A High-Level Ontology Network for ICT Infrastructures

Oscar Corcho[1(✉)], David Chaves-Fraga[1], Jhon Toledo[1], Julián Arenas-Guerrero[1], Carlos Badenes-Olmedo[1], Mingxue Wang[2], Hu Peng[2], Nicholas Burrett[2], José Mora[2], and Puchao Zhang[2]

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
{ocorcho,dchaves,jatoledo,cbadenes}@fi.upm.es, julian.arenas.guerrero@upm.es

[2] Huawei Research Ireland, Dublin, Ireland
{wangmingxue1,patrick.hupeng,nicholas.burrett,jose.mora, zhangpuchao}@huawei.com

Abstract. The ICT infrastructures of medium and large organisations that offer ICT services (infrastructure, platforms, software, applications, etc.) are becoming increasingly complex. Nowadays, these environments combine all sorts of hardware (e.g., CPUs, GPUs, storage elements, network equipment) and software (e.g., virtual machines, servers, microservices, services, products, AI models). Tracking, understanding and acting upon all the data produced in the context of such environments is hence challenging. Configuration management databases have been so far widely used to store and provide access to relevant information and views on these components and on their relationships. However, different databases are organised according to different schemas. Despite existing efforts in standardising the main entities relevant for configuration management, there is not yet a core set of ontologies that describes these environments homogeneously, and which can be easily extended when new types of items appear. This paper presents an ontology network created with the purpose of serving as an initial step towards an homogeneous representation of this domain, and which has been already used to produce a knowledge graph for a large ICT company.

Keywords: Configuration management database · Ontology network · Knowledge graph

Resource type: Ontology
License: CC BY 4.0 International
DOI: 10.5281/zenodo.4701264
URL: http://w3id.org/devops-infra

1 Introduction

Most ICT organisations (IT service providers, cloud providers, telecom industry, etc.) are witnessing, in recent years, the growing amount and interdependencies of hardware and software components that they need to handle as part of their infrastructure. Distinctions between hardware and software-related functionalities are sometimes blurring due to virtualisation: some hardware items may now be virtualised as software (e.g., virtual machines, DNSs). Terms like infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS), etc., are now part of the ICT infrastructure jargon, and new terms are appearing (e.g., AI as a service -AIaaS-). This makes these environments increasingly difficult to track and understand.

Information about all these physical or virtual components has been traditionally handled in several types of (often loosely interconnected) databases: configuration management databases (CMDB), IT Service Management (ITSM) systems, IT Asset Management (ITAM) databases and tools, etc. The first group of databases (CMDBs) store and provide access to relevant information on these components and their relationships, providing organised views of configuration data and dynamic views on their functioning. These databases have evolved in recent years to reflect not only those components, but also the DevOps universe of technology and practices, including software configuration scripts, containers, cloud resources, etc. The second group (ITSM) is focused more exclusively on service management KPIs. The latter (ITAM) is usually more static and provide general information about the lifecycle of hardware components (purchase information, suppliers, disposal, etc.). There are many other types of products, tools and databases that provide support for other parts of the global ICT architecture of an organisation, following general architectural frameworks such as those identified in the Open Group Architecture Forum (TOGAF) [19].

Our work focuses on the description of the items and relationships normally covered under the umbrella of CMDBs and ITSMs, and the four major tasks that these systems and databases provide support to, according to the IT Infrastructure Library (ITIL) service management framework [2]:

– **Discovery**. Identify and catalogue the resources and groups of resources (also known collectively as configuration items) that are managed by the organisation or influence in the delivery of products and services.
– **Security**. Control that data can only be accessed and/or changed by those individuals or services that are authorised to do so.
– **Maintenance and Reporting**. Record, maintain, update and report the current status of all the handled resources (e.g., a server is up and running or idle, an IP address may be assigned or not).
– **Auditing and Recovery**. Verify that the data about all resources is accurate and identify causes of errors, so that remedial actions can be taken (by humans or by software). For instance, identify which systems may be affected by an outage and which groups of actions should be taken to repair its negative consequences.

Our expectation is that a set of ontologies focused around these main entities will allow organisations to have a global unified view of all of their resources, and provide better support for the aforementioned tasks, abstracting away from the characteristics of the underlying data sources. Knowledge graphs created according to these ontologies may also allow connecting the data commonly used in CMDBs and ISTMs with other data sources (product details from providers, product and service databases, CRM and ERP systems, etc.).

Contribution. The main contribution of our work is the development of **an ontology network that identifies and captures high-level entities (configuration items, resources and resource groups) that are common across configuration and IT Service management databases, together with the most common relationships among them**. This network is composed of a top-level ontology, describing general characteristics of all configuration items, and a set of 9 interconnected ontologies to represent: organisations, product and service descriptions, data centers, server infrastructure, network components and services, software, databases, hardware components and network security (including digital certificates). This ontology network results from the joint work of a team of ontology engineers and domain experts for approximately one year, following state-of-the-art ontology development practices. The resulting ontology network reflects the shared agreement on the core types of resources dealt with in this domain, and has been used as the basis for the creation of a knowledge graph related to the cloud and DevOps operations at a telecommunications company (Huawei). We expect this ontology network to serve as an initial step towards the standardisation of the main resources to be managed in the context of CMDBs and ISTMs in the future.

