data

MDPI

Article

Using Semantic Web Technologies to Query and Manage Information within Federated Cyber-Infrastructures

Alexander Willner [1,2,*,†,‡], Mary Giatili [3,†], Paola Grosso [4,†], Chrysa Papagianni [3,5,†], Mohamed Morsey [4,†] and Ilya Baldin [6,†]

1 Fraunhofer FOKUS, 10589 Berlin, Germany
2 Next Generation Networks Department, Technische Universität Berlin, 10587 Berlin, Germany
3 School of Electrical and Computer Engineering, National Technical University of Athens, Zografou, Athens 15780, Greece; mgiatili@netmode.ntua.gr (M.G.); chrisap@noc.ntua.gr or chrisap@isr.umd.edu (C.P.)
4 Informatics Institute, University of Amsterdam, 1098 XH Amsterdam, The Netherlands; p.grosso@uva.nl (P.G.); mohamed.mabrouk@live.com (M.M.)
5 Institute for Systems Research, University of Maryland, College Park, MD 20742, USA
6 Renaissance Computing Institute (RENCI), University of North Carolina at Chapel Hill, Chapel Hill, NC 27517, USA; ibaldin@renci.org
* Correspondence: alexander.willner@fokus.fraunhofer.de or alexander.willner@tu-berlin.de; Tel.: +49-30-3463-7116
† These authors contributed equally to this work.
‡ Current address: Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany.

Received: 29 March 2017; Accepted: 10 June 2017; Published: 23 June 2017

Abstract: A standardized descriptive ontology supports efficient querying and manipulation of data from heterogeneous sources across boundaries of distributed infrastructures, particularly in federated environments. In this article, we present the Open-Multinet (OMN) set of ontologies, which were designed specifically for this purpose as well as to support management of life-cycles of infrastructure resources. We present their initial application in Future Internet testbeds, their use for representing and requesting available resources, and our experimental performance evaluation of the ontologies in terms of querying and translation times. Our results highlight the value and applicability of Semantic Web technologies in managing resources of federated cyber-infrastructures.

Keywords: Semantic Web; federated infrastructures; RDF; OWL; SPARQL

1. Introduction

Cloud computing supports many distributed applications that are vital to the economy and to the advancement of science. The rising popularity of cloud computing and the diversity of available resources create an urgent need to match those resources optimally to the requests of end-users.

The desired level of self-serve operation within the cloud obviates the need for intervention by IT departments, allowing end-users direct and independent access to the computational infrastructure. As a result, end-users can deploy the necessary infrastructure and software solutions rapidly. Toward this end, accurate modeling of the infrastructure must support abstract representation of various resources, simplify interactions with them, and expose the right levels of information. The next frontier in cloud computing lies in supporting widely distributed clouds and the various aspects of the architectures needed to manage resources across multiple administrative domains. These problems are also closely related to future Internet research in academia, as well as to emerging commercial technologies like the Internet of Things (IoT) [1].

Modeling cloud infrastructures in a manner that supports effective matching of users' requests with available resources is a challenging task. The issue becomes even more complex in the context of distributed cloud systems with multiple infrastructure owners. In the academic research the same problem is encountered when trying to describe computational resources, scientific instruments and testbeds, which belong to different institutions and must be used by inter-disciplinary and inter-institutional collaborative teams. In such environments each infrastructure owner may model resources using their particular information and data modeling approach to set up the system quickly and attract users. The end-result, however, is user lock-in and inability to easily leverage available resources if they belong to different owners. Thus, resource matching and recommendation *based on common models* becomes of great importance.

This paper describes Open-Multinet (OMN) [2], a set of ontologies that rely on Semantic Web (Semantic Web) [3] technologies. It was designed by an international team of academic researchers who are intimately familiar with the related problems. The OMN researchers are also involved in multiple efforts to design a federation of Future Internet and cloud testbeds spanning the US and the EU, to be used for at-scale experimentation with novel concepts in networking and distributed systems. While we briefly introduced the ontology set in [2] and presented a preliminary description of its application in the context of a federated cloud environments in [4], in this paper we complement our previous work by an extended description of the OMN ontology set and we further added new evaluation results of the overall OMN framework.

Motivation for our work comes largely from our experience with the growth of academic networking, including the proliferation of cloud testbeds. Their ad hoc attempts to federate with each other, i.e., to make their resources available to wider communities of users through common interfaces, suffer from a lack of common models to describe available resources. Testbed owners use such models chiefly to provide their users with information about available resources, e.g., compute nodes, storage, network switches, and wireless access points. Each user, in turn, employs similar models to request resources from the testbeds, describing in some detail the exact configuration of available resources needed from the testbed.

Most testbeds are small when they first launch. Their designers often spend little time thinking through the information model that they wish to use to present resource information to users. Testbeds frequently rely on home-brewed solutions utilizing syntactic schema specifications serialized using XML or JSON, sometimes referred to as *RSpecs*, although RSpec is also a name of a specific XML dialect used by a subset of testbeds in the US. Documents expressed in those languages are passed between the users and the testbed management software in order to describe the available resources and to request specific configurations of resources for the experiments. While the built-in mechanisms in those languages allow for straightforward verification of document syntax, few mechanisms are available for validation of semantic correctness. These solutions typically rely on structure-implied semantics to validate correctness by associating semantic meaning rigidly with the position of information elements within the document.

These approaches tend to work in early phases of the design. As the diversity of resources grows, however, and as the sophistication of users increases, the need arises for extension mechanisms. Demand emerges for more powerful resource descriptions. The extension mechanisms then inevitably relax the structure-implied semantics, thus making validation of documents progressively more difficult. We observed this development first-hand in the case of US Global Environment for Network Innovations (GENI) [5] and EU Future Internet Research and Experimentation (FIRE) [6] testbed-federation efforts. XML schema extensions were introduced to allow different federation members to describe the unique attributes of their cloud testbeds. The extensions, we found, made it possible to create syntactically valid but semantically invalid documents requesting resources from a testbed, e.g., by requesting that a particular operating-system image be attached to a network interface instead of to a compute node.

Informed by these experiences, we decided to adopt Semantic Web technologies, which provided us with a number of advantages:

- A common standardized model is used to describe cloud and testbed infrastructures. The extensibility of this model is built into it from the start in the form of additional ontologies that describe new types of resources. The machinery to deal with extensions is built into standard semantic web toolkits, leaving the designers free to think about the information model while affixing the data model.
- Different resources and descriptions easily can be related and connected semantically. Semantic web mechanisms intuitively represent computer network graph structures. Network topologies are embedded into the RDF graph using graph homeomorphisms and then are annotated with additional information, addressing *structural and semantic constraints* in a single structure.
- Model errors can be detected early, before testbed resources are provisioned, by using many standard inference tools.
- Rules can in particular be used to complement queries. Rules for harmonizing relationships should to be defined and applied on the federation level. This is where specialties and commonalities of the involved testbeds are known and this approach lifts the burden from users to formulate complex queries.
- The annotation process, i.e., the conversion from XML-based RSpecs to RDF-based graphs, is automatic and configurable to take testbed specific extensions and federation-wide agreements into account.
- Using standard Semantic Web tools, complex queries can be formulated to discover resources. A common way for testbeds to operate is by ingesting JSON/XML or other encoding of the user request or resource advertisement and then converting it into a non-portable native form on which queries and embeddings are performed. Semantic web tools allow us to store testbed-state information natively in RDF and to operate on that information using a multitude of native inference and query tools, thus simplifying and abstracting many parts of testbed operations.
- Once cloud resources are described semantically, they can be interlinked to other Linked Open Data (LOD) [7] cloud data sets. These linkages provide additional information about resource availability or constraints and help to link resources, e.g., to policies governing their allocation.
- Semantic resource descriptions support convergence from multiple syntactic-schema based representations of testbed resources to a single semantically enriched representation that combines information from multiple sources. Such sources include various RSpecs describing testbed resources, out-of-band knowledge that may be encoded in resource names or contained in human-readable online Web pages, an approach consistent with Ontology-based Data Access (OBDA). Encoding this information in a structured way into a single representation prepares it for direct analysis, without need of an intermediate representation. Answers are derived by matching resources required by the user to those available at one or more different testbeds, federating the testbeds automatically, with minimal human intervention.

We believe that our approach represents an interesting application of OBDA to a novel area of use that combines information search and retrieval with active infrastructure-resource management.

The OMN development effort consisted of several phases, starting with the upper ontology design, followed by the design of several critical subordinate ontologies for, e.g., resource monitoring. We relied heavily on previous work in this area, directly incorporating, for instance, the Infrastructure and Network Description Language (INDL) [8,9] ontology for describing computer networks and the Networking innovations Over Virtualized Infrastructures (NOVI) [10] ontology for federated infrastructures. We then started developing tools that utilize the ontology, including converters from the various testbed resource description formats to OMN, inference rules for knowledge extension to complement the conversion process, and rule sets for semantic validation of the documents. We also

developed standard queries that assist the testbed resource-matching algorithms in extracting needed information from the testbed resource descriptions.

The remainder of the paper is structured as follows. We give a brief overview of related work in the context of (federated) heterogeneous computing infrastructures in Section 2. In Section 3, we present the OMN ontology set. Section 4 shows how we extract information from RSpecs and annotate it using additional knowledge extraction from out-of-band information. Querying and validation using traditional semantic web stools are then performed by the tools built on this framework. Section 5 shows the performance and applicability of our tools. Finally, we close in Section 6 with conclusions, considerations, and a description of future work.

2. Related Work

Many application disciplines shifted the focus from tree-based data models (e.g., XML-based syntactic schemas) to semantic models. This change is reflected in development of ontologies to support, for example, the operation of Grids, Clouds, and now the Internet of Things. These efforts have informed our own OMN development. In the coming section, we provide an overview of these efforts.

2.1. Semantic Models For Grids, Clouds and IoT

In the context of Grid Computing, the Grid Laboratory for a Uniform Environment (GLUE) [11] schema was started 15 years ago to support interoperability between Grid projects by defining a schematic vocabulary with serializations to XML, LDAP, and SQL [12]. A lack of formalism and a consequent inability to perform logical operations on data motivated the transition to Semantic Open Grid Service Architecture (S-OGSA) [13].

Semantic Web service discovery [14,15] addresses the automated discovery of Web services that satisfy given requirements. The discovery process uses a matchmaking algorithm to find potential Web services that might solve the problem at hand. Such methods, however, are inadequate to handle the complex interconnected computing infrastructures addressed by our work. Research on matching concentrates mainly on Web services [16], specifically, on semantic similarities between input and output parameters of various services. Our resource-matching involves more than matching available resources to the requirements of the end-user. We also need to identify homeomorphic embeddings of requested topologies within available resource topologies. The combination of such semantic and structural constraints leads to a substantially greater challenge. Pedrinaci et al. introduced Linked USDL (Linked USDL) [17], a vocabulary that applies research conducted on Semantic Web services to USDL [18,19]. Linked USDL provides comprehensive means to describe services in support of automated processing. It focuses only on services, and is unsuited to the description of cloud infrastructures. Ontologies such as the Semantic Markup for Web Services (OWL-S) [20] or Good Relations (GR) [21], however, are of interest to our work, and are referenced in part in our ontology.

