna, F. Spiessens, L. Helsen, Reinforced model predictive control (rl-mpc) for building energy management, Appl. Energy 309 (2022) 118346.
[38] H.T. Walnum, I. Sartori, M. Bagle, Model predictive control of district heating substations for flexible heating of buildings, in: International Conference Organised by IBPSA-Nordic, 13th–14th October 2020, OsloMet. BuildSIM-Nordic 2020. Selected Papers, SINTEF Academic Press, 2020, pp. 123–130.
[39] D. Wang, W. Zheng, Z. Wang, Y. Wang, X. Pang, W. Wang, Comparison of reinforcement learning and model predictive control for building energy system optimization, Appl. Therm. Eng. 228 (2023) 120430.
[40] J. Arroyo, F. Spiessens, L. Helsen, Comparison of optimal control techniques for building energy management, Front. Built Environ. 8 (2022) 849754.
[41] F. Bünning, C. Pfister, A. Aboudonia, P. Heer, J. Lygeros, Comparing machine learning based methods to standard regression methods for mpc on a virtual testbed, in: Building Simulation 2021, vol. 17, IBPSA, 2021, pp. 127–134.
[42] C.A. Faulkner, R. Lutes, S. Huang, W. Zuo, D. Vrabie, Simulation-based assessment of ashrae guideline 36, considering energy performance, indoor air quality, and control stability, Build. Environ. 240 (2023) 110371.
[43] E. Zanetti, D. Kim, D. Blum, R. Scoccia, M. Aprile, Performance comparison of quadratic, nonlinear, and mixed integer nonlinear mpc formulations and solvers on an air source heat pump hydronic floor heating system, J. Build. Perform. Simul. 16 (2023) 144–162.
[44] R. Yan, Z. Ma, Y. Zhao, G. Kokogiannakis, A decision tree based data-driven diagnostic strategy for air handling units, Energy Build. 133 (2016) 37–45.
[45] G. Stamatescu, I. Stamatescu, N. Arghira, I. Fagarasan, et al., Data-driven modelling of smart building ventilation subsystem, J. Sens. (2019) 2019.
[46] G. Stamatescu, Hvac air handling units: one-year data from medium-to-large size academic building, IEEE Dataport (2019).
[47] S.P. Melgaard, K.H. Andersen, A. Marszal-Pomianowska, R.L. Jensen, P.K. Heiselberg, Fault detection and diagnosis encyclopedia for building systems: a systematic review, Energies 15 (2022), https://doi.org/10.3390/en15124366.
[48] O.E. Gundersen, S. Shamsaliei, R.J. Isdahl, Do machine learning platforms provide out-of-the-box reproducibility?, Future Gener. Comput. Syst. 126 (2022) 34–47.
[49] C. Miller, P. Arjunan, A. Kathirgamanathan, C. Fu, J. Roth, J.Y. Park, C. Balbach, K. Gowri, Z. Nagy, A.D. Fontanini, et al., The ashrae great energy predictor iii competition: overview and results, Sci. Technol. Built Environ. 26 (2020) 1427–1447.
[50] C. Miller, A. Kathirgamanathan, B. Picchetti, P. Arjunan, J.Y. Park, Z. Nagy, P. Raftery, B.W. Hobson, Z. Shi, F. Meggers, The building data genome project 2, energy meter data from the ashrae great energy predictor iii competition, Sci. Data 7 (2020) 368.
[51] A.-D. Pham, N.-T. Ngo, T.T.H. Truong, N.-T. Huynh, N.-S. Truong, Predicting energy consumption in multiple buildings using machine learning for improving energy efficiency and sustainability, J. Clean. Prod. 260 (2020) 121082.
[52] Z. Dong, J. Liu, B. Liu, K. Li, X. Li, Hourly energy consumption prediction of an office building based on ensemble learning and energy consumption pattern classification, Energy Build. 241 (2021) 110929.
[53] M.S. Piscitelli, R. Giudice, A. Capozzoli, A holistic time series-based energy benchmarking framework for applications in large stocks of buildings, Appl. Energy 357 (2024) 122550.
[54] B. Gunay, D. Darwazeh, N. Torabi, S. Shillinglaw, Ahu doctor: an inverse model-based software platform for commissioning controls hardware and sequences in vav ahu systems, in: eSim 2022, 2022.
[55] J. Pereira, M. Silveira, Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention, in: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2018, pp. 1275–1282.
[56] Y. Chen, E. Crowe, G. Lin, J. Granderson, Integration of fdd data to aid hvac system maintenance, 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. 492–495.