The remainder of this paper is structured as follows: Sect. 2 motivates our work with a typical use case related to disaster recovery for a cloud service provider. Section 3 describes our methodological approach for the development of a high-level network of ontologies in the area of configuration management and IT service management. Section 4 describes the ontology network, which consists of a top-level ontology and a set of 9 interconnected ontologies. Section 5 details how the ontology network has been used to drive the creation of a knowledge graph (KG) from Huawei's CMDB, using declarative RDB2RDF mappings and related technology, and how the KG can be exploited using SPARQL queries and a natural language question answering system, which is planned to be integrated in a chatbot. Section 6 presents some related work, pointing to previous efforts on the development of ontologies in this area, and their main limitations. And Sect. 7 outlines the main conclusions derived from the ontology development process and from the current set of ontologies, and describes future lines of work.

2 Motivational Example

Our work is motivated by the real-world challenges problems that a large cloud provider has to face in an increasingly demanding context where continuous integration and continuous deployment are more frequent and automated. As discussed in the introduction, the increase in the complexity of the underlying infrastructures and the runtime constraints imposed by DevOps practices increases the amount of problems and the costs associated to infrastructure maintenance and recovery operations. And this is expected to grow even more in the near future with the addition of AIOps [5] on top of current DevOps.

A typical example is the scenario where a sudden drop in performance (e.g., data throughput) is detected in the provision of a cloud-based product or service (e.g., in a video transmission application that uses a content delivery network). For this performance drop to be dealt with, site reliability engineers (SREs) need to inspect first the topology of services and microservices (e.g., object storage services, domain name services, elastic cloud servers) that are used by the product where the problem is detected. These services and microservices are running in specific servers and clusters and are based on a specific software module version, available in some software directory. At the same time, those servers (commonly virtual servers) are running on specific configurations of hardware items (including hardware servers, network cards, etc.) that are hosted in a data center. And this should be done independently of which specific types of infrastructures, hardware and software providers are being used.

By having a comprehensive view of all the components that are involved in this context, the SREs (usually in conjunction with AI algorithms prepared for that purpose) can perform the described actions with fewer and shorter queries and fewer actions, which is especially important when time is a pressing matter. Shorter queries and fewer actions implies less potential human errors, e.g. overlooking small groups of software or hardware elements as larger groups are generally more relevant.

3 Methodological Approach for Ontology Development

We have built the ontology network following the Linked Open Terms (LOT) methodology [17], whose main actors, activities and artefacts are depicted in Fig. 1. This methodology is rooted on NeOn methodology [18] and inspired by agile software development techniques, with sprints and iterations representing the main workflow organisation. In addition, the methodology focuses on the publication of the ontology according to Linked Data principles, together with all of its associated intermediate and final products (requirements, HTML documentation, etc.), so as to facilitate reuse.

The LOT methodology defines iterations over a basic workflow composed of the following activities: (1) ontological requirements specification; (2) ontology implementation; (3) ontology publication; and (4) ontology maintenance. In this section, we describe the process that we have followed, while Sect. 4 describes the final published outputs. We have used OnToology [1] connected to the GitHub repository of the ontology network (as discussed in Sect. 4) as the continuous integration environment to provide technological support during the ontology development process, making use of tools like Widoco [10] for ontology documentation and Oops! [15] for ontology evaluation.

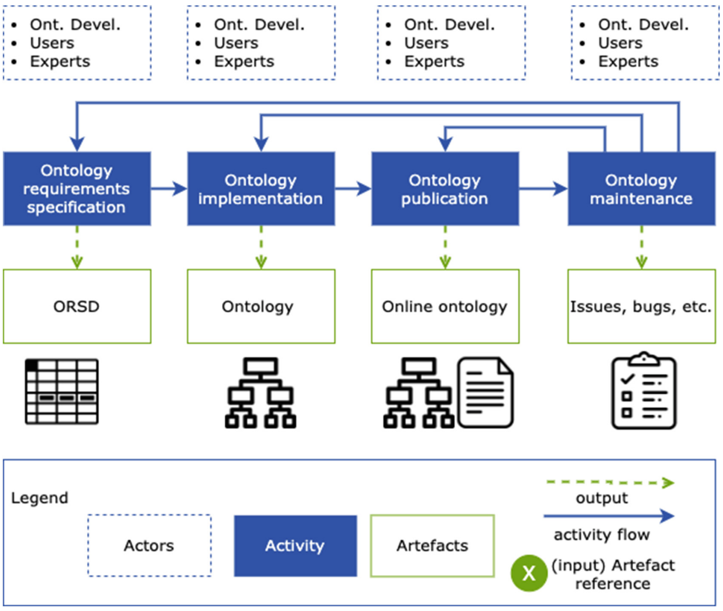


Fig. 1. LOT methodology basic workflow of processes [17]

