Ontology Integration for Building Systems and Energy Storage Systems

Fang He
The Hong Kong Polytechnic Univ.
Kowloon, Hong Kong
fangf.he@connect.polyu.hk

Dan Wang
The Hong Kong Polytechnic Univ.
Kowloon, Hong Kong
dan.wang@polyu.edu.hk

Yaojie Sun
Fudan Univ.
Shanghai, China
yjsun@fudan.edu.cn

ABSTRACT

A building ontology defines the concepts and organization of building data. Such knowledge can be assistance with automatic data access and support data-driven applications in buildings. With technological advances in batteries and energy storage, an increasing number of data-driven building applications now involve both building systems and energy storage systems (ESS), e.g., peak load shaving (PLS). However, existing building ontologies, e.g., Brick, are not designed to include concepts from ESS systems. Given the emergence of building-ESS applications, it has become important to develop ontologies that can cover knowledge about both building and ESS systems.

Building systems and ESS systems fall under different industry sectors and there are building ontologies and ESS ontologies that have been developed independently. To maximally reuse existing knowledge, we leverage ontology integration technologies. We present a building-energy storage ontology integration (BESOI) system that can extend a building ontology with appropriate ESS ontologies. Our system handles ambiguity, incoherence, and redundancy problems in ontology integration. We evaluate BESOI on four building-ESS applications by extending Brick, a notable building ontology, with different ESS ontologies. The results show that BESOI can extend the coverage of Brick from 68.09% to 95.74% on the concepts of applications.

CCS CONCEPTS

• **Information systems → Data Management System**.

KEYWORDS

Building Application, Energy Storage System, Metadata, Ontology Integration

ACM Reference Format:
Fang He, Dan Wang, and Yaojie Sun. 2023. Ontology Integration for Building Systems and Energy Storage Systems. In *The 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '23), November 15–16, 2023, Istanbul, Turkey*, 4 pages. https://doi.org/10.1145/3600100.3623720

BuildSys '23, November 15–16, 2023, Istanbul, Turkey

ACM ISBN 979-8-4007-0230-3/23/11…$15.00
https://doi.org/10.1145/3600100.3623720

1 INTRODUCTION

We have seen an increasing number of data-driven applications in building systems. The amount and diversity of the data are increasing. There is an unprecedented demand to organize data in effective ways. Current building automation systems develop *ontologies* which defines the concept (naming conventions), and data organization models. With an ontology, building data can be automatically accessed. To allow applications to be portable across different building systems, standard ontologies have been developed, e.g., Brick, Google DBO, Haystack, and others.

Along with technical progress in batteries and energy storage, an increasing number of data-driven building applications involve both building systems and ESS systems [11]. E.g., a PLS application [8] requires the power data of a chiller and the state data of a li-ion battery. Unfortunately, existing building ontologies, e.g., Brick, are not elaborate enough to represent the concepts of ESS systems (e.g., the concept of "li-ion battery" is not included in Brick).

The building industry and the ESS industry are two distinct industries. The ESS industry has developed its own ontologies, e.g., OntoPowsys [4], SEMANCO [12], SARGON [7], EM-KPI [10] independent of building ontologies.

To support building-ESS applications, we need to manage the coverage shortage of building ontologies. In this paper, we take the approach of ontology integration to maximally reuse knowledge from existing ESS ontologies. We present the current ontology-assisted data access work flow and review ontology integration techniques in §2. We present the concept of the coverage shortage problem and the solution of an ontology integration system BESOI[1] with ontology matching, merging, and repair in §3. We evaluate BESOI in §4 on four building-ESS applications by using BESOI to extend Brick with different ESS ontologies. The results show that BESOI can significantly extend the coverage of Brick.

