energies

MDPI

Article

Microservices Model to Enhance the Availability of Data for Buildings Energy Efficiency Management Services

Muhammad Aslam Jarwar, Sajjad Ali and Ilyoung Chong *

Department of Information and Communications Engineering, Hankuk University of Foreign Studies, Seoul 02450, Korea; aslam.jarwar@hufs.ac.kr (M.A.J.); sajjad@hufs.ac.kr (S.A.)

* Correspondence: iychong@hufs.ac.kr; Tel.: +82-10-3305-5904

Received: 12 December 2018; Accepted: 18 January 2019; Published: 23 January 2019

Abstract: In the Internet of Things (IoT)-supported energy data management infrastructure, objects from various energy generation and consumption terminals in buildings produce a tremendous amount of data. However, this data is not useful unless it is available on-time for services that discover meaningful information in order to provide intelligent decisions. The microservices-based data caching, data virtualization, data processing, data analysis, and data ingestion methods can be applied to enhance the data availability for energy efficiency management services provision across buildings. To foster building energy efficiency management services (BEEMS), Web of Objects (WoO) provides data abstraction, aggregation, and ingestion mechanism with virtual objects (VOs) and composite virtual objects (CVOs) by using ontologies and availability and scalability of services with microservices. This article proposes the use of data processing microservices modeling to enhance data availability and expose services capabilities with microservices for BEEMS. We present a semantic web agent based on an ontology for linking, enhancement, reusability, and availability of data-objects, services, and microservices. For the evaluation, we present a use case, which includes heterogeneous data collection and processing and provision of various BEEMS. A prototype for the use case scenario has been built and the results have been evaluated in the laboratory to mimic the enhanced data availability for BEEMS.

Keywords: energy data; energy management services; data and service availability; data processing microservices; internet of things (IoT)

1. Introduction

In the ubiquitous Internet of Things (IoT) infrastructure, millions of heterogeneous and distributed objects including very simple ones like sensors, actuators, RFID tags, and smart appliances as well as more complex objects, including self-driving autonomous vehicles and smart robots are connected and coordinated based on the various types of data availability and processing services [1,2]. The data availability is the dimension of data quality [3] and its techniques ensure that sufficient data continuously available for the provision of services. Moreover, the role of such type of services has been extended to many IoT-related applications, i.e., healthcare, self-driving autonomous vehicles, and energy management in buildings for curtailing the effects of climate change and to enhance the comfort of dwellers. In these areas, building energy efficiency management services (BEEMS) are a key aspect of sustainable economic growth and in reducing the global greenhouse gas emissions [4,5]. In 2017 buildings and the commercial sector in the United States consumed 39% of the total energy [6]. Similarly, the European energy consumption of buildings was approximately 40% of the total consumption and they emitted 36% of the CO_2 [7].

BEEMS solutions are required to collect, integrate, and analyze heterogeneous data (i.e., (1) energy data, (2) environmental data, including indoor and outdoor data, and (3) individual living pattern data) in a timely manner to enhance data availability for the efficient provision of services. Further, the BEEMS systems perform the profiling of buildings and apply the energy policies in order to enhance the efficiency of buildings in terms of cost, comfort and negative effects on the climate. The profiling and policies include benchmarking, rating, and labeling with respect to the type of energy. In the BEEMS, the failure of data provision services or insufficient data might have severe consequences such as the breakdown of energy to the buildings or the increase in billing cost and a decrease in the comfort of the living environment. The enhanced data availability and reusability can be achieved with the availability of quality of energy-related data-objects and correctly mapping of these objects with data processing and analysis services [8]. In the article, we use data-objects interchangeable with VOs and CVOs.

For the intelligent and efficient BEEMS provision, on-time data availability with lightweight services is crucial because data processing and analysis services enable people and smart agents to gain insights into the data to discover useful information for taking the right decision at the right time [9]. The enhanced data availability and services reliability and scalability are the non-functional requirements as in other information systems [10]. However, these are important for BEEMS; and can be modeled with microservices. The microservices-based BEEMS must provide efficient services with a lightweight mechanism in order to consume less energy at all times and should remain operational even it experiences the failure of some components or unavailability of sufficient data [11,12]. Data processing and analysis microservices are a set of lightweight granular services which can replace large monolithic systems [13,14]. Monolithic data processing systems are complex, they take many efforts to scale or upgrade and the failure of a component can downgrade the whole system [15]. Microservices-based BEEMS have great potential to meet the demand for 24/7 availability and can be measured as the percentage of time that the service is available with its corresponding data-objects.

To meet the requirement of enhanced data availability for such a demanding BEEMS request, there are many challenges. For example the dynamic discovery of existing, newly created, and alternate data-objects; identifying the context of BEEMS requests; scaling the existing services to accommodate more objects; orchestration of data processing microservices to suitably select, readjust and link data-objects, etc.

There are popular projects and commercially available platforms such as oneM2M, iCore, NiCE, and PLEEC in the smart city building energy efficiency management services domain [16–19]. However, there is a dearth of such a comprehensive platform which can leverage the data capture from various types of data-objects and provide on-time processed data-objects for BEEMS with high availability, reliability, and scalability with an efficient and modular microservice approach.

In order to resolve the aforementioned challenges, we propose a microservices model in the Web of Objects (WoO) platform, which leverages the enhanced data availability, and provision of services in the building environment. WoO is a simple but efficient layered architecture for the creation and deployment of effective and modular smart services [20–23]. It facilitates the collection and abstraction of data from a variety of heterogeneous data sources and fosters the service provision for energy efficiency management in buildings [4,24,25]. In WoO-enabled BEEMS, the service is modularized into a set of granular microservices which perform data collection, data cleaning, data transformation, and data analysis. For an efficient, and scalable BEEMS, the service load can also be distributed to granular data processing microservices which contain many instances and related processed data-objects.

Moreover, to foster enhanced data availability for BEEMS we propose a WoO architecture based on efficient, scalable and reusable data processing microservices. It includes semantic web agents based on ontology for readjustment, enhancement, and availability of data-objects, an information model for the creations of microservices template and services discovery and an execution model. Our contributions may be summarized as follows:

- We cover energy efficiency services availability, and scalability with the enhanced data processing and analysis microservices modeling in a layered WoO architecture;
- We proposed enhanced data processing microservices template information model in order to develop, deploy, discover, and manage BEEMS with enhanced data availability;
- We provide a BEEMS discovery and execution model with microservices, CVOs, and VOs data to accommodate and adjust the characterization of newly available real-world objects;
- We present a semantic web agent model based on ontology to detect missing data-objects and map the highly available data-objects to microservices.

The paper is organized as follows: Section 2 presents the background and related work; Section 3 describes microservices in the WoO architecture to enhance data availability for BEEMS; Section 4 discusses data availability in real-world scenario; Section 5 elaborates our microservices model for BEEMS data availability; Section 6 presents details of a use case, experiments and data description, and a prototype; Section 7 covered results and discussions; finally, Section 8 concludes the article.

2. Background and Related Work

In the early days data processing, data analysis, and service provision systems were developed with a monolithic approach [26]. In recent years, the new trends in data processing services are fostering the concept of microservices-based distributed and scalable data processing modules. These modules enhance the data availability for the provision of smart services because microservices foster efficient, reliable, and coherent functionality to enhance the data availability by processing the data at edge-level [27]. By using the microservices approach, we can scale and extend the service-oriented (SOA) architecture-based systems because the scalability and availability of IoT services is feasible [28]. Salvadori et al., [29] used data-driven microservices for the alignment of ontologies in a cross-domain scenario. As in the proposed model, we used semantic web notions, including Resource Description Framework (RDF), Web Ontology Language (OWL), Extensible Markup Language (XML) and SPARQL in VOs, CVOs-based data modeling and data processing microservices in order to provide modular, scalable, and efficient smart services with enhanced data availability. The data availability can be enhanced by reducing the services requests for the similar types of data and pre-processed the required data before the service request.

In [30,31], the authors used the quasi-copies method and cognitive data engines to enhance the data availability in information retrieval and IoT systems, respectively. The data and services modeling languages support runtime adaptability, reusability, availability, and heterogeneity in order to provide efficient IoT services [32]. Xively is a cloud-based platform for reliable data availability and services to support large numbers of IoT objects with RESTful APIs [33]. However, it does not provide service modularity with microservices and semantic relations among the services by using ontologies. The OneM2M standard has been developed to support a wide range of services across the application domains with enhanced data availability, and scalability. Its supported common service layer can be embedded in different software and hardware modules [34].

In order to increase the accessibility and availability of IoT data, the data needs to be virtualized with the VOs by using web technologies as the VO is a digital representation of IoT object over the web and the crucial component of modern IoT platforms [17,35,36]. In [37] the authors proposed an IoT framework for resolving the issues of unreliable associated services in a smart city use case scenario. In this framework, the authors divided their framework into VO, CVO, and service layer. However, the authors do not discuss the issues of service modularity, scalability, and availability in the case where one of service supported data-objects experiences a failure. In the case of missing data-objects, a similar data-object needs to mapped with the services automatically in order to enhance the data availability for services. The smart agent is another idea for mapping and providing an interaction among the data-objects and service-objects [38]. By extending the idea of mobile cloud computing, Bellavista et al., [39] proposed the human-driven edge computing framework to process the data near

the data-source for the provision of efficient services. This idea can also be applied to enhance the data availability for BEEMS by utilizing edge computing capabilities as microservices.

Energy efficiency management services based on big data analytics have been discussed in [10,40], for monitoring and optimizing the energy consumption in buildings. To monitor the energy load and consumption with respect to appliances various datasets and algorithms have been proposed [41,42]. The energy consumption data processing in real-time enhances the availability of processed data in order to provide BEEMS in a timely manner that can also enhance the user satisfaction level with less energy cost. Fadi et al., [43] purposed IoT middleware for the collection and integration of energy data in order to make timely production management decisions. However, in the architecture the authors analyze the energy data with a monolithic data processing and analytics software module. For the prediction of energy consumption in buildings physical and data-driven models have been widely used [44]. To evaluate the energy efficiency management in the buildings in terms of forecasting the energy demand and supply, k-means clustering techniques and IoT data analytics have been applied to over one-year energy data collected from buildings [45,46].

3. Microservices in Web Objects Architecture to Enhance Data Availability for BEEMS

The microservices in WoO foster highly available and scalable energy efficiency management services. For highly available and reliable smart services, the energy-related data from the real-world objects are collected, virtualized and annotated with the VOs. The energy-related data from VOs is aggregated and synthesized by the CVOs in order to extract the actionable knowledge for the execution of energy efficiency features. To achieve scalable and highly available BEEMS with enhanced data availability, the service has been granulated into the data processing and analysis microservices. The microservices-based WoO architecture is shown in Figure 1.

The VO layer and service layer provide an application programming interface (APIs) for the collection of data from third-party services (i.e., weather service) and provide interfaces to the BEEMS, respectively. In order to foster the enhanced data availability, data caching and data agility for analysis services, each layer contains separate databases. These databases are created and deployed in a triple store graph database software, which stores the resource description framework (RDF) and web ontology language (OWL)-based documents. In WoO one of the major reasons for using microservices is to enhance data availability and agility for analysis services and to overcome the limited scalability, availability, and agility of monolithic-based BEEMS.

The BEEMS API module contains the functions of RESTful Create, Read, Update and Delete (CRUD) operations. When a smart service request is brokered at the BEEMS API module, the request is authenticated by the services authentication module so that a tailored BEEMS can be provided to the user. After the service request authentication, the group of microservices is discovered based on templates for the execution of smart features inside lightweight containers. The containers provide an isolated environment and operating system level virtualization for controlling resources and processes to execute microservices with high availability and scalability [47]. In WoO, containers loaded with data analysis microservices support libraries, data processing methods and machine learning models in the form of microcode. In this article, we consider microcode as a small logic for a microservice to perform a certain task.

When the BEEMS requests arrive, the microservices discovery mechanism searches the related granular services by matching service context parameters. Initially, the discovery mechanism search for the static granular services from the microservices database because these microservices are already composed and instantiated. Static type microservices are fully loaded with filtered and aggregated data and these are fast in execution compared with the dynamically created microservices. Service composition (SC) has been used to combine granular services and related CVOs, and VOs data by using ontologies for composing a smart service from the group of microservices. SC module continuously works in a loop for checking, creating, and updating the services and

microservices templates based on the available, and reliable CVOs, and VOs data with corresponding real-world objects.