3.1 Ontological Requirements Specification Process

The process of obtaining ontological requirements for the ontology network has been bootstrapped from three main types of sources:

- A set of competency questions related to the types of questions posed by our domain experts (site reliability engineers). This includes questions and requests such as "What is the current status of servers in a data center?", "Find the service topology for service X", etc. 31 questions and 88 facts have been collected, and are made available in the ontology network website and GitHub repository[1]. These have been further used for the validation and exploitation of the knowledge graph, as described in Sect. 5.2.
- The data model used by Huawei's Configuration Management and IT Service Management Database[2]. This data model was provided in SQL, and contains 82 tables that represent entities (e.g., Server, Software Module, Service, Data Center, etc.), 85 tables that represent relationships between entities (e.g., a service is running in one or serveral servers), and 59 tables representing views that connect different entities (e.g., lists of services and servers running in a data center).
- Several CSV files commonly used by domain experts from the AI-DevOps team at Huawei as additional intermediate views of specific entities from the

[1] https://github.com/oeg-upm/devops-infra.

[2] This data model is not available publicly for confidentiality reasons.

aforementioned database. Even though these CSV files are obtained using queries to the relational database, they do not follow exactly the same structure as the views in the data model.

All these resources come from several organisational departments inside a single company. During the ontology development process we have also checked further online resources describing database models used for this purpose [4,9] and contrasted with domain experts from other organisations so as to make sure that the design decisions were not biased towards the data models or practices used by a specific company.

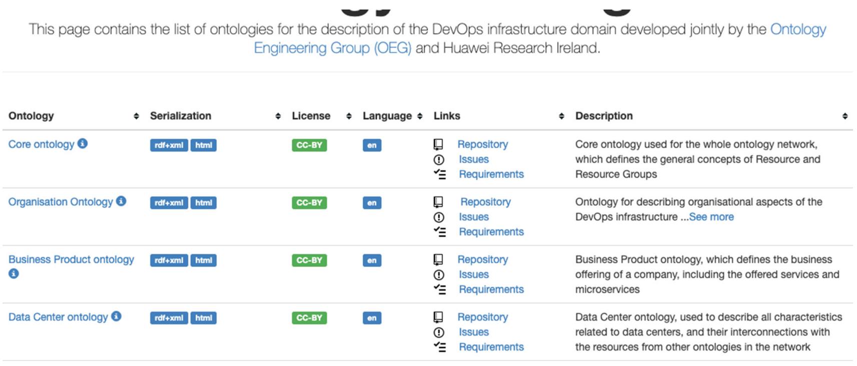
The process has run during 10 iterations with domain experts. An initial set of requirements were proposed from the initial set of queries identified by domain experts and from the entities and attributes available in the data model and CSV structures. We used the spreadsheet-based structure available at https://github.com/oeg-upm/ORSD-template so as to determine the classes to be created, their associated data and object properties, the enumerations to be transformed into SKOS thesauri, and the SPARQL queries to be generated once the ontology implementation would be ready. From an initial set of 165 competency questions (121 facts and 44 questions), 119 (88 and 31 respectively) were kept. The final set of competency questions is available at the GitHub repository.

3.2 Ontology Implementation Process

The ontology implementation process has followed a traditional approach. Our team of ontology engineers, who were already involved in the ontology requirement specification process, analysed the requirements and divided them into modules for the ontology network, considering the different areas of specialisation that domain experts would normally have in an organisation offering cloud services (as described in Sect. 4). This organisation into modules is also created with the objective of facilitating the evolution of the ontology network in the future. It has been validated (and refined) with some of the domain experts that were involved in the requirement specification process. We created and discussed conceptual models with them, following the graphical representations proposed in CHOWLK[3], transformed them into OWL, edited further with Protégé and maintained the different versions of the OWL ontologies in GitHub.

In order to facilitate the governance process afterwards, a set of ontology development guidelines and principles have been considered and deployed in the top-level ontology (the so-called core ontology), which describes the most generic items of Configuration Item, Resource and Resource Group. General properties to be used for the description of any Configuration Item have been determined, including the use of `rdfs:label` for names, `dct:identifier` for identifiers, and specialisations of `dct:created` and `dct:modified` for the creation and update times. Although these properties are not available as attributes in all the data models examined for all resources, this provides an initial level of homogeneisation for these simple properties. Our main goal on the decision for the ontology

[3] https://chowlk.linkeddata.es.



This page contains the list of ontologies for the description of the DevOps infrastructure domain developed jointly by the Ontology Engineering Group (OEG) and Huawei Research Ireland.