2 BACKGROUND

Building-Energy Storage System applications: Intrinsically, ESS can assist with the interior and exterior dynamics of building electricity usage. Building-ESS applications have been developed at different levels [8]. The aim of peak load shaving is to minimize the electricity costs of buildings facing grid dynamics by operating the ESSs. PV-battery optimization (PVB) further takes the PV-battery system into consideration. A thermal economic analysis with ESS (TAESS) analyzes the thermal economics of buildings at the district level. Model predictive control with ESS (MPCESS) is about controlling both the electrical appliances and ESSs in buildings.

[1]We make our codes available: https://github.com/fangger4396/BESOI.

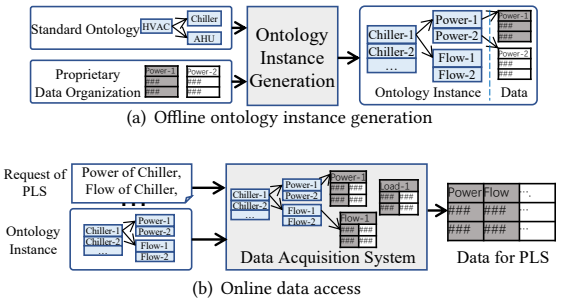

Figure 1: Ontology-assisted data access

Ontology-assisted data access: Applications rely on data. To access the raw data of an entity, *metadata* are used to describe entities (of concepts). The definition of metadata and the organization of metadata are called an *ontology*. For building systems, a notable ontology is Brick [1]. Brick defines the metadata of a set of entity classes (e.g., heating ventilation air condition (HVAC) system, Chiller, and others), and a set of relationship classes (e.g. hasPart, hasPoint, etc.) that represents the entities in buildings and their relationship. Brick uses a Resource Description Framework (RDF) graph to organize the Brick metadata.

An ontology allows an application to automatically access data. The general flow of ontology-assisted data access is as follows (Figure 1). In an offline process, each specific building can generate an *ontology instance* that follows a standard ontology, e.g., Brick. The ontology instance represents a standard logic view of the data organization of this specific building. Tools [5, 9] have been developed for such a process of generation. In the online data access process, an application, e.g., PLS, makes a request for data (e.g., the power data and load data), and a data acquisition system [2, 6] can follow the ontology instance to access the raw data.

ESS Ontologies: The building industry and the ESS industry belong to two industry sectors. Different industries develop their own ontologies without covering the concepts of other industries. The ESS industry has developed ontologies with different objectives: (1) OntoPowsys [4] was developed for ESS systems when working with power systems, where concepts related to energy systems in view of grids are included; (2) EM-KPI [10] was developed to enhance energy management at the district and building levels, where the main concepts are concepts referring to energy evaluations; (3) SEMANCO [12] was developed to capture concepts on diverse sources at the building level for managing energy, and (4) SARGON [7] was developed to describe concepts related to IoT devices, data points, and functions in buildings.

Ontology integration: is a process to generate a coherent ontology from multiple input ontologies [13]. Ontology integration has been widely used in biomedicine, collaborative manufacturing, and geographic information systems.

Ontology integration is about reusing the knowledge of existing ontologies. It typically consists of three steps: 1) *ontology matching*, i.e., to identify the overlapping concepts between two ontologies. For example, identifying that the "ethanol" of a chemical ontology may be the same concept with the "alcohol" of a pharmaceutical ontology; 2) *ontology merging*, i.e., to logically merge the matched correspondence elements. For example, using an equivalence relation to link the concept of "ethanol" and "alcohol"; 3) *ontology repair*, i.e., to improve the results of previous steps. For example, finding that the relation "belongTo" and "hasOwner" are redundant relations for "alcohol". Ontology integration can involve the use of one ontology as a base ontology and the integration of the other into the base, or the equal alignment of two ontologies and their reconciliation into on ontology.

3 BUILDING-ENERGY STORAGE SYSTEM ONTOLOGY INTEGRATION

3.1 Design Overview

Problem Statement: We aim to solve the ontology coverage problem for building-ESS applications, i.e., existing building ontologies cannot cover the concepts of ESS systems. We leverage ontology integration techniques and develop a system to automatically extend existing building ontologies with ESS concepts with the objective of maximizing coverage and reducing human involvement.