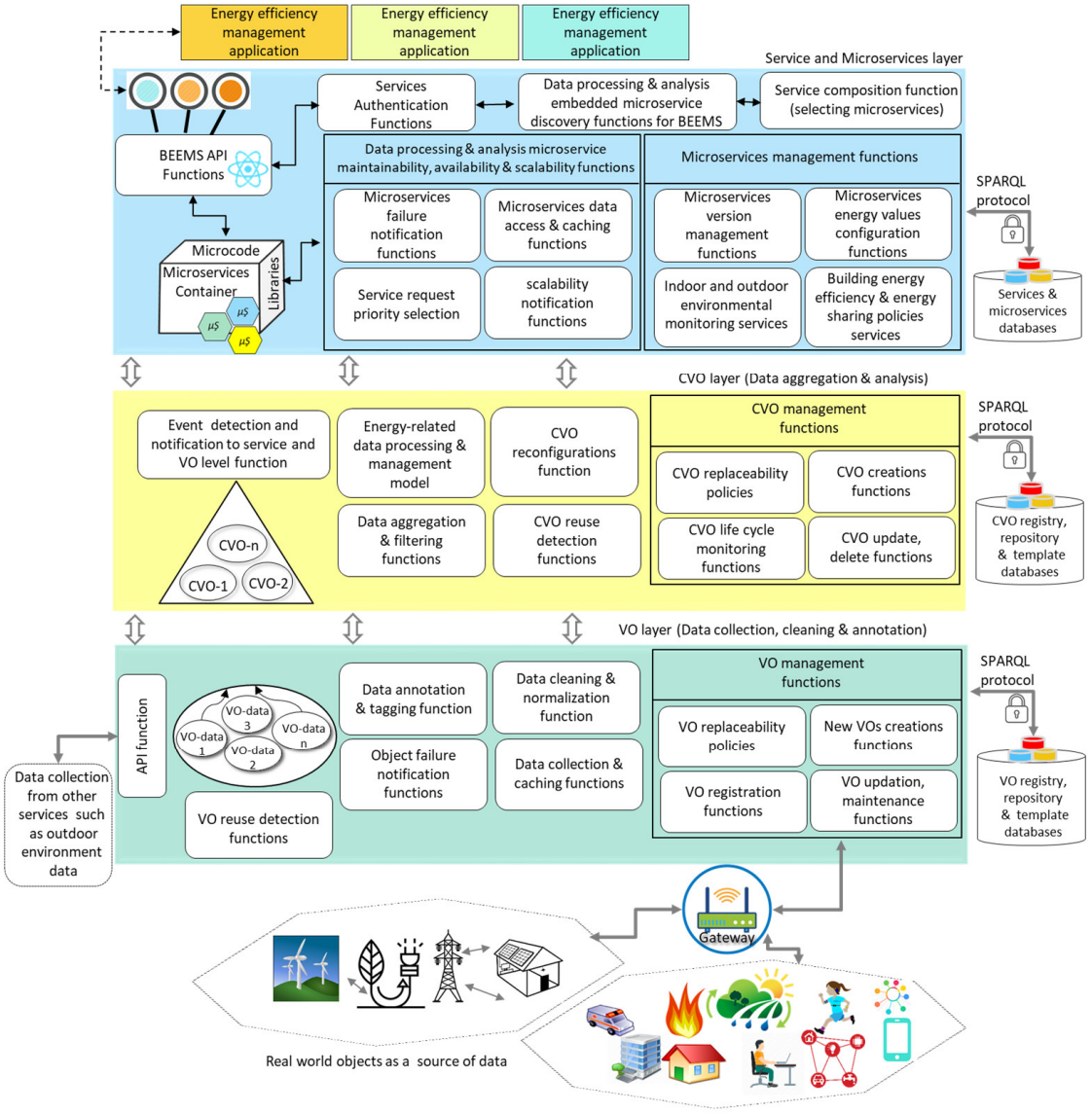


Figure 1. Microservices-based functions in Web Objects architecture to enhance data for buildings energy efficiency management services.

3.1. Microservices for Management Functions

To enhance the availability of the various type of energy-related data, including that from new and old objects for BEEMS, microservices version management is another important aspect. In WoO, the version management of microservices is part of the microservices architecture for continuous delivery of smart services based on various types of data such as environmental and user living pattern data. For the backward and forward compatibility, the major and minor versions of microservices are stored in the separate codebase. The microservices version management functions also handle different quality of services (service demand vs service cost) for BEEMS. To self-configure and manage microservices for different BEEMS requirement, the microservice management module includes the microservices energy values configuration function.

In the BEEMS provision, the service requests use a group of data processing and analysis microservices, and in that group, sometimes the response of one or more microservices is delayed

due to its corresponding data-objects. Due to the slow response of microservices, the BEEMS request may run for a longer interval. Therefore, it is necessary that the system should check the status of microservices and relevant real-world data-objects for services efficiency. In WoO, the health checking microservices module performs the logging in the microservice registry database for each microservice instance, and generates alerts upon the failure of instances, so the discovery and SC module can re-route the BEEMS requests to favorable microservice instances and available data-objects for the next service request. In the management microservices module, the functions continuously check the indoor and outdoor environment in order to enhance the energy efficiency management for the buildings. The microservices energy policy module applies the energy saving policies on the appliances with respect to indoor and outdoor environment parameters and dwellers' satisfaction level. In WoO the heterogeneous energy-related data from many sources are collected and harmonized with the VOs, and CVOs by using ontologies. The data processing functions use different types of task-oriented microservices to process VOs, CVOs annotated data and performs descriptive and predictive analytics based on energy efficiency use cases requirements. WoO data processing and analysis microservices are categorized as per the data analytics pipeline; which includes data extraction, cleaning, normalization, and energy consumption pattern identification.

3.2. Data Processing and Analysis Microservices Functions to Enhance Data Availability

One of the significance of microservices is that we can isolate inconsistent data-objects and execute the service partially. In WoO for reliable BEEMS, it is necessary that other dependent modules of the systems be notified about the inconsistency of data-objects and data caching and data virtualization must be incorporated. WoO provides object failure notification functions, which notify the service discovery module, and SC module about any failure of microservices, VOs, and CVOs. The link to the inconsistent objects will be disconnected, and this will prevent the use of these objects for the subsequent service requests.

In the case of the failure of a microservice, CVOs or VOs, a similar microservice, and the relevant CVOs and VOs will be remapped and reused in the BEEMS provision. The microservices data access and cache functions decrease the network cost, improve the service response, decrease the energy consumption by the objects, and ensure the availability of data for BEEMS. For the priority energy efficiency management service use case scenarios, the service priority selection module checks and assigns the priorities to the smart services from the priority mapping table. The service priority is assigned based on service level agreements and the business aspects of the service such as service quality vs cost. A service priority mechanism is also useful in resolving conflicts among two microservices which wants to use the VO with the highest data availability or the VO which represents the actuator. Further, all service requests cannot be redirected to the VO with highest data availability as the selection of VOs is based on service context parameters which include service priority and requested service quality.

To connect, accumulate and process a large amount of energy data and to provide smart services to the building dwellers, the service scalability represents an arduous task for BEEMS. WoO foster scalability modules which scale the number of service instances in three dimensions by following the principle of scale cube phenomena [48]. The scalability notification module scales the microservices, CVOs, and VOs at all three layers and sends a notification to each relevant module. The scalability module also supports in the service load balancing management when similar microservices, VOs and CVOs are reused at the same time into multiple service requests. Further, the interactions among the modules in WoO layered architecture to execute the BEEMS features with microservices, CVOs, and VOs is shown in Figure 2.

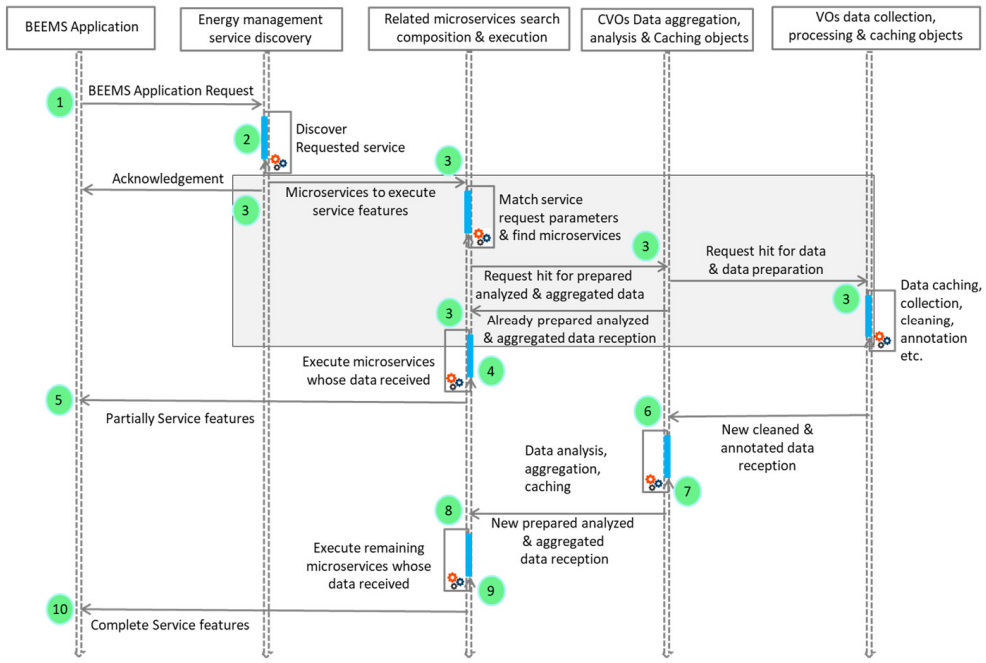


Figure 2. Interactions among the microservices-based modules in the WoO layered architecture to improve the data availability for BEEMS.

4. Data Availability in Real-World Scenarios

In the real-world, the continuity of a business or service depends on the availability of data with freshness and accuracy at the right time and at the right place. Energy management services use various types of energy data in order to provide efficient and reliable service. This energy data includes core energy data such as energy generation, distribution, and consumption data and other supplementary data such as environmental data, user occupancy data, and weather data. In the real-world scenario, the energy management services are using all these types of data to perform some task such as energy demand forecasting and management of comfortability in home buildings. If some attributes of the relevant data are missing that are required for the provision of requested service become unavailable, then the services may not perform their tasks well.

For example in a home building energy management scenario, the heating, ventilation, and air conditioning (HVAC) service controls and monitors the environmental condition of a room. This service uses many types of sensors such as temperature sensors, humidity sensors, smoke detection sensors, occupancy detection sensors, indoor and outdoor air quality checking sensors, etc. If the room temperature sensor becomes faulty and its current data is not available, should the HVAC service be stopped or will it manage the room environment wrongly because of unavailability of temperature data? There can be many solutions to mitigate this situation of unavailability of temperature sensor data. The most favorable are: (1) the HVAC service uses a virtual sensor to estimate the current temperature value based on previously available data; (2) use the temperature readings from similar sensors installed in another area of the home building, and (3) use the current temperature value from a weather service API. By using a similar type of temperature sensor values (similar VOs and CVOs) would support HVAC service to continue its tasks with less quality even though one of its data-object remains unavailable.

In the energy management services, the actors can be system administrator which manage the building energy management system, companies that produce and distribute energy, the government which defines policies, buildings users who consume energy and energy applications. The devices include sensors, actuators, smart meters, and microcontrollers such as Raspberry Pi which sense the environment, collect data and perform some actions on the things in the environment, respectively. The software can be a MQTT server which mediates the communication between the middleware and

devices, semantic agents which map and link the available data-objects to the services, databases to store energy data and data analytics tools to analyze energy data.

5. Microservices Model for BEEMS Data Availability

5.1. Data Processing and Analysis Microservices

A data processing and analysis microservice is a small, independent and autonomous minimal software module that can be developed and deployed independently by adding data mining and machine learning tasks. In a WoO architecture smart services for BEEMS are created from the group of data processing microservices and related VOs and CVOs data. Machine learning models are embedded in the microservices and may contain one or more CVOs and VOs data depending on the selected model and task. The more fine-grained data filtering and analysis results are formulated by reducing the models in each next step. In this model four types of data have been characterized for BEEMS, including data from user activities in the home and office building (ds1), indoor and outdoor environmental data (ds2), appliances' wise energy consumption data (ds3), and energy generation data (ds4). The data processing and analysis microservices functions to improve the data availability for BEEMS are illustrated in Figure 3.