In the domain of Cloud Computing, researchers are working to ensure interoperability on a semantic level. Since 2008, work has progressed in the development of ontologies and of tools for semantic cloud computing [22–24]. Haase et al. [25], for example, introduced an approach to administration of enterprise cloud environments, using semantic Web technologies. They proposed a Semantic Web-based product called eCloudManager, which incorporates an ontology to model its cloud data. However, the system and its ontology only focus on the management aspect of cloud systems, and the data are not open for usage. In another example, Haak et al. [26] proposed an ontology-based optimization methodology that enables cloud providers to detect the best resource set to satisfy a user's request. Their framework handles only a single administrative domain, whereas we seek to cover a distributed set of provider domains.

A paradigm shift is in progress in favor of Intercloud Computing. For instance, 20 approaches to this new challenge are presented in [27]. Within this context, Manno et al. proposed the use of the semantic Federated Cloud Framework Architecture (FCFA) [28] to manage resource life cycles

based on formal models. In contrast, the Intercloud architecture developed within the Institute of Electrical and Electronics Engineers (IEEE) Standard for Intercloud Interoperability and Federation (P2302) [29,30] Working Group uses graphs to describe and to discover cloud resources based on the existing Open-Source API and Platform for Multiple Clouds (mOSAIC) [31] ontology. Both approaches are being considered as domain-specific extensions to our work. In addition, Santana-Pérez et al. [32] proposed a scheduling algorithm that was suitable for federated hybrid cloud systems. The algorithm applies semantic techniques to scheduling and to matching tasks with the most suitable resources. The information model is based on the Unified Cloud Interface (UCI) project ontologies, which cover a wide range of details but which cannot handle Intercloud systems. Le and Kanagasabai [33,34] also proposed ontology-based methodologies to discover and to broker cloud services. They use Semantic Web technologies for user requirements and for cloud provider advertisements, and then apply an algorithm to match each requirement list to advertised resource units. Multiple levels of matching are defined, ranging from an exact match to no match. These methodologies concentrate only on Infrastructure as a Service (IaaS) provisioning. Moreover, they typically neither export their data nor provide a SPARQL Protocol And RDF Query Language (SPARQL) [35] endpoint, thereby hindering reuse of and access to data.

Interest has soared recently in the uses and challenges of the Internet of Things in which many heterogeneous devices from different administrative domains communicate with each other. Semantic models are needed for the IoT. The European Research Cluster on the Internet of Things (IERC) has established Activity Chain 4—Service Openness and Interoperability Issues/Semantic Interoperability (AC4) [36], and semantic models such as the Semantic Sensor Network (SSN) [37] ontology have been developed. Support for semantics in Machine-To-Machine Communication (M2M) [38] has received further attention [39]. The primary applicable standardization activity from the European Telecommunications Standards Institute (ETSI) M2M Working Group has identified the need for a semantic resource descriptions in [40]. The successor, oneM2M (http://onem2m.org) [41], already has established the OneM2M Working Group 5 Management, Abstraction and Semantics (MAS). With the recent establishment of the World Wide Web Consortium (W3C) Web of Things (WoT) [42] Working Group, semantic vocabularies will be developed to describe data and interaction models.

2.2. OMN Background

Development of our approach, the OMN ontology set, started within the Federation for FIRE (Fed4FIRE) [43] project. The aim was to extend and to harmonize related work for the FIRE initiative, which has been developed within the context of federated networks and e-infrastructures. Our main motivation was the state of the art in the Future Internet (FI) experimentation context, which considers only simple schema-based models. The Slice-based Federation Architecture (SFA) [44] is the de-facto standard Application Programming Interface (API) for testbed federation. It uses XML-based RSpecs to describe, to discover, and to manage resources in a simple declarative way. However, it cannot support complex queries combining structural and semantic constraints or knowledge analysis. OMN ontology design reuses concepts previously defined in RSpecs, but also leverages significant prior efforts to define ontologies targeting cyber-infrastructure management.

The Open Grid Forum (OGF) Network Mark-Up Language (NML) [45] is a well established ontology for modeling computer networks. It provides a framework for definition and description of topologies ranging from simple networks comprising a few nodes and connections to massive, complex networks with hundreds or thousands of nodes and links. The model underwent a thorough review and definition process, finally becoming an OGF standard. While NML lacks concepts and properties required to describe federated infrastructures, OMN adopts NML in order to model the networking aspects of the infrastructure.

In comparison with NML, the INDL addresses virtualization of resources and services. It supports description, discovery, modeling, composition, and monitoring of those resources and services. The INDL actually imports NML to describe attached computing infrastructures in a manner

that is independent of technology and vendor. It offers the capacity to extend coverage to emerging network architectures. The INDL, however, does not support infrastructure federation, in which several different testbeds are interconnected experimentally.

Semantic models developed within the European NOVI and GEYSERS [46] projects have been used to describe federated infrastructures, user requests, policies, and monitoring information. They also support virtualization concepts for computing and networking devices. They have been adopted by OMN where their incorporation is appropriate. In the first project, the proposed Information Modeling Framework (IMF) [47] represents resources from the same or from different infrastructure providers.

In parallel to this, within the GENI initiative, the Network Description Language based on the Web Ontology Language (NDL-OWL) [48–51] model specifies capabilities to control and to manage complex networked testbed infrastructures. Lessons learned from live deployments of NDL-OWL in GENI proved informative in OMN modeling discussions.

Efforts related to describing APIs via OWL-S or DAML-S are not applicable directly to our problem, since they focus on the description of Web-service APIs. The testbeds in our research converged on a set of simple API calls like *createSlice* (requesting that a desired network topology be created) or *listResources* (requesting information about available infrastructure resources). The complexity lies in the parameters passed in those calls, not in the diversity of types of parameters serving as inputs or outputs to the APIs. Those parameters often are represented by XML documents describing requested or available testbed resource topologies. Our goal is to replace those syntactic-schema-based representations with semantic-web based views. We want them to include enough information to support native querying based on both structural and semantic constraints, either by the users or by testbed management algorithms.

3. Open-Multinet Ontology Set

Following Noy and McGuiness [52], the first step for defining a formal information model is to determine the specific domain and scope of the proposed ontology. As stated in Sections 1 and 2, the initial objective was to support resource management in federated infrastructures for experimentation. The related phases to this management effort are depicted in Figure 1. Each step embodies a wide range of requirements and challenges. We particularly highlight the first phase in this paper; however, our approach was to provide a hierarchical set of ontologies to cover the whole resource life-cycle.

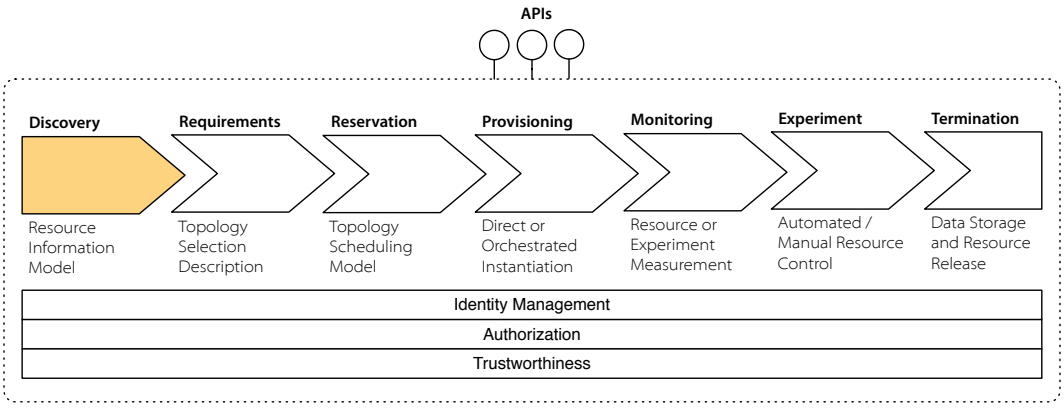

Figure 1. The experiment life-cycle phases and protocols.

3.1. Design

After identifying the scope of the reusable work (cf. Section 2), we have defined the significant concepts and properties. Consequently, our ontology bundle consists of nine ontologies, specifically

the *omn* upper ontology and eight descendant ontologies (cf. Figure 2): *omn-federation*; *omn-lifecycle*; *omn-resource*; *omn-component*; *omn-service*; *omn-monitoring*; *omn-policy*; and domain-specific extensions called *omn-domain-xxx*.

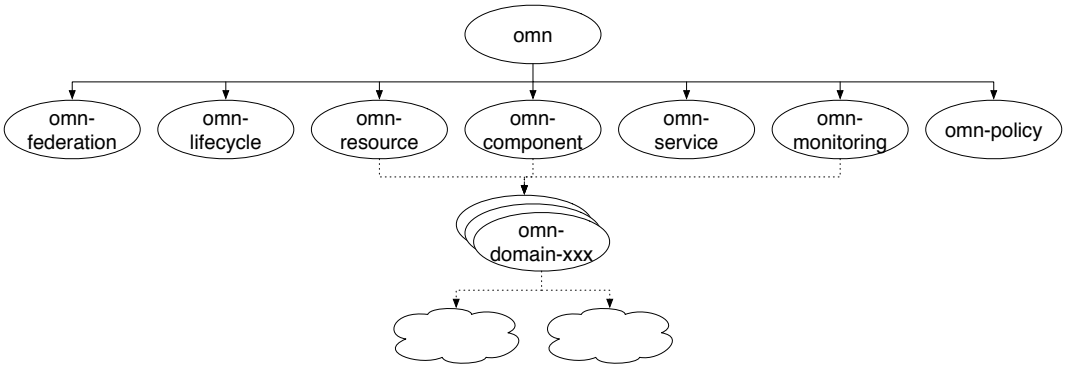

Figure 2. Open-Multinet ontology hierarchy [4].

These ontologies can be used to describe formally a federation of e-infrastructures, including types and attributes of resources as well as services available within the federation. Ontologies also describe the life-cycle phases of usage. The various ontologies extend the upper OMN ontology (solid lines), which contains common concepts used within the other models. To describe concrete resources within a particular infrastructure, domain-specific ontologies might need to be defined that use and extend selected subsets of the OMN ontologies (dotted lines, see Section 3.1.9).

3.1.1. OMN Upper Ontology

The *omn* upper ontology defines the abstract terms required to describe federated infrastructures in general.

It includes a set of classes representing concepts providing general terms to model federated infrastructures, along with their respective components and services. These concepts are as follows:

- *Resource*: a stand-alone component of the infrastructure that can be provisioned, i.e., granted to a user such as a network node.
- *Service*: is a manageable entity that can be controlled and/or used via either APIs or capabilities that it supports, such as a SSH login.
- *Component*: constitutes a part of a *Resource* or a *Service*, such as a port of a network node.
- *Attribute*: helps to describe the characteristics and properties of a specific *Resource*, *Group*, *Resource*, or *Component*, such as Quality of Service (QoS).
- *Group*: is a collection of resources and services, for instance, a testbed or a requested network topology logically grouped together to perform a particular function.
- *Dependency*: describes a unidirectional relationship between two elements such as *Resource*, *Service*, *Component*, or *Group*. It may define, for example, an order in which particular resources need to be instantiated: first, a network link, and then, the compute nodes attached to it. This class opens up the possibility of adding more properties to a dependency via annotation.
- *Layer*: describes a place within a hierarchy to which a specific *Group*, *Resource*, *Service*, or *Component* can adapt. Infrastructure resources naturally fall into layers, with resources at higher layers requiring presence of resources at lower layers in order to function.
- *Environment*: the conditions under which a *Resource*, *Group*, or *Service* is operating, as in, e.g., concurrent virtual machines.
- *Reservation*: a specification of a guarantee for a certain duration. Hence, it is a subclass of the ”Interval” class of the W3C Time ontology [53].

