A Reasoning Engine Architecture for Building Energy Metadata Management

Panagiotis Kapsalis
Decision Support Systems Laboratory, School of Electrical and Computer Engineering National Technical University of Athens
Athens, Greece
pkapsalis@epu.ntua.gr

Giorgos Kormpakis
Decision Support Systems Laboratory, School of Electrical and Computer Engineering National Technical University of Athens
Athens, Greece
gkorbakis@epu.ntua.gr

Konstantinos Alexakis
Decision Support Systems Laboratory, School of Electrical and Computer Engineering National Technical University of Athens
Athens, Greece
kalexakis@epu.ntua.gr

Evangelos Karakolis
Decision Support Systems Laboratory, School of Electrical and Computer Engineering National Technical University of Athens
Athens, Greece
vkarakolis@epu.ntua.gr

Spiros Mouzakitis
Decision Support Systems Laboratory, School of Electrical and Computer Engineering National Technical University of Athens
Athens, Greece
smouzakitis@epu.ntua.gr

Dimitris Askounis
Decision Support Systems Laboratory, School of Electrical and Computer Engineering National Technical University of Athens
Athens, Greece
askous@epu.ntua.gr

***Abstract* — During the Buildings' lifecycle, massive amounts of data, that contain information related to their energy consumption, are generated. Towards the creation of smart building networks, this produced information must be intercepted and harmonized according to building ontologies and schemas. The pattern recognition from building metadata is based on inferencing and intelligent querying, that can be achieved with the utilization of graph and property databases that deploy and host building information. This paper presents a Reasoning Engine Architecture implemented in the context of the H2020 project called MATRYCS that persists building semantic information. It will be leveraged to support real life applications by improving the inference operations.**

Keywords — Inference analytics, building metadata management, semantic enrichment, reasoning engine, energy efficiency, intelligent querying, graph knowledge database

I. Introduction

The environmental crisis (climate change, energy supply) is the main problem that European Union (EU) is called to solve. Almost 40% of EU energy consumption is produced from buildings [1] across different member states, thus the building sector plays a solid role in the creation of effective climate and environmental policies[2]. The constantly generated metadata from the building lifecycle [3] offer a huge opportunity for improving the energy efficiency in the building sector by leveraging the logical connections between them. All the above-mentioned factors have led to an increasing momentum of technologies that could be utilized for managing the energy data generation [4] and consumption results [5], which can push towards the creation of smart and energy–aware building networks [6]. Data are expected to play an important role and various data analysis techniques [7] (including, among others, optimization, forecasting, classification, metadata analysis and management and clustering) can be applied for the extraction of meaningful information from building data and for making data-driven decisions that effectively support environmental policies.

Despite the number of existing solutions for the management of building data [8], there is not a standardizable, interoperable and modular architecture that is able to combine metadata belonging to different buildings. The lack of a semantic framework for buildings data interoperability is severe, as the need for semantic management of buildings technical measurements, exogenous context-based data (such as weather conditions), geographical or energy networks (district, heating, power networks) related data pushes towards the direction of an approach to supports multi-sector integrated buildings-centered analytics.

This work is mainly focused on the building metadata management. More specifically, a Reasoning Engine architecture that will be leveraged to extract and infer logical consequences from a set of asserted facts or axioms is proposed. The utility of a mechanism like this, is to provide richer sets of information to work with, by enabling RDF information [9], graph databases [10] and building standards as well as ontologies such as the BRICK schema [11] in order to standardize the physical, virtual and logical assets that are included in building operations. The BRICK schema was developed on top of Project Haystack [12] and the SAREF Ontology[13] in order to support more assets and data models. The schema and standards that BRICK provides may represent information for HVAC, Lighting, Electrical, Sensor systems, Spatial information, Formal definitions, building operational and control relationships.

Neo4j can store undirected, weighted graphs and hyper-graphs, in contraction to RDF triple stores that only store and manage the RDF building information as records. Moreover, Neo4j graph database provides theNeosemantics plugin [14]that can be used for importing RDFs and its associated vocabularies and for performing inferencing, by using imported ontologies and schemas axioms. In this research, Neo4j and Neosemantics deploy the Brick schema for conducting inferences and intelligent querying over building information and metadata.

