applied sciences

MDPI

Article

Semantic Data Management for a Virtual Factory Collaborative Environment

Artem A. Nazarenko [1,*], Joao Sarraipa [1], Luis M. Camarinha-Matos [1], Oscar Garcia [2] and Ricardo Jardim-Goncalves [1]

[1] Faculty of Sciences and Technology & UNINOVA-CTS, Nova University of Lisbon, 2829-516 Monte Caparica, Portugal; jfss@uninova.pt (J.S.); cam@uninova.pt (L.M.C.-M.); rg@uninova.pt (R.J.-G.)
[2] Information Catalyst SL, Ptda Tosal de la Cometa, 1-F-1-D, BL 3501 (Imperial Park), 03710 Calpe, Spain; oscar.garcia@informationcatalyst.com
* Correspondence: a.nazarenko@campus.fct.unl.pt

Received: 23 October 2019; Accepted: 12 November 2019; Published: 16 November 2019

Featured Application: Materials presented in this article can be used for knowledge base creation and reasoning algorithms in industrial applications.

Abstract: Recent developments in the area of cyber-physical systems (CPSs) and Internet of Things (IoT) are among the drivers for the emergence of the Industry 4.0 concept, setting new requirements for the architecture, technology, and design approaches of modern industrial systems. Industry 4.0 assumes a higher level of intelligence, and thus autonomy of the systems and subsystems, and a larger focus on the analysis of gathered data for further utilization. The Virtual Factory Open Operating System (vf-OS) project is intended to respond to some of these key challenges, in particular for the smart factory application domain. Complementarily, data and knowledge storage and processing are also in the scope of vf-OS. This article introduces the semantic management component of vf-OS, which aims to analyze the interrelations among stored entities, as well as to define the closeness among them to generate meaningful suggestions, which can be later used by other subsystems or operators in a user-friendly way. The semantic managing system makes use of non relational approaches, namely a graph database, which enables data to be represented as graphs for further semantic querying. The developed prototype and an illustrative application case are also presented.

Keywords: cyber-physical systems; Industry 4.0; semantic management; vf-OS; collaborative environment

1. Introduction

Cyber-physical systems (CPSs) have found their application in many areas of human activity, from home automation to large industrial complexes containing hundreds of systems and subsystems. The growing complexity of the modern CPS forces to cope with challenges such as interoperability, scalability, and increasing volumes of generated data, which leads to the need of new system architectures and frameworks. CPSs [1] have become one of the key elements of the Industry 4.0 concept, which is gaining more and more attention from both academia and industry. For instance, in [2] the authors offer insight on how the concept of Industry 4.0 influences the overall state of manufacturing in Europe, with respect to socio-economical aspects such as improved product manufacturability, reduced production time, and reduced negative impact on the environment. In line with this trend, several key projects have been launched in the framework of the European Horizon 2020 program (e.g., Virtual Factory Open Operating System (vf-OS), DISRUPT, SAFIRE, ENACT, Boost 4.0, and many others) to support the implementation of ideas brought up in Industry 4.0. Smart manufacturing is one of the topics of

focus for some of these projects, contributing to a convergence of the digital and physical worlds in the manufacturing domain [3]. The vf-OS project precisely targets issues related to these technologies, enabling the fulfilment of some key requirements and needs of Industry 4.0 by developing a set of modular services for better integration of manufacturing and logistics processes [4].

According to [5], smart manufacturing is moving from knowledge-based manufacturing towards data-driven or knowledge-enabled manufacturing. Thus, the data generated over the entire complex and distributed manufacturing units become a resource that can be utilized to improve the system's efficiency, as well as to find the system's weak and strong points. This makes data processing and analysis a critically important challenge for advanced CPSs, requiring novel Information and communications technology (ICT) support platforms.

An important requirement for platform development that supports the ongoing digital transformation in different application domains is modularity, both in terms of the involved technological process and autonomy. Moreover, for platforms developed for Industry 4.0, a hybrid approach can be implemented [6]. This approach assumes that every subsystem or system entity deployed in a certain location performs actions independently, while some form of centralized data assembly is performed. However, the modularity requirement sets several challenges in terms of orchestration and interoperability. For instance, some of the facilities might be dependent on the output from another facility, and data should be "understandable" to all parties.

Furthermore, with the increasing autonomy and embedded intelligence of components, CPSs are becoming complex networks of collaborative units, leading to the notion of collaborative CPSs [7]. In fact, collaboration issues are at the heart of most challenges in modern industrial systems. The effective materialization of the fourth industrial revolution strongly depends on properly addressing collaborative organizational structures, processes, and mechanisms along the six dimensions of Industry 4.0: vertical integration, horizontal integration, through-engineering across value chain, acceleration of manufacturing, digitalization, and new business models [8].

To better explore the data generated by complex systems, methods and techniques coming from semantic theory are used. Semantic theory assumes that words often appear in text with other related words. This principle can be applied not only in studying text structures but also complex systems such as CPSs, as the language is just "the reflection of objective reality" [9]. CPS systems typically contain a lot of heterogeneous components that communicate with each other and generate data. In order to perform reasoning actions over the collected data, there is a need to understand the interdependencies among different objects and concepts, which can be supported by a knowledge base. An example can be found in [10], where authors established a knowledge base that allowed mapping of functional blocks of different standards in the automation domain. Structured data can be efficiently represented through ontologies; in turn, these structures can be represented as a graph [11]. The importance of ontologies for automated web service composition and provision is underlined in [12]. The main goal of an ontology is to provide structural representations of concepts, properties, and the relations among them. However, this representation is largely dependent on the purpose of the ontology. For instance, the "chair" and "lemon" concepts could be interrelated if they are sold in the same shopping mall. Thus, for text analysis alone, it might be not enough to rely on the co-occurrence of words [9]. However, in this work, we mostly focus on semantic knowledge graphs, which are, according to [13], "extensive networks of objects or concepts with properties and semantic types, and relationships between objects/concepts providing information about a specific domain". However, the proposed approach is not limited to a specific domain; rather, it allows integration of various ontologies from various application areas.

The issue of data interoperability, and more specifically semantic interoperability, is also in the scope of this work. Key research challenges in the area of semantic interoperability have been pointed out in [14] such as: (i) data modeling and exchange, (ii) ontology matching and merging, (iii) data/event semantic annotation, (iv) knowledge representation, (v) knowledge sharing, (vi) knowledge revision, (vii) semantic discovery of data and services, (viii) semantic routing and publishing/subscribing, (ix)

reasoning, and so on. Furthermore, the issue of storage can be added to this list of challenges [15], especially in the context of data-rich environments. Most of the efforts presented in this article focus on semantic discovery, storage, and ontology matching and merging. In the context of this research, semantic discovery is performed through linking the concepts from different ontological models, which, by analogy with semantic routing, have higher preferences or reputation indexes [16].

As such, the main research questions addressed in this work are:

- Which tools and approaches can contribute to the improvement of semantic interoperability of heterogeneous data?
- How can the import and combination of various data models and ontologies help to enrich the mapping process?
- How does a collaborative approach contribute to forming the knowledge base and improving the mapping process?

To underline the core ideas highlighted in this work, the following topics are addressed: limitations of current semantic discovery architectures, definitions clarifying the difference between semantic similarity and contextual representation, description of the graph-based approach and its advantages over the vector-based approach, semantic distance as an identifier of concept closeness, application of pathfinding algorithms to map data models of concepts, utilization of a weighted approach allowing tracking of evolving semantic relations, and identification of the need to import and integrate ontologies from different sources in order to enrich the knowledge base.

The remainder of this article is organized as follows. Section 2 is devoted to a brief discussion of relevant literature, and then the connection of this work to vf-OS is drawn in Section 3. Section 4 is devoted to the semantic management component of vf-OS and some relevant services that are delivered by this component. The next section covers a test-case scenario of how the semantic management component can be used, and related details are discussed. At the end of the article conclusions are made.

2. Related Literature

Complex systems require support environments that can provide assistance in various aspects, such as data and knowledge storage, process modeling and simulation, and so on. For instance, three main support environments are identified in [17]: (i) e-infrastructure, covering computing, storage, and network capabilities; (ii) research infrastructure, allowing handling of assets from various domains; and (iii) virtual research environments, enabling user-centric support for data selection and discovery. In this work we target some functional aspects from all three categories, namely storage, export of ontologies from different areas, and data selection and discovery mechanisms.

Several limitations have been identified in current semantic discovery systems [18]. The involved architectures are limited or weighted ontologies are not considered, and there is poor performance with aggregated data from various knowledge bases. Adaptive techniques or collaborative knowledge base formation could be key in overcoming the difficulty of possible lack of knowledge about a specific concept. For instance, in one ontology concept, “BMW” is a “car”, whereas in another it is a “vehicle”. If we aggregate these ontologies, the concept “vehicle” can be associated with the concept “car” through the concept “BMW”. This might significantly affect the search results, depending on the types of routing algorithms. Two main routing algorithms are identified: single-keyword and multi-keyword [19]. The same authors proposed an approach for the IoT domain, where each IoT device has its core functionality described using one keyword, while other related aspects are described as “attributes”. Single-keyword algorithms might not be enough for some applications, not only in terms of expressing the functionality, but also attributes of the object or concept. Sometimes, instead of a requested concept, object, or service, another one, which is close enough, can be suggested. In order to define “relatedness” between two concepts, the semantic in-between distance needs to be calculated. According to [20], the semantic distance is used to capture the closeness between two pieces of text

for the case of "concepts". There are two main types of semantic distance: lexical-based measures and distributional measures of word distance. The first one relies on the structure of the knowledge source, whereas the second is based on the word occurrence rate. The distributional measure of the word-distance approach does not require a lexical database, which makes it more flexible in the case of higher data heterogeneity [21].