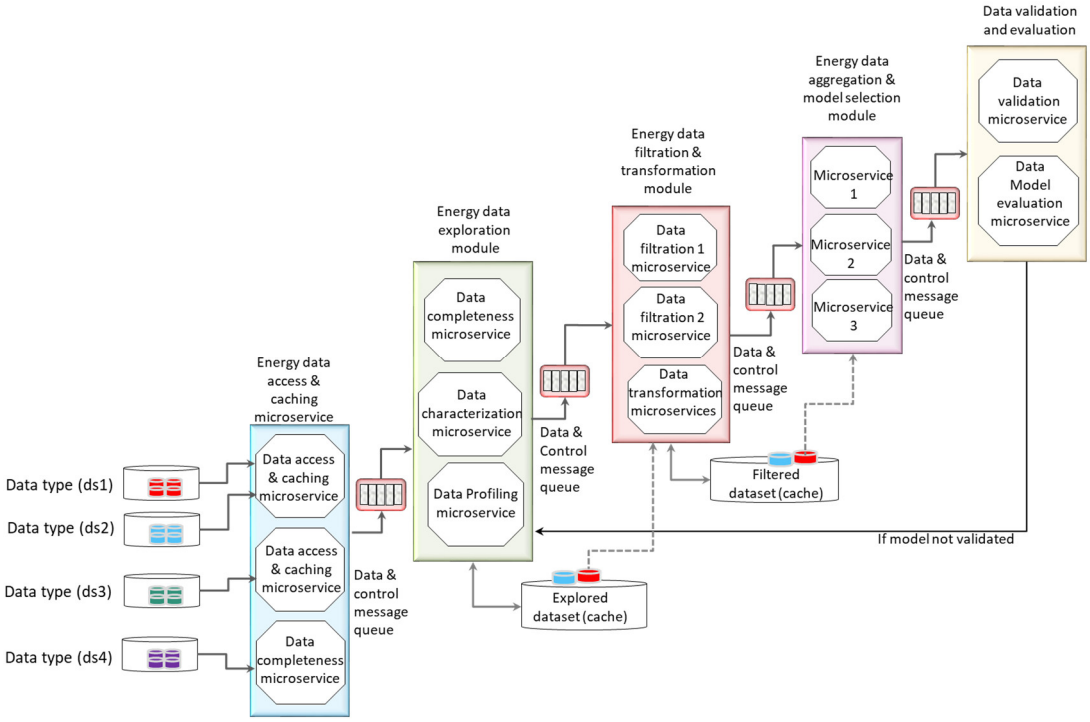


Figure 3. Microservices-based data processing and analysis functions to improve data availability for the services.

Data access and caching microservices use SPARQL queries to access the energy related-data forms ds1, ds2, ds3, and ds4 in a distributed manner. The data access and caching microservices cache the multiple data chunks with multiple service instances and if one instance fails then another instance takes on the responsibility. Data access and caching microservices foster the availability of data for the subsequent process in order to allow the timely provision of BEEMS. Data exploration microservices include the functionality of data characterization, data completeness checking, filling missing data values, and data profiling. The data exploration microservices also identify the patterns and relationships among the data by using VOs and CVOs ontologies. The processed data from the data exploration stage is stored in the temporary cache dataset in order to speed up the process for the next stage.

The data filtration and transformation microservices have been used for transforming explored data in the required format. These microservices transform the data according to given policies such as batch transformation, real-time transformation and normalization, fixing outliers and mapping transformation. Multiple instances of microservices can be applied in large dataset transformation to reduce failover and speed up the process. Data aggregation microservices combine the data-objects based on the similar data types and context by using statistical tools and methods according to the given policies. In order to automate the process, the data aggregation and model selection policies can be represented in the ontology. In the last, the selected model validated according to the required model accuracy threshold and executed with multiple microservices.

The significance of using the microservices model for data processing and analyzing is to modularize the process of execution of a large batch of SPARQL queries with lightweight microservice APIs in order to achieve the high availability and scalability of data for BEEMS. Microservices-based models support decentralized data management and execution of complex monolithic SPARQL queries in a simple way by minimizing the usage of join-based queries.

5.2. Microservices Template Information Model

In a WoO platform the microservice template is a crucial component along with VO and CVO templates for the creation, maintenance and discovery of data processing microservices in order to provide highly available data for BEEMS. These microservices are created, discovered, and maintained based on the microservices template. The template contains microservices metadata which is stored in the microservices template database. An information model for the microservices template is shown in Figure 4.

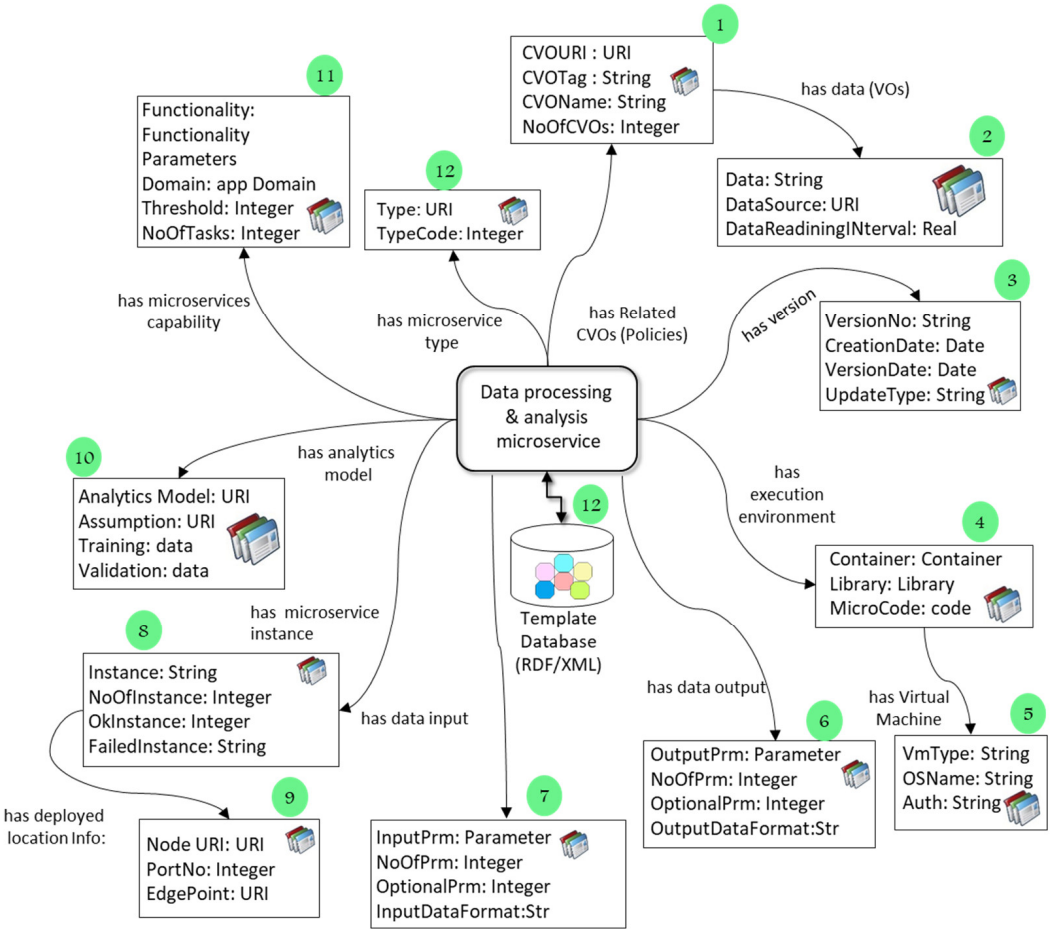


Figure 4. Microservice template's information model to create and discover microservices for BEEMS.

In the microservices template information model, node 01 contains the relevant CVO and VO information which provides data and metadata to the microservice, such as CVO URI, CVO name, CVO unique tag and number of CVOs required in the particular microservice. The CVO URI and CVO tag uniquely identify the CVO, and the CVO name is a more contextual name for human understanding whereas the CVO tag and URI is for the machine-to-machine (M2M) communication and understanding. The node 02 contains the information about the source of the data, which is required to accumulate, preprocess and feature selection for the analytics as a service. Node 03 contains the version information which also includes data processing and analysis microservice creation date, current version updating date, and semantic update type (major, minor, patch). To ensure scalability, and fault tolerance data processing and analysis microservices, each microservice deployment needs a separate execution environment. The execution environment information node (04 and 05) includes container information, microservice related support library information, and relevant microcode. The microservices input and output information nodes (06 and 07) contains input and output properties. These properties foster the SC, microservices discovery, and availability of data and services for BEEMS.

For the availability of BEEMS, the microservice instance information node (08 and 09) contains the detail that whether microservice has another instance, how many instances are currently available and reliable, and how many instances are inconsistent or have poor health. The sub-node 09 also has location information (LI) as sub-nodes. LI sub-node contains the microservice deployed URI, port number and URI of edge, in case of edge deployment. The microservices template also contains information about the analysis model, assumptions for the model, and what is the training data and validation data for applying the model (node 10).

5.3. Microservices Based BEEMS Discovery and Execution Model

To foster modular and scalable WoO-enabled BEEMS with enhanced data availability, we distribute the service into granular microservices. Each microservice contains data processing models and data-objects in the form of VOs and CVOs. The microservices are selected based on the service context, and priority. In a WoO-enabled data processing and analysis microservices model, each service object has a predefined threshold and priority of execution. The predefined execution thresholds and service priorities can be different for a similar type of service because of microservice execution cost, energy efficiency monitoring and management feature cost and service demands. In the model, the energy efficiency management service is requested with contextual parameters, which includes service priority and execution threshold.

Upon receiving the service request, the *findSmartService* function takes the input as a service request, and it searches the requested service from the services database and returns the service request object (*sro*). The *sro* contains a BEEMS identifier, BEEMS priority, BEEMS application domain, and BEEMS execution threshold. The service identifier is the unique id which recognizes the service for the user in WoO enabled BEEMS. In *sro*, every BEEMS request contains the service priority information. For example, the fire detection in smart building and closing of all relevant appliances service has more priority than setting the air-conditioning thermostat.

The *sro* contains the information regarding the application domain of the service because the same service can be reused in different application domains with different rules and threshold. As these rules are defined in relevant CVOs and executed by the microservices, for example, 39° temperature has a different meaning in weather and healthcare domain services. The *sro* also contains the service execution threshold, that shows the maximum execution time limit of the service. The *sro* of a service can be defined as: $sro = \{sid \mid sid = (spi, spd, sth)\}$, where *sid* is the energy efficiency management service identifier, *spi* is the service priority, *spd* is the specific application domain, and *sth* is the threshold of service execution.

The *findDataProAnalysisMicroservices* function takes the *sro* as input for searching the relevant microservices and CVOs by semantically matching the *sid* and microservices application domain. This *findDataProAnalysisMicroservices* function issues the SPARQL query to the microservices

and CVO databases for the microservices and relevant object discovery. The function *findDataProAnalysisMicroservices* returns the microservices object (*µŞo*) which includes building environment application domain-specific data processing microservices and CVOs with execution threshold. In this case, *µŞo* contains the list of execution threshold for each microservice in the group and its relevant CVOs. Thus we can describe the *µŞo* for *sro* as $\mu Şo = \{\mu Şgid \mid \mu Şgid = \sum(\mu Şth, CVOth)$, where *µŞgid* is the group identifier of the data processing microservices for the BEEMS request, *µŞth* is the microservice execution threshold and CVO*th* is execution threshold of the CVO.

The output (*µŞo*) from the *findDataProAnalysisMicroservices* function and output (*sro*) from the *findSmartService* function have been sent to the *checkAvailability* function. The *checkAvailability* function takes the *sro*, and *µŞo* as input parameters and returns the group of microservices and related service policies in the form of CVOs which match the execution threshold of the requested service. If the BEEMS threshold is matched with the sum of a selected group of data processing microservices and their relevant CVOs, the service features will be executed with the evolutionary energy consumption predictive model, otherwise, the "service not available message" payload will be sent in the response. The evolutionary energy consumption predictive model learns the energy consumption situation from the environmental data, weather data, user living pattern data, and energy consumption history data. The discovery and execution model of microservices based BEEMS is shown in Figure 5.

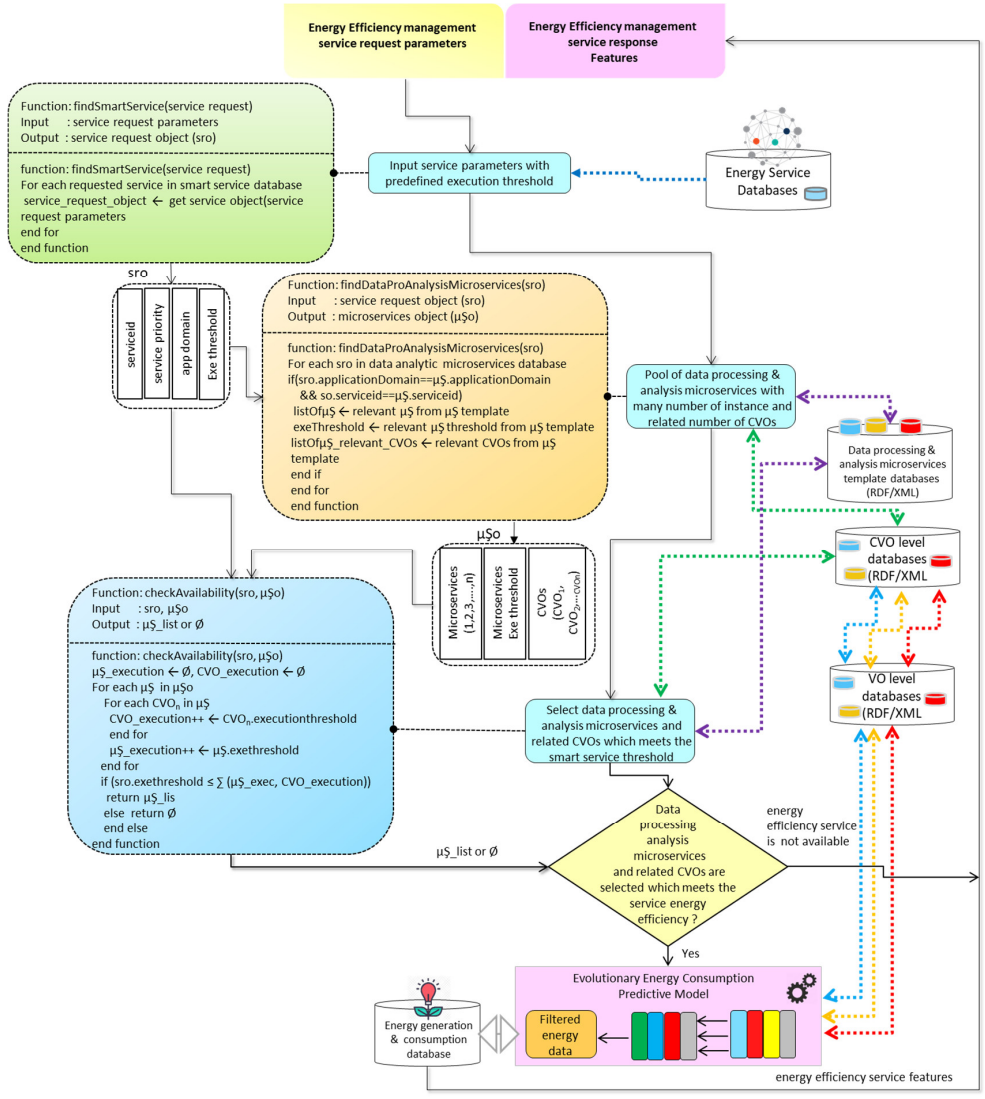


Figure 5. Microservices-based services discovery and execution model for the provision of BEEMS request.