The OMN upper ontology has 23 properties, of which the following are the most significant:

- *hasAttribute*: the Attribute associated with a *Component*, *Resource*, *Service*, or *Group*; e.g., CPU speed, or uptime.
- *hasComponent*: links a *Component* , *Resource*, or *Service* to its subcomponent.
- *hasGroup*: connects a *Group* to its subgroup; it is the inverse of *isGroupOf*.
- *hasReservation*: relates *Group*, *Resource* or *Service* to its *Reservation*.
- *hasResource*: declares that a specific *Group* has a *Resource*.
- *hasService*: declares that a *Group*, *Resource* or *Service* provides a *Service*.
- *withinEnvironment*: defines the "Environment" in which a *Group*, *Resource*, *Service*, or *Component* operates. An example of environment is the operating system under which a resource works.

To support rich querying and inferences, inverse counterparts have been declared for most properties. Figure 3 illustrates the key concepts and properties of the OMN ontology.

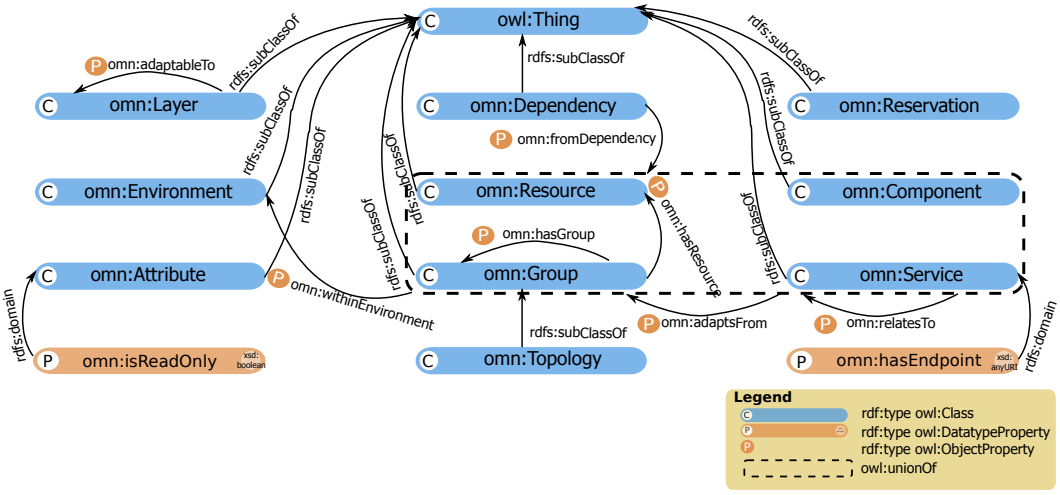

Figure 3. The key concepts and properties of the omn upper ontology.

3.1.2. OMN Federation

A crucial part of the developed ontology set is the formal description of the relationship between the involved e-infrastructures (see Figure 4). This allows to describe how resources relate to each other from the highest organizational level and depicts the starting point to discover capabilities and offered services. Federated providers maintain their autonomy in making resource-allocation decisions; however, they inter-operate with some federation-level functions, such as identity management or resource advertisement. To model these aspects, the *omn-federation* ontology introduces the concepts of a *Federation*, *FederationMember*, and *Infrastructure*, along with properties *hasFederationMember* and *isAdministeredBy*. The first two are subclasses of the schema:Organization class, which allows them to be described by properties of the Schema vocabulary. The latter concept relates infrastructures to a federation or to its members, and finally subclasses the *Group* concept which allows infrastructures to expose services with endpoints, such as an SFA Aggregate Manager (AM).

3.1.3. OMN Life Cycle

Another important ontology is the *omn-life cycle*, which addresses life-cycle management of a collection of resources and services (e.g., a requested network topology) that are grouped together to perform a particular function (e.g., to conduct an experiment or to deploy a service architecture). The life-cycle of the resources is described by a set of allocation and operational state changes such as *Allocated*, *Provisioned*, *Unallocated*, *Pending*, *Ready*, *Started*, and *Stopped*. The life-cycle of the collection of resources reflects the first four phases of Figure 1:

1. the infrastructure provider advertises an *Offering* describing the available resources;
2. the user forms a *Request* defining the required collection of resources to the infrastructure provider;
3. the *Confirmation* contains an agreement by the provider, termed bound (tied to a specific set of physical resources) or unbound, to provide the requested resources;
4. and, finally, a *Manifest* describes the provisioned resources and their properties.

Each of these stages is represented as a subclass of the *Group* concept.

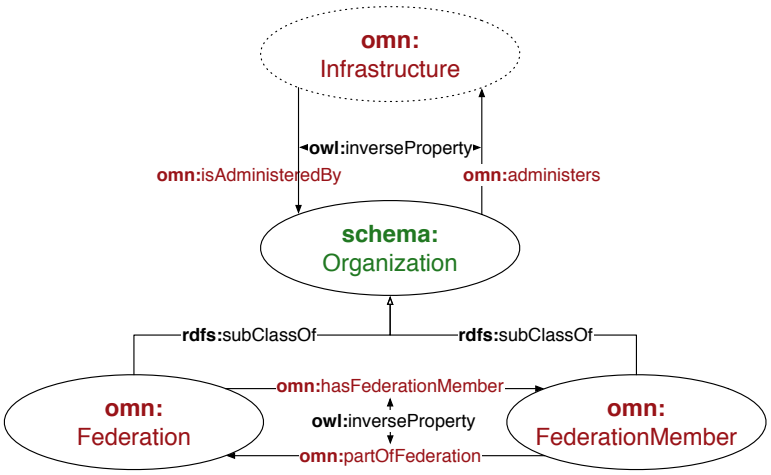

Figure 4. Open-Multinet (OMN) Federation Ontology.

3.1.4. OMN Monitoring

It describes the main concepts and relations to support monitoring services within federated infrastructures. It includes, therefore, multiple classes and properties that define measurement *Metrics*, their *Data*, *Unit*, *Tool*, and further *Generic-Concepts*. The monitoring ontology therefore comprises an upper-level ontology. It describes the common, basic concepts and properties, which then are reused and specialized in the subjacent ontologies.

3.1.5. OMN Resource

The OMN Resource ontology deals with the networking aspect of the infrastructure. It supports the creation of complex networks of interconnected resources. It include concepts and properties, e.g., *Node*, *Link*, and *IPAddress*, which are required for defining complex networks. It also supports defining single or bi-directional links, which can be utilized for defining the direction of packet flow across the link(s).

3.1.6. OMN Component

This ontology describes any entity that is part of a *Resource* or a *Service*; however, in itself, it is not a *Resource* or a *Service*. The OMN Component ontology describes concepts that are subclasses of the *Component* class defined in the OMN Upper ontology. It covers several classes to describe a set of basic entities in any Information and Communication Technology (ICT) infrastructure such as *CPU*, and *Memory*. Any class or instance of these can be the range of the property *hasComponent* that has a *Resource*, *Service*, or even another *Component* as a domain.

3.1.7. OMN Service

This ontology deals with ICT services. Any entity that delivers a value for its user is considered by the OMN Service ontology as a service. Examples can be services that offer APIs or login capabilities such as SSH. This ontology includes a set of classes to describe those services being

used in ICT infrastructures. The current version covers a set of services being used and implemented by OMN ontology within the context of the application area addressed in this paper, namely the FI experimentation.

3.1.8. OMN Policy

This ontology will cover policy-related concepts and relations. We consider the NOVI policy ontology as a starting point for its design, as it supports [10]:

- Authorization policies that specify authorization rights of users within the federation.
- Event-condition-action policies that enforce control and management actions upon certain events within the managed environment.
- Role-based-access control policies that assign users to roles, with different permissions/usage priorities on resources.
- Mission policies that define inter-platform obligations in a federation.

3.1.9. OMN Domain Specific

OMN provides a way to define domain-specific ontologies, which customize the definition of concepts and relations for a particular ICT application. This allows a set of concepts and relations that are specific to a particular domain to be grouped along with some concepts and relations from other OMN ontologies to form a domain-specific ontology. Examples of these ontologies include, for instance, OMN Wireless ontology and OMN Cloud ontology, used to define the behavior of wireless networks and of cloud infrastructures, respectively. Another example includes the specification of an operating system (OS) version within a disk image, using *omn-domain-pc:hasDiskimageVersion*.

3.2. Use of Existing Ontologies

As described in Section 2 the OMN ontology set is inspired by and based on a number of existing formal information models. As an indicator, in Listing 1 a list of referenced vocabularies are shown that are used within the upper OMN ontology. An OMN Service, for example, has relationships to *novi:Service*, *dctype:Service*, *gr:ProductOrService*, *service:Service*, *schema:Service*, *nml:Service*, and *owl-s:Service*.

Listing 1: Used ontologies within omn.ttl

```
@prefix : <http://open-multinet.info/ontology/omn#> .
@prefix cc: <http://creativecommons.org/ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix gr: <http://purl.org/goodrelations/v1#> .
@prefix nml: <http://schemas.ogf.org/nml/2013/05/base#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix indl: <http://www.science.uva.nl/research/sne/indl#> .
@prefix move: <http://www.ontologydesignpatterns.org/cp/owl/move.owl#> .
@prefix novi: <http://fp7-novi.eu/im.owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix vann: <http://purl.org/vocab/vann/> .
@prefix voaf: <http://purl.org/vocommons/voaf#> .
@prefix color: <http://geni-orca.renci.org/owl/app-color.owl#> .
@prefix owl-s: <http://www.daml.org/services/owl-s/1.0DL/Service.owl#> .
@prefix dctype: <http://purl.org/dc/dcmitype/> .
@prefix schema: <http://schema.org/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix service: <http://purl.org/ontology/service#> .
@prefix collections: <http://geni-orca.renci.org/owl/collections.owl#> .
```

3.3. Implementation

We selected OWL2 to encode the OMN ontology suite due to its expressiveness, wide acceptance, and available tools. To ensure quality, changes to the ontologies are automatically checked using Apache Jena Eyeball inspectors; other validators such as the OntOlogy Pitfall Scanner (OOPS) [54] are executed manually.

As part of the design process we are taking steps to ensure the broadest possible dissemination of the ontologies. As a result, we are using Dublin Core (DC), Vocabulary for Annotating Vocabulary Descriptions (VANN), and Vocabulary of a Friend (VOAF) vocabularies to describe the associated meta information. We are publishing the files by following best practices (http://www.w3.org/TR/swbp-vocab-pub/). The URL http://open-multinet.info/ontology/omn provides both a human-readable documentation and machine-readable serializations. We have registered the permanent identifier https://w3id.org/omn and published the root ontology to the Linked Open Vocabulary (LOV) repository (http://lov.okfn.org/dataset/lov/vocabs/omn). Additionally, we have registered the *omn* name space (http://prefix.cc/omn). The source code of, and an issue tracker for, the ontologies are publicly available (https://github.com/w3c/omn).

In order to make the work recognizable to the international community, we established the Open-Multinet Forum, which is named after the ontology, and created the W3C OMN Community Group (https://www.w3.org/community/omn).

4. Information Querying and Validation

4.1. DBcloud Application For Federated Experimental Infrastructures

Most of the requirements for the development of OMN are rooted in research issues within the life-cycle management of resources across federated experimental infrastructures. In such a distributed environment, resource discovery is highly constrained, as it is based on (multi-) attribute matching. It requires an increased level of coordination between users and infrastructure providers as well as among infrastructure providers in the federation. For this purpose, we essentially propose a federation-wide knowledge layer over the federated infrastructures to support semantic representation of such information and to facilitate semantic-based resource discovery.