The rest of the paper is structured as follows. Section II analyses the architecture and implementation of the Reasoning Engine for building metadata. Section III demonstrates the Reasoning Engine queries and capabilities. Finally, Section V summarizes the key issues arisen in this work

II. ARCHITECTURE & IMPLEMENTATION

The proposed Reasoning Engine Architecture for building data presented in this research [15] is mainly focused on the management of streaming messages that contain building information, rather than using ontologies and RDF schemas for building information Our architecture is focused on the management of connected and linked data (e.g., RDF triples) and building ontologies (e.g., BRICK) that can be leveraged in to homogenize the information, create logical inferences from building entities and extract hidden patterns that will facilitate beneficiary parties to perceive and comprehend their infrastructures' needs towards improved energy efficiency.

A Reasoning Engine for linked data is proposed in order to handle building ontologies and triples, as well as extract information and entities from them by leveraging the functionalities of graph databases and object storages. The above-mentioned engine is consisted of components that are responsible for the management and the version control of the linked datasets, the storage of RDF triples, the processing and the homogenization of these data under BRICK ontology, their transformation to graph entities, and the intelligent querying over the stored graph information via REST APIs. The Reasoning Engine for building linked data is a microservices – oriented solution, meaning that all the components are independent and separately deployed from the others, but, of course, properly connected to each other in order to harmonically collaborate. This approach provides resilience, flexibility and scalability. Figure 1depicts its architecture.

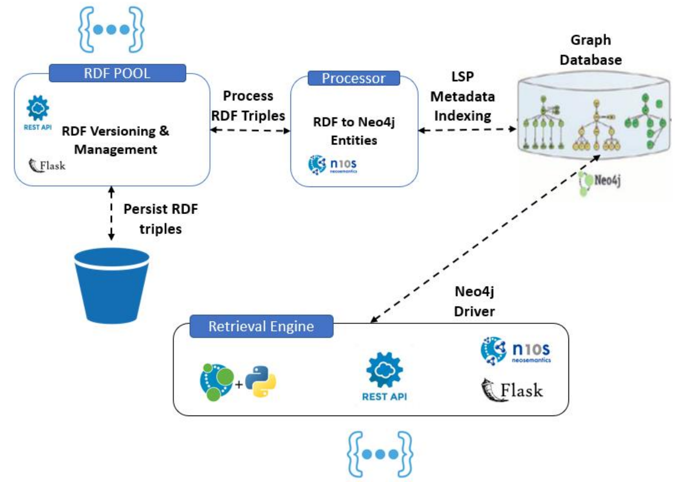

Figure 1: Proposed Architecture for building linked data

As demonstrated, the Reasoning Engine is a multi-component framework, consisted of four components, namely:

- the ***RDF Pool***: Object storage where Building RDF are stored and managed
- the ***RDF processors***: Component responsible for transforming RDF information to Neo4j entities
- the ***Graph database***: Neo4j graph database
- the ***Retrieval Engine***: Collection of REST APIs that receive JSON and, on the background, transform the JSON input to CYPHER for querying the stored building graph information. The result is also a JSON instance.

A. RDF Pool

The RDF Pool is the service responsible for receiving RDF triples and handling them before processing them [16]. It is a component implemented in Python 3[1] and it leverages the capabilities of Python libraries such as Flask REST Framework[2], MinIO[3] driver and Neo4j[4] driver to create REST APIs for receiving RDFs, to persist and manage these datasets and to enable the Brick ontology. Figure 2 demonstrates the functionality of the RDF Pool and how the data are inserted in the Reasoning Engine framework in RDF format.

[1] https://www.python.org/download/releases/3.0/
[2] https://flask-restful.readthedocs.io/en/latest/
[3] https://min.io/
[4] https://neo4j.com/

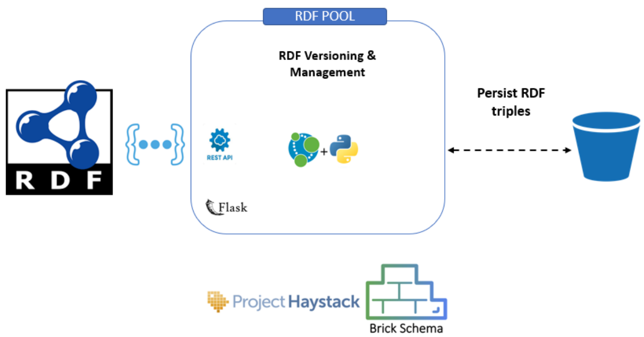

Figure 2: RDF file consumption

More specifically, the RDF Pool component enables the Brick Schema during its initialization. This process is performed for harmonizing all the incoming RDF files and the consumed building information in one common data model. Following, the RDF files are stored into MinIO object storage.