5.4. Microservices Model to Enhance Data Availability and Services for Building Energy Efficiency Management Applications

5.4.1. Formal Description and Motivation of the Microservices Model

In WoO-enabled BEEMS, the functions of services have been modularized and distributed into several granular data processing and analysis microservices. The BEEMS provision model has been created and deployed with many numbers of microservices instances and replicated VOs in order to minimize the risk of service failure and ensure the maximum availability of data for services with CVOs and VOs. One of the main motivations of a microservices model is to enhance data availability for services through reusability of similar VOs, CVOs, and microservices in multiple services and the continuity of service in any situation when some of the data-objects become unavailable. The reusability of objects and continuity of service improves the system in terms of scalability and reliability of services.

In the model, each data processing and analysis microservice contains relevant CVOs and VOs for data collection, (the data collected from sensors, actuators, buildings, smart meters, and appliances) data caching, data representation, data exploration, data filtration, data aggregation, and data ingestion. A WoO-enabled microservices model that enhances the availability of data and services for BEEMS is shown in Figure 6.

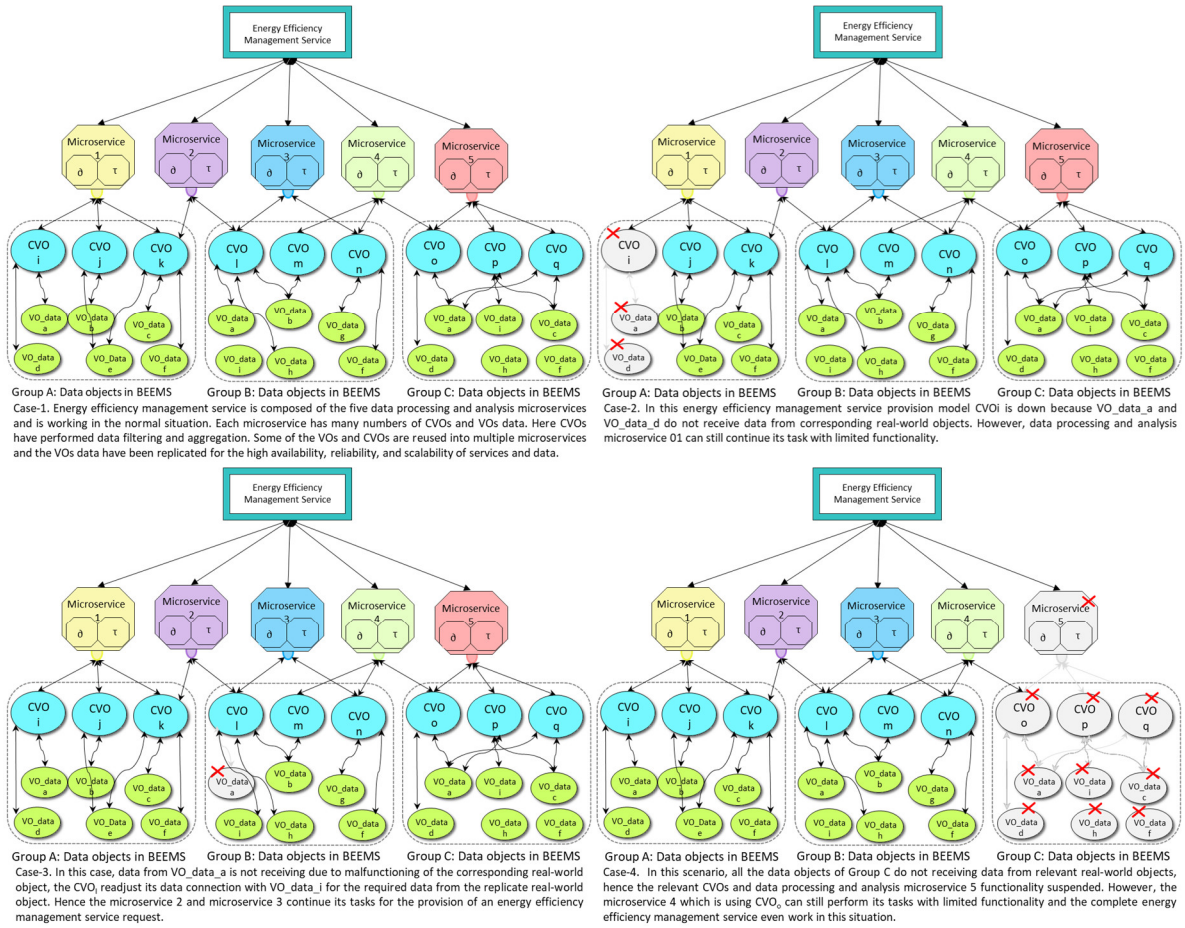


Figure 6. Microservices model to enhance data availability and services for BEEMS.

5.4.2. Mathematical Description and Representation of Microservices and Related Data-Objects with Different Cases

For ease of understanding, we illustrate the enhanced data availability model with five microservices in four different cases as shown in Figure 6. In the model, the BEEMS request tasks have

been distributed to five microservices. Therefore, the BEEMS request (*esr*) can be defined as a set of data processing microservices as follows:

$$esr = \{\mu\S_1, \mu\S_2, \mu\S_3, \mu\S_4, \mu\S_5\} \tag{1}$$

The data filtered and aggregated by CVOs and the corresponding VO data have been modeled into three groups in each case for the clear illustration of enhanced availability and reusability of different data-objects with microservices. In the model, the microservices are also reused in multiple BEEMS requests with their VOs and CVOs. Similarly many VOs and CVOs are reused in multiple microservices. The reusability of similar VOs and CVOs [22,24] does not negatively affect the services load balancing and execution because the scalability function in the WoO architecture scales the VOs and CVOs into many instances as per the requirement of microservices. For simplicity, we present four different cases of a BEEMS request as follows:

Case 1. A BEEMS request has been executed in a normal situation. The normal situation is a situation in which all the data-objects in the model are available for the provision of service. In the normal situation, the service request is accomplished with full capacity, i.e., with five microservices and all relevant CVO and VO data. The *esr* with five microservices and all their relevant data-objects can be defined as in Equations (2a) to (2e):

$$uS_1 = \{(\text{CVO}i \in \text{VO}a \wedge \text{VO}d),\ (\text{CVO}j \in \text{VO}b \wedge \text{VO}e),\ (\text{CVO}k \in \text{VO}c \wedge \text{VO}e \wedge \text{VO}f)\} \tag{2a}$$

$$uS_2 = \{(\text{CVO}k \in \text{VO}c \wedge \text{VO}e \wedge \text{VO}f),\ (\text{CVO}l \in \text{VO}a \wedge \text{VO}b \wedge \text{VO}h)\} \tag{2b}$$

$$uS_3 = \{(\text{CVO}l \in \text{VO}a \wedge \text{VO}b \wedge \text{VO}h),\ (\text{CVO}n \in \text{VO}f \wedge \text{VO}g)\} \tag{2c}$$

$$uS_4 = \{(\text{CVO}m \in \text{VO}b),\ (\text{CVO}n \in \text{VO}f \wedge \text{VO}g),\ (\text{CVO}o \in \text{VO}a \wedge \text{VO}d)\} \tag{2d}$$

$$uS_5 = \{(\text{CVO}o \in \text{VO}a \wedge \text{VO}d)(\text{CVO}p \in \text{VO}a \wedge \text{VO}c \wedge \text{VO}i),\ (\text{CVO}q \in \text{VO}a \wedge \text{VO}c)\} \tag{2e}$$

The microservice 1 ($\mu\S_1$) contain three CVOs (CVO*i*, CVO*j*, and CVO*k*) and each CVO contains different types of VOs (VO*a*, VO*b*, VO*c*, VO*d*, VO*e*, and VO*f*) as mentioned in Equation (2a). The microservice 2 ($\mu\S_2$) representation is shown in Equation 2b and it contains two CVOs (CVO*k*, CVOl) and each CVO contains different types of VOs as in $\mu\S_1$. The VOs in $\mu\S_2$ includes VO*a*, VO*b*, VO*c*, VO*e*, VO*f*, VO*h*, and VO*l*. The microservice 3 ($\mu\S_3$) also contains two different CVOs (CVO*l* and CVO*n*) and many numbers of VOs as shown in Equation (2c). In the $\mu\S_3$, CVO*l* is the composition of VO*a*, VO*b*, and VO*h*; and CVO*n* is the composition of VO*f* and VO*g*. The composition of microservice 4 ($\mu\S_4$) is shown in the Equation 2d. In this microservice, CVO*m* includes only a single VO (VO*b*) and CVO (CVO*n*). The last microservice 5 ($\mu\S_5$) is composed of three CVOs (CVO*o*, CVO*p*, and CVO*q*) and four VOs (VO*a*, VO*c*, VO*d*, VO*i*) as shown in Equation (2e).

Case 2. In this case, $\mu\S_1$ depends on data-objects (CVO$i \in$ VO$a \wedge$ VOd), (CVO$j \in$ VO$b \wedge$ VOe), (CVOk $\in$ VO$c \wedge$ VO$e \wedge$ VOf), but the data from the objects including one CVO and two VOs (CVO$i \in$ VOa $\wedge$ VOd) become unavailable due to disconnection with the corresponding real-world objects in the buildings' environment. In this case, the functionality of $\mu\S_1$ will be degraded. However, the data for the whole service may not be unavailable and the service will continue its operation. The related pending tasks with $\mu\S_1$ will be queued and processed upon the availability of data from the set of data-objects (CVO$i \in$ VO$a \wedge$ VOd). In this model, the set of unavailable data-objects can be represented with the prime symbols as shown in Equation (3) and all the remaining available data-objects will be the same as in Equation (2a).

$$\text{CVO}i' = \left(\text{VO}a' \wedge \text{VO}d'\right) \tag{3}$$

Case 3. In the third case, $\mu\S_2$, and $\mu\S_3$ both are using data from a set of data-objects (CVO$l \in$ VO$a \wedge$ VO$b \wedge$ VOh), but the VOa does not receive data from the corresponding object. Therefore, the CVOl

start its self-healing process for initiating the disconnection with the VO*a* and connected with the VO*i*. Because in this group VO*i* provides the replicated data from the other real-world object in place of VO*a*. So now, the new set of objects for the $\mu\S_2$ and $\mu\S_3$ can be represented as in Equations (4a) and (4b). Hence the complete *esr* continues its requested features with $\mu\S_2$ and $\mu\S_3$ without degrading the functionality of the service:

$$uS_2 = \{(\text{CVO}k \in \text{VO}c \wedge \text{VO}e \wedge \text{VO}f),\ (\text{CVO}l \in \text{VO}a' \wedge \text{VO}i \wedge \text{VO}b \wedge \text{VO}h)\} \tag{4a}$$

$$uS_3 = \{(\text{CVO}l \in \text{VO}a' \wedge \text{VO}i \wedge \text{VO}b \wedge \text{VO}h),\ (\text{CVO}n \in \text{VO}f \wedge \text{VO}g)\} \tag{4b}$$