A large amount of semistructured information is available describing the GENI and FIRE testbed federations, including details about the testbeds involved and about the heterogeneous resources offered, reservation information, and monitoring data. This information is encoded mainly as human-readable text on websites as well as in the forms of JSON and XML trees via secured API calls. To extract this information and to make it semantically accessible on the Web, we previously introduced the OMN extraction framework [4].

In essence, the OMN extraction framework (Figure 5) follows the design of the DBpedia extraction framework [55]. Information is retrieved from the infrastructures, calling periodically according to methods of the SFA AM API (http://groups.geni.net/geni/wiki/GAPI_AM_API_V3_DETAILS). The downloaded documents are translated into a semantically annotated Resource Description Framework (RDF) [56] graph using the *OMN translator* and the OMN ontology suite. To extend the knowledge encoded in this graph, the Apache Jena inference engine is used within this process by applying infrastructure-specific rules (Section 4.2).

Finally, the resulting knowledge graph is written in an in-memory triplet database (Sesame v.2.8.6) and in a Turtle (TTL) [57] serialized file (DBcloud Dump). A SPARQL endpoint on top of the triplet data store implements a federation-wide lookup service that enables resource discovery by end-users. The result is currently available at http://lod.fed4fire.eu using, among others, the Vocabulary of Interlinked Datasets. The knowledge base currently describes approximately 100 aggregates, 3000 nodes, 30,000 links, and about 25,000 interfaces. This consists of 4.1 million statements, with the potential to grow substantially as new testbeds join the federation.

The *OMN translator* is a Java-based extensible translation mechanism introduced in [2], allowing the automated transformation of semi-structured data into an OMN based knowledge graph. It translates statelessly between GENI, Resource Specifications (RSpecs), and OMN; applies inferencing rules for validation and knowledge injection; and has been extended to support Topology and Orchestration Specification for Cloud Applications (TOSCA) [58] and Yet Another Next Generation (YANG) [59] data models as well.

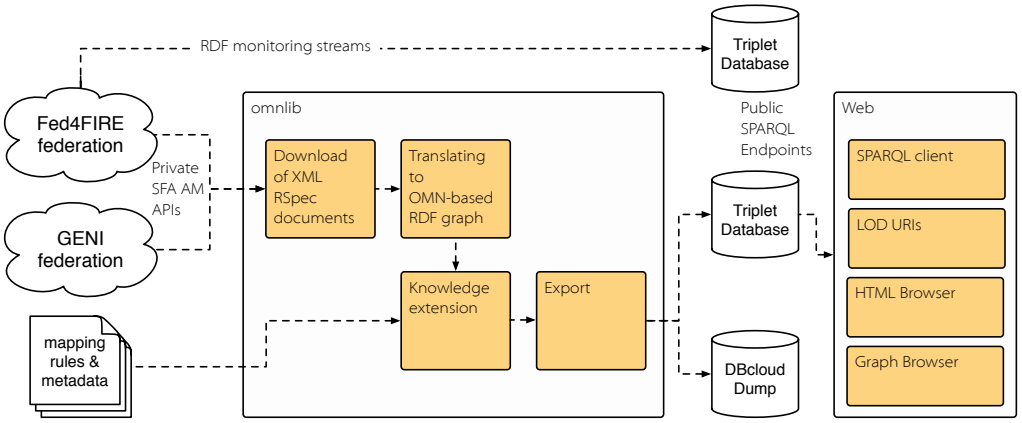

Figure 5. OMN extraction framework and lookup service (based on [4,55]).

The implementation of the translation tool follows a Test Driven Development (TDD) approach, is included in a Continuous Integration (CI) environment with test coverage analytics, and is offered as a Java-based open-source library ("omnlib") in a public maven repository. It uses the Java Architecture for XML Binding (JAXB) and Apache Jena to map between XML, RDF, and Java objects. It supports a number of APIs: (i) a native API to be included in other Java projects; (ii) a CLI to be used within other applications; and (iii) a REST-based API to run as a Web service.

The *OMN translator* parses the XML tree and converts the tags and attributes to their corresponding classes or properties. To give a better understanding of this translation process, we provide an illustrative example for the conversion of a GENI Advertisement RSpec used to publish available resources within a federation of experimental infrastructures.

The example in Listing 2 shows a single node of type *PC* that can provision the sliver type *PLAB-VSERVER* (virtual server for PlanetLab). Traditionally, hardware type and sliver type used to be simple strings, but unique Uniform Resource Identifiers (URIs) are used here to provide machine-interpretable information.

Listing 2: RSpec Advertisement (excerpt)

```
<rspec xmlns="http://www.geni.net/resources/rspec/3" type="advertisement">
<node component_manager_id="urn:publicid:IDN+ple+authority+cm"
 component_id="urn:publicid:IDN+ple:netmodeple+node+stella.planetlab.ntua.gr"
 exclusive="false" component_name="stella.planetlab.ntua.gr"> <- - - - - - - - - - - - - - - - - - - - Node Name
 <hardware_type name="http://open-multinet.info/ontology/resources/pc#PC"/> <- - - - - - - - - - Hardware Type
 <sliver_type name="http://open-multinet.info/ontology/resources/plab-vserver#PLAB-VSERVER"/> <- Provisions Sliver Type
 <available now="true"/>
</node>
</rspec>
```

Listing 3 shows the converted graph, serialized in Turtle. The overall approach is to define an *omn:Topology* (the subclass *omn-lifecycle:Offering* is used in this case) that contains pointers to the offered resources. Each resource is an individual of a specific type that *can implement* (i.e., can provision) one or more specific sliver types.

Listing 3: OMN Offering

```
<urn:uuid:7eb7b551−7566−4d3c−ac5f−f41a63236baa>  <- - - - - - - - - - - - - - - - - - - - Offering
        a <http://open−multinet.info/ontology/omn−lifecycle#Offering> ;
        <http://www.w3.org/2000/01/rdf−schema#label> "Offering" ;
        <http://open−multinet.info/ontology/omn#hasResource> <urn:publicid:IDN+ple:netmodeple+node+stella.planetlab.ntua.gr> .

<urn:publicid:IDN+ple:netmodeple+node+stella.planetlab.ntua.gr>
        a <http://open−multinet.info/ontology/omn−resource#Node> ,
          <http://open−multinet.info/ontology/resources/plab−vserver#PLAB−VSERVER> ,
           <http://open−multinet.info/ontology/resources/pc#PC> ;
        <http://www.w3.org/2000/01/rdf−schema#label>
                "stella.planetlab.ntua.gr"^^<http://www.w3.org/2001/XMLSchema#string> ;
        <http://open−multinet.info/ontology/omn#isResourceOf>
                <urn:uuid:7eb7b551−7566−4d3c−ac5f−f41a63236baa> ;
        <http://open−multinet.info/ontology/omn−lifecycle#canImplement>  <- - - - - - - - - - Implements Sliver Type
                <http://open−multinet.info/ontology/resources/plab−vserver#PLAB−VSERVER> ;
        <http://open−multinet.info/ontology/omn−lifecycle#hasComponentName>
                "stella.planetlab.ntua.gr" ;
        <http://open−multinet.info/ontology/omn−lifecycle#managedBy>
                <urn:publicid:IDN+ple+authority+cm> ;
        <http://open−multinet.info/ontology/omn−resource#hasHardwareType>  <- - - - - Node Hardware Type
                <http://open−multinet.info/ontology/resources/pc#PC> ;
        <http://open−multinet.info/ontology/omn−resource#hasSliverType>
                <http://open−multinet.info/ontology/resources/plab−vserver#PLAB−VSERVER> ;
        <http://open−multinet.info/ontology/omn−resource#isAvailable>
                true ;
        <http://open−multinet.info/ontology/omn−resource#isExclusive>
                false .

<http://open−multinet.info/ontology/resources/plab−vserver#PLAB−VSERVER>
        a <http://open−multinet.info/ontology/omn−resource#SliverType> ;
        <http://open−multinet.info/ontology/omn−lifecycle#hasSliverName>
                "http://open−multinet.info/ontology/resources/plab−vserver#PLAB−VSERVER" .

<http://open−multinet.info/ontology/resources/pc#PC>
        a <http://open−multinet.info/ontology/omn−resource#HardwareType> ;
        <http://www.w3.org/2000/01/rdf−schema#label>
                "http://open−multinet.info/ontology/resources/pc#PC" .
```

4.2. Knowledge Extension and Information Querying

Having described the framework for semantic-based resource discovery in the context of federated experimental infrastructures, we will now focus on the specifics of the discovery process. Given a user request (query) and the aforementioned knowledge base, the resource-discovery problem amounts to automatically finding the resources from the triplet data store that match the query requirements along with policies set by infrastructure providers, since a request can be expressed at different levels of abstraction (*Resource Matching*). The adoption of the OMN ontology suite provides the necessary flexibility of expression as well as tools for querying and inference that simplify the typical problems encountered in the process of resource matching. Rules can capture domain background knowledge or infer resource requirements from the request model; specifically regarding the latter these are added as additional information to the initial request model. In addition, they can be used to check the request model's validity [49]. These benefits are highlighted in the following text.

4.2.1. Knowledge Extension

Background knowledge captures additional knowledge about the domain. This information can be used in matching a request with available resources. Knowledge is expressed in terms of rules that use the vocabulary of the ontology to add axioms. The knowledge graph can be extended by applying such rules.

For example, infrastructure providers in the federation do not advertise explicitly the hardware configurations of their resources in the RSpec XML documents provided. Such data are not translated into RDF. Instead, the information is encoded in each resource's *hardware type*, arbitrarily set by the

infrastructure provider as highlighted in the advertisement excerpt provided in Listing 2, (i.e., *hardware type: PC*).

In Table 1, we provide sample hardware specifications for a subset of the federated experimental infrastructures as they are described by the corresponding infrastructure providers, namely, NETMODE (http://www.netmode.ntua.gr/testbed) and Virtual Wall 2 (http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html) testbeds.

Table 1. Resource Specifications.

HW Type	Description
alix3d2	500 MHz AMD Geode LX800, 256 MB DDR DRAM, 1 GB flash card storage
pcgen3	2x Hexacore Intel E5645 (2.4 GHz) CPU, 24 GB RAM, 250 GB harddisk

Figure 6 depicts a rudimentary *offering* (advertisement) excerpt from the NETMODE infrastructure provide. For the sake of readability, only a single advertised resource is depicted (*omf.netmode.node1*). Moreover, the diagram does not show all the details of the resource description, although it identifies the distinct OMN ontologies used for this purpose, in the upper part the figure. In the excerpt provided the offered resource *omf.netmode.node1* is *managedBy* the infrastructure provider *omf:netmode* (*AMService*) and is part of (*isResourceOf*) the offering (advertisement) identified by *urn:uuid:c9c34c9c-08d6-4dc6-91e2-2e5fac9dd418*. The resource is related via the object property *hasHardwareType* to the *HardwareType* individual with the label *alix3d2*. It is associated (*hasSliverType*) to the *SliverType* individual, with the label *miniPC*, attributed with specific *Disk Image* properties (e.g., OS Voyage). As noted in this example, infrastructures advertise node capacities by their hardware type name (*alix3d2* in this case).