The problem acquiring semantic information to find interdependencies among sensors and actuators is raised in [22], where the approach to detect sources of errors through path tracking is proposed. However, these authors mostly focused on the narrow area of physical object dependencies without considering interdependencies among concepts and virtual entities.

Another aspect is that, because of the increasing autonomy and intelligence of CPS components, the need for supporting tools to facilitate the establishment of a collaborative environment has emerged, where the entities need additional semantic information to choose the "best" options based on factors such as, for instance, closest position, similar functionality, previous history, and so on. The issues of self-organization for CPS components are raised in [23], where the authors also present an approach for interoperability provision among resources and services in smart spaces. The basic idea is that each resource contributes to semantic interoperability when it connects to the smart space by uploading its ontology. Semantic interoperability is a key requirement for the development of agile design methodologies for collaborative CPSs, such as the one proposed in [7] based on the design science method. Another work [24] presents a study of the semantic enrichment of manufactured products, with the main objective to enhance the generation and circulation of product-related knowledge among all involved parties. This work departs from these ideas and goes deeper into the subject of data harmonization, including clustering, when basic terms and concepts are combined based on the similarities they possess and are organised based on their semantic and contextual relationships. The use of ontologies to represent machine behavior through the interrelation of input signals and output controls is also discussed in [25].

It is important to distinguish between semantic similarity and contextual representation. According to [26], semantic similarity is a broad term, which considers not only synonyms, but also meronymy (PC and keyboard, keyboard is a part of the PC) and hyponyms (animal and dog, dog is a part of the animal class). Whereas, contextual representation is described [27] as knowledge about how the word is used (i.e., an association between concepts and some attributes common for a specific context). The solution proposed in this work is intended to provide a broader basis for further reasoning, when two concepts might be interrelated or associated not only based on belonging to the same class or one concept being part of another, but also how these two concepts are used together. As an example, we can consider a smart home, where a smart fridge is located in the "kitchen", and a mobile vacuum cleaner can move among different rooms, including the "kitchen"; thus, the smart fridge might be associated with the vacuum cleaner based on a common location, even if temporarily. This aspect raises another modeling challenge, as some of the concepts or entities might have strong or weak associations indicating how close they are related to each other.

Regarding the issue of data representation, two main approaches are discussed in the literature [28]: vector-based and graph-based approaches. The graph-based approach has been chosen for this work based on the following reasons:

- it can be used to represent order and structural features of the considered topic;
- it allows decomposition of complex topics and representation of interdependencies among subtopics;
- it allows hierarchical representation of topics, features, and so on; and
- it offers pattern detection capabilities.

Additionally, in Table 1 we include other literature sources devoted to other aspects that are covered in this work, namely (i) ontology or data model integration, (ii) delivering concept "relatedness",

(iii) generation of queries for further discovery and visualization, (iv) collaborative knowledge base formation, (v) mapping of data models and ontologies, and (vi) consideration of weighted edges.

Table 1. Topics covered in this work and related research works.

	Henkel et al., 2015	Kivikangas & Ishizuka, 2012	Hoppen et al., 2017	Hor et al., 2018	Nguyen et al., 2017	Hristovski et al., 2015	Song et al., 2018	Malliaros & Skianis, 2015	Abay et al., 2015	Rashidy et al., 2018	Xia et al., 2019	Alshammari et al., 2019
Ontology or Data Model import	x			x	x				x	x		
Delivering concept "relatedness"	x	x				x			x		x	x
Generation of Queries for further discovery and visualization											x	
Mapping of data models and ontologies	x	x	x	x	x		x					
Consideration of weighted edges								x				

Most of the works mentioned in Table 1 [29,30] are focused on mapping different data structures, models, and ontologies in order to represent them as graphs in a graph database, but the core logic is maintained [31]. The often-used format for ontology representation is the resource description framework [32,33], sometimes extended by the web ontology language [27,29]. Articles devoted to identifying "relatedness" of concepts to each other using the path between them do not cover the weighted approach [29,30,34,35]. However, in [36] the authors raise the issue of using the weighted approach for text categorization, whereas each document is considered as a graph to apply a categorization algorithm. This allows identifying the importance of the term based on co-occurrence. In our work, the weighted approach is used to provide a basis for reasoning systems by enabling detection of the importance of the concepts inside the data models to each other. Most current work in the literature is rather focused on contextual representation, considering "relatedness" identification rather than semantic similarity [33], which allows interrelating the concepts without a need for predefined knowledge about synonyms, hyponyms, or meronymy. For instance, in [35] the authors solely target the issue of finding the hyponyms in a specific ontology, which limits the approach to a specific ontology containing specific relation types. The issue of importing ontologies is raised in several works (e.g., [29,31,37]), whereas in [38], the authors propose to use a graph database as a "hub" to aggregate the data for further analysis.

Some specific application areas (e.g., utilization of graph database as advisor for academic collaboration [34], mapping documents containing the smart city information into the graph nodes [37], or how ontologies mapping can contribute to the cross standard interoperability in automation [39]) are further considered in the literature. Though, our aim is to utilize the framework presented in this work for multiple domains. Furthermore, the aim is also to provide different functionalities, offer a good level of modularity, and allow new features to be added or existing ones to be extended on demand. Finally, the establishment of mechanisms for collaborative knowledge base formation is also in the scope of this work.

Moreover, a challenging issue itself is the conversion of the so called nonsemantic data. For instance, in [40] the authors propose a platform that can transform the data from a relational database

(DB) into a semantic resource description framework (RDF) format. The usage of a graph DB can help to avoid additional difficulties in the way of conversion, allowing data to be stored in an RDF compatible way.

3. Relation to vf-OS

The component presented in this article is part of the vf-OS project [41], which aims to provide an open platform for virtual factories (VFs) and a collaborative environment to support the manufacturing processes at various stages. The project is closely related to the concepts of Industry 4.0, including CPS and Internet of Things (IoT), whereas other relevant subjects are covered such as data processing, data visualization, and cloud computing. The key vf-OS components are: system kernel, virtual factory i/o, data and connect, open applications development kit, service provision framework, and platform environment. The solution proposed in this work belongs to the data management component (DMC) (Figure 1), which aims to manage data flows on a large scale. Another goal is to provide analytical operators for data analysis. DMC is composed of four modules: data infrastructure middleware, data harmonization, data storage, and data analytics [42].

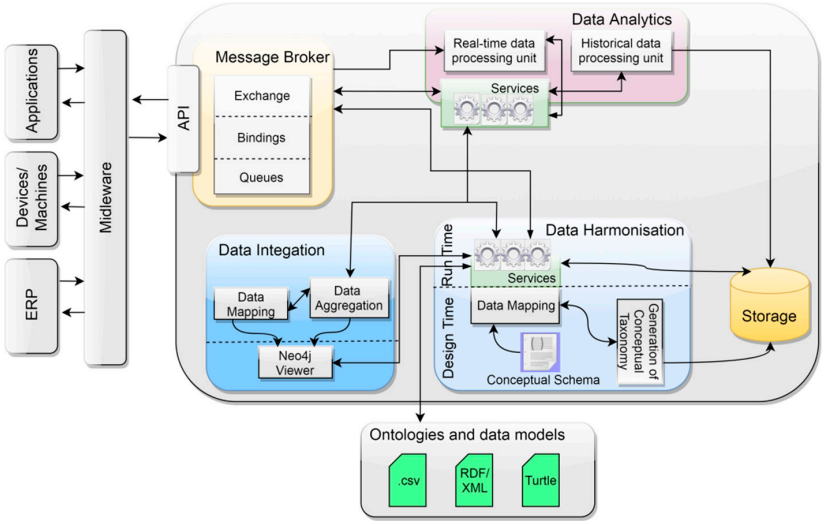

Figure 1. Data management component (DMC) overview.

More specifically, this work is focused on data harmonization and partially on the data storage modules. The data harmonization module is responsible for implementing data mapping mechanisms extended with ontology import interfaces and data visualization capabilities. It is divided into design time and run time submodules. The design time submodule maps the data models, whereas run time is responsible for data transformation. The data storage issue is addressed only partially, as vf-OS requires different types of storage for different types of data, thus using the advantages of various storage approaches in the best way [42]. For the storage part of this component, a Not only SQL (NoSQL) type of database, namely a graph DB, has been chosen. The chosen NoSQL DB, which is Neo4j, is able to cope with large amounts of data and provides agile querying mechanisms, allowing its use in smart storage, which not only "keeps" or stores data but also provides tools to perform reasoning operations on the stored data.

4. Semantic Management Component

The semantic management component contributes mainly to the design time part of the harmonization module, and it consists of two main parts (Figure 2): (1) storage and (2) harmonization components. It enables two-way communication processes, output formatting, and a set of services related to the graph DB. It also transforms the data into a format that can be consumed by other

components of the vf-OS, resorting to the mechanisms of the graph DB to perform reasoning actions right after the data are inserted. Furthermore, semantic management supports ontologies and data model import and integration. Some formats that are supported for import are RDF/XML (Extensible Markup Language), web ontology language (OWL) syntax extension for RDF, Turtle, and comma-separated value (CSV) data models.