Case 4. In this case, data from all the data-objects in group c become unavailable. Because of the malfunctioning of corresponding real-world objects or any other reason; and $\mu\S_5$ stops functioning, although, $\mu\S_4$ will still continue its tasks with degraded functionality and fulfill the service request with the high availability and reliability. The autonomic semantic web agent (Section 5.5) now considers the $\mu\S_4$ and $\mu\S_5$ data-objects with notations (′) as unavailable as shown in Equations (5a) and (5b):

$$uS_4 = \{(\text{CVO}m \in \text{VO}b),\ (\text{CVO}n \in \text{VO}f \wedge \text{VO}g),\ (\text{CVO}o' \in \text{VO}a\prime \wedge \text{VO}d')\} \tag{5a}$$

$$uS_5{}' = \{(\text{CVO}o' \in \text{VO}a' \wedge \text{VO}d')(\text{CVO}p' \in \text{VO}a' \wedge \text{VO}c' \wedge \text{VO}i'),\ (\text{CVO}q' \in \text{VO}a' \wedge \text{VO}c')\} \tag{5b}$$

In all the cases we can find the data availability for CVOs by using Equation (6). In the Equation, CVO*x* represents any CVO which is used as aggregation function in the microservice that required a number of VOs such as in Equation (6) the data availability from the derivative of two VOs. The same way we can also find the data availability for the microservice based on its relevant CVOs as mentioned in Equation (7). In Equation (7) uSx represents any microservice which contains many types of CVOs (i.e., CVO*x*, CVO*y*, CVO*z*) and the single quotation (′) symbol represents the derivative of the term:

$$\text{CVO}x = \left\{1 - \left(1 - (\text{VO}x.\text{VO}y)'\right)^2\right\} \tag{6}$$

$$uSx = \left\{1 - \left(1 - (\text{CVO}x.\text{CVO}y)'\right)^2\right\} \tag{7}$$

The total data availability for the BEESMS application request can be calculated based on the total number of microservices used in service execution. The decision to execute the BEEMS request with full or partially features can be decided based on the threshold value of the required microservices as indicated in Equation (8), where *fs* is a function of BEEMS which represent that the required microservices are available with their data-objects:

$$fs = \begin{cases} 1, & \frac{1}{m}\sum_{x=1}^{m}|uSx| \geq threshold_{av} \\ 0, & otherwise \end{cases} \tag{8}$$

In the microservices-based enhanced data availability model, all the calls among the objects are asynchronous with a timeout and circuit-breaker functionality in order to enhance data availability and avoidance of recursive failures. Further, the pseudo-code to enhance the data availability for the efficient microservices based services executions is detailed in Algorithm 1.

Algorithm 1 Algorithm the enhance data availability for efficient microservices-based service execution

Given Ψ (service request parameters) and case (ϒ)

Task To-do: Find related μŞ (microservices) and highly available data-objects to execute service

```
1.  Procedure exe_svc_data_objects(Ψ, ϒ)
2.  μŞØ→ microservices template database
3.  CVØ→ CVO template database
4.  VOØ→ VO template database
5.  μ[]→ microservice list
6.  Ω[] → CVO (Composite Virtual Object) list
7.  ∂[] → VO (Virtual Objects) list
8.  Ƭ → flag (initially false)
9.  fs → function indicate data availability for service through microservices
10. for (j=1; j= Ψ in μŞØ; j++)
11.     μ[j] ← get matched microservice based on Ψj
12.     for (k=1; k= SizeOf (μŞ[j]) in CVØ; k++)
13.         Ω[k] ← get matched CVOs based on μŞ[k]) templates in CVØk
14.         for (m=1; m= SizeOf (CV[k]) in VOØ; m++)
15.             ∂[m] ← get matched VOs based on CV[k]) templates in VOØm
16.         end for
17.     end for
18. end for
19. if μŞ1, μŞ2, μŞ3, μŞ4, and μŞ5 are in μ and ϒ and case-1 then
20.         do_service_request_execution and exit()
21. else if μŞ1, μŞ2, μŞ3, μŞ4, and μŞ5 are in μ and ϒ and case-2 then
22.         Ƭ←fs(CVOi in Ω && VO_data_a, VO_data_d in ∂)
23.             if (Ƭ) then do_service_request_execution and exit()
24.             end if
25.     else ischeckreplicatefor(VO_data_a, VO_data_d) and map to CVOi
26.     else send_temp_suspend_message_for μŞ1 and do_service_request_execution_partially
27. else if μŞ1, μŞ2, μŞ3, μŞ4, and μŞ5 are in μ and ϒ and case-3 then
28.         Ƭ←fs(CVOi in Ω && VO_data_a in ∂)
29.             if (Ƭ) then do_service_request_execution and exit()
30.             end if
31.     else map CVOi to VO_data_i and update and do_service_request_execution
32. else if μŞ1, μŞ2, μŞ3, μŞ4, and μŞ5 are in μ and ϒ and case-4 then
33.         Ƭ←fs(CVOo, CVOp and CVOq in Ω && VO_data_a, VO_data_i, VO_data_c in ∂)
34.             if (Ƭ) then do_service_request_execution and exit()
35.             end if
36.     else ischeckreplicatefor(VO_data_a, VO_data_i, VO_data_c) and map to CVOo, CVOp
            and CVOq
37.     else send_temp_suspend_message_for μŞ5 and do_service_request_execution_partially
38. end if
39. end Procedure
```

5.5. Semantic Web Agent

In order to enhance the availability of data and services for BEEMS requests, the semantic web agent extracts the knowledge, map and maintain the link among the data-objects, microservices, and services based on the ontology. The web agent semantically navigates in the large and complex energy-related knowledge base and extracts the knowledge of the available and connected data-objects. In order to solve the problem of data-objects linking and produce more smart features for BEEMS, the semantic web agent has been designed based on a WoO layered architecture, to efficiently identify the available microservices, CVOs, and VOs data. It uses semantic rules designed in Semantic Web Rule Language (SWRL), RDF and OWL. If there is any data-object does not available or missing, it tries to map the link with another related data-object in the knowledge graph.

The semantic web agent model considers services, the data processing microservices, CVOs, and VOs data as a first class citizens and their instances as individuals. To execute service features for BEEMS request, the rules have been defined with a minimum number of necessary highly available data-objects and reliable microservices. For example, the rules $\mu\S_1 \rightarrow$ *hasCVO some CVOs*, $\mu\S_1 \rightarrow$ *hasCVO min* 2 *CVOs*, *esr* $\rightarrow$ *has* has$\mu\S$ *some* $\mu\S$*s and esr* $\rightarrow$ has$\mu\S$ *min* 4 $\mu\S$*s*; represent that $\mu\S_1$ has some CVOs, $\mu\S_1$ has required at least two CVOs to execute assigned tasks, energy service request has some $\mu\S$*s* and this service required at least four microservices to successfully execute the smart service features respectively. Figure 7 shows an excerpt of the semantic web agent ontology model for the navigation in an energy-related data and knowledge graph.

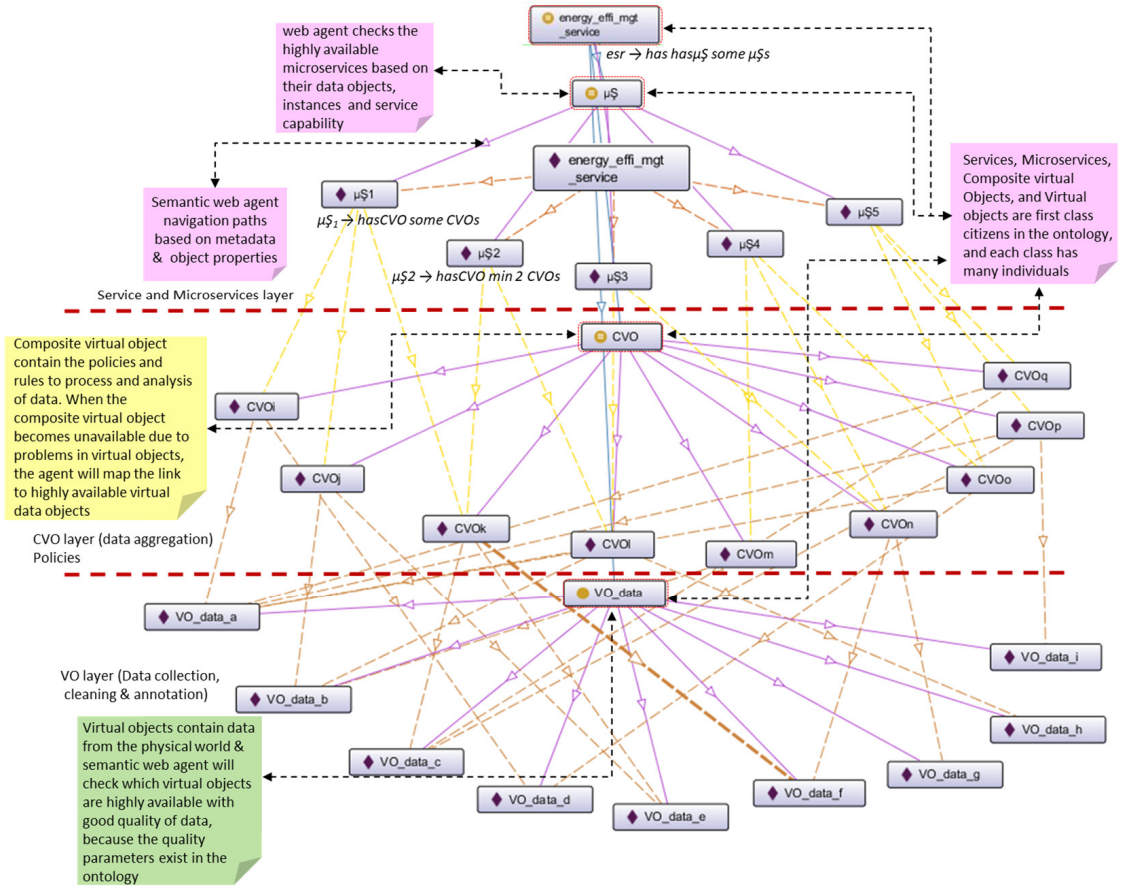


Figure 7. Semantic web agent interaction model illustration for semantically navigation in energy-related data and knowledge graph.

6. A Use Case and Prototype

In this section, we present a use case in which a variety of energy-related data is collected and processed by using microservices, CVOs, and VOs. The microservices model to enhance the data availability for BEEMS is a generic and can be applied to a range of use cases.

6.1. Use Case Details

In the energy management process data is generated in a huge quantity but due to the lack of coordination and workflow mechanism among the different entities it is not processed and utilized efficiently for purposeful BEEMS. In order to enhance the data availability, data caching, data aggregation and data analysis for the BEEMS, we create data processing microservices. These microservices uses data aggregation policies in the form of CVOs over the VOs energy-related data and apply data mining pipelines and machine learning models. The demonstrated use case

includes many main entities. Each entity has sub-entities with separate smart meters, energy sensors, and appliances actuators, as illustrated in Figure 8.

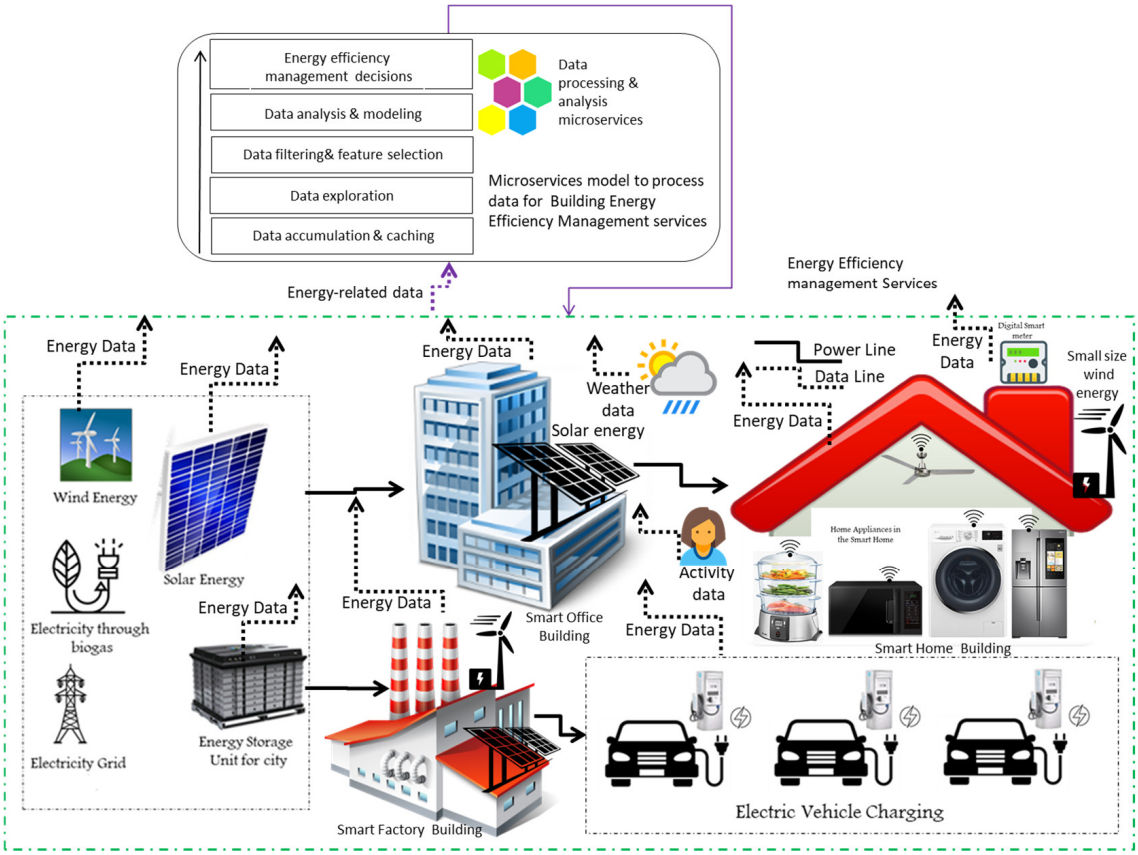


Figure 8. An illustration of use case scenario shows a variety of energy-related data which needs to be enhanced for buildings energy efficiency management services.