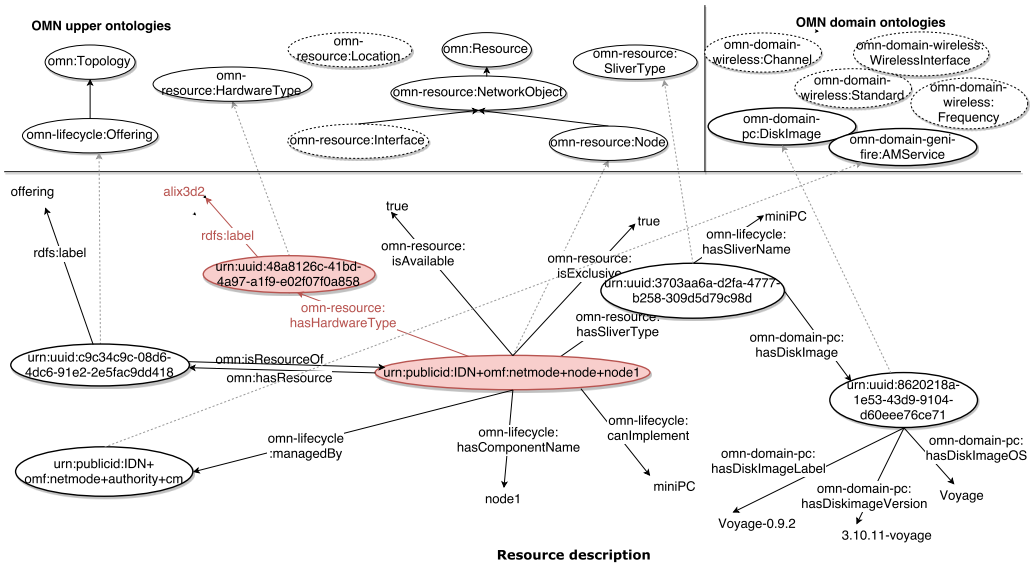

Figure 6. Partial NETMODE offering.

A simple example of background knowledge on the context of the "hardware type" is provided in Listing 4. The listing represents a subset of the rules used to expand the knowledge base with CPU-related information regarding *pcgen3* nodes listed in Table 1. Such information can be used in the resource matchmaking process. In the specific application, it is the responsibility of the federator, which maintains/provides the extraction framework, to apply such rules.

Listing 4: Infrastructure knowledge 1 (excerpt)

```
[rule1:
(?node omnres:hasHardwareType ?hwtype)
(?hwtype rdfs:label ?label)
regex (?label , "pcgen0?3.*")          <- For every compute node with a hardware
makeTemp(?cpuComp)                        type that has a label matching "pcgen0?3.*
->
(?cpuComp rdf:type owl:NamedIndividual)
(?cpuComp rdf:type omncomp:CPU)
(?cpuComp rdfs:label "Intel E5645 CPU")
(?cpuComp omn:hasModelType "Intel E5645")   <- Insert standard information about this node
(?cpuComp rdfs:label "Hexa Core Processor")    type: CPU Type, Core Count, CPU Frequency
(?cpuComp dbp:fastest "2.4"^^xsd:double)
(?cpuComp dbp:fastUnit <http://dbpedia.org/resource/GHZ>)
(?cpuComp omncomp:hasCores 6)
(?cpuComp dbp:arch <http://dbpedia.org/resource/X86-64>)
(?node omn:hasComponent ?cpuComp)   <- Link new information to the compute node
]
```

In our second example, shown in Listing 5, rules 1 to 3 mandate that each node identified by hardware type *alix3d2* have the hardware capacity described in Table 1 in terms of CPU, memory, and storage. Rules 4 to 6 link this information to the node.

Listing 5: Infrastructure knowledge 2 (excerpt)

```
[rule1: uriConcat(omncomp:,"alix3d2_mem", ?memComp) noValue(?memComp rdf:type owl:NamedIndividual)->
(?memComp rdf:type owl:NamedIndividual)
(?memComp rdf:type omncomp:MemoryComponent)(?memComp omnmonunit:hasValue "256000000"^^xsd:integer) ]
[rule2: uriConcat(omncomp:,"alix3d2_cpu", ?cpuComp) noValue(?cpuComp rdf:type owl:NamedIndividual)->
(?cpuComp rdf:type owl:NamedIndividual)
(?cpuComp rdf:type omncomp:CPU) (?cpuComp omnmonunit:hasValue "500000000"^^xsd:integer) ]
[rule3: uriConcat(omncomp:,"alix3d2_sto", ?stoComp) noValue(?stoComp rdf:type owl:NamedIndividual)->
(?stoComp rdf:type owl:NamedIndividual)
(?stoComp rdf:type omncomp:StorageComponent)(?stoComp omnmonunit:hasValue "1000000000"^^xsd:integer) ]
[rule4: (?node omnres:hasHardwareType ?hwtype) (?hwtype rdfs:label "alix3d2"^^xsd:string)
    uriConcat(omncomp:,"alix3d2_mem", ?memComp) -> (?node omncomp:hasComponent ?memComp) ]
[rule5: (?node omnres:hasHardwareType ?hwtype) (?hwtype rdfs:label "alix3d2"^^xsd:string)
  uriConcat(omncomp:,"alix3d2_cpu", ?cpuComp) -> (?node omncomp:hasComponent ?cpuComp) ]
[rule6: (?node omnres:hasHardwareType ?hwtype) (?hwtype rdfs:label "alix3d2"^^xsd:string)
  uriConcat(omncomp:,"alix3d2_sto", ?stoComp) -> (?node omncomp:hasComponent ?stoComp) ]
```

4.2.2. Information Querying

Having applied the rules in Listing 4, a user may make a request for cloud resources with, for example, specific CPU requirements. In the sample SPARQL query provided in Listing 6, the user submits a request for two virtual machines with a specific number of CPU cores and OS type, e.g., Fedora:6cores. The results are shown in Listing 7.

Listing 6: SPARQL query 1

```
SELECT ?resource1 ?resource2 WHERE {
?resource1 rdf:type omnres:Node .   <- Find me two hosts, resource1 and resource2
?resource2 rdf:type omnres:Node .
?resource1 omnres:hasSliverType/omndpc:hasDiskImage/omndpc:hasDiskimageOS ?os1.
?resource2 omnres:hasSliverType/omndpc:hasDiskImage/omndpc:hasDiskimageOS ?os2.
?resource1 omn:hasComponent ?cpuComp1.   <- Both with 6 cores
    ?cpuComp1 rdf:type omncomp:CPU.
  ?cpuComp1 omncomp:hasCores ?cpuvalue1.FILTER (?cpuvalue1 = "6"^^xsd:integer).
?resource2 omn:hasComponent ?cpuComp2.
    ?cpuComp2 rdf:type omncomp:CPU.
  ?cpuComp2 omncomp:hasCores ?cpuvalue2.FILTER (?cpuvalue2 = "6"^^xsd:integer).
FILTER (xsd:string(?os1) = "Fedora"^^xsd:string).   <- Both running Fedora
FILTER (xsd:string(?os2) = "Fedora"^^xsd:string).
FILTER (?resource1 = ?resource2)limit 1 !   <- Just one answer, please
```

Listing 7: Query results

```
RESULTS                                                                    1
urn:publicid:IDN+wall2.ilabt.iminds.be+node+n095−05a                       2
urn:publicid:IDN+wall2.ilabt.iminds.be+node+n096−02                        3
TIME EXECUTION: 0.016sec                                                   4
```

In a more complex example, a user may submit a request for *two nodes running a Linux distribution, with specific hardware requirements; e.g., 256MB of RAM and storage capacity greater than 500 MB*. The query is described in Listing 8. The resource-matching process is not straightforward, as it was in the previous case, even if we apply the rules in Listing 5. In most cases, Infrastructure Providers advertise the exact Linux Distribution (e.g., Voyage in Figure 6). Thus, the condition for *Linux OS variant* needs to be either incorporated into the request requirements or advertised explicitly by the testbeds. We follow the first approach in this case; additional rules are added to infer automatically the resource characteristics, e.g., acceptable Linux distribution, without explicit statements needed from the user, as proposed in [60]. The rule set is an appropriately defined set of axioms from which additional implicit information can be derived. A sample rule used is provided in Listing 9 stating Linux compatibility (Voyage is a Linux-variant OS).

Listing 8: Initial SPARQL query 2

```
SELECT ?node1 ?node2 WHERE {                                                                                                                    1
?node1 rdf:type omn_resource:Node.  <- - - - - - - - - - - - - - - - - - - - Find me two hosts, node1 and node2                                  2
?node2 rdf:type omn_resource:Node.                                                                                                              3
?node1 omn:hasComponent ?memComp1.                                                                                                              4
?node2 omn:hasComponent ?memComp2.                                                                                                              5
?memComp1 rdf:type omn_component:MemoryComponent.                                                                                               6
?memComp2 rdf:type omn_component:MemoryComponent.                                                                                               7
?memComp1 omn_monitoring_unit:hasValue ?mvalue.                                                                                                 8
FILTER (?mvalue >= "256000000"^^xsd:integer)  <- - - - - - - - - - Both with RAM greater than 256 MB                                            9
?memComp2 omn_monitoring_unit:hasValue ?mvalue.                                                                                                 10
FILTER (?mvalue >= "256000000"^^xsd:integer)                                                                                                    11
?node1 omn:hasComponent ?stoComp1.                                                                                                              12
?node2 omn:hasComponent ?stoComp2.                                                                                                              13
?stoComp1 rdf:type omn_component:StorageComponent.                                                                                              14
?stoComp2 rdf:type omn_component:StorageComponent.                                                                                              15
?stoComp1 omn_monitoring_unit:hasValue ?svalue1.                                                                                                16
FILTER (?svalue1 >= "500000000"^^xsd:integer)  <- - - - - - - - - - Both with disk storage greater than 500 MB                                  17
?stoComp2 omn_monitoring_unit:hasValue ?svalue2.                                                                                                18
FILTER (?svalue2 >= "500000000"^^xsd:integer)                                                                                                   19
?node1 omn_resource:hasSliverType/omn_domain_pc:hasDiskImage/omn_domain_pc:hasDiskimageOS ?os1.  < Both running Linux                           20
?node2 omn_resource:hasSliverType/omn_domain_pc:hasDiskImage/omn_domain_pc:hasDiskimageOS ?os2.                                                 21
FILTER (xsd:string(?os1) = "Linux"^^xsd:string)                                                                                                 22
FILTER (xsd:string(?os2) = "Linux"^^xsd:string)                                                                                                 23
FILTER (?node1 = ?node2)LIMIT 1                                                                                                                 24
```

Listing 9: IT background knowledge (excerpt)

```
[rule7:(?node rdf:type omn−resource:Node)                                  1
(?node omn−resource:hasSliverType ?stype)                                  2
(?stype omn−domain−pc:hasDiskImage ?dimage)                                3
(?dimage omn−domain−pc:hasDiskimageOS "Voyage"^^xsd:string) −>             4
(?dimage omn−domain−pc:hasDiskimageOS "Linux"^^xsd:string)]                5
```

Once the rules are applied, OR-AND clauses are built and added to the initial request [60]. Given the additional information injected into the graph, Listing 10 shows the new, expanded SPARQL query, with OR-AND clauses included in Lines 22–25. The results are restricted to one feasible matching solution, which is shown in Listing 11.