We analyzed different building-ESS applications and different ESS ontologies. We observed that as compared to building systems, where the ontologies have comprehensive coverage for building systems, and thus support for the applications on building system operations and controls; there is currently a lack of comprehensive ESS ontologies. Different ESS ontologies have different objectives and a different fit for different applications. In this paper, we develop in §3 and evaluate in §4 a generic ontology integration system that can integrate different ESS ontologies. We will discuss potential customized extensions of the generic system in §5.

We now present the Building-Energy System Ontology Integration (BESOI). BESOI can be used as a module to generate the standard ontology module in Fig. 1. The modular design of BESOI is shown in Figure 2. BESOI takes the building ontology and an ESS ontology as inputs. BESOI outputs a Building-Energy System Ontology (BESO)). BESOI consists of three modules:

1) **Semantic-based ontology matching** (§3.2): The objective is to discover the overlapping concepts between the two ontologies and solve the ambiguity of the concepts in both ontologies. We will develop a semantic-based ontology matching algorithm to match the entity classes, find the overlapping classes, and solve the ambiguity of the concepts by understanding their semantics.

2) **Coherence-based ontology merging** (§3.3): The objective is to logically merge the concepts and the input ontologies into a single ontology, and solve the incoherence of the concepts among ontologies. We will develop an incoherence resolution algorithm to detect and resolve coherence violations, which will cause logical conflicts in the merged ontology.

3) **Redundancy-based ontology repair** (§3.4): The objective is to repair the merged ontology and solve redundant relations. We will develop a redundant relation elimination algorithm to discover and eliminate relation redundancy.

3.2 Semantics-based ontology matching

A *correspondence* is the entity classes that refer to the same concept in different ontologies. For example, both Brick and OntoPowsys ontology have an entity class to represent the concept of "energy storage system" (although it is represented in different names of classes). The "energy storage system" is a correspondence between the two ontologies. Ontology matching is the process of discovering correspondences between entity classes from different ontologies.

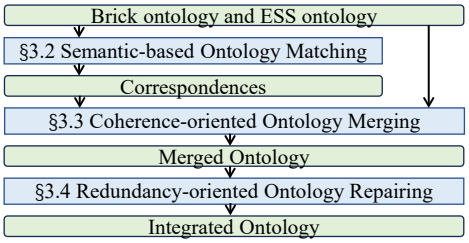

Figure 2: An overview of the system

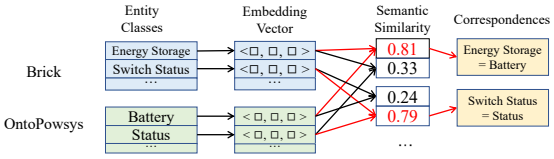

Figure 3: Ontology matching by using semantic similarities

Discovering correspondences is challenging since different ontologies are develop with different naming conventions and thus the concepts naturally have *entity ambiguity*. E.g., the concept of "energy storage system" is represented as the class "energy storage" in Brick and as "battery" in OntoPowsys; thus, "energy storage system" is a concept with entity ambiguity for these two ontologies.

There are numerous methods of correspondence discovering, and they fall into two categories: string-based and semantics-based correspondence discovering. String-based correspondence discovering involves conducting string-level processes on entity classes. String-based correspondence discovering is simple, but the performance is usually poor since it ignores the semantics between entity classes. Semantic-based correspondence discovering can provide more complete and reliable results since it can capture the similarity in semantics between entity classes. In this paper, we focus on finding the equivalence relations and we have selected the semantics-based correspondence discovering approach.