B. RDF Processor

The next layer is the RDF to Neo4j Processor, which is the service responsible for transforming the processed RDF information to Neo4j entities. This processor acts as a middleware between the RDF Pool and the Neo4j Graph database. It receives RDF data from MinIO and transforms the RDF data to CYPHER[5] commands to insert the information to Neo4j Graph database. Neosemantics[6] toolkit is used for the deployment of the RDFs to the graph database. The pipeline of RDF deployment is demonstrated in Figure 3.

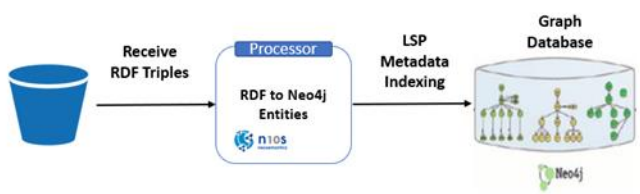

Figure 3: RDF Deployment Pipeline to Neo4j

C. Graph Database

The selected graph database technology is Neo4j, which is a distributed database that translates its records to graph nodes and the relationships between records are intercepted as graph connections. The graph database is selected for persisting the RDF information because it is capable of conducting logical inferences over data relationships, extracting hidden patterns and creatin a network of building data aligned to BRICK Ontology. Figure 4 demonstrates the network of persisted buildings in Neo4j. These building data have been inserted through the Reasoning Engine components, from RDF Pool to Processor and afterwards to the Graph indexing. Neosemantics toolkit achieves the transformation of the RDF to graph entities.

D. Retrieval Engine

The last component which exposes the persisted entities to the external world is the Reasoning Engine's Retrieval Engine. It is a collection of REST APIs based on Flask Rest Framework for the APIs generations and the routing functionality. The querying functionalities are derived from the usage of Python Neo4j driver and the Neosemantics toolkit which interacts with the stored network of building information.

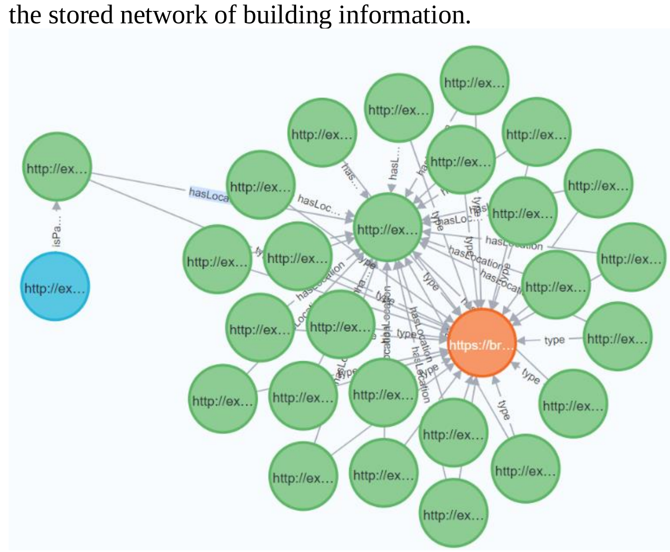

Figure 4: Network of Buildings aligned to BRICK Schema

The Retrieval Engine is responsible for the transformation of JSON[7] input received from the API collection. On the backend side the delivered information is translated to CYPHER queries that are finally sent to the Neo4j graph database.

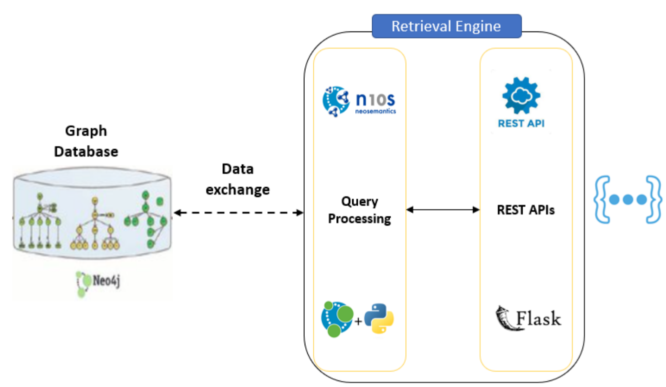

Figure 5: Retrieval Engine Interaction with the Neo4j Graph Database

Figure 5 depicts the interaction of the Retrieval Engine and the incoming input from the external applications and users of MATRYCS ecosystem. The Retrieval Engine is consisted of two sub-components: the REST APIs that collect the external information and distribute it to the Query Processing mechanism, and the Query Processing mechanism itself, which

[5] https://neo4j.com/developer/cypher/
[6] https://neo4j.com/labs/neosemantics/
[7] https://www.json.org/json-en.html

process and polishes the incoming events to CYPHER commands in order to query the existing network of buildings in the graph database.