Listing 10: SPARQL query 2

```
SELECT ?node1 ?node2 WHERE {
?node1 rdf:type omn_resource:Node.  <- - - - - - - - - - - - - - - - - - - - Find me two hosts, node1 and node2
?node2 rdf:type omn_resource:Node.
?node1 omn:hasComponent ?memComp1.
?node2 omn:hasComponent ?memComp2.
?memComp1 rdf:type omn_component:MemoryComponent.
?memComp2 rdf:type omn_component:MemoryComponent.
?memComp1 omn_monitoring_unit:hasValue ?mvalue.
FILTER (?mvalue >= "256000000"^^xsd:integer)  <- - - - - - - - - - - Both with RAM greater than 256 MB
?memComp2 omn_monitoring_unit:hasValue ?mvalue.
FILTER (?mvalue >= "256000000"^^xsd:integer)
?node1 omn:hasComponent ?stoComp1.
?node2 omn:hasComponent ?stoComp2.
?stoComp1 rdf:type omn_component:StorageComponent.
?stoComp2 rdf:type omn_component:StorageComponent.
?stoComp1 omn_monitoring_unit:hasValue ?svalue1.
FILTER (?svalue1 >= "500000000"^^xsd:integer)  <- - - - - - - - - - - Both with disk storage greater than 500 MB
?stoComp2 omn_monitoring_unit:hasValue ?svalue2.
FILTER (?svalue2 >= "500000000"^^xsd:integer)
                                                                                    Running Linux
?node1 omn_resource:hasSliverType/omn_domain_pc:hasDiskImage/omn_domain_pc:hasDiskimageOS ?os1.  <-
?node2 omn_resource:hasSliverType/omn_domain_pc:hasDiskImage/omn_domain_pc:hasDiskimageOS ?os2.   Variant
FILTER (xsd:string(?os1) = "Voyage"^^xsd:string ||xsd:string(?os1) = "Fedora"^^xsd:string || xsd:string(?os1) = "Ubuntu"
^^xsd:string || xsd:string(?os1) = "Linux"^^xsd:string)
FILTER (xsd:string(?os2) = "Voyage"^^xsd:string ||xsd:string(?os2) = "Fedora"^^xsd:string || xsd:string(?os2) = "Ubuntu"
^^xsd:string || xsd:string(?os2) = "Linux"^^xsd:string)
FILTER (?node1 = ?node2)LIMIT 1
```

Listing 11: Query results

```
RESULTS
:node1=> <urn:publicid:IDN+omf:netmode+node+node18>,
:node2=> <urn:publicid:IDN+omf:netmode+node+node14>
TIME EXECUTION: 0.299sec
```

4.3. Validation

Documents created using OMN vocabularies can be validated semantically in part by using traditional OWL entailments, which verify that the domains and ranges of properties used in a particular model match those defined in the vocabulary. We found, however, that the expressivity of those mechanisms was not always sufficient to validate the user requests being sent to the testbed. Procedural verification is not portable. It is hard to ensure correctness and consistency across implementations. To supplant traditional OWL mechanisms, we developed Datalog rule-sets that trigger inference errors when processing a document that either lacks specific information or is semantically ambiguous. In this section, we explore several examples of such rules.

For instance, if a user is attempting to request a network connection that loops to the same node on which it started, a request may be represented by a valid OMN model; however, semantically, it doesn't make sense to the resource-matching algorithm that is attempting to reproduce the topology. To guard against cases like this, we validate the user's request using the following Datalog rule in Listing 12.

Listing 12: Validating self-looping links in requests

```
(?Z rb:violation error('Connection Validation', 'Connection cannot loop on itself', ?Y))
  <– (?X rdf:type pc:PC), (?X nml:hasOutboundPort ?P1), (?X nml:hasInboundPort ?P2),
  (?Y rdf:type nml:Link), (?P1 nml:isSink ?Y), (?P2 nml:isSource ?Y) ]
```

In some requests by end-users, every Virtual Machine (VM) node must specify an OS image to be booted. At the same time, a VM -server node does not need an image, since it operates using only a pre-determined image. The *pc* : *hasDiskImage* property is defined for all *PC* types, including VM Servers and VMs, so a cardinality restriction cannot be used in this case. This request validation rule is expressed as follows in Listing 13.

Listing 13: Validating presence of OS image in VM requests

```
(?Z rb:violation error("Validating that VM nodes have OS images", ?R)) <– (?R rdf:type pc:VM),
noValue(?R, pc:hasDiskImage, ?I)
```

It is important to emphasize that the set of the rules that we use continues to evolve with the schema and with the resource-matching algorithms used to allocate CI resources for the users. For example, as the algorithms become more sophisticated, they are able to function without some of the guards protecting them from poorly formed requests, reducing the need for some rules. Nonetheless, the designing of resource-matching and of embedding algorithms in testbeds is an active field of study. The availability of declarative rule-based semantic validation significantly simplifies the continuing evolution of these algorithms by clearly associating a particular algorithm with its own set of validation rules that prevent errant executions and simplify the algorithm code.

5. Performance Evaluation

By adopting formal information models and semantically annotated graphs, our approach allows operations to link, relate, enhance, query, and conduct logical manipulations of heterogeneous data, all of which would be impossible otherwise.

One of the most important measure for the applicability of our work is the amount of time required to translate and to query resources using our ontology. This time needs to range in a practicable span for the given context. Our initial work [4] looked at the sizes of the advertisements for testbeds in the FIRE and GENI projects and evaluated the performance of the translation to RDF of the respective XML files. The novel work we present in this paper show a more comprehensive comparison of the queries performance; namely, we look at the time needed to translate resource information to the one needed to list resources, as well as the performance of queries of different complexity.

We have analyzed the result of the *ListResources* method call of the 99 SFA AMs that are monitored (https://flsmonitor.fed4fire.eu/) within the Fed4FIRE project. This list contains 82 valid XML based GENI RSpec replies with 762.634 XML elements in total, of which 3.043 are *Nodes*, 31.155 are *Links*, and 25.493 *Interfaces*. Figure 7 shows the side of the RSpec advertisements in the testbeds we considered.

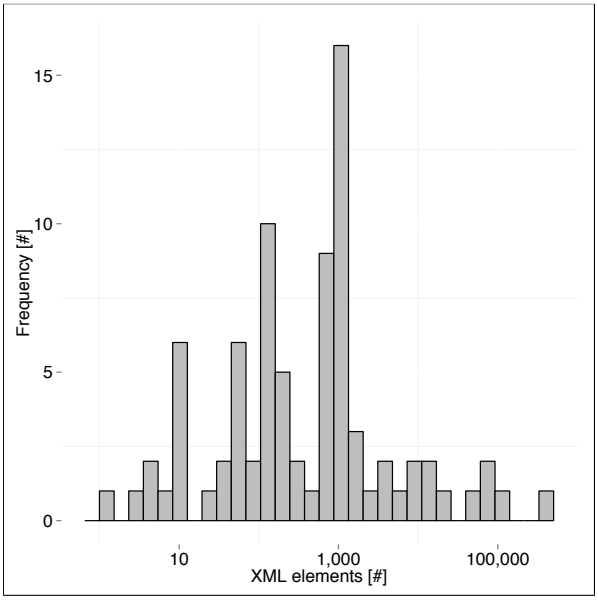

Figure 7. Size distribution of RSpec Advertisements (logarithmic).

To estimate the time needed to translate the advertisements, the actual RSpecs from these testbeds has been downloaded. The XML files were then translated to TTL serialized RDF graphs using the OMN translator. Of great importance to the potential scalability of our approach is the time taken for such translations, particularly with regard to the number of XML elements involved. 100 Advertisement RSpecs had been extracted, of which six contained errors, e.g., not adhering to the RSpec XML Schema Definition (XSD) file, and could not be translated without manual changes. Tests were run on a MacBookPro with OS X Yosemite, a 2.8 GHz Intel Core i7 processor, and 8 GB of RAM. Running a translation over all correct RSpecs produced median values of 24 milliseconds from XML to Java Architecture for XML Binding (JAXB) and 20 milliseconds from JAXB to RDF, yielding a total median translation time of 44 milliseconds from XML to RDF. As shown in Figure 8, translation times appear to be roughly linearly correlated with the number of XML elements translated, with a median of 180 elements and a maximum of 159,372 translated. This linear correlation indicates upwards scaling should be possible, although more data are required to confirm this point. At this stage, no major limiting factors have been identified, and, given appropriate processing power, translation should be possible in most foreseeable use cases.

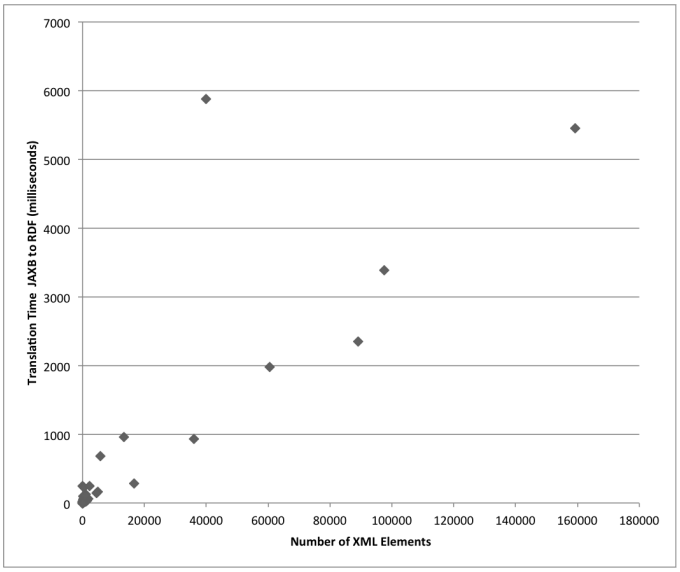

Figure 8. JAXB to RDF translation times versus number of XML elements [4].

To put the duration needed for the translation of an RSpec *Advertisement* into relation with the duration of the underlying function call needed in the FI experimentation context, we quantified the query and translation time for a single testbed. As indicated in Figure 7, about 95% of the testbeds expose fewer than 20.000 XML elements; therefore, we have used the CloudLab Wisconsin testbed (https://www.cloudlab.us), which exposes 19.371, for our measurements. The results in Figure 9 show that the average translation time of 583 ms $\pm$ 9 ms (95% CI) would add about 10% to the average response time of 5453 ms $\pm$ 131 ms (95% CI). This effect, however, could be mitigated by translating in advance or by distributing the work load. The delay of over 5 seconds for listing resources using a single API call, is influenced by mainly two factors. First, the available bandwidth to transmit the resulting XML document from the testbed to the caller. Second, the testbed internal communication architecture to gather the required information, as CloudLab is a distributed infrastructure itself that is composed by three different sites.

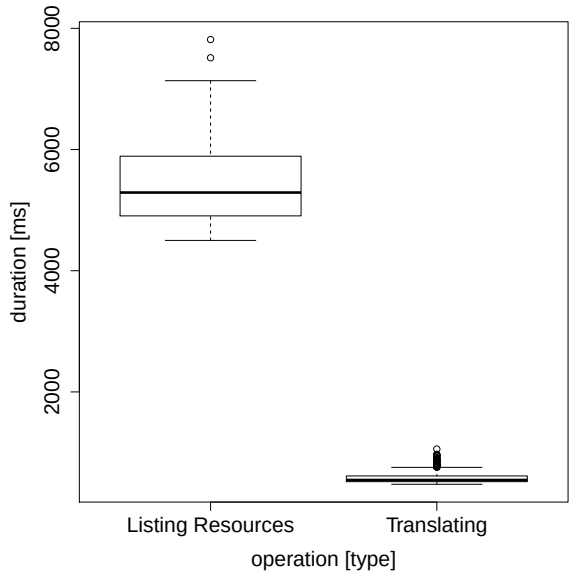

Figure 9. Listing/translating resource information.