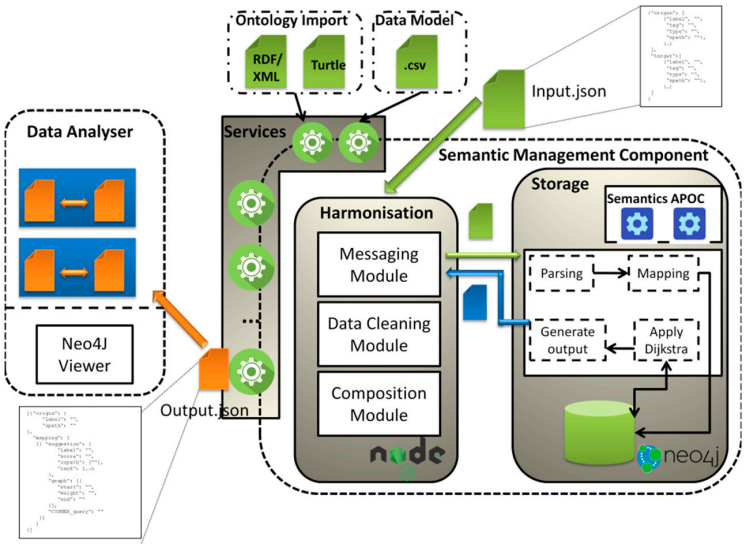

Figure 2. General view of the semantic management component.

The core of the storage component is a Neo4j graph DB. Neo4j provides a number of tools not only for storing the data in a graph format, but also for applying a wide variety of algorithms such as path finding, similarity, link prediction, and so on. Neo4j uses the Cypher querying language, which enables core operations of processing, reasoning, and retrieving the data. To be able to use these algorithms, the APOC plugin is used, and for importing RDF ontologies, a semantics plugin is adopted. The system also accepts data to be inserted in various formats such as JavaScript Object Notation (JSON) or Comma-separated Values (CSV), which contributes to overall interoperability without much effort to transform the data.

The harmonization module of the semantic management component includes messaging, data cleaning, and composition modules.

- The messaging module is responsible for maintaining communication between the storage and harmonization components. Moreover, it provides a Representational State Transfer (REST)-based interface for outer applications. Between the harmonization and storage components, communication is accomplished through bolt connectivity, which is the network protocol running over a Transmission Control Protocol (TCP) connection. In other words, the messaging module allows requests to be received from outer applications, following the REST architectural approach, and retranslates them to storage in a bolt-compatible format.
- The data cleaning module is needed to cope with complex output objects returned after the storage component processes a request. Neo4j-returned objects are parsed, and the required data are extracted.
- The composition module is used to align the data that was prepared by the data cleaning module with the output format and to generate a Cypher query which can be visualized in the Neo4j viewer environment to represent the output in a human-friendly format. However, this module, as well as the data cleaning one, is only needed for some of the services, as not all of them are assumed to return a JSON output.

The semantic management component provides a set of services that are used by the data analyzer responsible for further data mapping and visualization. Some of the services, such as importOntology or importDataModel, are intended to provide an interface for uploading and integration of RDF/XML and Turtle ontologies and data models in CSV format.

In the following sections these modules are described in more detail, and a test case is used with some chosen services (e.g., importOntology, importDataModel, and getMappingSuggestion). Detailed descriptions of these services are given below.

4.1. Get MappingSuggestion Service

The "getMappingSuggestion" provides mechanisms to map two data models. The goal of the service is to identify how close the concepts in one model are interrelated to concepts in another model. The result of mapping, as mentioned above, can be reused for further reasoning or discovery. In Figure 3, the models submitted for mapping are identified as input, which is a JSON document containing both models. From now on, we use the terms "origin" and "target" to identify the two parts of the input data model, as also shown in Figure 3. It is important to mention that the "getMappingSuggestion" service is developed for use within the vf-OS framework; thus, it has a structure based on JSON format. This is necessary to represent, if needed, the data models as a complex, structured tree of concepts with appropriate relations. In order to better understand the involved models, we introduce the metastructure of the input models (Figure 3). These models comprise several fields as follows:

- "label"—which introduces the name of the capillary/single concept;
- "tag"—this field can be used to identify the attributes of the concept. The inspiration for introducing the "tag" field came from the topic of folksonomy, described in [43], where a collaborative tagging system is presented that is composed of three core entities: users, resources, and tags. Tags, compared to, for instance, resources, are in no way limited by predefined vocabulary [44]. In this work, tags are used to describe the concepts that, to some extent, can be compared with resources in folksonomy;
- "xpath"—is used to show the parent concepts of the capillary/single concept. This field allows expressing the hierarchy of concepts from a specific concept to the root;
- "type"—used to express the type of the concept, for instance: string.

```
{
        "origin":[
                {"label":"origin_Concept_1","tag":[“Property_1”,
                “Property_n”], "type": "type_of_concept", "xpath":
                [“Parent_1”, “Parent_n”]},
                ……
                {"label":"origin_Concept_m","tag":[“Property_1”,
                “Property_n”], “type":"type_of_concept",“ xpath":
                [“Parent_1”,“Parent_n”]}
        ],
        "target":[
                {"label":"target_Concept_1","tag":[“Property_1”,
                “Property_n”], "type":"type_of_concept","xpath":[]},
                ……
                {"label":"target_Concept_m","tag":[“Property_1”,
                “Property_n”], "type":"type_of_concept","xpath":
                [“Parent_1”, “Parent_n”]}
        ]
}
```

Figure 3. General schema of the input data models for getMappingSuggestion.

The input file containing the origin and target models is delivered to the harmonization component, which is responsible for correct communication with the graph DB, proper parsing, and integration of the input data model.

In order to clarify the process of delivering and mapping the data, we use definitions derived from the set theory and graph theory. The payload, or, in other words, data within the origin and target models, is represented as two sets:

$$\mathrm{M_S} = (A, T_S, Xp_S, Type) \; and \; \mathrm{M_T} = (B, T_T, Xp_T, Type), \tag{1}$$

where $\mathrm{M_S}$ stands for the origin or source data model, and $\mathrm{M_T}$ stands for the target. Subsets A and B contain the capillary concepts derived from the "*label*" field. The related concepts from the "*path*" and "*tag*" fields are expressed through the $\mathrm{T_S}$ and $\mathrm{T_T}$ terms and $\mathrm{Xp_S}$ and $\mathrm{Xp_T}$ subsets. And finally, the "*Type*" subset represents the variety of types of concepts from A and B subsets.

As the graph DB deals with nodes and relationships, all data being part of the input are converted. Thus, the concepts become nodes and relationships that are generated according to the data model schema. The data model schema assumes that all concepts from the fields "*tag*" and "*xpath*" are directly related to the concept presented in the "*label*" field (Figure 3). After the concepts provided in the origin and target models are inserted and converted into nodes, and appropriate relationships are created, the phase of data mapping starts. As mentioned in Section 2, the pathfinding approach for mapping follows the Dijkstra algorithm implemented as the APOC.algo extension of Neo4j [45]. It supports calculation of the "distance" considering the weights of the path segment. This is important in order to cope with requirements for the weighted approach stated in Section 2. For the operations within the mapping process, a graph theory apparatus is used, which allows handling nodes (called vertices) and relationships (called edges). For further convenience, the term "concept" is used as equivalent to "node" (a graph DB term) and "vertices" (a graph theory term), and "relationship" is used as equivalent to "edges" (a graph theory term). Before applying Dijkstra's algorithm, we need to formulate concept pairs. In other words, ordered pairs of concepts are built from subsets A and B in order to set source (subset A) and target (subset B) concepts for path finding. To perform mapping between the origin data model and the target data model, a Cartesian product between the origin data model, defined as set A containing elements $\{a_1, a_2, ..., a_\mathrm{m}\}$, and the target data model, defined as set B with elements $\{b_1, b_2, ..., b_\mathrm{m}\}$, is built. The Cartesian product for origin and target data models gives:

$$A \times B = \{(a,\ b) \mid a \in A,\ b \in B\}, \tag{2}$$

$$i.e.,\ A \times B = \{(a_1,\ b_1),\ (a_1, b_2), \ldots, (a_1, b_n), (a_2,\ b_1), (a_2, b_2), \ldots, (a_2,\ b_n), \ldots, (a_m, b_1), (a_m,\ b_2),\ \ldots, (a_m, b_n)\}. \tag{3}$$

However, because each element of the origin data model needs to be represented separately (in order to provide the best output of Dijkstra's algorithm for each concept in A) rather than for the whole origin data model, the concepts from A are treated one by one. Thus, the goal is not to compare the path from a_1 to b_1 with the path from a_2 to b_n, but rather a_1 to b_1 with a_1 to b_2 and a_1 to b_n. This also allows sparing the computational resources for excessive matching operations. This results in the following:

$$a_1 \times B = \left\{(a_1, b) \middle| a_1 \in A_{a_1},\ b \in B\right\},\ \text{where } A_{a_1} \in A, \tag{4}$$

$$a_1 \times B = \{(a_1, b_1),\ (a_1, b_2), \ldots, (a_1, b_n)\}. \tag{5}$$

The origin model set is represented through a number of concepts:

$$A = (a_1,\ a_2, \ldots, a_m). \tag{6}$$

Let the resulting nondirected weighted graph be $G = (V, E, w)$, where V is a set of concepts and E is a set of relationships with $w : E \rightarrow R$ as a weight assigned to edges. After applying Dijkstra's algorithm, we obtain a set of suggested paths P for each concept in the origin model. For instance,

path $p \in P$ from a_1 to b_1 in G is a list of edges $((V_0, V_1), (V_1, V_2),..., (V_{k-1}, V_k))$, where $V_0 = (a_1)$ and $V_k = (b_1)$, and k is the total number of concepts that are part of the path (see illustration in Figure 4).