Table 1 demonstrates the input received from the APIs.

Table 1: JSON input for receiving building KPIs

```
POST /leif/rdf/multiple/kpi/buildings
{
"kpi_list": ["Action_value", "Renovation_cost"]
}
```

This input is then transmitted to the Query processing, where are utilized to covert JSON to CYPHER. Table 2 depicts the CYPHER command generated from the input in Table 1.

Table 2: CYPHER command generated from JSON input

```
MATCH (b:Building)-[r:value]->(kpi)
WHERE kpi.label[0] in ["Action_value", "Renovation_cost"]
RETURN b.uri as building_uri, kpi.label[0] as label, kpi.hasUnit[0] as hasUnit, kpi.value[0] as value
```

The output of the Retrieval Engine is the result of query demonstrated in Table 2, which fetches the information of the stored buildings for the provided KPIs, and it is sent back in JSON format through HTTP[8] response (Table 3).

Table 3: Response returned from Retrieval Engine

```
[
  {
    "building_uri": "http://example.com/Leif#Incukalns_1",
    "label": "Renovation_cost",
    "hasUnit": "euro",
    "value": 328899.5
  },
  {
    "building_uri": "http://example.com/Leif#Incukalns_2",
    "label": "Action_value",
    "hasUnit": "action_value",
    "value": 3.01
  },
…..
]
```

E. Methodology

The design and the implementation of the proposed architecture was based on the results of the state-of-the-art analysis and research, as well as the undoubted need that arises for extracting information by leveraging rules and axioms from Building RDF information [17]. The next step was the identification of the BRICK ontology and of the Neosemantics toolkit which respectively are the technologies used for the homogenization of linked data and their graphical representation. Following, the technical phase was initiated with the exploration of available RDF building files and the creation of the Building graph. Finally, the materialization of the Reasoning Engine component using Python 3 accomplished.

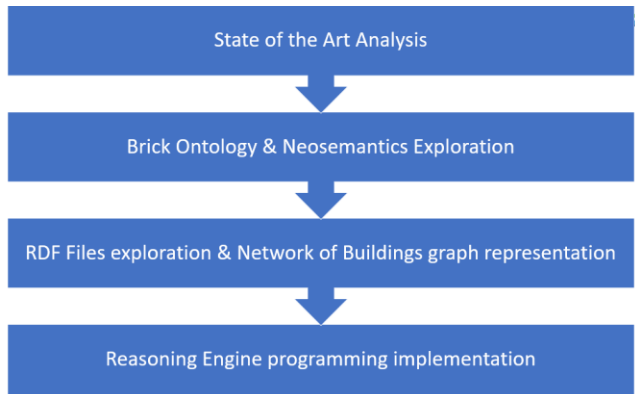

Figure 6: Implementation Steps

III. Demonstration

The Reasoning Engine proposed in this paper is leveraged to apply logical inferences and axioms over energy building data. The knowledge base, where our architecture will be used to perform logical consequences, is part of the MATRYCS[9] H2020 large scale pilots.

The data used for the testing of the proposed architecture were RDF triples containing information of buildings, such as building's construction year, building's total area, building's coordinates, building's KPIs before and after renovation, name of the project that funds the building's renovation activities, etc.

The insertion of RDF building data is conducted through the RDF Pool file uploading service. Before using this functionality, it is obligatory to define the RDF file that contains the building data that will be uploaded in the Reasoning Engine structures, the version number and the identifier of the file. This information are needed for characterizing the directory, where the input will be stored, in the RDF Pool object storage (MinIO bucket). Table 4 presents the uploading file functionality.