We develop a *semantic-based correspondence discovering algorithm*, which consists of three steps: 1) To obtain the semantics of entity classes of input ontologies, the entity classes are transformed into embedding vectors by using a pre-trained language model. Here, we use the Universal Sentence Encoder (USE) [3] to encode the texts of entity classes. The USE is a well-developed language model that computes context-aware representations of words in a sentence that takes into account both the ordering and identity of other words. 2) To compare the similarities between entity classes from two input ontologies, the cosine similarity of the embedding vectors of entity classes is calculated; 3) Correspondences are generated between entity classes with high semantic similarity. A pre-defined threshold is used to control the scale of the correspondences that are generated. We follow the OWL standards to represent the correspondences by using the relation `owl:equivalentClass` (e.g., <Brick:energy_storage, `owl:equivalentClass`, OntoPowsys:battery> represents that the "energy storage" of Brick is equal to the "battery" of OntoPowsys).

3.3 Coherence-based ontology merging

A *coherence violation* occurs when an entity class of the original ontology can never be satisfied, i.e., when no instance in practice can meet all of the requirements to be a member of the class. E.g., in OntoPowsys, the "thermal storage system" is logically disjointed from the class "Battery". This means that an entity can not be a "thermal storage system" and a "battery" at the same time. However, if another entity class, "energy storage", of Brick is equivalent to "thermal storage system" and "battery" concurrently, then this violates the disjointed relation in OntoPowsys and results in *incoherence*.

We developed an *incoherence resolution algorithm* that is comprised of two steps: 1) *detecting* the incoherence between the correspondences and the original ontologies, where the parent classes of an entity class are traversed to determine if there is incoherence since incoherence not only occurs between a pair of entity classes themselves but also between their parent classes(e.g., if "setpoint" is disjointed from "status", then "load setpoint" should be disjointed frp, "load status"); and 2) *resolving* the detected incoherence by removing the causal correspondences. We defined two types of incoherence to be detected: 1) disjointness-based incoherence, caused by the relation of disjointness between two distinct classes, and 2) inheritance-based incoherence, where a class cannot equal to a parent class and its child class at the same time (e.g., "energy storage" can not equal to "battery" and "li-ion battery" simultaneously, since "battery" and "li-ion battery" have a parent-child relation).

3.4 Redundancy-based ontology repair

A *relation redundancy* occurs when there is a relation of an ontology that can be subsumed by a relation in another ontology. E.g., the "hasStatus" is a relation between "battery" and the "status" in OntoPowsys, and "hasPoint" is another relation between "energy storage" and the "switch status" in Brick. Once "battery" and the "status" class correspond to "energy storage" and the "switch status" class, then the "hasStatus" and "hasPoint" are redundant relations.

We developed a *redundant relation elimination algorithm* that contains three steps: 1) to extract the relations in the merged ontology, for each entity class, the merged ontology is traversed to find its possible relations; 2) to detect the relation redundancy, for each extracted relation, a check is carried out to determine if there is anther relation that can be used with the same entity it relates to; 3) to eliminate the detected relation redundancy, a subsumption relation is added that is, by using the owl relation `rdfs:subPropertyOf` to describe the redundancy (i.e., <OntoPowsys:hasStatus, `rdfs:subPropertyOf`, Brick:hasPoint> represents the situation that the "hasStatus" of OntoPowsys is a subproperty of the "hasPoint" of Brick).

4 EVALUATION

We evaluate BESOI across a set of building-ESS applications with expressiveness in concepts: PLS, PVB, MPCESS, and TAESS. Our focus is on the coverage of integrated ontologies for the concepts required by the applications. We also evaluate the automated operations of BESOI for ontology integration. We use two manual ontology aligning strategies as the baseline. We use the BESOI to extend Brick with four different ESS ontologies mentioned above.

4.1 Concept Coverage

Metrics We evaluate the usability of integrated BESO by using the coverage of concepts for applications. Formally, the coverage of concepts is defined as $C = N_o/N_a$. Here, N_o is the number of concepts that an ontology contains for an application, and N_a is the number of concepts that an application requires.