Ontology	Serialization	License	Language	Links	Description
Core ontology	rdf+xml html	CC-BY	en	Repository Issues Requirements	Core ontology used for the whole ontology network, which defines the general concepts of Resource and Resource Groups
Organisation Ontology	rdf+xml html	CC-BY	en	Repository Issues Requirements	Ontology for describing organisational aspects of the DevOps infrastructure ...See more
Business Product ontology	rdf+xml html	CC-BY	en	Repository Issues Requirements	Business Product ontology, which defines the business offering of a company, including the offered services and microservices
Data Center ontology	rdf+xml html	CC-BY	en	Repository Issues Requirements	Data Center ontology, used to describe all characteristics related to data centers, and their interconnections with the resources from other ontologies in the network

Fig. 2. Excerpt from the landing page of the ontology network

modules to structure the ontology network was to have a reduced number of classes and properties (in the range of tens as a maximum) so as to make them more manageable and easily extensible in the future. Figure 3 provides a general conceptual view of the main classes and properties of the ontology network (more details for each module are provided in the corresponding HTML documentation of each of the ontologies in the network).

3.3 Ontology Publication

In terms of ontology publication, we have followed usual practices proposed for Linked Data publication for ontologies. This has been facilitated by the usage of our suite of tools for ontology publication and evaluation, as aforementioned (OnToology, Widoco and Oops!).

More specifically, the ontology network landing page is published at http://w3id.org/devops-infra, as shown in Fig. 2. It follows the layout commonly used for other ontology networks (ontology name and URI, serialisation, license, language, links to the GitHub repository, open issues and requirements, and a brief description). Thanks to the content negotiation capabilities provided by W3id, the ontologies are dereferenced in HTML and OWL (both in RDF/XML and Turtle serialisations) in their corresponding URIs, so that they can be easily imported by ontology editors.

The ontology network is also archived in Zenodo [6], following usual practices in Open Science.

3.4 Ontology Maintenance

Our setup is now prepared for the ontology maintenance phase for all ontologies, with the possibility of submitting issues (bugs, requests for additions, etc.) for each of the ontologies in the network, so as to facilitate discussions that may arise during future standardisation processes (as discussed in Sect. 7) or ontology usage by other organisations. Indeed, the creation of declarative mappings for the

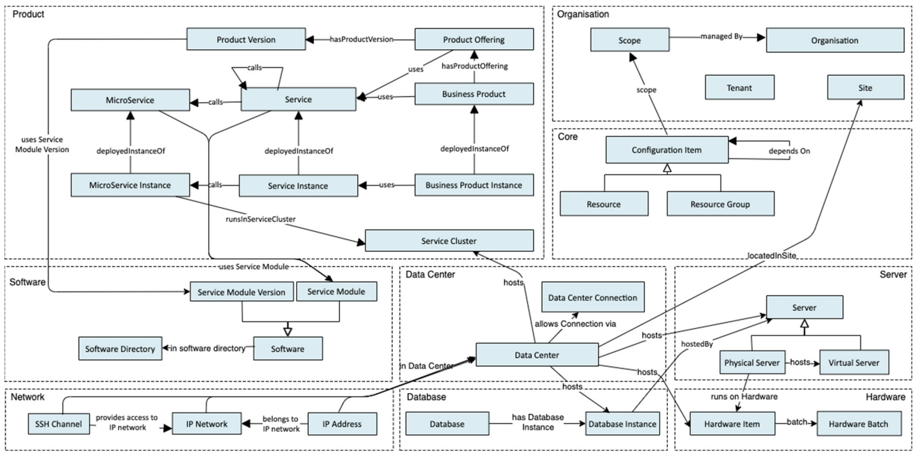


Fig. 3. High-level conceptual view of the main concepts and relationships in the ontology network

construction of Huawei's knowledge graph did already rise several issues (e.g., URI modifications and homogeneisation, common properties that were moved to the core ontology, etc.) that have been dealt with.

4 An Ontology Network for ICT Infrastructures

In this section we describe the current version of the implementation of this network of ontologies, and the main decisions taken during their development. Figure 3 shows an overview of the main concepts and relationships of our ontology network (except for the Certificate Ontology), as discussed in Sect. 3.2.

Core Ontology. The core ontology describes the most general concepts and properties that are used across the ontology network. Namely, it defines the concepts of `Resource` and `ResourceGroup`, as subclasses of the most general concept `ConfigurationItem` (the term that is commonly used in CMDBs), which describe any type of individual resource or group of resources that may need to be identified and managed in this context. Resources include databases, physical and virtual servers, data centers, different types of network equipment, services, microservices, products, etc. Besides the object properties that allow describing dependencies among resources and resource groups (`dependsOn`, which can be further specialised by the other ontologies in the network), relating a resource to a resource group (`belongsToResourceGroup`) and relating a resource group to another one (`parentResourceGroup`), this ontology proposes the use of other general data and object properties to ensure homogeneous descriptions of resources and resource groups across the ontology network, namely `created`, `modified`, `version` and `status`. When applicable, these properties are specialisations of the corresponding Dublin Core terms.