Assuming that a testbed accepts the potentially enhanced response time, in favor of the added value of merging its information into a global linked data set, its resources can be found by applying the aforementioned resource-matching queries. The translation of all available tree data structures into an RDF-based graph, using our OMN vocabulary and rules, resulted in a set of 2.911.372 statements. It builds the basis for our conclusion that adding further rules, infrastructures, and other data sources will increase the potential for significant growth.

In Figure 10, the duration of Listing 10 against this graph is shown. To assess the performance impact of the complexity of the query, it has been compared with a simpler one, which is shown in Listing 14 together with its result in Listing 15. While finding the three largest aggregates took on average 129 ms $\pm$ 3 ms (95% CI), the matching query took on average 168 ms $\pm$ 1 ms (95% CI) and therefore took about 30% longer, yet much less time than a single *ListResources* call in a single testbed. Finally, we have summarized our findings in Table 2.

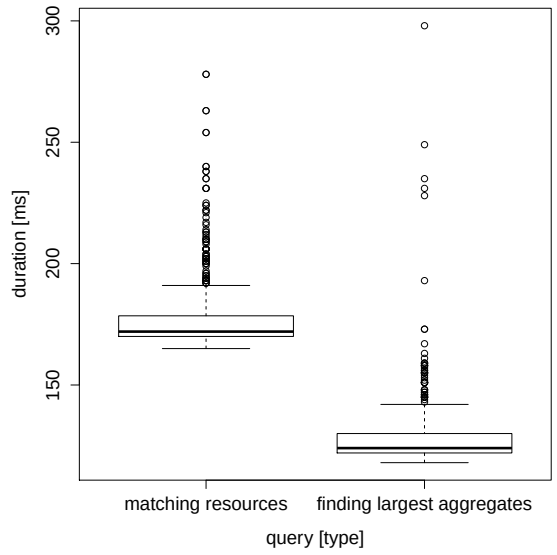

Figure 10. Performance comparison of queries.

Listing 14: Finding the largest aggregate via query

```
SELECT (COUNT(?am) as ?fre) ?am WHERE {
  ?node omn−lifecycle:managedBy ?am .
} GROUP BY (?am) ORDER BY DESC (?fre) LIMIT 3
```

Listing 15: Largest aggregates

```
?fre ?am
719 <urn:publicid:IDN+emulab.net+authority+cm>
326 <urn:publicid:IDN+utah.cloudlab.us+authority+cm>
255 <urn:publicid:IDN+ple+authority+cm>
```

Table 2. Results of the performance evaluation.

Median Duration [ms]	Phase
24	Translation from XML to JAXB (on average)
20	Translation from JAXB to RDF (on average)
583	Translation of 19.371 XML elements (CloudLab)
5453	Listing resources (CloudLab)
129	Querying three largest aggregates (Listing 10))
168	Matching resources (Listing 14)

6. Conclusion and Future Work

The OMN set of ontologies that we presented in this article has been developed to support resource management in federated and distributed computing infrastructures. OMN provides a federation-wide knowledge layer that eases the process of resource selection and matching.

In this article, we described the OMN framework, which allows the extraction of underlying information from tree-based data structures. It exposes this information in the form of OMN triples to interested parties via the Web. DBcloud is an application developed in support of the federation of experimental cyber-infrastructures which relies on OMN translators that automatically transform semi-structured data into OMN graphs. An important aspect that we have assessed is the performance of such translations, as this is crucial to OMN usability and adoption. We have shown that the translation and query times require additional time (on the order of 10% in our experiment), which, however, we expect to be acceptable to all resource providers given the added value of merging information.

We have also shown how users can query OMN information that represents the resources available in the underlying infrastructures and match them with their own computational requirements. In such case, we evaluated the time needed to find matching resources. We have shown that more complex queries complete within times that are acceptable to end-users.

In the long run, we expect that our contributions will outlive the specific use case of the cloud testbed resource management. We believe that it will be accepted by the broader community of academic and commercial cloud providers. It will help to create an ecosystem of flexible, extensible tools and mechanisms that will see the use of cloud platforms become even more pervasive. We expect it to open up the marketplace to competing cloud providers, large and small, catering to specific market niches. We are also promoting adoption of OMN in new domains such as the Internet of Things (IoT). As a specific Industrial Internet of Things (IIoT) [61] example, things, services and data can be connected between federated manufacturing facilities. As an analog to the federation of testbeds the involved facilities, the digital factories, their available APIs and services, have to be described formally to allow for matchmaking capabilities required for the envisioned autonomous production within the fourth industrial revolution. Our ontology set could act as a basis. Following discussions within the German initiative Plattform Industrie 4.0 (PI4.0), linking information based on the LOD paradigm and using interfaces such as the W3C WoT could build a technological base for implementing this vision. Another focus area for the OMN ontologies is the integration with ontologies defining

data-access policies among cooperating entities that make use of the cloud infrastructures. The support provided by OMN for the definition of complex usage of heterogeneous resources will be the backbone for novel kinds of open data services, both in industrial and commercial settings, as well as in the scientific community.

Acknowledgments: Research for this paper was financed in part by the Fed4FIRE (#318389) and Fed4FIREplus (#732638) projects, by the US NSF award CNS-1526964 and GENI funding, and by the Dutch national program COMMIT. We thank our project partners for their contributions and for their collaboration in this work.

Author Contributions: A.W. initiated development of the presented ontology set and of the DBcloud extraction framework; contributed to related work as well as to the sections about the ontology and validation; and evaluated the performance of the system. M.G. contributed to the development of the upper and domain-specific ontologies, of knowledge extension, and of information querying. P.G. contributed to the development of the ontologies. C.P. contributed to the design and development of the presented ontology set as well as to the ontology, application, knowledge extension, and information-querying sections. M.M. contributed to the development of the ontologies and wrote parts of the ontology section. I.B. contributed to the upper ontology design and provided feedback based on using NDL-OWL ontology in GENI, particularly w.r.t. RSpec translation and request validation.

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the collection, analyses, or interpretation of the data, in the writing of the manuscript, nor in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AC4	Activity Chain 4—Service Openness and Interoperability Issues/Semantic Interoperability
AM	Aggregate Manager
API	Application Programming Interface
DC	Dublin Core
ETSI	European Telecommunications Standards Institute
FCFA	Federated Cloud Framework Architecture
Fed4FIRE	Federation for FIRE
FI	Future Internet
FIRE	Future Internet Research and Experimentation
GENI	Global Environment for Network Innovations
GLUE	Grid Laboratory for a Uniform Environment
GR	Good Relations
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IERC	European Research Cluster on the Internet of Things
IIoT	Industrial Internet of Things
IMF	Information Modeling Framework
INDL	Infrastructure and Network Description Language
IoT	Internet of Things
JAXB	Java Architecture for XML Binding
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
LOD	Linked Open Data
M2M	Machine-To-Machine Communication
MAS	OneM2MWorking Group 5 Management, Abstraction and Semantics
mOSAIC	Open-Source API and Platform for Multiple Clouds
NDL-OWL	Network Description Language based on the Web Ontology Language
NML	Network Mark-Up Language
NOVI	Networking innovations Over Virtualized Infrastructures
OGF	Open Grid Forum
OMN	Open-Multinet
OOPS	OntOlogy Pitfall Scanner

OWL-S	Semantic Markup forWeb Services
P2302	Standard for Intercloud Interoperability and Federation
PI4.0	Plattform Industrie 4.0
QoS	Quality of Service
RDF	Resource Description Framework
RSpec	Resource Specification
S-OGSA	Semantic Open Grid Service Architecture
SFA	Slice-based Federation Architecture
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
SSH	Secure Shell
SSN	Semantic Sensor Network
TOSCA	Topology and Orchestration Specification for Cloud Applications
TTL	Turtle
UCI	Unified Cloud Interface
URL	Uniform Resource Locator
VANN	Vocabulary for Annotating Vocabulary Descriptions
VOAF	Vocabulary of a Friend
VoID	Vocabulary of Interlinked Datasets
W3C	WorldWide Web Consortium
WoT	Web of Things
XML	Extensible Markup Language
XSD	XML Schema Definition
YANG	Yet Another Next Generation

References

1. Ashton, K. That 'Internet of Things' Thing. *RFID J.* **2009**, *6*, 4986.
2. Willner, A.; Papagianni, C.; Giatili, M.; Grosso, P.; Morsey, M.; Al-Hazmi, Y.; Baldin, I. The open-multinet upper ontology towards the semantic-based management of federated infrastructures. In Proceedings of the 10th EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, Vancouver, BC, Canada, 24–25 June 2015; p. 10.
3. Berners-Lee, T.; Hendler, J.; Lassila, O. The Semantic Web. *Sci. Am.* **2001**, *284*, 34–43.
4. Morsey, M.; Willner, A.; Loughnane, R.; Giatili, M.; Papagianni, C.; Baldin, I.; Grosso, P.; Al-Hazmi, Y. DBcloud: Semantic Dataset for the cloud. In Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, USA, 10–15 April 2016; pp. 207–212.
5. Berman, M.; Chase, J.S.; Landweber, L.; Nakao, A.; Ott, M.; Raychaudhuri, D.; Ricci, R.; Seskar, I. GENI: A federated testbed for innovative network experiments. *Comput. Netw.* **2014**, *61*, 5–23.
6. Gavras, A.; Karila, A.; Fdida, S.; May, M.; Potts, M. Future internet research and experimentation. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 89–92.
7. Bauer, F.; Kaltenböck, M. *Linked Open Data: The Essentials*; Edition Mono/Monochrom: Vienna, Austria, 2011.
8. Ghijsen, M.; van der Ham, J.; Grosso, P.; de Laat, C. Towards an infrastructure description language for modeling computing infrastructures. In Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, Madrid, Spain, 10–13 July 2012; pp. 207–214.
9. Ghijsen, M.; van der Ham, J.; Grosso, P.; Dumitru, C.; Zhu, H.; Zhao, Z.; de Laat, C. A Semantic-Web Approach for Modeling Computing Infrastructures. *Comput. Electr. Eng.* **2013**, *39*, 2553–2565.
10. van der Ham, J.; Stéger, J.; Laki, S.; Kryftis, Y.; Maglaris, V.; de Laat, C. The NOVI information models. *Future Gener. Comput. Syst.* **2015**, *42*, 64–73.
11. Andreozzi, S.; Burke, S.; Field, L.; Fisher, S.; Konya, B.; Mambelli, M.; Schopf, J.M.; Viljoen, M.; Wilson, A. *GFD 147: Glue Schema Specification*; Open Grid Forum (OGF): Muncie, IN, USA, 2007.
12. Drozdowicz, M.; Ganzha, M.; Paprzycki, M.; Olejnik, R.; Lirkov, I.; Telegin, P.; Senobari, M. Ontologies, agents and the grid: An overview. In *Parallel, Distributed and Grid Computing for Engineering*; Saxe-Coburg Publications: Stirlingshire, UK, 2009; pp. 117–140.