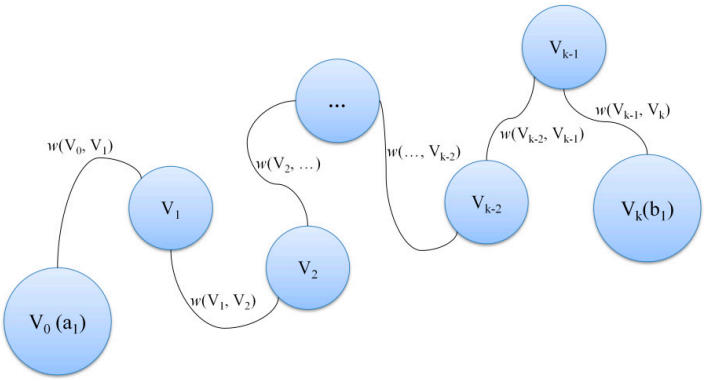

Figure 4. Example path with concepts, relationships, and weights.

The resulting weight of the path is the sum of the weights of all relationships and is calculated as follows [46]:

$$w(p) = \sum_{i=0}^{k-1} w(v_i,\ v_{i+1}). \tag{7}$$

The number of resulting paths can vary, but for the current implementation, we have chosen four for each concept in the origin submodel. Each path built from origin to target concept is called a suggestion. As only four suggestions are required, the resulting graph includes a maximum of four subgraphs, based on the distance parameter. In the theory of semantics, this parameter can be described as the semantic distance and defined as the shortest path connecting two concepts. For instance, for a_1 and b_1 [47]:

$$d(a_1, b_1) = \min\{w(p) : p = ((v_0, v_1), (v_1, v_2)\ , \ldots, (v_{k-1}, v_k)),\ \forall i : (v_{i-1}, v_i) \in e, a_1 = v_0, b_1 = v_k\}. \tag{8}$$

Thus, after the mapping process is accomplished, the result can be represented through a graph that consists of a set of subgraphs representing the suggestions for each of the origin concepts

$$G = (G_{a_1}, G_{a_2}\ , \ldots, G_{a_m}), \tag{9}$$

where m is the number of origin concepts, and $G_{a_1}, G_{a_2}\ , \ldots, G_{a_m}$ are the graphs containing suggested paths for the corresponding origin concepts. The suggestions or suggested paths contain the set of concepts, relationships, and weights

$$G_{a_1} = (V_{a_1},\ E_{a_1},\ w_{a_1}),\ G_{a_2} = (V_{a_2},\ E_{a_2},\ w_{a_2}), \ldots,\ G_{a_m} = (V_{a_m},\ E_{a_m},\ w_{a_m}). \tag{10}$$

(Note: if no path exists between the source and target concepts, the result is nil.)

If the service getMappingSuggestions is called again, and the input is the same as in the previous time or contains some common connected vertices, the weight of the relationship between the inputs decreases. This enables a dynamic approach to evaluate the semantic distance, compared to [11], for instance, where the authors also consider a semantic distance, but the value of each relationship is static and equal to 1. The static weight of a relationship means that the semantic distance cannot vary regarding the number of cases in which the concept was used with the related counterpart. Instead, we consider that the importance or "closeness" between interrelated concepts grows with each run of the getMappingSuggestion service with repeated relationships inside the input. An example can be coffee and cake. For instance, a person asking for a coffee also takes a cake, this means that these two concepts are, from now, interrelated, and their "closeness" grows each time someone orders coffee and

cake. As Dijkstra's algorithm finds the weighted shortest path between two concepts, after each match, the weight of the corresponding relationship decreases according to:

$$w_j = \frac{w_{j-1}}{1 + w_{j-1}}, \tag{11}$$

where w_j is the current value of the weight between two concepts, and w_{j-1} is the value of weight before this iteration. In this way, the value of the weight will constantly decrease with each iteration; on the contrary, the "closeness" grows. This can be applied for two standalone concepts, or, if it is a model with a set of interconnected concepts, the weight of all segmental relationships decreases, and the "closeness" of the connected concepts grows correspondingly.

Algorithm 1 builds the suggestions for each concept in the origin model and is shown below:

Algorithm 1 Suggestion building for each concept in the origin model

INPUT: origin and target labels A and B, nondirected graph $G = (V, E, w)$, with relationships weights $w_e \in R$ for all $w \in E$, source and target concepts $a_m,\ b_n \in V$
for each *a* in A **do**
define all possible paths from *a* to each *b* in B
if there is a path *p* form *a* to *b* in *G* **then**
apply ***Dijkstra's algorithm*** to all *a*-*b* pairs
order resulting paths by distance (or semantic distance)
if the *number* of resulting paths < 4 **then**
collect *number* shortest paths
else
collect 4 shortest paths
end if
else
assign nil
end if
end for each
OUTPUT: *number* ordered paths with assigned distance

Based on the suggestions built for the concepts in origin model, an output document containing the suggestions is generated, as shown in Figure 5:

- "origin"—which contains the subfields "label" for the name of the concept and "XPath" containing the parent nodes;
- "mapping"—represents mapping the origin and target data models, containing all suggestions for each specific concept. The subfields are:
- "suggestion", with "label" naming the target concept, "score" reflecting the importance or weight of the path between "origin" and "target" concepts, "XPath" for the parent concepts, and "rank", which serves to identify which suggestion is the closest one;
- "graph"—representing the full path from the "origin" to the "target" concept in a segmental way, being composed of triples of the form "start" -> "weight" -> "end"; and
- "CYPHER query"—a field that contains the generated Cypher query, which can be used for visualizing and checking the suggested path.

```
[{"origin": {
              "label": "Emergency Stop",
              "xpath": ["Manufacturing", "Conveyor"]},
"mapping": [
[{"suggestion": {
                        "label": "Assembly",
                        "score": 0.6000000000000001,
                        "xpath": [],
                        "rank": 1
              },
              "graph": [{
                        "start": "Emergency Stop",
                        "weight": 0.2,
                        "end": "Conveyor"
              }, {
                        "start": "Conveyor",
                        "weight": 0.2,
                        "end": "Manufacturing"
              }, {
                        "start": "Manufacturing",
                        "weight": 0.2,
                        "end": "Assembly"
              }],
      "CYPHER_query": " MATCH (Emergency_Stop:concept{name: 'Emergency Stop'}) MATCH
      (Conveyor:concept{name: 'Conveyor'}) MATCH (Manufacturing:concept{name: 'Manufacturing'}) MATCH
      (Assembly:concept{name: 'Assembly'}) RETURN (Emergency_Stop)-[:link]-(Conveyor), (Conveyor)-[:link]-
      (Manufacturing), (Manufacturing)-[:link]-(Assembly)"
}],
...
]
}]
```


Figure 5. The output of the getMappingSuggestion service.

The output provides a set of suggestions for each of the origin concepts, and the Cypher query is used for visualizing and checking the suggestions. Once receiving the JSON input, the semantic manager identifies the origin and target concepts. If these two models, after insertion into the graph DB, are connected through some intermediary concepts (i.e., crossing points), a set of suggestions or the shortest paths from each origin to a set of target concepts are returned. The number of returned suggestions depends on the size, complexity, and depth of the graph; however, for this particular test case, it cannot exceed four suggestions. Nevertheless, the number of returned suggestions can be changed on demand.

The getMappingSuggestion service finds the interrelation between concepts placed in the origin and the target data models. Data models are inserted into the graph DB and, thus, are converted to a graph, enabling the use of the shortest path algorithms. The concept of interrelation is based on the "closeness", which is calculated as the shortest weighted path between each concept in the origin and a set of concepts in the target data models. The difference between the approach used in this work and previous solutions is that the weighted approach is used, which distinguishes the weight, and thus "importance", of relationships between concepts. Moreover, the weight is not static; it changes with each repetition of the relationship that is based on the co-occurrence principle. This principle assumes that if two related concepts are repeated, the "closeness" of them to each other grows, so the result of the mapping can be different in different time stamps. Another novel point is that getMappingSuggestion compares the "closeness" of a set of concepts to one chosen concept and orders them accordingly. This has good potential for further reasoning, allowing patterns in the data to be found and related concepts to be discovered.

4.2. Import Data Model Service

To enrich the output results of the getMappingSuggestion service, models built based on data generated by machines and sensors can be imported for further processing and analysis. The format for importing models is comma-separated values (CSV), which presumes data storage in a tabular way. It offers a simple way for data exchange in a tabular form. From the name itself, the data are represented in rows and columns, where the first row usually represents a set of headers that determines the meaning of the data in each column. This format can be adopted to exchange the data among the applications using the proprietary format. One of the main advantages of this format is its

simplicity. The standard which specifies CSV is RFC 4180 [48], although separation can be made using, for instance, semicolons instead of commas.

An example of a CSV file can be a document containing data on the functioning of a machine or its parts. For instance, data about the functioning of cylinders of a tabber stringer machine may contain headers such as the current time stamp, number of cylinders, movement types, and maximal and minimal speeds. CSV files are imported into Neo4j by loading the headers first followed by all other records regarding the corresponding header. If the value in the record is complex, in other words, it consists of several parts, it can be separated and parts can be inserted independently. This service is just an auxiliary or supporting service that can enable data in CSV format to be loaded, thus enriching the knowledge base with the new data. In principle, this service treats the imported data as graph entities to apply the getMappingSuggestion service or other services that can be added on demand.

