Semantic Ontology-based NFV Service Modeling

Sang Il Kim
Electronics and Communications Engineering
KwangWoon University, Seoul, Korea
rlatkd234@kw.ac.kr

Hwa Sung Kim
Electronics and Communications Engineering
KwangWoon University, Seoul, Korea
hwkim@kw.ac.kr

Abstract- **Recently, research on methods for reducing CAPEX and OPEX by the use of automation technology in the NFV environment has been conducted actively. In particular, technologies for the standardization and automatic generation of description formats, which describe information on VNF and network service, are actively investigated. This paper conducted data modeling for VNFs management automation and automatic network service generation, using NSD(Network Service Description) data and ontology, and similar VNF reasoning by the use of ontology reasoning technologies was carried out to define and test relationships between ontology nodes**

Keywords— **NFV, SDN, Ontology, Metadata, Semantic**

I. INTRODUCTION

Recently, demand for the expansion of virtualized network infrastructure has been on the increase according as network usage increases, and investigation into virtualized network technologies in response to the demand is actively carried out. NFV, one of typical virtualized network technologies, virtualizes all types of resources necessary for networking, and makes software-based management and control possible, thereby enabling common carriers to reduce CAPEX and OPEX. Among the components of NFV, the network service consists of one or more VNFs (Virtual Network Functions), and all information on the network service is described in NSD (Network Service Description) and VNFD (Virtual Network Function Description).[1]

Information on VNFFG(VNF Forwarding Graph) defined by the network service is also included in NSD, describing the relationship between VNFs. Cloud computing can be more useful and applicable if it is possible to define automatically all the life cycle including the relationships of generation, management, and interoperability with respect to application services; and OASIS (Organization for the Advancement of Structured Information Standards), an organization for standardization, developed a standard language called TOSCA (Topology and Orchestration Specification for Cloud Application)[2-3] for the automation of NSD management.

TOSCA provides the definition of service components and the relationships to every component, using a service topology, and describes processes that create or modify services, thereby enabling the automated management of service creation and modification. While TOSCA makes it possible to define relationships and to automate the management of service creation and modification, however, it only defines the relationships between service nodes, but not relationships between parameters such as service category, necessary bandwidth, and service vendor.

That is, TOSCA automatically performs functions such as VNF forwarding graph creation based on relationships subjected to automatic service creation, modification and definition according to process and information described in a relevant descriptor by the use of *TOSCA-based NSD*, it does not support functions provided on the basis of semantic information between parameters, such as the recommendation of similar VNF, the automatic generation of VNFFG between similar services, and search for the existing similar network services. Recommendation based on search for similar service, or automatic service synthesis has the advantages that they can satisfy users' diverse requirements by enhancing the reuse rate and efficiency of the existing services, and reduce common carriers' CAPEX and OPEX further. Therefore, this paper constructed the ontology of semantic information between parameters through the annotation process that adds semantic information to parameters described in NSD, and made possible the search for similar service by the use of ontology reasoning and the automated network service composition.

This paper is organized as follows: Chapter 2 reviews TOSCA and technologies for semantic service. Chapter 3 describes the annotation process that adds semantic information to the existing NSD. Chapter 4 tests the ontology through ontology reasoning simulation. And Chapter 5 concludes this study.

II. BACKGROUND

This chapter describes semantic Web service technologies for adding, to TOSCA and the parameters of NSD, and discovering semantic information.

A. TOSCA

TOSCA defines a Meta model for defining IT services. The Meta model can define matters related to service structure as well as methods for the management of services. In the structure of TOSCA Service Template, the Topology Template defines the service structure, and the Plane defines a process model that is used for service management, generation, and deletion during the whole life cycle. The *Topology Template* consists of *Node Template* and Relationship Template. The Node Template represents a node of graph, and specifies the node types of components for each service. The *Node Type* defines the properties of components,

978-1-5386-4646-5/18/$31.00 ©2018 IEEE

and is used to define operations needed to operate components. The Relationship Template is used to specify the relationships between nodes, and every Relationship Template represents *Relationship Types* used to define the properties or meanings of relationships. Fig. 1 shows the template structure of TOSCA.[4]

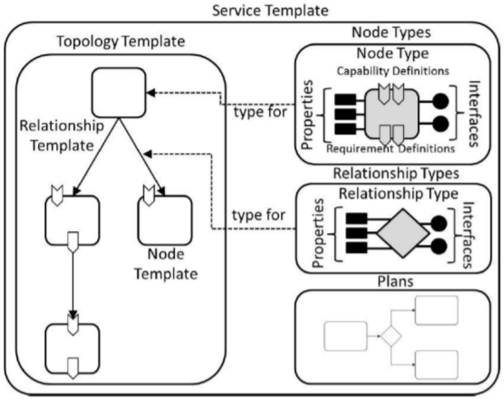


Figure 1. Template structure of TOSCA**.**

B. Semantic Web Service

The existing Web services provided convenient methods for the expression and delivery of information, thereby contributing to the wide use of the Internet. However, colossal information accumulated on the Web and the permission of access to the accumulated data only by means of keyword has resulted in numerous unnecessary data in case of information retrieval. To resolve this problem, research on the semantic Web service has been on the increase. The semantic Web service consists of the following technological components.[5]