---

[8] https://developer.mozilla.org/en-US/docs/Web/HTTP

[9] https://matrycs.eu/

Table 4: RDF uploading service

```
POST /upload/ttl/files HTTP/1.1
Host: reasoning_engine
Content-Length: 401
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file";
filename="/ /LEIF.ttl"
Content-Type: <Content-Type header here>
(data)
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="version"
2
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file_id"
leif
----WebKitFormBoundary7MA4YWxkTrZu0gW
```

After the file uploading, the *Processor* sub-component begins to function. More specifically, the RDF triples are transformed to Neo4j entities (nodes and connections) by enabling the BRICK schema and Neosemantics toolkit. The RDF artifact is selected from the MinIO bucket and, as the process continues,it is polished and processed through the Processor cleansing procedures. Firstly, the homogenization process is triggered and the building properties in RDF triples are remodeled according to BRICK standards and attributes. Consequently, Neosemantics and its Python driver are enabled to convert the input data to Neo4j graph entities.

Table 5: Trigger Processing for specific RDF file

```
POST /enable/ttl HTTP/1.1
Host: reasoning_engine
{
  "ttl_id": "leif",
  "ttl_version": 1,
  "ttl_name": "LEIF.ttl",
  "ontologies":[""http://qudt.org/vocab/unit/", "http://www.w3.org/2001/XMLSchema#"] }
```

In Table 5 the REST API that triggers the procedures responsible to convert the file with name "LEIF.ttl" is demonstrated. As shown, this specific file has an identifier with the value "leif" and its version number is "1". The resulting graph from the previous file is depicted in Figure 7.

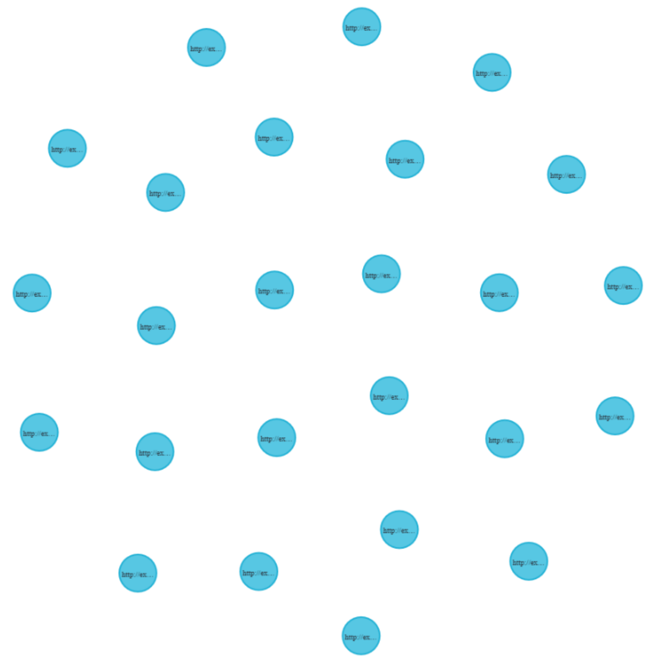

Figure 7: RDF Triples as Neo4j Entities

The building information is stored as graph entities and the Retrieval Engine is responsible for exposing these data to analytics applications in order to enhance their accuracy. The intelligent querying is offered from the persisted BRICK schema which is utilized from the Neosemantics toolkit to manage the building attributes effectively and to provide insights and capabilities to analytics services for reducing the size and the volume of their input.

Data produced from buildings are massive and their load is vast [18], building analytics services are responsible for handling that load for predicting and handling building KPIs and metrics [19], such as CO_2 emissions, electricity consumption. The proposed Reasoning Engine manipulates these metadata as graph entities and provides mechanisms for translating them to inputs easy to be consumed from other services such as JSON and reducing their size via metadata and ontology management. The analytics services receive the pattern extraction from Reasoning Engine REST APIs as JSON data. The following table (Table 6) contains a series of APIs that trigger the pattern and insights extraction from building graph entities.

Table 6: List of Reasoning Engine REST APIs

Description	Functionality
List of stored buildings URI	*GET /leif/rdf/buildings HTTP/1.1* *Host: reasoning_engine:5000*
List Project Sites	*GET /leif/rdf/sites HTTP/1.1* *Host: reasoning_engine5000*