In the use case, the energy is produced by urban small size wind turbines, solar grid panels, and building waste in the form of biogas. Then subsequently, this energy is stored in energy storage units and distributed to the buildings through national grids. By the utilization of BEEMS, the use of energy shall be optimized in order to save energy. The saved energy can be transferred to other consumers through national grids. During this process of energy production, distribution, and consumption, a big amount of energy-related data is generated. In the use case, all this data is recorded in a real-time and transported with lightweight IoT protocols (i.e., MQTT and COAP) to the data servers. This energy-related data is monitored and semantically aggregated and annotated with CVOs, and VOs and processed with analysis microservices. The special monitoring mechanism is created with data processing and analysis microservices for different uses such as for the home consumer, electric vehicle charging stations and energy prosumers. The scalable energy efficiency management services based on enhanced data processing microservices can accommodate more data-objects through VOs and CVOs in the future. This BEEMS-based system contains replicated data-objects, data caching mechanism and data processing microservices instances so the data can be highly available for service provision.

6.2. Prototype Description

In order to demonstrate the proposed model, we designed a prototype to mimic the use case in the laboratory environment. In the prototype, we used four virtual machines running on the Ubuntu operating system. The first and second virtual machines have been used to store the energy-related data. We used the OpenTSDB database, which is suitable for the scalability and massive amount of

time series energy-related data. We also used lightweight messaging MQTT and COAP protocol with RabbitMq and data caching service for efficient and reliable data feeds.

The third virtual machine has been used for the deployment of VOs and CVOs templates, ontologies, and data-objects. The VOs and CVOs ontologies have been created with the Protégé and stored in the Apache Jena triple store [49,50]. To access VOs, and CVOs data-objects from the triple store, the Apache Jena SPARQL engine has been used. In the third virtual machine, we also hosted the codes for the functions of data caching, cleaning and annotation. The data from all the relevant VOs have been cleaned, filtered and aggregated by the data processing, aggregation and reusability microservices, and theses microservices also reuses the VOs data at the placement of non-available VOs by using semantic web agent. To predict energy generation and consumption in real-time, and to know the consumption pattern (households energy consumption pattern from their lifestyle, vehicle energy consumption pattern from the driving style of the user), we used Apache Spark's machine learning libraries (ASMLL) because ASMLL supports open source fault tolerant, scalable, and real-time big data analysis and prediction models.

The VOs and CVOs data-related functionality from the first and second virtual machine has been exposed as RESTful services. We developed data processing and analysis microservices by using Spring Framework and Apache Spark's machine learning libraries. The data processing and analysis microservices were created depending on the BEEMS applications requirements by using microservices templates. The developed microservices have been deployed on Docker version 18.05.0-he including Apache HTTP Server with WebSphere application server in the fourth virtual machine. Some of the energy efficiency management services created from the group of microservices are: (S1) Energy Generation & Load Estimation Analytics for Home, (S2) Home building Appliances Energy Consumption Prediction, (S3) Electric Vehicle Charging Billing & Reporting, (S4) Energy Sharing Billing & Reporting to the Prosumers, (S5) Energy Optimization based on User Movement Tracking in Home & Office, and (S6) Energy Saving Prediction in Home building Environment. The porotype model for the use case is shown in Figure 9.

6.3. Experiment and Data Description

To evaluate the proposed model and use case scenario we created an experimental setup for prototyping with synthesized and real energy data. The synthesized energy-related data generated with Log-Synth tool [51], and a large publically available energy dataset GREEND [41] is used as real data. We used synthesized energy-related data in order to evaluate the performance of the microservices model in terms of high availability of data in near real-time for BEEMS provision. The reason to use synthesized energy-related data is to assume that we are receiving this data from smart meters and sensors in near real-time. In order to analyze and identify the relationship among the energy generation, and consumption we distribute the data into two data types: (1) the energy generation data and (2) energy consumption data, and converted these datasets into RDF/OWL formats using VOs and CVOs ontologies. The energy generation data attributes includes: energy_generator_building_id, generation_type, max_generation_capacity, actual_generated (KWH or BTUs), and timestamp. We further distribute the energy consumption dataset attributes into two sets: total energy consumed by the building and energy consumption by the appliances separately in a house building or office building. Data attributes for the total energy consumption includes building_id, no,_of_electric_appliances, no_of_gas_appliances, consumption_1 (KWH), consumption_2 (m^3), and timestamp. Appliances wise energy consumption data attributes include building_id, appliances_id, consumption_1 (KWH), consumption_2 (m^3), and timestamp. The Log-Synth tool [51] and GREEND [41] data are initially stored in the energy generation and consumption databases respectively. The stored data semantically annotated with VOs ontologies and stored in RDF/OWL format in VOs database. For this experiment, we created separate energy generation data VOs with a different time interval and the type of energy such as wind, hydro or biofuel.

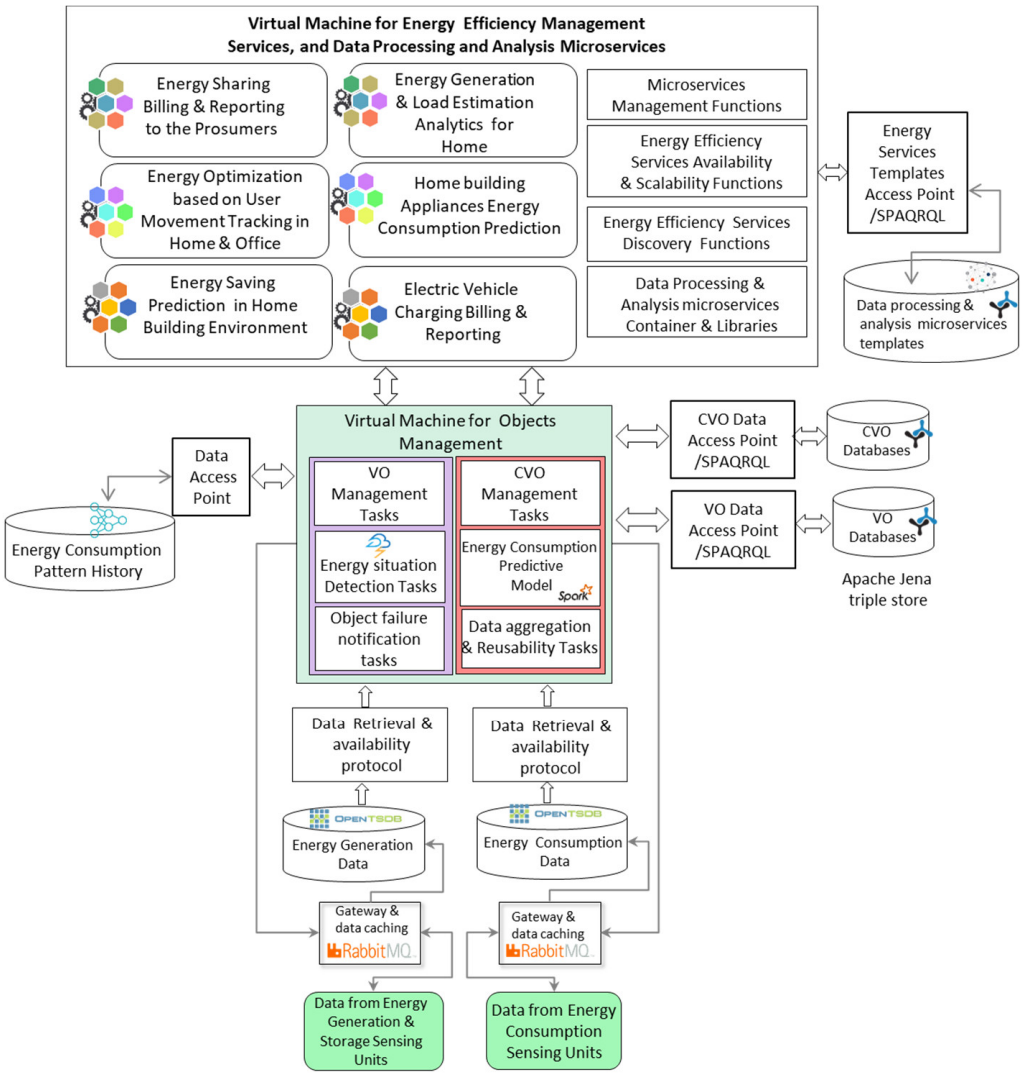


Figure 9. A prototype model for the use case scenario.

We aggregated the data-objects from similar types of energy data as CVOs for the energy generation and consumption data. The reason for combining and aggregating the energy data into VOs and CVO data-objects is to identify the semantic relationship among the energy generation and consumption data and to reuse that data-objects in order to enhance the availability of processed data for other relative services. The real-world energy generation and consumption situation information have been detected by using data processing and analysis microservices, with respect to the time of the day, the number of appliances, type of appliances, and weather condition. In the experiment, we also reused our created VOs and CVOs data-objects in multiple microservices based on energy consumption pattern by the households and similar appliances in different homes. This reusability of VOs and CVOs data creates more knowledge and enhances the availability of processed data for BEEMS.

Based on the use case scenario and keeping the understanding simple, we created six BEEMS from the group of data processing microservices. The templates of these microservices have been stored in the template database and used for the discovery and execution of service features. For learning and activating the energy saving policies from the generation and consumption history data, microservices also used the CVOs rules and VOs processed data. In this case, an energy optimization CVO contain the parameters (VOs) of the consumers and appliances such as building_id appliance_id, appliance_start_time, appliance_close_time, weather_condition, number of dwellers, etc. The data

processing and analysis microservices for office building energy consumption and optimization used the CVOs rules which contain policies for various office electric accessories.

In the experiment, we created many instances of scalable data processing and analysis microservices, CVOs, and VOs for ensuring the availability, and scalability of the system in consideration of the availability of energy management services and data. To better understand and compare the effectiveness of BEEMS from the group of microservices, we also created monolithic services for the use case scenario. The monolithic BEEMS have been created and executed as a single large service without using containers. We tested our experiments by increasing the number of concurrent requests to the services, by using the Apache HTTP server benchmarking tool [52] and the performance of the energy efficiency management services have been tested and evaluated with the Funkload [53]. We used different types of VOs and CVOs with respect to each microservice and the context of service in which they were requested and executed in the BEEMS. The examples of SPARQL queries to insert energy consumption data into the database and select SPARQL statements to inspect energy generation aggregated data is shown in Figure 10.

```
#insert query example energy data
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl_1:  <http://www.w3.org/2002/07/owl#>
PREFIX owl: <http://www.localhost.com/sagent/woosvcagent.owl#>
INSERT
{
owl:building_consumption rdf:type owl:dishWasher_VO
owl:building_consumption owl:id Building_H_001
owl:building_consumption owl:hasConsumption 23.5
owl:building_consumption owl:has_timestamp_VO 1434556972
owl:building_consumption owl:description 'data for the dishwasher energy_consumption'
}
#inserted example energy data
<owl:NamedIndividual
rdf:about="http://www.localhost.com/sagent/woosvcagent.owl#building_consumption">
<rdf:type
rdf:resource="http://www.localhost.com/sagent/woosvcagent.owl#dishWasher_VO"/>
<description>data for the dishwasher energy_consumption</description>
<hasConsumption rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">23.5</hasPrice>
<has_timestamp_VO rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1434556972</hasPrice>
<building_id rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">Building_H_001</id>
</owl:NamedIndividual>
#*************************************************************************
#example to select generated energy data in the building
#with particular buidling id and given date time
Select ?generation_type_VO ?actual_generated_VO ?timestamp_VO
from <http://www.localhost.com/sagent/woosvcagent>
where
{
?building_id_VO:building_id;
FILTER( ?timestamp_VO>= "1433109195"^^xsd:date
&& ?timestamp_VO <= "1434556972"^^xsd:date )
}
#*************************************************************************
#example energy data aggregation
select ?building_id_VO ?generation_type_VO
(AVG(?actual_generated_VO) as generated_energy)
from <http://www.localhost.com/sagent/woosvcagent>
WHERE
{
FILTER( ?timestamp_VO>= "1433109195"^^xsd:date
&& ?timestamp_VO <= "1434556972"^^xsd:date )
}
GROUP BY ?building_id_VO ?generation_type_VO ?actual_generated_VO
```

Figure 10. Excerpt examples of SPARQL queries used in the experimentation.