[57] K. Bruton, D. Coakley, P. Raftery, D.O. Cusack, M.M. Keane, D. O'sullivan, Comparative analysis of the ahu info fault detection and diagnostic expert tool for ahus with apar, Energy Effic. 8 (2015) 299–322.
[58] N. Torabi, H.B. Gunay, W. O'Brien, R. Moromisato, A holistic sequential fault detection and diagnostics framework for multiple zone variable air volume air handling unit systems, Build. Serv. Eng. Res. Technol. 43 (2022) 605–625.
[59] 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) 108900.
[60] G. Lin, A. Casillas, M. Sheng, J. Granderson, Performance evaluation of an occupancy-based hvac control system in an office building, Energies 16 (2023) 7088.
[61] S. Pachuta, J. Dean, A. Kandt, K.N. Cu, Field Validation of a Building Operating System Platform, Technical Report, National Renewable Energy Lab. (NREL), Golden, CO (United States), 2022.
[62] D. Coraci, S. Brandi, T. Hong, A. Capozzoli, Online transfer learning strategy for enhancing the scalability and deployment of deep reinforcement learning control in smart buildings, Appl. Energy 333 (2023) 120598.
[63] C. Duarte Roa, P. Raftery, R. Sun, L. Paul, A.K. Prakash, M. Pritoni, G. Fierro, T. Peffer, Towards a Stronger Foundation: Digitizing Commercial Buildings with Brick to Enable Portable Advanced Applications, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2022.
[64] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, et al., Brick: metadata schema for portable smart building applications, Appl. Energy 226 (2018) 1273–1292.
[65] 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 (2021) 2024.
[66] P. Pauwels, G. Fierro, A reference architecture for data-driven smart buildings using brick and lbd ontologies, in: CLIMA 2022 Conference, 2022.
[67] C. Quinn, J. McArthur, Comparison of brick and project haystack to support smart building applications, arXiv preprint, arXiv:2205.05521, 2022.
[68] S.W.C. Group, Resource description framework (rdf), https://www.w3.org/RDF/, 2014.
[69] S. Wan, M. Zhao, Y. Chen, S. Yang, D. Qiu, L.J. Lo, A novel data-driven relationship inference approach for automatic data tagging in building heating, ventilation and air conditioning systems, Build. Environ. 246 (2023) 110968.
[70] M. Ihlenburg, N. Réhault, S. Herkel, G. Benndorf, Towards a digital representation of building systems controls, Int. Sustain. Energy Conf., Proc. 1 (2024), https://doi.org/10.52825/isec.v1i.1217.
[71] 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, 2023, pp. 127–131.
[72] V. Kukkonen, Method for using information models and queries to connect hvac analytics and data, J. Comput. Civ. Eng. 37 (2023) 04023034.
[73] C. Duarte Roa, P. Raftery, R. Singla, M. Pritoni, T. Peffer, Detecting Passing Valves at Scale Across Different Buildings and Systems: a Brick Enabled and Mortar Tested Application, Technical Report, Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States), 2022.
[74] F. de Andrade Pereira, L. Paul, A. Casillas, A. Prakash, W. Huang, M. Pritoni, C. Shaw, S. Martin-Toral, D. Finn, J. O'Donnell, Enabling portable demand flexibility control applications in virtual and real buildings, J. Build. Eng. (2024) 108645.
[75] G. Fierro, M. Pritoni, M. AbdelBaky, D. Lengyel, J. Leyden, A. Prakash, P. Gupta, P. Raftery, T. Peffer, G. Thomson, et al., Mortar: an open testbed for portable building analytics, ACM Trans. Sens. Netw. 16 (2019) 1–31.
[76] O. Nehasil, L. Dobiášová, V. Mazanec, J. Širokỳ, Versatile ahu fault detection–design, field validation and practical application, Energy Build. 237 (2021) 110781.
[77] S. Frank, G. Lin, X. Jin, R. Singla, A. Farthing, J. Granderson, A performance evaluation framework for building fault detection and diagnosis algorithms, Energy Build. 192 (2019) 84–92.
[78] M.N. Fekri, H. Patel, K. Grolinger, V. Sharma, Deep learning for load forecasting with smart meter data: online adaptive recurrent neural network, Appl. Energy 282 (2021) 116177.
[79] I. Žliobaitė, M. Pechenizkiy, J. Gama, An overview of concept drift applications, in: Big Data Analysis: New Algorithms for a New Society, 2016, pp. 91–114.

[80] D. Coraci, S. Brandi, A. Capozzoli, Effective pre-training of a deep reinforcement learning agent by means of long short-term memory models for thermal energy management in buildings, Energy Convers. Manag. 291 (2023) 117303.