13. Corcho, O.; Alper, P.; Kotsiopoulos, I.; Missier, P.; Bechhofer, S.; Goble, C. An overview of S-OGSA: A reference semantic grid architecture. *Web Semant. Sci. Serv. Agents World Wide Web* **2006**, *4*, 102–115.
14. Junghans, M.; Agarwal, S.; Studer, R. Towards practical semantic web service discovery. In Proceedings of the European Semantic Web Conference, Heraklion, Greece, 30 May–3 June 2010; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–29.
15. Stollberg, M.; Keller, U.; Lausen, H.; Heymans, S. In *Two-Phase Web Service Discovery Based on Rich Functional Descriptions, Proceedings of the 4th European Semantic Web Conference, Innsbruck, Austria, 3–7 June 2007*; Franconi, E., Kifer, M., May, W., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; pp. 99–113.
16. Paolucci, M.; Kawamura, T.; Payne, T.R.; Sycara, K. In *Semantic Matching of Web Services Capabilities, Proceedings of the International Semantic Web Conference, Sardinia, Italy, 9–12 June 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 333–347.
17. Pedrinaci, C.; Cardoso, J.; Leidig, T. Linked USDL: A vocabulary for web-scale service trading. In *The Semantic Web: Trends and Challenges*; Springer: Cham, Switzerland, 2014; pp. 68–82.
18. Cardoso, J.; Barros, A.P.; May, N.; Kylau, U. Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments. In Proceedings of the 2010 IEEE International Conference on Services Computing (SCC), Miami, FL, USA, 5–10 July 2010; pp. 602–609.
19. Oberle, D.; Barros, A.P.; Kylau, U.; Heinzl, S. A unified description language for human to automated services. *Inf. Syst.* **2013**, *38*, 155–181.
20. Martin, D.; Burstein, M.; Hobbs, J.; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Parsia, B.; Payne, T.; et al. OWL-S: Semantic Markup for Web Services. *Word Wide Web Consortium (W3C)*. Available online: https://www.w3.org/Submission/OWL-S/ (accessed on 20 June 2017).
21. Hepp, M. GoodRelations: An ontology for describing products and services offers on the web. In *Knowledge Engineering: Practice and Patterns*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 329–346.
22. Youseff, L.; Butrico, M.; Da Silva, D. Toward a unified ontology of cloud computing. In Proceedings of the IEEE Grid Computing Environments Workshop, Austin, TX, USA, 16 November 2008; pp. 1–10.
23. Han, T.; Sim, K.M. In *An ontology-enhanced cloud service discovery system. Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, China, 17–19 March 2010; Springer: Berlin/Heidelberg, Germany, 2010; Volume 1, pp. 17–19.
24. Ma, Y.B.; Jang, S.H.; Lee, J.S. Ontology-Based Resource Management for Cloud Computing. In *Intelligent Information and Database Systems*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 343–352.
25. Haase, P.; Mathäß, T.; Schmidt, M.; Eberhart, A.; Walther, U. Semantic technologies for enterprise cloud management. In Proceedings of the International Semantic Web Conference, Shanghai, China, 7–11 November 2010; pp. 98–113.
26. Haak, S.; Grimm, S. Towards custom cloud services—Using semantic technology to optimize resource configuration. In Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011), Paris, France, 20–25 October 2011; pp. 345–359.
27. Grozev, N.; Buyya, R. Inter-Cloud architectures and application brokering: Taxonomy and survey. *Softw. Pract. Exp.* **2014**, *44*, 369–390.
28. Manno, G.; Smari, W.W.; Spalazzi, L. FCFA: A semantic-based federated cloud framework architecture. In Proceedings of the International Conference on High Performance Computing & Simulation (HPCS), Madrid, Spain, 02–06 July 2012; pp. 42–52.
29. Bernstein, D.; Deepak, V.; Chang, R. *Draft Standard for Intercloud Interoperability and Federation (SIIF)*; Technical Report IEEE P2303; IEEE Computer Society: Washington, DC, USA, 2015.
30. Martino, B.D.; Cretella, G.; Esposito, A.; Willner, A.; Alloush, A.; Bernstein, D.; Vij, D.; Weinman, J. Towards an ontology-based intercloud resource catalogue—The IEEE P2302 intercloud approach for a semantic resource exchange. In Proceedings of the International Conference on Cloud Engineering, Tempe, AZ, USA, 9–13 March 2015; pp. 458–464.
31. Moscato, F.; Aversa, R.; Di Martino, B.; Fortis, T.; Munteanu, V. An analysis of mOSAIC ontology for cloud resources annotation. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Szczecin, Poland, 18–21 September 2011; pp. 973–980.

32. Santana-Pérez, I.; Perez-Hernández, M.S. A semantic scheduler architecture for federated hybrid clouds. In Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, 24–29 June 2012; pp. 384–391.
33. Dastjerdi, A.V.; Tabatabaei, S.G.H.; Buyya, R. An effective architecture for automated appliance management system applying ontology-based cloud discovery. In Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), Washington, DC, USA, 17–20 May 2010; pp. 104–112.
34. Ngan, L.D.; Kanagasabai, R. OWL-S based semantic cloud service broker. In Proceedings of the 2012 IEEE 19th International Conference on Web Services, Honolulu, HI, USA, 24–29 June 2012; pp. 560–567.
35. Aranda, C.B.; Corby, O.; Das, S.; Feigenbaum, L.; Gearon, P.; Glimm, B.; Harris, S.; Hawke, S.; Herman, I.; Humfrey, N.; et al. SPARQL 1.1 Overview. *Word Wide Web Consortium (W3C)*. Available online: https://www.w3.org/TR/2012/PR-sparql11-overview-20121108/ (accessed on 20 June 2017).
36. Serrano, M.; Barnaghi, P.; Cousin, P. *IoT Semantic Interoperability: Research Challenges, Best Practices, Solutions and Next Steps*; Technical Report for European Research Cluster on the Internet of Things (IERC): Brussels, Belgium, 2013.
37. Compton, M.; Barnaghi, P.; Bermudez, L.; García-Castro, R.; Corcho, O.; Cox, S.; Graybeal, J.; Hauswirth, M.; Henson, C.; Herzog, A.; et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semant. Sci. Serv. Agents World Wide Web* **2012**, *17*, 25–32.
38. Wu, G..; Talwar, S.; Johnsson, K.; Himayat, N.; Johnson, K.D. M2M: From mobile to embedded internet. *IEEE Commun. Mag.* **2011**, *49*, 36–43.
39. Čačković, V.; Popović, Ž. Abstraction and Semantics support in M2M communications. In Proceedings of the 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2013; pp. 404–408.
40. European Telecommunications Standards Institute. *Machine-to-Machine Communications (M2M); Functional Architecture*; Technical Report for ETSI: Valbonne, France, 2013.
41. Swetina, J.; Lu, G.; Jacobs, P.; Ennesser, F.; Song, J. Toward a standardized common M2M service layer platform: Introduction to oneM2M. *IEEE Wirel. Commun.* **2014**, *21*, 20–26.
42. Raggett, D. Web of Things Framework. *Word Wide Web Consortium (W3C)*. Available online: https://www.w3.org/2015/05/wot-framework.pdf (accessed on 20 June 2017).
43. Vandenberghe, W.; Vermeulen, B.; Demeester, P.; Willner, A.; Papavassiliou, S.; Gavras, A.; Quereilhac, A.; Al-Hazmi, Y.; Lobillo, F.; Velayos, C.; et al. Architecture for the heterogeneous federation of future internet experimentation facilities. In Proceedings of the Future Network and Mobile Summit (FNMS), Lisboa, Portugal, 3–5 July 2013; pp. 1–11.
44. Peterson, L.; Sevinc, S.; Lepreau, J.; Ricci, R. *Slice-Based Federation Architecture*, version 2; GENI: West Hollywood, CA, USA, 2010.
45. Van der Ham, J.; Dijkstra, F.; Lapacz, R.; Zurawski, J. *GFD 206: Network Markup Language Base Schema*; Technical Report for Open Grid Forum (OGF): Muncie, IN, USA, 2013.
46. Escalona, E.; Peng, S.; Nejabati, R.; Simeonidou, D.; Garcia-Espin, J.A.; Riera, J.F.; Figuerola, S.; de Laat, C. GEYSERS: A novel architecture for virtualization and co-provisioning of dynamic optical networks and IT services. In Proceedings of the Future Network and Mobile Summit, Warsaw, Poland, 15–17 June 2011; pp. 1–8.
47. Garcia-Espin, J.A.; Riera, J.F.; Figuerola, S.; Ghijsen, M.; Demchenko, Y.; Buysse, J.; de Leenheer, M.; Develder, C.; Anhalt, F.; Soudan, S. Logical infrastructure composition layer, the GEYSERS holistic approach for infrastructure virtualisation. In Proceedings of the Terena Networking Conference (TNC), Reykjavík, Iceland, 21–24 May 2012; pp. 1–16.
48. Baldine, I.; Xin, Y.; Mandal, A.; Renci, C.H.; Chase, U.C.J.; Marupadi, V.; Yumerefendi, A.; Irwin, D. Networked cloud orchestration: A GENI perspective. In Proceedings of the Globecom Workshops, Miami, FL, USA, 6–10 December, 2010; pp. 573–578.
49. Yufeng, X.; Baldine, I.; Chase, J.; Anyanwu, K. *TR-13-02: Using Semantic Web Description Techniques for Managing Resources in a Multi-Domain Infrastructure-as-a-Service Environment*; Technical Report for RENCI: Durham, NC, USA, 2013.

50. Xin, Y.; Hill, C.; Baldine, I.; Mandal, A.; Heermann, C.; Chase, J. *Semantic Plane: Life Cycle of Resource Representation and Reservations in a Network Operating System*; Technical Report for RENCI: Durham, NC, USA, 2013.
51. Xin, Y.; Baldin, I.; Chase, J.; Ogan, K.; Anyanwu, K. *Leveraging Semantic Web Technologies for Managing Resources in a Multi-Domain Infrastructure-as-a-Service Environment*; Technical Repor for RENCI: Durham, NC, USA, 2014.
52. Noy, N.F.; Mcguinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Technical Report for Stanford University: Stanford, CA, USA, 2001.
53. Hobbs, J.R.; Pan, F. Time Ontology in OWL. *Word Wide Web Consortium (W3C)*. Available online: https://www.w3.org/TR/owl-time/ (accessed on 20 June 2017).
54. Poveda-Villalón, M.; Suárez-Figueroa, M.C.; Gómez-Pérez, A. Validating ontologies with OOPS! In *Knowledge Engineering and Knowledge Management*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 267–281.
55. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; et al. DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web J. SWJ* **2014**, *5*, 1–29.
56. Cyganiak, R.; Wood, D.; Lanthaler, M. Resource Description Framework (RDF) 1.1 Concepts and Abstract Syntax. *Word Wide Web Consortium (W3C)*. Available online: http://travesia.mcu.es/portalnb/jspui/handle/10421/2427 (accessed on 20 June 2017).
57. Beckett, D.; Berners-Lee, T.; Prud'hommeaux., E. Turtle-Terse RDF Triple Language. *Word Wide Web Consortium (W3C)*. Available online: https://www.w3.org/TeamSubmission/turtle/ (accessed on 20 June 2017).
58. OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA)*, version 1; OASIS Standard; Advancing Open Standards for the Information Society (OASIS): Burlington, MA, USA, 2013.
59. Bjorklund, M. *RFC 6020: YANG—A Data Modeling Language for the Network Configuration Protocol (NETCONF)*; RFC 6020 (Proposed Standard); RFC Editor: Los Angeles, CA, USA, 2010.
60. Urbani, J.; van Harmelen, F.; Schlobach, S.; Bal, H. QueryPIE: Backward reasoning for OWL Horst over very large knowledge bases. In Proceedings of the 10th International Semantic Web Conference (ISWC'11), Berlin/Heidelberg, Germany, 23–27 October 2011; pp. 730–745.
61. Jeschke, S.; Brecher, C.; Song, H.B.; Rawat, D.B. *Industrial Internet of Things*; Springer: Cham, Switzerland, 2017.