4.3. Import Ontology Service

Another core service allows us to import RDF ontologies to the semantic management component. One of the main purposes of the semantic management component is to provide an extensible environment for ontology development, improvement, and evolution. The basic principles of ontology development can be summarized as follows [49]:

- reusability, when the concepts introduced in existing reliable ontologies can be reused;
- semantic alignment, referring mainly to ontology interoperability, to integrate concepts from imported ontologies, as well as newly created concepts into an existing structure;
- ontology design pattern usage, to ensure that the concept generation procedure can be applied not only to a single concept but to a group of concepts; and
- community extensibility, assuming a collaborative perspective when one ontology, covering few use-cases, can be extended by other users of a community and, thus, be applied to more use-cases.

The process of importing ontologies should correlate with these ontology development principles. Thus, in the present work, the concepts imported within an ontology are reusable, integrated in the existing structure, and extended using other services delivered within the semantic management component.

To introduce the process of ontology import, some definitions from the graph theory are adopted. For the ontology or data model, which is already inside the graph DB, the term "initial" is used, and for ontology loaded from outside, the term "imported" is used. Both initial and imported ontologies are represented as a graph with E edges and V vertices, $G = (V, E)$. If the database is not empty, the process of importing an ontology can be described through the logical disjunction or union. Let us assume that the database is not empty; thus, the initial data model structure that is already inside the database is $G_{init} = (V_{init}, E_{init})$, and the imported ontology is $G_{imp} = (V_{imp}, E_{imp})$.

Using this definition, edges are imported is as follows:

$$E_{init} \cup E_{imp} = \left(e_1^{init}, e_2^{init}, \ldots, e_n^{init}\right) \cup \left(e_1^{imp}, e_2^{imp}, \ldots, e_m^{imp}\right), \quad (12)$$

where n is the number of relationships connecting concepts in the initial ontology, and m is the number of relationships connecting concepts in the imported ontology.

The importing of concepts can be represented through:

$$V_{init} \cup V_{imp} = \left(v_1^{init}, v_2^{init}, \ldots, v_i^{init}\right) \cup \left(v_1^{imp}, v_2^{imp}, \ldots, v_j^{imp}\right), \quad (13)$$

where i is the number of concepts in the initial ontology, and j is the number of concepts in the imported ontology.

The resulting graph is represented by G_{res}, which is the union of the initial ontology graph (already inside the DB) and the imported ontology, and therefore contains the concepts V_{init} and V_{imp} and relationships E_{init} and E_{imp}:

$$G_{res} = G_{init} \cup G_{imp} = \Big(\big(E_{init} \cup E_{imp}\big), \big(V_{init} \cup V_{imp}\big)\Big). \tag{14}$$

Based on these definitions, the algorithm for importing an ontology to a nonempty graph DB is developed (Algorithm 2). If empty, the imported model is inserted as is. However, the most interesting is the first case, when the DB already contains some data models or ontologies with some common concepts and/or relationships. For this case the algorithm is as follows:

Algorithm 2 Suggestion building for each concept in the origin model

INPUT: nondirected graph $G_{init}=(V_{init}, E_{init}, w)$, with $e^{init} \in E_{init}$ edges and $(v_1^{init}, v_2^{init}, \ldots, v_i^{init}) \in V_{init}$, vertices and edge weights $w_{e^{init}} \in R$ for all $w \in E_{init}$, nondirected graph $G_{imp}=(V_{imp}, E_{imp})$, with $e^{imp} \in E_{imp}$ edges and $(v_1^{imp}, v_2^{imp}, \ldots, v_j^{imp}) \in V_{imp}$ vertices
for each e^{init} *in* E_{init} **do**
if $e^{imp} \in E_{init}$ **then**
imported edge exists in the initial model → skip
else
$E_{init} \cup E_{imp}$
end if
end for each
for each (v_i^{init}, v_j^{init}) *in* V_{init} **do**
if $(v_k^{imp}, v_l^{imp}) \in V_{init}$ **then**
imported vertices exist in this initial model → skip
else
$w_{e^{init}} = \frac{w_{e^{init}-1}}{1+w_{e^{init}-1}}$
end if
end for each
OUTPUT: G_{res}

As mentioned before, if the database is not empty and the imported ontology contains some concepts with the same names as in the database, in other words they equally labelled, they are ignored by importOntology, but the relationships are created or updated. The importOntology can be used in conjunction with the getMappingSuggestion service, which provides the necessary design patterns to apply to groups of concepts. Finally, all users of vf-OS having the permit to create and update the ontologies and can contribute to the evolution of the ontology development environment. The procedure of requesting the importOntology service is presented in Figure 6.

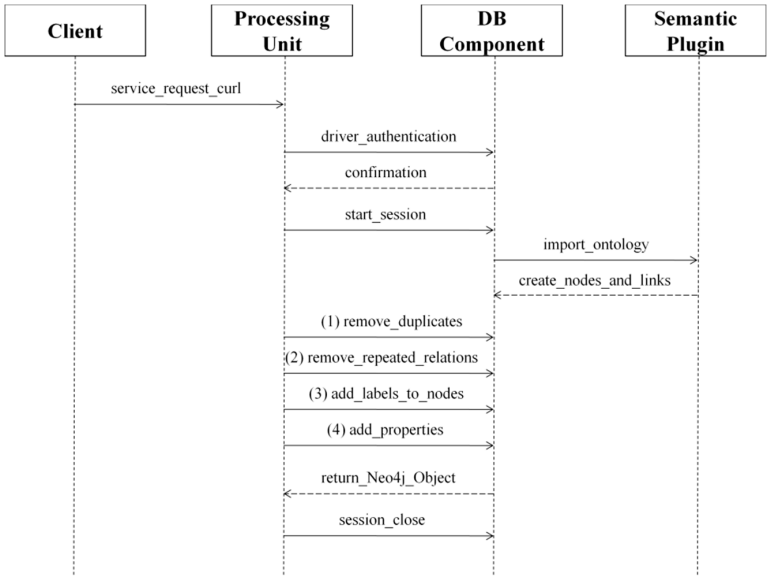

Figure 6. Procedure for requesting the importOntology service.

The importOntology service algorithm has a similar starting point compared to all implemented services; when a request has been sent, an authentication procedure is passed, and the session is launched. Then, the semantic plugin is required to parse and insert the ontology, with all concepts and relations, into Neo4j. After this is accomplished, duplicate concepts are removed, and, if relationships are identical between the imported ontology and the database, the weight of the relationship is updated. In the next step, each concept is labelled within Neo4j, and the name property is extracted.

Considering the treatment of replicas, as in the example given in [50], concepts with equal names are removed. However, while importing two ontologies with similar concepts and applying unification procedures, the resulting concept class possesses all properties from the initial ontologies. In this work, all concepts with equal names are also removed, although we follow the approach that all properties of the concepts are inserted as separate concepts. For instance, the concept of "cake" has the property "sweet", which, after insertion, appears as another concept having a relation to "cake". This allows concepts with similar properties to be interrelated and, thus, enriches the knowledge base and avoids the creation of redundant entities. Moreover, a matching approach is made possible, which is commonly and widely used. This approach is illustrated in [51], where two ontologies are merged combining equal root concepts while differentiating between similar, but not equivalent, concepts. The process of matching ontologies is also considered in [52]. However, the authors were mostly focused on ontology matching or alignment from the same domain, and they did not consider importing cross-domain ontologies.

The importOntology service enables RDF/OWL ontologies to be imported. One of the tasks of the importOntology service, as well as of the importDataModel, is to enrich the knowledge base with data structures aggregated from various sources, such as, for instance, professional communities. The key feature of the proposed service is that it treats the properties of a concept as separate concepts, thus enabling different concepts to be interrelated based on their similar properties, which is crucial when using the graph-based approach. This might be used, for instance, as the basis for building service discovery mechanisms for offering the appropriate service. It is important to mention that, to explore the functionality of importOntology efficiently, it is used in conjunction with getMappingSuggestion. Moreover, importOntology has no domain restrictions as long as an RDF specification is used.

Considering contribution of this service to enhance collaborative mechanisms, importOntology allows aggregation of ontologies developed within various professional communities and supports identification of closely related concept patterns, which is important for facilitating a common understanding of the used terms and definitions. Another relevant feature is that importOntology,

in conjunction with other developed services, allows the ontology to make and change necessary additions. For instance, the concept "cake" is related to the concept "sweets", but the definition can be improved, whereas "cake" is also "food" and "consumable thing". Thus, all members of a community can contribute to the enrichment of the knowledge base.

5. Test Case Scenario and Discussion

The goal of this work is to enable a knowledge base that can accumulate various ontologies and data models, whereas a solution to the main challenges of semantic discovery architectures can be provided [18]. This can be accomplished by considering weighted ontologies and adaptive techniques for collaborative knowledge base formation. Therefore, the aim is not to improve the timing of existing methods and solutions, but rather to focus on functionality. Another goal is to show how the export of ontologies in conjunction with collaborative knowledge base formation can contribute to improving the discovery mechanisms.

As a proof-of-concept test case, we decided to import an ontology developed in the framework of the Federated Interoperable Semantic IoT Testbeds and Applications (FIESTA)-IoT project, which aims at covering the core notions of the IoT [53]. The ontology uses the resource description framework (RDF) graph vocabulary and schema with some core constructs such as class, property, type, subClassOf, subPropertyOf, domain, and range [54]. Moreover, web ontology language (OWL) syntax is integrated into the RDF schema to extend its functionality and thus enrich it with new meanings of triples [55]. To implement the mapping, the semantic plugin for Neo4j was used [56], which ensures proper export of the ontology into the database.