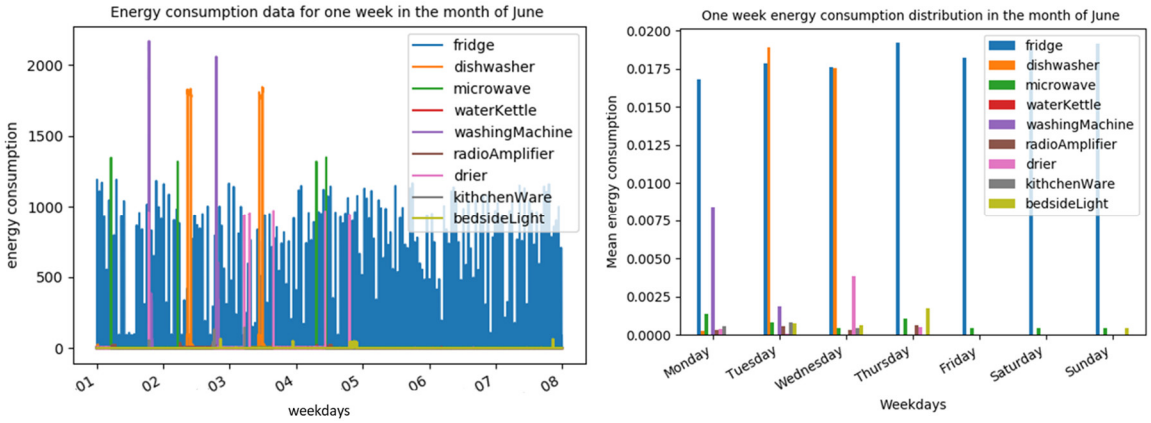
7. Results and Discussion

To evaluate the proposed framework and analyze all possible behaviors of the services including CVOs and VOs data-objects all the services have been tested with a repeated number of experiments. The execution of BEEMS requests with the microservices has also been tested by exponentially increasing of service requests in order to know the availability of data. During the results analysis, missing attribute values of the dataset have been assumed that these data-objects are not available, the details about the experiment and data can be found in Section 6.3. Figure 11a,b show excerpt data

processing and analysis results of the GREEND [41] dataset. The first graph shows the home building energy consumption data with respect to appliances for the first eight days of June. During the result analysis, it is discovered that the fridge consumed more energy and the bedside-light consumed less energy.

In the second graph it is shown that the dwellers use the dishwasher only on Tuesday and Wednesday and they do not use the water kettle during this period. From the results, it is also extracted that the fridge and dishwasher are consuming more energy than the other appliances. Therefore, by adjusting the fridge temperature, we can reduce the energy consumption. Further, these energy consumption data processing and analysis results can also help to know the living behavior of dwellers, and we can predict their occupancy in the building for more energy-optimized services with comfortable living.

The results presented in Figure 11a,b show that even though some of the data attributes and values are missing (i.e., data-objects), still we can execute microservices and services. In this case, our microservices can be categorized as data processing microservices for the energy data visualization.



(**a**) (**b**)

Figure 11. (**a**) Home building energy consumption data processing and analysis with respect to each appliance for the period of eight days in the month of June (**b**) Mean energy utilization distribution and comparison in the weekdays.

Figure 12a shows the availability of energy efficiency management service based on enhanced data availability for the 04 cases presented in Section 5.4. The results in Figure 12a indicate that the service can be available fully or partially even though its data-objects are not available. In this case, similar or virtualized data-objects can be used for the provision of service. In Figure 12a the number of data-objects is the independent variable and service availability is a dependent variable. The decrease in the availability of data-objects may decrease the availability of service if the similar data-object is not available and in another case, the missing data-object cannot be virtualized.

During the results analysis, the availability of the services for Case 1 was 100% both in microservices and monolithic approach because all the VOs and CVOs data-objects were continuously available. The service availability for Case 2 was 80% in the microservices approach and 72% in the monolithic approach. For this case, data from two VOs data was unavailable due to the malfunction of the corresponding real-world objects. In this case, even though the data of two VOs was not available, the service still continues its execution. The service availability in the monolithic approach in Case 2 decreased to 72% because of the tightly coupled dependent tasks in the data and services workflow. In Case 4, where the data-objects for the microservice 5 are not available, its service availability is degraded to 25% in the microservices approach, whereas, the service becomes completely unavailable in the monolithic approach because of the tight dependency among the service tasks and the unavailability of some important data-objects.

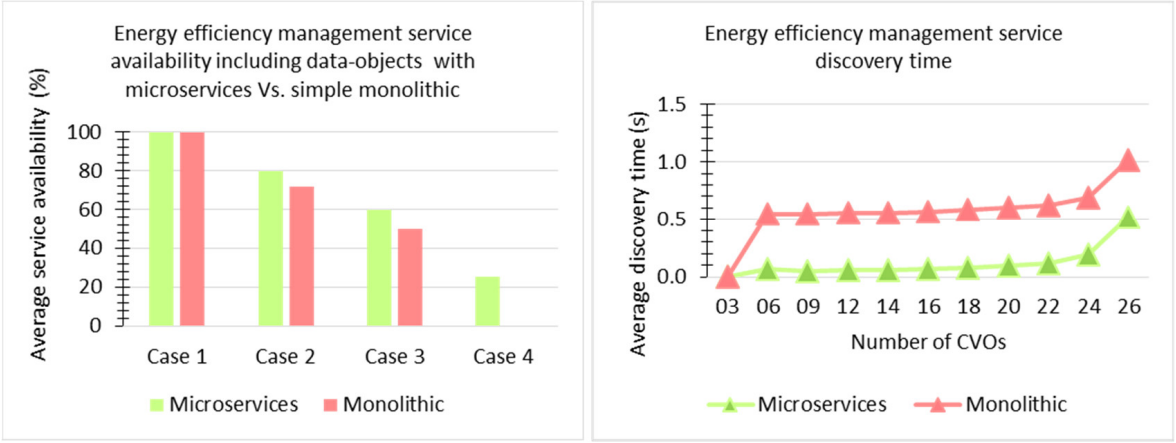


Figure 12. (**a**) Comparison of energy efficiency management service availability in microservices vs. monolithic, and (**b**) Comparison of energy efficiency management service discovery time in microservices vs. monolithic with a number of CVOs.

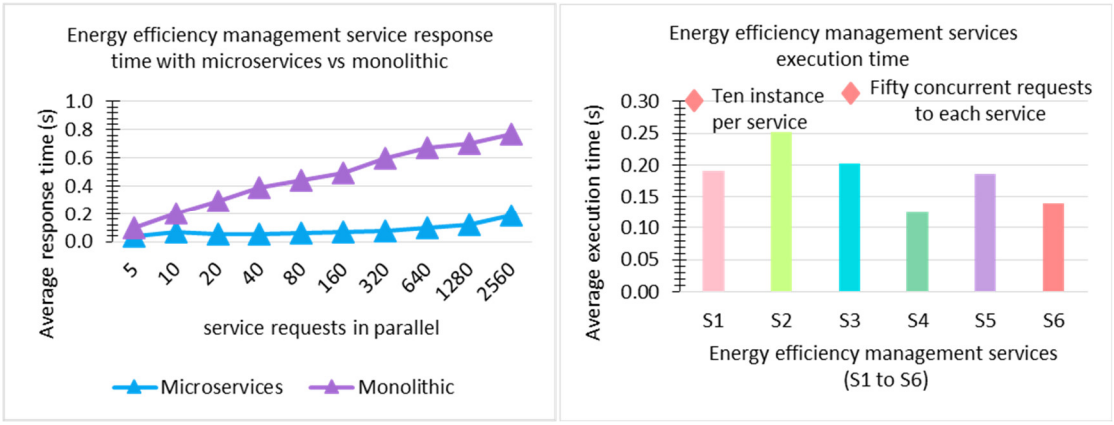
The results of BEEMS discovery have been compared in the microservices and monolithic approaches as shown in Figure 12b. Figure 12b shows the service average discovery time which is obtained when the semantic web agent (the semantic web agent is described in Section 5.5) checks that the whether the service relevant objects are available and when an application requests the service. The results in Figure 12b answer the question of how much time is required to know the availability of a service based on its data-objects because a service that requires more data-objects to execute its features takes more time to know its availability with the reference of data-objects. For obtaining the results indicated in Figure 12b we kept the number of CVOs as an independent variable and average discovery time as a dependent variable.

In each approach, concurrent BEEMS requests have arrived exponentially and are evaluated with a number of VO and CVO energy-related data-objects. The service discovery time increased gradually with the increase in the number of VOs and CVOs data and concurrent service requests. However, the results in microservices implementation were much better than the monolithic one because of the use of lightweight container technologies, RDF-based lookup templates, and enhanced data-objects.

In the BEEMS application requests, the service response time is very crucial for the enhanced data and services availability in an emergency situation or for user satisfaction and comfortable living. We also calculate the service response time in order to know the time agent take: (1) to map and link the similar data-objects in place of missing data-object and (2) the time taken if the similar data-object is not available and it attempts the virtualization of values based on the last available data for specific real-world object. The results in Figure 13a are obtained by keeping the number of parallel service requests as the independent variable and average service response time as a dependent variable.

In the experiment, we carefully analyze the service response time in both the microservices and monolithic approaches by exponentially increasing the number of concurrent BEEMS requests from five to 2560. The average response time for all of the concurrent service requests was about 0.083626 s in the microservices-based implementation and 0.465 s in the monolithic one. However, for five or more concurrent service requests, the average response time delay starts from 0.04 and 0.12 s in the microservices and monolithic approaches, respectively. In both approaches, the service response time was increased significantly with the increasing number of concurrent service requests. However, the services response time in the microservices approach was satisfactory compared with the monolithic one. During the experiments there was no service timeout in the microservices approach, however, in the monolithic one, the service timeout ratio was 0.05% when the number of concurrent BEEMS requests reached 640. The reason for the better services response time in the microservices approach is that every data processing microservice is considered as a separate server process with a

data caching mechanism and they share their data through loosely coupled message queues, whereas, in the monolithic approach, all the processes remains tightly coupled and any failure or delay of one data-object will affect the other dependent processes.



(**a**) (**b**)

Figure 13. (**a**) Comparison of energy efficiency management service response time in microservices vs. monolithic with a number of service requests (**b**) Average end-to-end execution time for six energy efficiency management services with many instances and concurrent requests.

We also analyzed the end-to-end average execution of BEEMS requests for our six implemented services. The end-to-end average BEEMS request execution time includes: (1) the time for the discovery of the service, data processing and microservices analysis; (2) time taken for the retrieval of VOs, CVOs data; (3) data aggregation and applying of modeled policies over the energy data at CVO level; (4) readjustment time taken by the semantic web agent in case of non-availability of some data-objects, and finally (5) the time required for the execution of the requested service features. The average execution time result is shown in Figure 13b, where S1, S2, S3, S4, S5, and S6 represent the BEEMS created with the group of microservices and the names of these services have been presented in Section 6.2.

In the experiment we created 10 instances per service and tested them with 50 concurrent service requests. During this experiment, service requests with high priority such as emergency services have been assigned a VO with high data availability and services with less priority such as entertainment services have been assigned VOs which are less reliable in terms of data availability. In the results, service S1, S2, S3, S4, S5, and S6 took 0.190384, 0.251513, 0.2016, 0.125503, 0.186217, 0.14 s average end-to-end execution time, respectively. The service m2 has the longest execution time whereas m4 has the lowest end-to-end average execution time. The reason m2 took the longest execution time because of a large number of VOs and CVOs energy-related data accumulation, data filtering, data aggregation and applying of the data analysis model. The application service m4 took less end-to-end execution time because it took only shared energy data through relevant CVOs and VOs data, and send the billing details to the prosumers.

We compared our approach with some other related platforms in terms of data availability and service provisions such as iCore [17], Xively [33], and one M2M [16]. These platforms did not discuss data and service availability, scalability, and modularity with data processing microservices, but our framework supports these functionalities. We believe that the results of our proposed approach show the great significance and major contribution in the data and services aspects for the building energy efficiency management services provision literature because it fosters enhanced data availability, modularity, and scalability with the microservices approach, and which will open doors to new ideas to handle data availability, data processing, service modularity, service scalability, and service efficiency in energy management services and other similar range of use cases.

8. Conclusions

In this article, we have presented amicroservices model to enhance data availability for the provision of energy efficiency management services in the building environment with reliability and scalability. The key concept in this paper is to enhance the data availability, linking of various energy-related data-objects and ingestion of these objects for buildings energy efficiency management services (BEEMS), by using microservices. In order to accumulate, processes and analysis of energy-related data to enhance data availability for various BEEMS, we presented a microservices model in the Web of Objects (WoO) platform. Out microservices model supports decentralized data management with improved SPARQL queries, by minimizing the usage of joins and exposes the service capability with modularity and reliability from the group of granular services. The data processing and analysis microservices template information model for the creation of BEEMS, and service discovery and execution model with respect to service request priority and service contextual parameters have been presented. For readjustment, enhancement, reusability, and availability of data-objects, services, and microservices semantic web agent based on ontology has been covered. The provision of BEEMS from the pools of reusable microservices with CVOs and VOs data has been discussed. For the evaluation of our approach, we described and demonstrated a real-world use case scenario with an energy consumption dataset and synthetically generated energy data.