Organisation Ontology. This ontology describes the concepts and properties that allow describing general organisational entities in this context, such as `Department` (which is a subclass of `org:Organization`), `Scope`, `Stage`, `Site` or `Tenant`. In general, a Resource, as defined in the core ontology, may belong to a `Scope` and a `Scope` is `managedBy` a `Department`. Basic data properties are provided for all these classes.

Business Product Ontology. This ontology focuses on the description of the catalogue of products (and product offerings associated to them and normally available in some catalogue) offered by the organisation, together with the services and microservices that they make use of. The most relevant concepts from this ontology are `BusinessProduct`, `Service`, `MicroService` and `ServiceCluster`. All of these are related by specialisations of the `dependsOn` property, which is definied in the core ontology, so that it would be easy to trace the topology of microservices that provide support to a business product, for example (something relevant for the maintenance and troubleshooting operations). Besides these concepts that can be used to describe these general entities, their instances deployed in a specific infrastructure can be represented, with their corresponding properties.

Data Center Ontology. This is a central ontology in the ontology network, since the `DataCenter` concept is related to many other concepts defined in other ontologies, especially those in the server and network infrastructure ontologies, as well as in the hardware ontology. Indeed, a `DataCenter` is understood as an entity that `hosts` different types of resources, is `locatedIn` a `Site`, `allowsConnectionVia` a `DataCenterConnection` and offers different network-related elements (`offersIPAddress`, `offersIPNetwork`, `offersNetworkSegment`). As in the previous ontologies, multiple data properties are defined to describe further the core concepts of `DataCenter` and `DataCenterConnection`.

Server Infrastructure Ontology. This ontology defines all those concepts that are strongly related to the physical and virtual infrastructure of servers that are handled in this context. This includes the concept `Server` and its two main subclasses `PhysicalServer` and `VirtualServer`, which may be further extended to account for specific types of servers handled by an organisation. It also defines the `HostImage` and different types of `HostConfiguration` for virtual servers, and the concept of `ServerLoadBalancer`.

Network Infrastructure Ontology. This ontology describes the resources that are relevant for the configuration of the network infrastructure, including aspects like `IPAddress` and all of its subclasses for public and private IP addresses, `IPNetwork`, `NetworkSegment`, `DNSDomain` and its other related entities, `FirewallCluster`, `PublicNATEntry` and `SSHChannel`. All of these entities are interconnected with the corresponding object properties.

Software Ontology. This ontology describes in a general manner all those components that can be characterised as `Software`, including the software that can be used to deploy some service (`ServiceModule`). Any `Software` item may be available in a `SoftwareDirectory` and may be represented as a `File`. This ontology serves as a generic ontology for the description of software, and may be specialised by other ontologies if more details are needed.

Database Ontology. This ontology is also created as a general ontology to describe general concepts related to databases, such as the concepts of `Database`, `DatabaseInstance`, `DatabaseReplica` and `DatabaseScanReport`.

Hardware Ontology. This ontology is the one that allows relating the concepts managed by a CMDB with those normally handled by an IT Asset Management (ITAM) database. It allows describing pieces of hardware equipment such as `Disk`, `Frame`, `ServerHardware`, `NetworkCard`, etc. All of these items may have been purchased in a `HardwareBatch`, which acts as the main link to such an ITAM database.

Certificate Ontology. This ontology is focused on the description of aspects related to the management of digital certificates (including `DigitalCertificate` as well as `DigitalCertificateBundle`, `DigitalCertificateDeployment` and `DigitalCertificateSigningRequest`). All the data properties defined in this ontology are focused on describing the main characteristics of such certificates, as commonly understood in existing standards.

Finally, a set of thesauri have been generated in the form of SKOS concept schemes in those cases where specific codelists or simple taxonomies are required (e.g., status of any resource, types of tenants, types of hardware, etc.). In general, these have been derived initially from the set of enumerated columns appearing in the CMDB data model that we have used as one of the starting points, which have been conveniently cleaned and aligned with other existing enumerations from other providers, so as to provide a more comprehensive set of values. It is expected that these thesauri will evolve in the future when the ontology starts to be used by additional organisations.

5 Ontology Usage Scenarios

This section describes how we have used this ontology network as the basis for the creation of a knowledge graph related to the ICT infrastructure of a company like Huawei, based on its current CMDB and ITSM, already mentioned in Sect. 3 as one of the resources used for the requirements specification. This database has been developed in-house and is being used for storing and monitoring the current ICT infrastructure used by the company for service and product provisioning. It consists of several thousands of products and services, hundreds of data centers across the world, and millions of running services and microservices.

The ICT infrastructure of the company is maintained in a relational database that stores data from 82 different types of entities. The specific configuration of this database is generated from a JSON-based configuration file that provides support for simple taxonomies of resources and resource groups, and which specifies the main attributes (columns) that are considered for each of these resources and resource groups. Given the fact that this database can be considered as a legacy database, the schema has been evolving over time and there are sometimes different names for attributes that refer to the same types of properties (e.g., simple descriptions, labels for resources, etc.), unlike what is done in the ontology, where these should be homogeneous.