[81] J. Yang, H. Rivard, R. Zmeureanu, On-line building energy prediction using adaptive artificial neural networks, Energy Build. 37 (2005) 1250–1259.
[82] J. Verhelst, G. Van Ham, D. Saelens, L. Helsen, Model selection for continuous commissioning of hvac-systems in office buildings: a review, Renew. Sustain. Energy Rev. 76 (2017) 673–686.
[83] 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, 2022, pp. 228–237.
[84] H. Knublauch, Shapes constraint language, https://www.w3.org/TR/shacl/, 2017.
[85] C. Fan, M. Chen, X. Wang, J. Wang, B. Huang, A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data, Front. Energy Res. 9 (2021) 652801.
[86] Timescale, Timescale database, https://www.timescale.com/, 2024.
[87] R. Rajagopalan, P.K. Varshney, Data aggregation techniques in sensor networks: a survey, Electr. Eng. Comput. Sci. (2006).
[88] F. Cucinella, Mass Scale Lightweight Remote Desktop Environments for Educational Purposes, Ph.D. thesis, Politecnico di Torino, 2021.
[89] K.S. Anderson, C.W. Hansen, W.F. Holmgren, A.R. Jensen, M.A. Mikofski, A. Driesse, pvlib python: 2023 project update, J. Open Sour. Softw. 8 (2023) 5994.
[90] A.F. Agarap, Deep learning using rectified linear units (relu), arXiv preprint, arXiv: 1803.08375, 2018.
[91] D.K. Dhaked, S. Dadhich, D. Birla, Power output forecasting of solar photovoltaic plant using lstm, Green Energy Intell. Transp. 2 (2023) 100113.
[92] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, M. Weyrich, A survey on long short-term memory networks for time series prediction, Proc. CIRP 99 (2021) 650–655.
[93] B. Meng, R. Loonen, J. Hensen, Leveraging dynamic power benchmarks and cusum charts for enhanced fault detection in distributed pv systems, Energy Convers. Manag. 314 (2024) 118692.
[94] T. Analytics, smana: Repairing Tool for Time Series with Weekly Seasonality, 2024.
[95] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, S. Zhu, Robuststl: a robust seasonal-trend decomposition algorithm for long time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 5409–5416.
[96] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, et al., Brick: towards a unified metadata schema for buildings, in: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, 2016, pp. 41–50.
[97] C. Quinn, J. McArthur, A case study comparing the completeness and expressiveness of two industry recognized ontologies, Adv. Eng. Inform. 47 (2021) 101233.
[98] B. Consortium, Brick consortium, https://brickschema.org/consortium/, 2024.
[99] M. Pritoni, M. Wetter, L. Paul, A. Prakash, W. Huang, S. Bushby, P. Delgoshaei, M. Poplawski, A. Saha, G. Fierro, M. Steen, J. Bender, P. Ehrlich, Digital and interoperable: the future of building automation is on the horizon. What's in it for me?, in: ACEEE Summer Study on Energy Efficiency, 2024, pp. 1–3.
[100] Z. Shi, W. O'Brien, Development and implementation of automated fault detection and diagnostics for building systems: a review, Autom. Constr. 104 (2019) 215–229.
[101] S. Lee, B. Cui, M.S. Bhandari, P. Im, Visual brick model authoring tool for building metadata standardization, Autom. Constr. 156 (2023) 105122.
[102] Clockworks Analytics, Building analytics comparison guide, https://go.clockworksanalytics.com/hubfs/FDD-White-Paper.pdf. (Accessed August 2024), 2022.
[103] D. Hong, H. Wang, J. Ortiz, K. Whitehouse, The building adapter: towards quickly applying building analytics at scale, in: Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, 2015, pp. 123–132.
[104] 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, 2020, pp. 70–79.
[105] M. Frommhold, R.N. Piris, N. Arndt, S. Tramp, N. Petersen, M. Martin, Towards versioning of arbitrary rdf data, in: Proceedings of the 12th International Conference on Semantic Systems, 2016, pp. 33–40.
[106] Brick, Brick versioning, https://brickschema.org/consortium/, 2024.
[107] OpenAI, Chatgpt large language model, https://chat.openai.com, 2024.
[108] Microsoft, Copilot ai, https://www.microsoft.com/microsoft-copilot, 2024.
[109] J. Frey, L.-P. Meyer, F. Brei, S. Gründer-Fahrer, M. Martin, Assessing the evolution of llm capabilities for knowledge graph engineering in 2023, in: ESWC, 2024, pp. 26–30.
[110] Y. Zhu, X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen, N. Zhang, Llms for knowledge graph construction and reasoning: recent capabilities and future opportunities, arXiv preprint, arXiv:2305.13168, 2023.