List of stored building data	*GET /leif/rdf/buildings/data HTTP/1.1* *Host: reasoning_engine:5000*
List stored building data for specific Project Site	*POST /leif/rdf/site/buildings/data HTTP/1.1* *Host: reasoning_engine:5000* *{* *"site_uri": "http://example.com/ProjectSite#KPFI_7_8"* *}*
Get Information for specific building	*POST /leif/rdf/building HTTP/1.1* *Host: reasoning_engine5000* *{* *"building_uri": "http://example.com/Leif#Riga_49"* *}*
Get KPIs for specific building	*POST /leif/rdf/building/totals HTTP/1.1* *Host: reasoning_engine:5000* *{* *"building_uri": "http://example.com/Leif#Riga_21"* *}*
Get Energy KPIs for specific Building	*GET /leif/rdf/energy/kpis HTTP/1.1* *Host: reasoning_engine:5000*
Get List of buildings along with their values for the provided list of KPIs	*POST /leif/rdf/multiple/kpi/buildings HTTP/1.1* *Host: reasoning_engine:5000* *{* *"kpi_list": ["Action_value", "Renovation_cost"]* *}*
Get Buildings that belong to the same cluster	*GET/leif/rdf/cluster/number/${cluster_number}/buildings HTTP/1.1* *Host: reasoning_engine:5000*
Get top N most similar buildings for the provided building URI	*POST /leif/rdf/list/similars HTTP/1.1* *Host: reasoning_engine:5000* *{* *"num": N,* *"building_uri": "http://example.com/Leif#Riga_21"* *}*
Consume Building RDF via URL	*POST /import/rdf HTTP/1.1* *Host: reasoning_engine:5000* *{* *"onto_url": "https://brickschema.org/schema/Brick#",* *"rdf_url": "${RDF_URL}"* *}*
Consume Building RDF stored in RDF Pool	*POST /enable/ttl HTTP/1.1* *Host: reasoning_engine:5000* *{* *"ttl_id": "leif",* *"ttl_version": 1,* *"ttl_name": "LEIF.ttl",* *"ontologies": ["http://www.w3.org/1999/02/22-rdf-syntax-ns#", "http://qudt.org/vocab/unit/", "http://www.w3.org/2001/XMLSchema#"]* *}*

IV. DISCUSSION

The component demonstrated before is an engine for metadata management, where the information is organized with a specific schema. If new data arrive that do not follow the prescribed schema/ontology the reasoning will fail and will produce poor results. The data enrichment process it is based only on BRICK schema, thus the need for additional schemas will expand the reasoning capabilities of the component. Furthermore, the existence of a user interface that presents the resulted operation is required as the user will be helped to understand more the reasoning process than the resulted command line operations.

V. CONLUSIONS

This paper introduced a Reasoning Engine architecture for buildings that enables the usage of the BRICK schema along with property storages for pattern extraction from building metadata. A system implemented on top of the graph database in order to provide a RDF Pool, as the staging area for the building metadata. The stored metadata are transformed from triples to logical entities under the BRICK schema and then the search capabilities and querying are offered via HTTP methods.

Potential insights derived from the building lifecycle could be used from decision makers to design policies based on data driven applications that will aid to solve complex problems that European Union deals with, such as energy efficiency and energy poverty across member states. The building domain is a starting point for using innovative methods based on analytics results and conduct energy prediction, multi-criteria decision support and assessment.

As future work, the Reasoning Engine for building metadata proposed in the context of MATRYCS H2020 project will be enriched with more RDF triples from other large-scale pilots involved in the project. Furthermore, in the near future our goal is to provide an admin console where the admin user will be capable to manage ontologies and schemas [20], for instance to add, delete and update RDF files and building ontologies to the MATRYCS Reasoning Engine.

ACKNOWLEDGMENT

This work has been co-funded from the European Union's Horizon 2020 research and innovation programs under the MATRYCS project 'Modular Big Data Applications for Holistic Energy Services in Buildings', grant agreement No 101000158, and the VesselAI project 'ENABLING MARITIME DIGITALIZATION BY EXTREME-SCALE ANALYTICS, AI AND DIGITAL TWINS', grant agreement No 957237.

The authors would like to thank all the MATRYCS consortium partners and especially Daniele Antonucci (EURAC) and Aija Zučika (LEIF) for the fruitful discussions, remarks and observations during project meetings.

REFERENCES

[1] H. Doukas and A. Nikas, "Decision support models in climate policy," *European Journal of Operational Research*, vol. 280, no. 1, pp. 1–24, Jan. 2020, doi: 10.1016/J.EJOR.2019.01.017.