Furthermore, the design decisions behind the creation of the database have considered that it was useful, for extensibility purposes, to have different types of tables and views, as briefly discussed in Sect. 3.1: tables describing resources and resource groups, tables describing relationships among them (instead of making use of primary and foreign key relationships across tables, so as to facilitate the possibility of extensions and generating m:n relationships as the database evolves without impacting the design of the underlying tables, even though some performance metrics may be compromised) and tables that represent views. This JSON-based file is processed by an ad-hoc system and generates a SQL schema, which is the basis for the storage of all the data maintained by this CMDB and ISTM database.

5.1 Knowledge Graph Construction

Taking into account this rather unusual database structure (although common across Configuration Management Database models), the generation of the knowledge graph has still been based on state-of-the-art techniques for the generation of RDF-based knowledge graphs, more specifically the usage of RML mappings [8] that are initially expressed in YARRRML [11]. To this extent, and to facilitate the generation of these mappings, we have created a specific piece of software[4] that generates YARRRML mappings from the OWL ontology implementation, as a way to bootstrap this process for the knowledge engineer that is creating and maintaining mappings. The current set of mappings cover 41 concepts, 61 object properties and 91 data properties from the ontology network, which are associated to 41 entity tables and 38 relationship tables[5].

The KG creation process has been supported with tools from the Morph suite[6]. We opted for the materialisation of the RDF dataset and its storage in a Virtuoso triple store for the purpose of facilitating the integration of these technologies into the software stack of the teams involved at the company, since the integration of virtualised KG creation tools would have required further integration efforts with additional software development teams, which were out of the scope of our initial prototyping phase. However, virtualisation and query translation techniques are not discarded for the near future.

5.2 KG Exploitation Using a Question Answering System

Our competency questions have been transformed into SPARQL queries, showing some of the advantages of using a global view over the underlying data

[4] https://github.com/oeg-dataintegration/owl2rml.

[5] The mappings are maintained in a private repository, for confidentiality reasons.

[6] https://morph.oeg.fi.upm.es/.

(something that will be more clear when other types of databases from other organisations and CMDB and ISTM vendors are also transformed according to this ontology network). Furthermore, in order to show the range of possibilities that the KG provides and how it could be integrated in a DevOps chatbot that is being currently deployed at Huawei, we have developed a knowledge-graph based question answering (KGQA) system. A KGQA system allows exploring a KB using queries expressed in natural language, such as those used in some of the competency questions. Natural language questions are transformed into SPARQL queries that can be used to retrieve the desired data.

We have developed an unsupervised KGQA system, since we did not have annotated data for this domain, and we wanted to have a system that may be applied to other domains (or extensions of this domain) if needed. Five tasks have been traditionally identified in this process [7]: *question analysis*, *phrase mapping*, *disambiguation*, *query construction* and *distributed knowledge querying*. In the following, we briefly describe these steps and how we have approached them, illustrating them with a simple example. Assume the user inputs the question: "Where is hosted the instance of the cores_db database?".

In the *question analysis* task, techniques based on syntactic features are used to extract information about the question. We create n-grams and annotate the part-of-speech to retrieve the relevant phrases. 1-gram nouns and verbs are identified, since the rest of grammatical forms are not covered in the KG resources. The following phrases are considered from the input question: 'hosted', 'instance', 'cores_db database', 'hosted the instance', 'instance of the cores_db', 'instance of the cores_db database', 'hosted the instance of the cores_db' and 'hosted the instance of the cores_db database'.

In the *phrase mapping* task we search for resources that may correspond to phrases. Since there is potentially more than one term that can be related to KG resources (e.g. the words 'server', 'service', or 'deployment' can refer to `devopsserver:Server`), we avoid having to identify all those synonyms by projecting the resources into a vector space based on word embeddings. Entities, predicates and instances are directly retrieved from the SPARQL endpoint. One or more labels are automatically associated to each resource, since they will be used to obtain its vector representation. The labels are created from their URI, either by obtaining the value of some properties (e.g. `skos:prefLabel`, `rdfs:label` or `skos:altLabel`) or directly by parsing the URL using regular expressions (e.g. 'hardware network card' from http://w3id.org/devops-infra/hardware#NetworkCard). Our embedding space is built on top of the Fasttext model[7] using 300 dimensions to describe vectors for each resource, and they were stored in a Nearest Neighbour-based index to be able to perform searches taking into account the cosine distance.

This way we can identify the resources, and their distance, with respect to the phrases from the previous step. For the input example they are: `devopsdb:Database` (0,718), `devopsdb:hostedInFrame` (0,814), `devopsserver:hostedBy` (0,657), `resource:database/Cores_DB` (0,718), `devopsdb:DatabaseInstance`

[7] https://fasttext.cc.