Table 1: Coverage of original ontologies (Brick and ESS ontology) and integrated BESO

Application	# of Concepts	Coverage of Original Ontology					Coverage of BESO			
		Brick	OntoPowsys	SEMANCO	SARGON	EM-KPI	Brick+OntoPowsys	Brick+SEMANCO	Brick+SARGON	Brick+EM-KPI
PLS	47	32 (68.09%)	16 (34.04%)	15 (31.91%)	12 (25.53%)	15 (31.91%)	**45 (95.74%)**	37 (78.72%)	35 (74.47%))	33 (70.21%)
PVB	71	43 (60.56%)	30 (42.25%)	15 (21.13%)	16 (22.54%)	14 (19.72%)	**68 (95.77%)**	51 (71.83%)	49 (69.01%)	53 (74.65%)
MPCESS	43	16 (37.21%)	16 (37.21%)	21 (48.84%)	11 (22.45%)	11 (22.45%)	30 (69.77%)	**36 (83.72%)**	22 (51.16%)	24 (55.81%)
EPPESS	39	31 (79.49%)	12 (30.77%)	14 (25.45%)	14 (23.73%)	17 (30.91%)	35 (89.74%)	34 (87.18%)	33 (84.61%)	**36 (92.31%)**

Results Table 1 shows the coverage of the original ontologies and the integrated BESOs across four different applications. We can see that the coverage of integrated BESOs is much higher than that of the original ontologies. The highest coverage of the integrated BESOs for the four applications is 95.74%, 95.77%, 83.72%, and 92.31%, respectively. As a comparison, a single Brick ontology has a coverage of 68.09%, 60.56%, 37.21%, and 79.49%, respectively. Due to the insufficient coverage of original ontologies, the coverage of integrated ontologies cannot reach 100%. This clearly shows that the integrated BESOs can provide more complete coverage for certain applications and thus have reliable usability. We can also see that the original ontology can influence the BESOs within specific applications. For example, the integrated BESO Brick+OntoPowsys has the highest coverage of the application PLS, since OntoPowsys is an ontology that was developed for power grids and contains numerous concepts related to the properties of ESS (e.g. the charging rate of a battery), which are required by the PLS.

4.2 Effectiveness in an Automated Process

Baseline We use two manual ontology alignment strategies as our baseline: 1) For developers familiar with ESS ontology: given each class of Brick, the developer can directly point out the most likely corresponding class in ESS ontology. 2) For developers familiar with Brick: given each class of Brick, from the root classes of ESS ontology the developer goes through the child classes and selects the most possible corresponding class from among the child classes until the final correspondence is determined.

Metrics We now study how much effort the BESOI can save by automating the ontology integration process. Since BESOI consists of three modules (ontology matching, merging, and repair), we count the number of steps of primary operations conducted in the three modules. For ontology matching, the dominant operation is calculating the semantic similarity of entity classes. For ontology merging and repair, the primary operation is to detect violations and redundancies by checking the entity classes in the correspondence.

Table 2: Effectiveness of ontology integration by using the BESOI and through the manual ontology alignment

Ontology	# of Entity Classes	# of Entity Classes in ESS	# of Overlapped Concepts	# of Operations for Matching	# of Operations for Merging	# of Operations for Repair	# of Operations in Total	# of Operations by Manual Strategy 1	# of Operations by Manual Strategy 2
Brick+ OntoPowsys	1682	246	124 (50.41%)	353,256	4,270	559	358,082	1,436	8,616
Brick+ SEMANCO	2357	921	236 (25.63%)	1,322,556	1,557	397	1,324,510	1,436	10,052
Brick+ SARGON	1593	157	91 (57.96%)	225,452	3,975	546	229,973	1,436	7,180
Brick+ EM-KPI	1558	122	73 (59.84%)	175,192	2,144	487	177,823	1,436	5,744

Results Table 2 shows number of operations of three modules in the ontology integration process. We can see that all four pairs of input ontologies were processed by a large number of automation processes. The total number of operations was around 358k, 1,325k, 230k, and 178k for the four different input ontologies. We can also see that the dominant module is the ontology matching module, where 353k, 1322k, 225k, and 175k operations are performed for the four pairs of inputted ontologies, respectively. We also count the numbers of operation of manual ontology aligning. An operation is that the developer compare two classes between the ontologies and decide whether to construct a correspondence. We can see that the numbers of operation of manual ontology aligning for strategy 1 is all 1,436, and for strategy 2 is 8,616, 10,052, 7,180, and 5,744. Note that all of these operations can be completely avoided by the automated process. We argue that these automated processes can significantly reduce efforts compared with manual development.