[2] V. Marinakis, H. Doukas, P. Xidonas, and C. Zopounidis, "Multicriteria decision support in local energy planning: An evaluation of alternative scenarios for the Sustainable Energy Action Plan," *Omega (Westport)*, vol. 69, pp. 1–16, Jun. 2017, doi: 10.1016/J.OMEGA.2016.07.005.

[3] V. Marinakis, A. G. Papadopoulou, G. Anastasopoulos, H. Doukas, and J. Psarras, "Advanced ICT platform for real-time monitoring and infrastructure efficiency at the city level," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2015, pp. 1–5. doi: 10.1109/IISA.2015.7387958.

[4] Y. Wei *et al.*, "A review of data-driven approaches for prediction and classification of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 1027–1047, Feb. 2018, doi: 10.1016/J.RSER.2017.09.108.

[5] K. Amasyali and N. M. El-Gohary, "A review of data-driven building energy consumption prediction studies," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 1192–1205, Jan. 2018, doi: 10.1016/J.RSER.2017.04.095.

[6] N. Naji *et al.*, "Actuator Networks Sensor and Energy-Aware Wireless Sensor Networks for Smart Buildings: A Review," 2021, doi: 10.3390/jsan10040067.

[7] V. Marinakis, H. Doukas, E. Spiliotis, and I. Papastamatiou, "Decision support for intelligent energy management in buildings using the thermal comfort model," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 882–893, Jan. 2017, doi: 10.2991/ijcis.2017.10.1.59.

[8] M. Petychakis, O. Vasileiou, C. Georgis, S. Mouzakitis, and J. Psarras, "34-47 A State-of-the-Art Analysis of the Current Public Data Landscape from a Functional, Semantic and Technical Perspective Journal of Theoretical and Applied Electronic Commerce Research ISSN 0718-1876 Electronic Version," vol. 9, pp. 34–47, 2014, doi: 10.4067/S0718-18762014000200004.

[9] O. Lassila, "Resource Description Framework (RDF) Model and Syntax Specification," 1998. [Online]. Available: http://www.w3.org/1998/10/WD-rdf-syntax-19981008

[10] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys*, vol. 40, no. 1, Feb. 2008, doi: 10.1145/1322432.1322433.

[11] B. Balaji *et al.*, "Brick: Towards a Unified Metadata Schema For Buildings", doi: 10.1145/2993422.2993577.

[12] V. Charpenay, S. Käbisch, D. Anicic, and H. Kosch, "An ontology design pattern for IoT device tagging systems," in *2015 5th International Conference on the Internet of Things (IOT)*, 2015, pp. 138–145. doi: 10.1109/IOT.2015.7356558.

[13] M. Poveda-Villalón and R. García-Castro, "Extending the SAREF ontology for building devices and topology." [Online]. Available: http://www.businessinsider.com/there-will-be-34-billion-iot-devices-installed-

[14] E. J. V. F. Moreira and J. C. Ramalho, "SPARQ ling Neo4J," in *OpenAccess Series in Informatics*, Sep. 2020, vol. 83. doi: 10.4230/OASIcs.SLATE.2020.17.

[15] P. Kapsalis, G. Kormpakis, K. Alexakis, and D. Askounis, "Leveraging Graph Analytics for Energy Efficiency Certificates," 2022, doi: 10.3390/en15041500.

[16] S. Ramanujam, A. Gupta, L. Khan, S. Seida, and B. Thuraisingham, "R2D: Extracting Relational Structure from RDF Stores," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2009, vol. 1, pp. 361–366. doi: 10.1109/WI-IAT.2009.63.

[17] V. Marinakis *et al.*, "From big data to smart energy services: An application for intelligent energy management," *Future Generation Computer Systems*, vol. 110, pp. 572–586, Sep. 2020, doi: 10.1016/J.FUTURE.2018.04.062.

[18] V. Marinakis, H. Doukas, K. Koasidis, and H. Albuflasa, "From Intelligent Energy Management to Value Economy through a Digital Energy Currency: Bahrain City Case Study", doi: 10.3390/s20051456.

[19] G. N. Papadakos, V. Marinakis, C. Konstas, H. Doukas, and A. Papadopoulos, "Managing the Uncertainty of the U-value Measurement Using an Auxiliary Set along with a Thermal Camera," *Energy and Buildings*, vol. 242, p. 110984, Mar. 2021, doi: 10.1016/j.enbuild.2021.110984.

[20] R. Calvo *et al.*, "Ontology-Based Production Simulation with OntologySim," 2022, doi: 10.3390/app12031608.