(0,879), `devopsdb:DatabaseReplica` (0,900), `devopsdb:hasDatabaseInstance` (0,655), The purpose of the *disambiguation* task is to determine which of the previous resources are the right ones. Our approach is unsupervised, so we adapt a technique that measures the relevance of terms using density-based clustering techniques [3]. Resources and their distances are organized in a bidimensional space. Candidates that are close, i.e. resources whose distance between them is less than the standard deviation of the set, are grouped in the same cluster and their relevance depends on the absolute value of their distances. The closer the distance is to 0, the higher the relevance. In the above example, the most relevant resources are `devopsdb:hasDatabaseInstance`, `devopsserver:hostedBy` and `resource:database/Cores_DB`.

The *query construction* task creates the SPARQL query to retrieve data. This part also covers the construction of queries that require special operators such as comparatives and superlatives. Our approach is based on SQG [21], a modular SPARQL Query Generator that discovers a minimal subgraph based on uncertain lists of predicates and resources. The original source code[8] was extended to handle queries that restrict the type of resources (e.g. "'Core_DB' database" implies that the Core_DB resource type is 'Database', which allows to extend the original query to other resources of the same type); and to browse any SPARQL endpoint using configuration files without having to develop specific source code for that endpoint. The source code is publicly available[9]. In the example above, the generated SPARQL query is listed in 1.1.

Listing 1.1. SPARQL query to retrieve the location of a database instance

```
PREFIX devops: <https://w3id.org/devops-infra/>

SELECT DISTINCT ?server WHERE {
    <http://database/Cores_DB> devops:database#hasDatabaseInstance  ?db .
    ?db devops:server#hostedBy ?server
}
```

The final result of our query in a dummy database that we have created to demonstrate our work, for confidentiality reasons, is `resource:server/5`.

6 Related Work

As far as we were able to determine after the initial literature search at the beginning of this ontology development process, as well as during the identification of ontological resources to be reused, this is the first comprehensive and fully documented effort for the generation of an ontology network in this area, which is born with the objective of serving further standardisation and community-driven initiatives around this domain.

That said, we can mention some previous approaches reported in the literature, where ontologies for some specific types of infrastructure are reported. However, in all of these cases the ontology implementations are not available

[8] https://github.com/AskNowQA/SQG.
[9] https://github.com/oeg-upm/nlp2sparql.

anymore, nor is there any comprehensive documentation associated to those reported ontologies.

One of the seminal papers on the topic of ontologies for cloud computing is [20], from 2008, where the authors propose an organisation of the domain of cloud computing in five layers: firmware/hardware (HaaS), software kernel, cloud software infrastructure - including computational resources (IaaS), storage (DaaS) and communications (CaaS) -, cloud software environment (PaaS) and cloud application (SaaS). Although the title of this paper may suggest that an ontology or a network of ontologies had been produced as a result of this paper, the reality is that this paper only discusses the main characteristics of these layers and identifies some examples of different types of systems available at that time that would fit into each of these layers. However, no implementation in a formal ontology language is provided nor discussed, even though ontology languages like OWL already existed. Something similar happens with the work presented a bit later in [12], which does not provide an implementation either, or in [14], which only provides some snapshots of the corresponding implementation in the Protégé editor.

The Cloud Computing Ontology (CoCoOn), described in [22], focused on the computational resources part (IaaS) of the cloud software infrastructure identified in the aforementioned paper, and used it for a recommender system. Even though this ontology was implemented in OWL, it is not accessible anymore at the corresponding URL at the Australian W3C chapter pages, and therefore its reuse has not been possible in our development. The recommender is not available either. However, it was useful for the differentiation of some of our ontologies in the network (such as the server, hardware and network ontologies), and for the identification of some of the data and object properties that have been included in our ontologies for the classes in these ontologies.

Finally, the latest work that we have been able to find in this context is the one presented in [13], which is indeed closer to the type of work that we have performed, since it focuses on the representation of some of the entities that are commonly found in the CMDB databases, in the context of DevOps processes. This work claims the usage of OWL and SWRL for the implementation of the ontology, but does not describe the generated ontology in sufficient detail nor does it provide any link to the corresponding implementation.

7 Conclusions and Future Work

As discussed in the previous Section, a clear problem that we identified in this domain when starting the ontology development process was the lack of implementations of ontologies or common data models in this area, which may benefit from a comprehensive and systematic representation and publication of ontologies according to best practices in ontology implementation and publication. Indeed, the state of the art analysis has clearly revealed that only some partial efforts had been done in the past, and those did not result into a sustained effort afterwards for its maintenance.

Therefore, our aim has been to provide such a systematic approach that could lead into further standardisation work by putting together more organisations that have already shown interest in having common models for the representation of many of the types of entities and data that they are handling in this context.

In terms of impact, therefore, we consider that this work and its results can fill an important gap that has not been addressed sufficiently in the state of art. This would be as well a resource of interest for the Semantic Web community, in general, demonstrating how ontologies and semantic technologies can be used in an area where data heterogeneity exists and that could hence benefit from this type of approach.