Besides the RDF graph extended with OWL syntax as the input for the semantic management component, as described in previous sections, the input model in JSON format was used. For testing purposes, importOntology and getMappingSuggestion services were chosen to demonstrate how the importOntology service can contribute to knowledge base enrichment, improvement of the mapping output, and introducing collaborative mechanisms for improving the knowledge base. Moreover, a second purpose is to demonstrate the capabilities of the combined use of getMappingSuggestion with importOntology. First, the importOntology service was executed, which loads the ontology to the graph DB. In the next stage, getMappingSuggestionService was activated to produce a complex mapping result with a set of suggestions for each concept in the origin submodel of the input document (Figure 7).

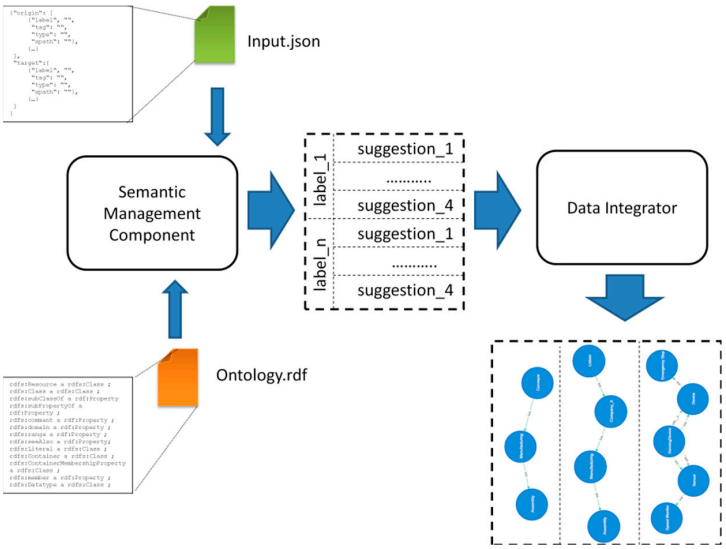

Figure 7. High-level view of the usage of the semantic management component and data integrator.

The input model, as described above, contains two main parts, the origin submodel and the target submodel, which are mapped. Following the definitions described in Section 4, the input model that is used for the test case is illustrated in Figure 8. This origin model is represented through set A = {*Manufacturing*, *Conveyor*, *Emergency Stop*, *Company_X*, *Lisbon*} and the target set as B = {*CoffeeMachine*, *Assembly*, *Company_Y*, *Porto*, *Speed Monitor*}.

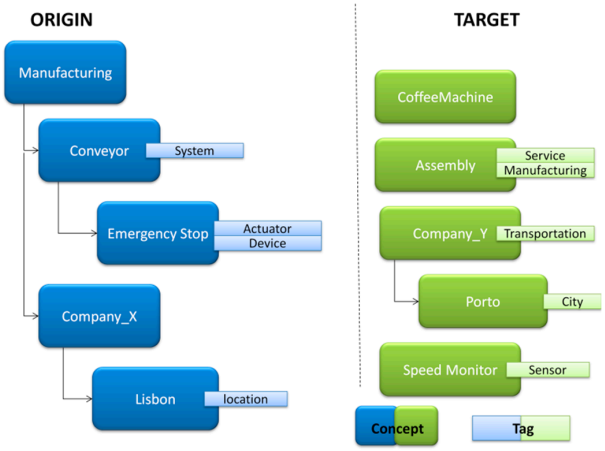

Figure 8. Schema of payload of the getMappingSuggestion service.

Each origin concept is mapped to every target concept and produces a suggestion that is ranked according to the weight of the path between the origin and target concepts:

$$\begin{aligned} A \times B = \{&(Manufacturing,CoffeeMachine),(Manufacturing,Assembly),\\ &(Manufacturing,Company\_Y),(Manufacturing,Porto),(Manufacturing,\\ &Speed\ Monitor),\ldots,(Lisbon,CoffeeMachine),(Lisbon,Assembly),\\ &(Lisbon,Company\_Y),(Lisbon,Porto),(Lisbon,Speed\ Monitor)\}. \end{aligned} \tag{15}$$

In the origin and target models, there are several IoT resources and services, but also location names and company names. In an extended ontology (not in FIESTA ontology), companies can be considered as users or service providers. Using the output from the semantic management component, further knowledge extraction can be made, and "optimal" mapping patterns can be found. To better describe and explain the process of ontology metamorphosis and generation of complex outputs, a conceptual architecture for semantic lifting [57] was adopted (Figure 9). This architecture is intended to provide the necessary basis for ensuring interoperability among model elements and ontology concepts.

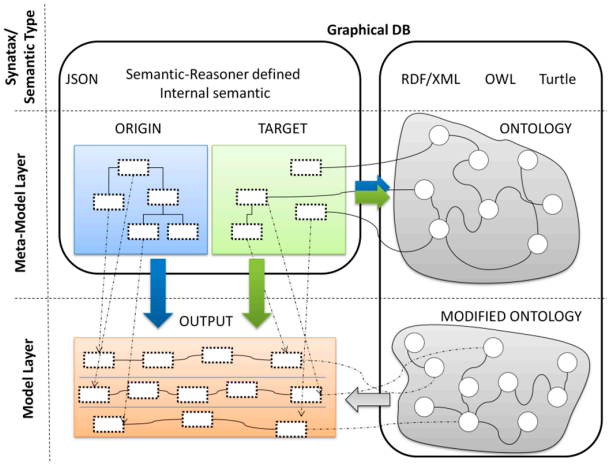

Figure 9. Metamorphosis of ontologies and output building.

The conceptual architecture contains three scales: the first reflects the syntax and semantic annotations used. The input model for the getMappingSuggestion service, containing origin and target data models, follows JSON formatting, with an internal semantic developed for the semantic management component to be consumed by other vf-OS framework components, including the data integrator. Ontologies that can be imported by the semantic management component are of two main types: RDF/XML extended with OWL syntax and Turtle. The metamodel layer represents the mapping of the imported ontology with origin and target data models. In the modeling layer, the initial ontology is modified in consideration with the origin and target models. Moreover, the output model is generated while combining both input models and modified ontology.

Resulting suggestions contain the first node acquired from origin model, the last node from the target model, and the sequence of the in-between nodes. These nodes can be acquired both from the input models and modified ontology, where all new concepts, if any, are already incorporated. The whole process is represented in Figure 9.

The first column in Figure 10 identifies the origin and target concepts (i.e., labels between the established mappings). For this specific case, one of the origin concepts has been chosen, namely the "*Emergency Stop*" actuating device, with appropriate suggestions that are presented in the target model of Figure 8. The second column is used to represent the Cypher queries generated to query the result. The concepts in the graph are marked as follows: (i) blue concepts were extracted from origin data model; (ii) green concepts were from the target data model; and (iii) grey concepts were extracted from imported ontology. The records in Figure 10 represent suggestions after the mapping process. Figure 10 also represents how the suggestions for the same origin and target concepts are changed if the relations among intermediary concepts evolve. As it can be seen, the number of intermediary nodes increased, though the distance or semantic distance between the nodes "*Emergency Stop*" and "*Assembly*" decreased, making the new suggestion preferable.

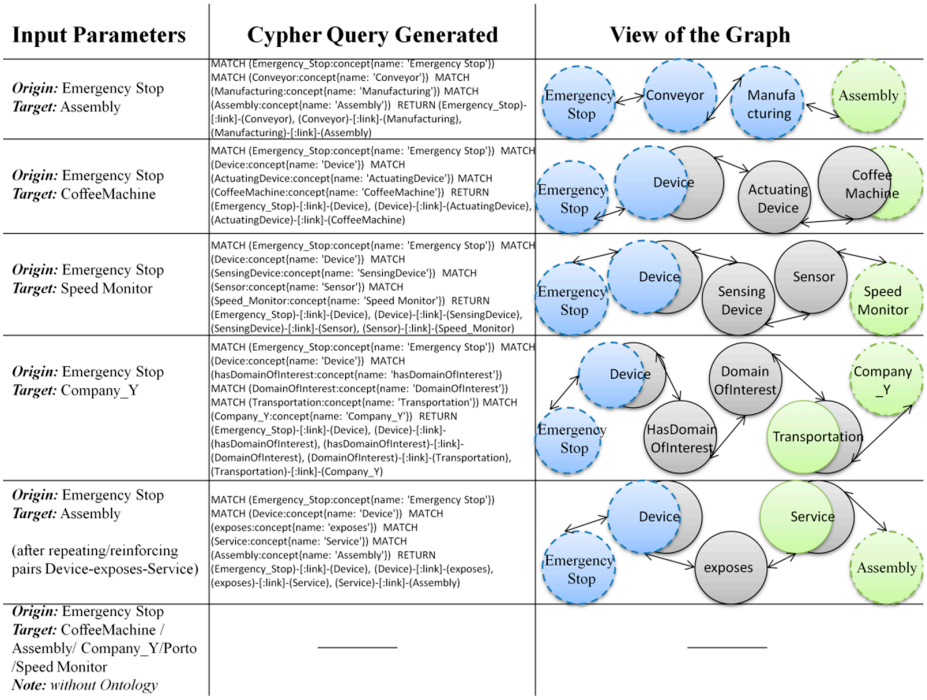

Figure 10. Illustrative results.