5 CONCLUSION AND FUTURE WORK

In this paper, we studied the coverage shortage problem of building ontology in covering energy storage system (ESS) concepts when supporting building-ESS applications. We developed BESOI, a building-ESS ontology integration system. Our evaluation showed that BESOI can effectively extend the concept coverage of a building ontology, Brick, in an automated integration process.

Many future studies can be conducted: (1) To develop a generic ontology integration system that can integrate multiple ESS ontologies, and even ontologies from other industry sectors, e.g., EV and PV are two close sectors related to building applications; (2) To develop specific modules that can solve individual problems such as the aforementioned "subsumption" relations. (3) To substantially update the developed integrated ontology following the evolution of the original ontologies.

ACKNOWLEDGMENTS

Dan Wang's work is supported by RGC-GRF 15210119, 15209220, 15200321, 15201322, from ITC via project No. K-BBY1, RGC-CRF C5018-20G, ITC ITF-ITS/056/22MX. Yaojie Sun's work is supported by Shanghai Engineering Research Center for Artificial Intelligence and Integrated Energy System (Grant No. 19DZ2252000).

REFERENCES

[1] B. Balaji, A. Bhattacharya, G. Fierro, et al. 2016. Brick: Towards a Unified Metadata Schema For Buildings. In *Proc. ACM BuildSys'16.*
[2] I. L. Bennani, A. K. Prakash, et al. 2021. Query relaxation for portable brick-based applications. In *Proc. of Buildsys' 21.*
[3] D. Cer, Y. Yang, , et al. 2018. Universal sentence encoder for English. In *Proc. of EMNLP-demos 2018.*
[4] A. Devanand, G. Karmakar, et al. 2020. OntoPowSys: A power system ontology for cross domain interactions in an eco industrial park. *Energy and AI* (2020).
[5] G. Fierro, J. Koh, Y. Agarwal, R. K. Gupta, and D. E Culler. 2019. Beyond a house of sticks: Formalizing metadata tags with brick. In *Proc. of ACM Buildsys' 19.*
[6] G. Fierro, M. Pritoni, et al. 2018. Mortar: an open testbed for portable building analytics. In *Proc. of ACM Buildsys' 18.*
[7] M. Haghgoo, I. Sychev, et al. 2020. SARGON–Smart energy domain ontology. *IET Smart Cities* (2020).
[8] R. Hidalgo-León, D. Siguenza, et al. 2017. A survey of battery energy storage system (BESS), applications and environmental impacts in power systems. In *2017 ieee second ecuador technical chapters meeting (etcm).* IEEE.
[9] J. Koh, B. Balaji, et al. 2018. Scrabble: transferrable semi-automated semantic metadata normalization using intermediate representation. In *Proc. BuildSys'18.*
[10] Y. Li, R. García-Castro, et al. 2019. Enhancing energy management at district and building levels via an EM-KPI ontology. *Automation in Construction* (2019).
[11] J. Liu, X. Chen, et al. 2020. Energy storage and management system design optimization for a photovoltaic integrated low-energy building. *Energy* (2020).
[12] L. Madrazo, A. Sicilia, et al. 2012. SEMANCO: Semantic tools for carbon reduction in urban planning. In *Proc. of ECPPM 2012.*
[13] I. Osman, S. B. Yahia, et al. 2021. Ontology integration: approaches and challenging issues. *Information Fusion* (2021).