We have not demonstrated yet any reuse of our ontology network, given that it has been only created recently. We expect, though, that there may be an interest in the context of standardisation and technical committees at IEEE and OASIS, as discussed earlier in this paper. As a result, we have already started contacting those that may be interested in this work, so as to show the potential advantages that such standardisation may bring in. Besides, the way in which the ontology network has been structured, together with the rich set of documentation provided for it, should facilitate such reuse and extensibility in the future, even for situations that have not been originally foreseen (product and service descriptions, root cause analysis, etc.).

The development has followed state-of-the-art practices in ontology development that we are applying in all of the ontology development projects that our group is involved in. This includes the LOT methodology and many of the ontology development support tools that we have been working on in the past years, and whose focus is to go further than just the implementation in OWL.

In terms of the availability, we cannot claim that our ontology network is yet completely FAIR compliant, especially considering that there is a strong debate in the state of the art on what the FAIR principles mean for ontologies (e.g., [16]), but we have at least followed what the community considers to be a good approach towards FAIRness: the ontology network is available in a permanent URI, thanks to w3id, it has an open license, all the resources are completely available online and in GitHub and archived in Zenodo (with a corresponding DOI). Indeed, at the time of writing, the usage of the FOOPS validator[10] provides a FAIRness score of 0.74.

References

1. Alobaid, A., Garijo, D., Poveda-Villalón, M., Santana-Perez, I., Fernández-Izquierdo, A., Corcho, O.: Automating ontology engineering support activities with OnToology. J. Web Semant. **57**, 100472 (2019)
2. AXELOS. ITIL Foundation: ITIL, 4th edn (2019)
3. Badenes-Olmedo, C., Redondo-García, J.L., Corcho, O.: Efficient clustering from distributions over topics. In: Proceedings of the Knowledge Capture Conference, K-CAP 2017. Association for Computing Machinery, New York (2017)

[10] https://foops.linkeddata.es/FAIR_validator.html.

4. BMC. Planning to populate BMC CMDB, version 20.08 (2020). https://docs.bmc.com/docs/ac2008/planning-to-populate-bmc-cmdb-929634733.html
5. Boasman-Patel, A., et al.: AIOps: a practical framework for AI driven operations in the telecom industry (2020). https://www.tmforum.org/resources/whitepapers/ai-operations-a-practical-framework-for-ai-driven-operations-in-the-telecom-industry/
6. Corcho, O., et al.: DevOps infrastructure ontology network (2021). https://doi.org/10.5281/zenodo.4701264
7. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowl. Inf. Syst. **55**(3), 529–569 (2018)
8. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: LDOW (2014)
9. Distributed Management Task Force (DMTF) Inc., Configuration Management Database (CMDB) Federation Specification, version 1.0.1 (2010). https://www.dmtf.org/sites/default/files/standards/documents/dsp0252_1.0.1_0.pdf
10. Garijo, D.: WIDOCO: a wizard for documenting ontologies. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 94–102. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_9
11. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Declarative rules for linked data generation at your fingertips! In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 213–217. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_40
12. Hoefer, C.N., Karagiannis, G.: Taxonomy of cloud computing services. In: 2010 IEEE Globecom Workshops, pp. 1345–1350 (2010)
13. McCarthy, M.A., Herger, L.M., Khan, S.M., Belgodere, B.M.: Composable DevOps: automated ontology based DevOps maturity analysis. In: 2015 IEEE International Conference on Services Computing, pp. 600–607, June 2015
14. Moscato, F., Aversa, R., Di Martino, B., Fortiş, T., Munteanu, V.: An analysis of mosaic ontology for cloud resources annotation. In: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 973–980 (2011)
15. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (OntOlogy pitfall scanner!): an on-line tool for ontology evaluation. Int. J. Semant. Web Inf. Syst. (IJSWIS) **10**(2), 7–34 (2014)
16. Poveda-Villalón, M., Espinoza-Arias, P., Garijo, D., Corcho, O.: Coming to terms with FAIR ontologies. In: Keet, C.M., Dumontier, M. (eds.) EKAW 2020. LNCS (LNAI), vol. 12387, pp. 255–270. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61244-3_18
17. Poveda-Villalón, M., Fernández-Izquierdo, A., Garcia-Castro, R.: Linked open terms (LOT) methodology (2019). https://doi.org/10.5281/zenodo.2539305
18. Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The NeOn methodology for ontology engineering. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 9–34. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24794-1_2
19. The Open Group. The TOGAF standard, version 9.2 (2020). https://publications.opengroup.org/standards/togaf
20. Youseff, L., Butrico, M., Da Silva, D.: Toward a unified ontology of cloud computing. In: 2008 Grid Computing Environments Workshop, pp. 1–10 (2008)

21. Zafar, H., Napolitano, G., Lehmann, J.: Formal query generation for question answering over knowledge bases. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 714–728. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_46
22. Zhang, M., Ranjan, R., Haller, A., Georgakopoulos, D., Menzel, M., Nepal, S.: An ontology-based system for cloud infrastructure services' discovery. In: 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp. 524–530, October 2012