As an example, the "*Emergency Stop*" concept from the input model is chosen. The resulting graph containing all 5 concepts from the origin data model is

$$G_{res} = (Manufacturing,\ Conveyor,\ Emergency\ Stop,\ Company\_X,\ Lisbon), \tag{16}$$

and the subgraph containing the suggestion for "*Emergency Stop*" is

$$\text{“}Emergency\ Stop\text{”} = \left(V_{pw},\ E_{pw},\ w_{pw}\right), \tag{17}$$

where V_{pw} is a set containing all the vertices forming the resulting graph based on four suggestions (Figure 10):

$$\begin{aligned} V\_pw = \{&Emergency\ Stop, Conveyor, Manufacturing, Assembly, Device,\\ &ActuatingDevice, CoffeeMachine, SensingDevice, Sensor, Speed\ Monitor,\\ &HasDomainOfInterest, DomainOfInterest, Transportation, Company\_Y\}. \end{aligned} \tag{18}$$

E_{pw} is a set containing all the edges between vertices in the path forming the suggestions:

$$\begin{aligned} E\_pw = \{&\{(Emergency\ Stop, Conveyor), (Conveyor, Manufacturing),\\ &(Manufacturing, Assembly)\}, \{(Emergency\ Stop, Device),\\ &(Device, ActuatingDevice), (ActuatingDevice, CoffeeMachine)\},\\ &\{(Emergency\ Stop, Device), (Device, SensingDevice), (SensingDevice, Sensor),\\ &(Sensor, Speed\ Monitor)\}, \{(Emergency\ Stop, Device),\\ &(Device, HasDomainOfInterest), (HasDomainOfInterest, DomainOfInterest),\\ &(DomainOfInterest, Transportation), (Transportation, Company\_Y)\}\}, \end{aligned} \tag{19}$$

and w_{pw} stands for a set of weights of the path segments in the suggestions. Thus, for the "*Emergency Stop*" concept, the set of segment weights for suggestions is as follows:

$$w_{pw} = \{\{(0.2, 0.2, 0.2)\},\ \{(0.2, 1, 1)\}, \{(0.2, 1, 1, 0.2)\}, \{(0.2, 1, 1, 1, 0.2)\}\}, \tag{20}$$

where four suggested paths are generated for each origin concept. For instance, the first element of the set is (0.2,0.2,0.2) which is the weights for segments "*Emergency Stop*" -> "*Conveyor*", "*Conveyor*" -> "*Manufacturing*" and "*Manufacturing*" -> "*Assembly*" of the first suggestion.

The distance from origin concept "*Emergency Stop*" to target concept "*Assembly*" is

$$d(Emergency\ Stop, Assembly) = \min\{w(p) : p_1 = (0.6)\}. \tag{21}$$

It is clear that some concepts overlap, as they are members of both the imported ontology and origin data or target data model. However, without the imported ontology, no relation among the concepts can be established, as the ontology extends the origin and target with missing relations to other concepts, which makes the mapping process possible. Without the imported ontology, the output result of getMappingSuggestion would be an empty object. This case is described in the last record in Figure 10, with the note that no ontology is being imported.

6. Conclusions

The aims of Industry 4.0 set a wide range of challenges both technical and organizational in nature. Many initiatives have been launched all around the world to enable the technological support for the fourth industrial revolution. CPSs are one of the main constituents, covering both physical and virtual components, that can generate and process rich ascending and descending data flows. Therefore, data and knowledge assets are turning into a resource that needs to be exploited to enable higher intelligence and autonomy of the systems and subsystems. In this work, we presented a part of the vf-OS initiative: the semantic management component. Some of the most important services of this component, such as importOntology, importDataModel, and getMappingSuggestion, were described and discussed. The focus has been made on data mapping and collaborative enriching of the data-rich environment. Further efforts were made to test the semantic management component through a combination of services, showing how imported ontologies or data models can influence the suggestions produced by the data mapping service. This is important since (i) relations among concepts are not static and can

evolve over time, and (ii) import of new ontologies and data models leads to introducing new concepts and relationships. These factors affect the outcome of the suggestions produced for a given concept.

As a part of further future work, some other services that add additional functionalities are to be developed. For instance, a service that can enable creation of ontologies within the graph DB as logical partitions is needed. Another service should enable reasoning over the knowledge base, using services developed within this work. An interface for integrating external data services for improving the mapping output is also in the scope of further research, as well as a service for integrating and reasoning over data models containing concepts that are synonyms or are very contextually interrelated. Furthermore, the developed services will also be tested with real-world data from smart home components and applications. Smart homes are a promising area, as it is crucial to find interdependencies among "things" represented as concepts and produce mappings between data models for various purposes such as optimization, hierarchy detection, and so on. Another promising task that is planned for the next stages is applying the semantic management component to support establishing smart things coalitions for the CPS.

Author Contributions: Conceptualization, A.A.N. and J.S.; Methodology, A.A.N., L.M.C.-M., and J.S.; Software, A.A.N.; Literature search and data sets, A.A.N., L.M.C.-M., J.S., and O.G.; Writing—original draft preparation, A.A.N.; Writing—review and editing, L.M.C.-M., J.S., O.G., and R.J.-G.; Supervision, L.M.C.-M. and J.S.; Project administration, J.S. and R.J.-G.

Funding: The research leading to these results received funding from the European Union H2020 Program under grant agreement No. 723710 "Virtual Factory Open Operating System" (vf-OS). This work was also funded in part by the Center of Technology and Systems (CTS) and the Portuguese Foundation for Science and Technology (FCT) through the Strategic Program UID/EEA/00066/2019.

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

References

1. Lee, J.; Azamfar, M.; Singh, J. A blockchain enabled Cyber-Physical System architecture for Industry 4.0 manufacturing systems. *Manuf. Lett.* **2019**, *20*, 34–39. [CrossRef]
2. Ancarani, A.; Di Mauro, C.; Mascali, F. Backshoring strategy and the adoption of Industry 4.0: Evidence from Europe. *J. World Bus.* **2019**, *54*, 360–371. [CrossRef]
3. Kavakli, E.; Buenabad-Chavez, J.; Tountopoulos, V.; Loucopoulos, P.; Sakellariou, R. WiP: An Architecture for Disruption Management in Smart Manufacturing. In Proceedings of the IEEE International Conference on Smart Computing, Taormina, Italy, 18–20 June 2018; pp. 279–281. [CrossRef]
4. Fraile, F.; Tagawa, T.; Poler, R.; Ortiz, A. Trustworthy Industrial IoT Gateways for Interoperability Platforms and Ecosystems. *IEEE Internet Things J.* **2018**, *5*, 2506–4514. [CrossRef]
5. Tao, F.; Qi, Q.; Wang, L.; Nee, A.Y.C. Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. *Engineering* **2019**, *5*, 653–661. [CrossRef]
6. Wermann, J.; Colombo, A.W.; Pechmann, A.; Zarte, M. Using an interdisciplinary demonstration platform for teaching Industry 4.0. *Procedia Manuf.* **2019**, *31*, 302–308. [CrossRef]
7. Nazarenko, A.; Camarinha-Matos, L.M. Basis for an Approach to Design Collaborative Cyber-Physical Systems. In *DoCEIS 2019: Technological Innovation for Industry and Service Systems*; Springer: Berlin/Heidelberg, Germany, 2019. [CrossRef]
8. Camarinha-Matos, L.M.; Fornasiero, R.; Afsarmanesh, H. Collaborative Networks as a Core Enabler of Industry 4.0. In *PRO-VE 2017: Collaboration in a Data-Rich World*; Springer: Berlin/Heidelberg, Germany, 2017. [CrossRef]
9. Wei, W.; Guo, C. A text semantic topic discovery method based on the conditional co-occurrence degree. *Neurocomputing* **2019**, *368*, 11–24. [CrossRef]
10. Dai, W.; Vyatkin, V. Transformation from PLC to Distributed Control using Ontology Mapping. In Proceedings of the IEEE 10th International Conference on Industrial Informatics, Beijing, China, 25–27 July 2012; pp. 436–441. [CrossRef]
11. Xia, H.; Hu, C.; Xiao, F.; Cheng, X.; Pan, Z. An Efficient Social-Like Semantic-Aware Service Discovery Mechanism for Large-Scale Internet of Things. *Comput. Netw.* **2019**, *152*, 210–220. [CrossRef]