We evaluated the results of the prototype with a repeated number of experiments by comparing the energy efficiency management services with the microservices approach and the monolithic based approach. In the results, we analyzed and compared the services data availability, services discovery time, and response time and end-to-end execution time. We believe that the results of our approach are prominent with microservices approach rather than the monolithic. As to the best of our knowledge, the data processing and analysis microservices approach was not previously discussed in the related platforms such as oneM2M, iCore, and Xively. In the future, we would like to evaluate the results in more detail, explore more features of semantic web agent for the readjustment of data-objects and provision of more efficient extended IoT based energy services to the other IoT platforms.

Author Contributions: The research work was conducted in collaboration with all authors. M.A.J. and I.C. defined the research theme and designed the research objectives. M.A.J., and S.A. implemented the prototype. M.A.J. wrote the article. M.A.J. and I.C. discussed and analyzed the results. All authors have contributed to, read, and approved the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-0-01396) supervised by the IITP (Institute for Information & communications Technology Promotion).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jarwar, M.A.M.A.; Ali, S.; Kibria, M.G.M.G.; Kumar, S.; Chong, I. Exploiting interoperable microservices in web objects enabled Internet of Things. In Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017; pp. 49–54.
2. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gen. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
3. Batini, C.; Cappiello, C.; Francalanci, C.; Maurino, A. Methodologies for data quality assessment and improvement. *ACM Comput. Surv.* **2009**, *41*, 1–52. [CrossRef]
4. Kibria, M.G.; Jarwar, M.A.; Ali, S.; Kumar, S.; Chong, I. Web objects based energy efficiency for smart home IoT service provisioning. In Proceedings of the International Conference on Ubiquitous and Future Networks, ICUFN, Milan, Italy, 4–7 July 2017.
5. Shaikh, P.H.; Nor, N.B.M.; Nallagownden, P.; Elamvazuthi, I.; Ibrahim, T. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renew. Sustain. Energy Rev.* **2014**, *34*, 409–429. [CrossRef]

6. How Much Energy is Consumed in U.S. Residential and Commercial Buildings?—FAQ—U.S. Energy Information Administration (EIA). Available online: https://bit.ly/2OXlHy7 (accessed on 22 November 2018).
7. Buildings—European Commission. Available online: https://bit.ly/1DP5nE2 (accessed on 12 November 2018).
8. Sharma, S.K.; Wang, X. Live Data Analytics with Collaborative Edge and Cloud Processing in Wireless IoT Networks. *IEEE Access* **2017**. [CrossRef]
9. Marjani, M.; Nasaruddin, F.; Gani, A.; Karim, A.; Hashem, I.A.T.; Siddiqa, A.; Yaqoob, I. Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges. *IEEE Access* **2017**. [CrossRef]
10. Al-Ali, A.R.; Zualkernan, I.A.; Rashid, M.; Gupta, R.; Alikarar, M. A smart home energy management system using IoT and big data analytics approach. *IEEE Trans. Consum. Electron.* **2017**, *63*, 426–434. [CrossRef]
11. Da Cruz, M.A.A.; Rodrigues, J.J.P.C.; Al-Muhtadi, J.; Korotaev, V.; Albuquerque, V.H.C. A Reference Model for Internet of Things Middleware. *IEEE Internet Things J.* **2018**, *5*, 871–883. [CrossRef]
12. Morley, J.; Widdicks, K.; Hazas, M. Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption. *Energy Res. Soc. Sci.* **2018**. [CrossRef]
13. Namiot, D.; Sneps-Sneppe, M. On micro-services architecture. *Int. J. Open Inf.* **2014**, *2*, 24–27.
14. Jarwar, M.A.; Ali, S.; Chong, I. Exploring Web Objects enabled Data-Driven Microservices for E-Health Service Provision in IoT Environment. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, South Korea, 17–19 October 2018; pp. 112–117.
15. Bhamare, D.; Samaka, M.; Erbad, A.; Jain, R.; Gupta, L. Exploring microservices for enhancing internet QoS. *Trans. Emerg. Telecommun. Technol.* **2018**, e3445. [CrossRef]
16. oneM2M—oneM2M is the global standards initiative for Machine to Machine Communications and the Internet of Things. Available online: https://bit.ly/2EjCsPM (accessed on 12 January 2018).
17. iCore: Internet Connected Objects for Reconfigurable Ecosystems, European FP7 Project. Available online: https://bit.ly/2R40fto (accessed on 15 November 2017).
18. NiCE—Networking Intelligent Cities for Energy Efficiency. Available online: https://bit.ly/2Krk4Z0 (accessed on 10 July 2018).
19. PLEEC-Planning for Energy Efficient Cities. Available online: https://bit.ly/1DDEZjfl (accessed on 10 July 2018).
20. Y.4452:Functional Framework of Web of Objects. Available online: https://bit.ly/2OWTvM2 (accessed on 22 November 2018).
21. Jarwar, M.A.M.A.; Chong, I. Exploiting IoT services by integrating emotion recognition in Web of Objects. In Proceedings of the International Conference on Information Networking, Da Nang, Vietnam, 11–13 January 2017; pp. 54–56.
22. Kibria, M.; Ali, S.; Jarwar, M.; Kumar, S.; Chong, I.; Kibria, M.G.; Ali, S.; Jarwar, M.A.; Kumar, S.; Chong, I. Logistic Model to Support Service Modularity for the Promotion of Reusability in a Web Objects-Enabled IoT Environment. *Sensors* **2017**, *17*, 2180. [CrossRef]
23. Kumar, S.; Kibria, M.G.M.G.; Ali, S.; Jarwar, M.A.M.A.; Chong, I. Smart spaces recommending service provisioning in WoO platform. In Proceedings of the 2017 International Conference on Information and Communications (ICIC), Hanoi, Vietnam, 26–28 June 2017; pp. 311–313.
24. Jarwar, M.; Kibria, M.; Ali, S.; Chong, I. Microservices in Web Objects Enabled IoT Environment for Enhancing Reusability. *Sensors* **2018**, *18*, 352. [CrossRef]
25. Ali, S.; Jarwar, M.A.; Chong, I. Design Methodology of Microservices to Support Predictive Analytics for IoT Applications. *Sensors* **2018**, *18*, 4226. [CrossRef]
26. Shen, C.; Singh, R.P.; Phanishayee, A.; Kansal, A.; Mahajan, R. Beam: Ending Monolithic Applications for Connected Devices. In Proceedings of the 2016 USENIX Annual Technical Conference (USENIX ATC '16), Denver, CO, USA, 22–24 June 2016.
27. Yu, D.; Jin, Y.; Zhang, Y.; Zheng, X. A survey on security issues in services communication of Microservices-enabled fog applications. *Concurr. Comput. Pract. Exp.* **2018**, e4436. [CrossRef]
28. Pautasso, C.; Zimmermann, O.; Amundsen, M.; Lewis, J.; Josuttis, N. Microservices in Practice, Part 1: Reality Check and Service Design. *IEEE Softw.* **2017**, *34*, 91–98. [CrossRef]
29. Salvadori, I.; Oliveira, B.C.N.; Huf, A.; Inacio, E.C.; Siqueira, F. An ontology alignment framework for data-driven microservices. In Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services—iiWAS '17, Salzburg, Austria, 4–6 December 2017; ACM Press: New York, NY, USA, 2017; pp. 425–433.

30. Alonso, R.; Barbara, D.; Garcia-Molina, H. Data caching issues in an information retrieval system. *ACM Trans. Database Syst.* **1990**, *15*, 359–384. [CrossRef]
31. Chen, M.; Qian, Y.; Hao, Y.; Li, Y.; Song, J. Data-Driven Computing and Caching in 5G Networks: Architecture and Delay Analysis. *IEEE Wirel. Commun.* **2018**, *25*, 70–75. [CrossRef]
32. Patel, P.; Cassou, D. Enabling high-level application development for the Internet of Things. *J. Syst. Softw.* **2015**, *103*, 62–84. [CrossRef]
33. Ray, P.P. A survey of IoT cloud platforms. *Futur. Comput. Informatics J.* **2016**, *1*, 35–46. [CrossRef]
34. Swetina, J.; Lu, G.; Jacobs, P.; Ennesser, F.; Song, J. Toward a standardized common M2M service layer platform: Introduction to oneM2M. *IEEE Wirel. Commun.* **2014**, *21*, 20–26. [CrossRef]
35. Nitti, M.; Pilloni, V.; Colistra, G.; Atzori, L. The Virtual Object as a Major Element of the Internet of Things: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1228–1240. [CrossRef]
36. Ali, S.; Kibria, M.G.M.; Jarwar, M.A.M.; Lee, H.K.H.; Chong, I. A Model of Socially Connected Web Objects for IoT Applications. *Wirel. Commun. Mob. Comput.* **2018**, *2018*. [CrossRef]
37. Vlacheas, P.; Giaffreda, R.; Stavroulaki, V.; Kelaidonis, D.; Foteinos, V.; Poulios, G.; Demestichas, P.; Somov, A.; Biswas, A.; Moessner, K. Enabling smart cities through a cognitive management framework for the internet of things. *IEEE Commun. Mag.* **2013**, *51*, 102–111. [CrossRef]
38. Mayer, S.; Ciortea, A.; Ricci, A.; Robles, M.I.; Kovatsch, M.; Croatti, A. Hypermedia to connect them all autonomous hypermedia agents and sociotechnical interactions. *Internet Technol. Lett.* **2018**, e50. [CrossRef]
39. Bellavista, P.; Chessa, S.; Foschini, L.; Gioia, L.; Girolami, M. Human-Enabled Edge Computing: Exploiting the Crowd as a Dynamic Extension of Mobile Edge Computing. *IEEE Commun. Mag.* **2018**, *56*, 145–155. [CrossRef]
40. Bao, K.; Mauser, I.; Kochanneck, S.; Xu, H.; Schmeck, H. A Microservice Architecture for the Intranet of Things and Energy in Smart Buildings. In Proceedings of the 1st International Workshop on Mashups of Things and APIs—MOTA '16, Trento, Italy, 12–16 December 2016; ACM Press: New York, New York, USA, 2016; pp. 1–6.
41. Monacchi, A.; Egarter, D.; Elmenreich, W.; D'Alessandro, S.; Tonello, A.M. GREEND: An energy consumption dataset of households in Italy and Austria. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 511–516.
42. Batra, N.; Kelly, J.; Parson, O.; Dutta, H.; Knottenbelt, W.; Rogers, A.; Singh, A.; Srivastava, M. NILMTK. In Proceedings of the 5th International Conference on Future Energy Systems—e-Energy '14, Cambridge, UK, 11–13 June 2014; ACM Press: New York, New York, USA, 2014; pp. 265–276.
43. Shrouf, F.; Miragliotta, G. Energy management based on Internet of Things: Practices and framework for adoption in production management. *J. Clean. Prod.* **2015**. [CrossRef]
44. Amasyali, K.; El-Gohary, N.M. A review of data-driven building energy consumption prediction studies. *Renew. Sustain. Energy Rev.* **2018**, *81*, 1192–1205. [CrossRef]
45. Pan, J.; Jain, R.; Paul, S.; Vu, T.; Saifullah, A.; Sha, M. An Internet of Things Framework for Smart Energy in Buildings: Designs, Prototype, and Experiments. *IEEE Internet Things J.* **2015**. [CrossRef]
46. Pérez-chacón, R.; Luna-romera, J.M.; Troncoso, A. Big Data Analytics for Discovering Electricity Consumption Patterns in Smart Cities. *Energies* **2018**, *11*, 683. [CrossRef]
47. Amaral, M.; Polo, J.; Carrera, D.; Mohomed, I.; Unuvar, M.; Steinder, M. Performance Evaluation of Microservices Architectures Using Containers. In Proceedings of the 2015 IEEE 14th International Symposium on Network Computing and Applications, Cambridge, MA, USA, 28–30 September 2015; pp. 27–34.
48. Abbott, M.L.; Fisher, M.T. *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*; Pearson Education: London, UK, 2009.
49. Protégé—A Free, Open-Source Ontology Editor and Framework for Building Intelligent Systems. Available online: https://protege.stanford.edu/ (accessed on 14 May 2018).
50. Apache Jena—SDB—Persistent Triple Stores Using Relational Databases. Available online: https://bit.ly/2S5xvjN (accessed on 14 May 2018).
51. Log-Synth: Schema-Driven Data Generation. Available online: https://bit.ly/1RvgY2m (accessed on 18 July 2018).

52. Apache Foundation Apache HTTP Server Benchmarking Tool—Apache HTTP Server Version 2.4. Available online: https://bit.ly/2AbEUXr (accessed on 30 May 2018).
53. FunkLoad Documentation Contents—FunkLoad 1.17.1 Documentation. Available online: https://bit.ly/2knJU7o (accessed on 30 May 2018).