12. Louge, T.; Karray, M.H.; Archimede, B.; Maamar, Z.; Mrissa, M. Semantic Web Services Composition in the astrophysics domain: Issues and solutions. *Future Gener. Comput. Syst.* **2019**, *90*, 185–197. [CrossRef]
13. Alshammari, M.; Nasraoui, O.; Sanders, S. Mining Semantic Knowledge Graphs to Add Explainability to Black Box Recommender Systems. *IEEE Access* **2019**, *7*, 110563–110579. [CrossRef]
14. Cousin, P.; Serrano, M.; Soldatos, J. Internet of Things Research on Semantic Interoperability to Address Manufacturing Challenges. In *Enterprise Interoperability: Interoperability for Agility, Resilience and Plasticity of Collaborations: IESA'14 Proceedings*; John Wiley & Sons: Hoboken, NJ, USA, 2015; pp. 21–30. [CrossRef]
15. Willner, A.; Diedrich, C.; Ben Younes, R.; Hohmann, S.; Kraft, A. Semantic communication between components for smart factories based on oneM2M. In Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017. [CrossRef]
16. Borch, N.T. Improving Semantic Routing Efficiency. In Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems, San Diego, CA, USA, 21 July 2015. [CrossRef]
17. Zhao, Z.; Liao, X.; Martin, P.; Maduro, J.; Thijsse, P.; Schaap, D.; Stocker, M.; Goldfarb, D.; Magagna, B. Knowledge-as-a-Service: A Community Knowledge Base for Research Infrastructures in Environmental and Earth Sciences. In Proceedings of the IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019; pp. 127–132. [CrossRef]
18. Mistry, S.K.; Kamal, M.H.; Mistry, D. Semantic Discovery of Web Services through Social Learning. *Procedia Technol.* **2012**, *3*, 167–177. [CrossRef]
19. Moeini, H.; Yen, I.-L.; Bastani, F. Service Specification and Discovery in IoT Networks. In Proceedings of the IEEE International Conference on Web Services (ICWS), Milan, Italy, 8–13 July 2019. [CrossRef]
20. Tsang, V.; Stevenson, S. A Graph-Theoretic Framework for Semantic Distance. *Comput. Linguist.* **2010**, *36*, 31–69. [CrossRef]
21. Quevedo, J.; Antunes, M.; Corujo, D.; Gomes, D.; Aguiar, R.L. On the application of contextual IoT service discovery in Information Centric Networks. *Comput. Commun.* **2016**, *89–90*, 117–127. [CrossRef]
22. Gries, S.; Hesenius, M.; Gruhn, V. Tracking Information Flow in Cyber-Physical Systems. In Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 2589–2590. [CrossRef]
23. Smirnov, A.; Kashevnik, A.; Shilov, N. Cyber-Physical-Social System Self-Organization: Ontology-Based Multi-Level Approach and Case Study. In Proceedings of the IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems, Cambridge, MA, USA, 21–25 September 2015; pp. 168–169. [CrossRef]
24. Goncalves, R.; Sarraipa, J.; Agostinho, C.; Panetto, H. Knowledge framework for intelligent manufacturing systems. *J. Intell. Manuf.* **2011**, *22*, 725–735. [CrossRef]
25. Cristani, M.; Demrozi, F.; Tomazzoli, C. ONTO-PLC: An ontology-driven methodology for converting PLC industrial plants to IoT. *Procedia Comput. Sci.* **2018**, *126*, 527–536. [CrossRef]
26. Martinez-Gil, J. An overview of textual semantic similarity measures based on web intelligence. *Artif. Intell. Rev.* **2014**, *42*, 935–943. [CrossRef]
27. Miller, G.A.; Charles, W.G. Contextual Correlates of Semantic Similarity. *Lang. Cogn. Process.* **1991**, *6*, 1–28. [CrossRef]
28. Song, S.; Lin, Y.; Guo, B.; Di, Q.; Lv, R. Scalable Distributed Semantic Network for knowledge management in cyber physical system. *J. Parallel Distrib. Comput.* **2018**, *118*, 22–33. [CrossRef]
29. Henkel, R.; Wolkenhauer, O.; Waltemath, D. Combining computational models, semantic annotations and simulation experiments in a graph database. *Database* **2015**, *2015*, bau130. [CrossRef]
30. Abay, N.C.; Mutlu, A.; Karagoz, P. A Path-finding Based Method for Concept Discovery in Graphs. In Proceedings of the 6th International Conference on Information, Intelligence, Systems and Applications (IISA), Corfu, Greece, 6–8 July 2015. [CrossRef]
31. Hoppen, M.; Rossmann, J.; Stapelbroek, A.-M.; Hiester, S. Managing Semantic World Models for eRobotics Applications Two Approaches Based on Object-Relational Mapping and on a Graph Database. *Int. J. Adv. Softw.* **2017**, *10*, 79–95. Available online: http://www.iariajournals.org/software/ (accessed on 7 September 2019).

32. Hor, A.-H.; Sohn, G.; Claudio, P.; Jadidi, M.; Afnan, A. A Semantic Graph Database for BIM-GIS Integrated Information Model for an Intelligent Urban Mobility Web Application. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *4*, 89–96. [CrossRef]
33. Song, S.; Sun, Y.; Di, Q. Multiple order semantic relation extraction. *Neural Comput. Appl.* **2019**, *31*, 4563–4576. [CrossRef]
34. Hristovski, D.; Kastrin, A.; Rindflesch, T.C. Semantics-Based Cross-domain Collaboration Recommendation in the Life Sciences. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 15), Paris, France, 25–28 August 2015. [CrossRef]
35. Kivikangas, P.; Ishizuka, M. Improving Semantic Queries by Utilizing UNL Ontology and a Graph Database. In Proceedings of the IEEE Sixth International Conference on Semantic Computing, Palermo, Italy, 19–21 September 2012. [CrossRef]
36. Malliaros, F.D.; Skianis, K. Graph-Based Term Weighting for Text Categorization. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 15), Paris, France, 25–28 August 2015. [CrossRef]
37. Nguyen, S.H.; Yao, Z.; Kolbe, T.H. Spatio-Semantic Comparison of Large 3D City Models in CITYGML Using a Graph Database. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *4*, 99–106. [CrossRef]
38. Rashidy, R.A.H.E.L.; Hughes, P.; Figueres-Esteban, M.; Harrison, C.; Van Gulijk, C. A big data modeling approach with graph databases for SPAD risk. *Saf. Sci.* **2017**, *110*, 75–79. [CrossRef]
39. Dai, W.; Dubinin, V.N.; Vyatkin, V. Migration from PLC to IEC 61499 Using Semantic Web Technologies. *IEEE Trans. Syst. Man Cybern.* **2014**, *44*, 277–291. [CrossRef]
40. Kibria, M.G.; Ali, S.; Jarwar, M.A.; Chong, I. A Framework to Support Data Interoperability in Web Objects Based IoT Environments. In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 18–20 October 2017; pp. 29–31. [CrossRef]
41. Pape, D.; Hinz, T.; Perales, O.G.; Fraile, F.; Flores, J.L.; Rubio, O.J. *vf-OS Architecture in Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability*; Zelm, M., Jaekel, F.-W., Doumeingts, G., Wollschlaeger, M., Eds.; ISTE Ltd.: London, UK, 2018; pp. 77–82. [CrossRef]
42. Nazarenko, A.A.; Giao, J.; Sarraipa, J.; Saiz, O.J.; Perales, O.G.; Jardim-Gonçalves, R. Data Management Component for Virtual Factories Systems. In *Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability*; Zelm, M., Jaekel, F.-W., Doumeingts, G., Wollschlaeger, M., Eds.; ISTE Ltd.: London, UK, 2018; pp. 99–106. [CrossRef]
43. Xie, M.; Lu, J.; Chen, G.; Wu, M.-Y. Folksonomy-Based Internet Object Profiling and Relation Extracting. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017. [CrossRef]
44. Solskinnsbakk, G.; Gulla, J.A.; Haderlein, V.; Myrseth, P.; Cerrato, O. Quality of hierarchies in ontologies and folksonomies. *Data Knowl. Eng.* **2012**, *74*, 13–25. [CrossRef]
45. Neo4j Labs. Available online: https://neo4j.com/docs/labs/apoc/3.4/overview/ (accessed on 25 July 2019).
46. Berkeley EECS. Available online: https://people.eecs.berkeley.edu/~{}nirkhe/cs38notes/graph.pdf (accessed on 3 September 2019).
47. Dietsel, R. *Graph Theory*; Electronic Edition; Springer: New York, NY, USA, 2000; Available online: http://www.esi2.us.es/~{}mbilbao/pdffiles/DiestelGT.pdf (accessed on 3 September 2019).
48. Common Format and MIME Type for Comma-Separated Values (CSV) Files, Network Working Group, Request for Comments: 4180, 2005. Available online: https://tools.ietf.org/html/rfc4180 (accessed on 7 August 2019).
49. Xiang, Y.; He, Z.; Zheng, J.; Lin, Y.; Overton, J.A.; Ong, E. The eXtensible ontology development (XOD) principles and tool implementation to support ontology interoperability. *J. Biomed. Semant.* **2018**, *9*, 3. [CrossRef]
50. Panagiotopoulos, I.; Kalou, A.; Pierrakeas, C.; Kameas, A. An Ontological Approach for Domain Knowledge Modeling and Management in E-Learning Systems. In Proceedings of the 8th International Conference on Artificial Intelligence Applications and Innovations (AIAI), Halkidiki, Greece, 27–30 September 2012; pp. 95–104. [CrossRef]
51. Arnold, P.; Rahm, E. Enriching ontology mappings with semantic relations. *Data Knowl. Eng.* **2014**, *93*, 1–18. [CrossRef]

52. Annane, A.; Bellahsene, Z.; Azouaou, F.; Jonquet, C. Building an effective and efficient background knowledge resource to enhance ontology matching. *J. Web Semant.* **2018**, *51*, 51–68. [CrossRef]
53. Agarwal, R.; Gomez Fernandez, D.; Elsaleh, T.; Gyrard, A.; Lanza, J.; Sanchez, L.; Georgantas, N.; Issarny, V. Unified IoT Ontology to Enable Interoperability and Federation of Testbeds. In Proceedings of the IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 12–14 December 2016; pp. 70–75. [CrossRef]
54. W3C Working Group Note. 24 June 2014. Available online: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/ (accessed on 23 June 2019).
55. OWL Web Ontology Language, Reference W3C Recommendation. 10 February 2004. Available online: https://www.w3.org/TR/owl-ref/ (accessed on 23 June 2019).
56. Barrasa, J. Building a Semantic Graph in NEO4J. April 2016. Available online: https://jbarrasa.com/2016/04/06/building-a-semantic-graph-in-neo4j/ (accessed on 15 June 2019).
57. Hinkelmann, K.; Gerber, A.; Karagiannis, D.; Thoenssen, B.; van der Merwe, A.; Woitsch, R. A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. *Comput. Ind.* **2016**, *79*, 77–86. [CrossRef]