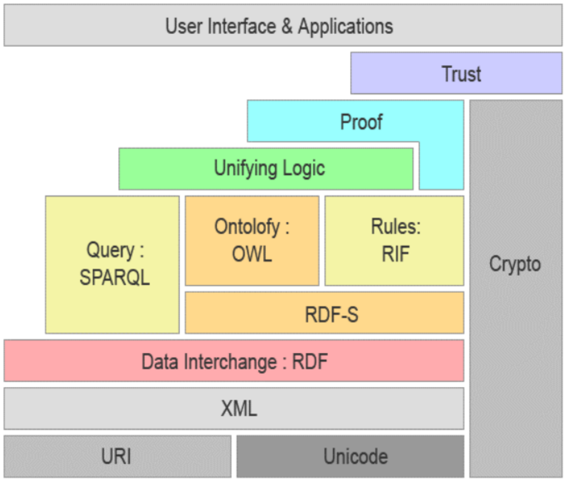


Figure 2. Structure of Semantic Web Service

C. Semantic annotation technology

Given that WSDL[6] (Web Service Description Language) documents, which mainly specify interface and input/output, contain insufficient information for semantic interpretation, the semantic annotation technology is a technology that adds the functional and non-functional information of ontology-based service to standard specification. The added information is used for semantic discovery and composition, and thus it may be said that the annotation technology is the most fundamental technology among semantic Web service technologies.

D. Semantic discovery technology

It is a technology that performs semantic service discovery with Web service specification in which semantic information is described in addition. There are semantic discovery technologies of various levels ranging from key word matching to service discovery that satisfies a natural language query.[7]

E. Semantic ontology technology

It is a method for expressing semantic information necessary for the discovery, mediation, and composition of the semantic Web service, and is a very useful technology for finding meanings among relationships due to its description of various relationships among data. As for ontology technology models, there are RDF (Resource Description Framework) and OWL[8] (Web Ontology Language) defined as standards by W3C, and WSML(Web Service Modeling Language) proposed by the ESSI WSML working group.

F. Semantic composition technology

According as the Internet users' requirements are diversified, methods for satisfying users' requirements by combining several services in terms of reuse and efficiency, rather than creating one Web service that satisfies all requirements, have been highlighted. The semantic composition technology is a technology that analyzes a requirement semantically, divides it into requirements of smaller units, discovers services that satisfy every requirement, and creates the order and logics of service execution. This technology is one that uses all the technologies of semantic annotation, semantic discovery, and semantic mediation, and it may be said that it is the ultimate technology pursued by the semantic Web service technology.

The WSDL used in the semantic Web service, however, has the problem that it cannot describe the NFV service clearly, and thus this paper uses OWL for the semantic annotation of NSD parameters. That is, the semantic information of parameters is annotated by the use of OWL, and the relationships of annotated parameters are defined with ontology. After then, similar VNF or similar network service is reasoned, using ontology reasoning SPARQL.[9]

III. Method for the semantic annotation of NSD information

This chapter looks into NSD, which describes all information about the network service, and describes the method for their annotation.

A. Semantic annotation of NSD information

Table 1 shows NSD[10] defined by ETSI. As shown in Table 1, NSD contains *ID*, *vendor*, and *version* for the network service; VNFD for VNFs forming the relevant service; the description of VNF forwarding graph linked into the service; and the dependency parameter that determines sequence for the operation of the network service. Among the relevant parameters, parameters that provide Non-functional information of the network service are classified as **Non-Functional Block**; *VNFD, VNFFGD, VLD, vnf_dependency, service_ deployment_flavor, connection_point, and PNFD* as **Functional Block** that provides functional information of the service; and the remaining parameters of *nsd_security, monitoring_parameter, lifecycle_event and auto_scale_policy* as **Optional Block**, respectively.

In addition, any category parameter that does not exist in the existing NSD is added to the Non-Functional Block so that network services of the same category may be managed. As for the *category*, it can be obtained from a developer and be defined by the administrator who manages network services on a portal. Table 2 shows the classification of parameters into Functional, Non-Functional, and Optional Blocks.

Table 1. NSD parameter

Parameter	Description
Id	ID of this Network Service Descriptor.
vendor	Provider or vendor of the Network Service.
version	Version of the Network Service Descriptor.
vnfd	VNF which is part of the Network Service,
vnffgd	VNFFG which is part of the Network Service.
vld	Virtual Link which is part of the Network Service,
lifecycle_event	Defines NS functional scripts/workflows for specific lifecycle events
vnf_dependency	Describe dependencies between VNF.
monitoring_parameter	Represents a monitoring parameter which can be tracked for this NS.
service_ deployment_ flavour	Represents the service KPI parameters and its requirement for each deployment flavour of the NS being described,
auto_scale_policy	Represents the policy meta data, which may include the criteria parameter & action-type. The criteria parameter should be a supported assurance parameter.
connection_point	This element describes a Connection Point which acts as an endpoint of the Network Service.
pnfd	PNFs which are part of the Network Service,
nsd_security	This is a signature of nsd to prevent tampering.

Table 2. Annotated NSD parameter

	Parameter
Functional Block	*VMFD*
	VLD
	VNFFG
	conntection_point
	service deployment_flavor
	PNFD
	vnf_dependency
Non-Functional Block	*name*
	id
	vendor
	version
	category
Optional Block	*auto_scaling_policy*
	monitoring_parameter
	lifecycle_event

B. NFV service ontology

This chapter constructs ontology by defining the relationships of parameters whose annotation was completed in the previous chapter. Fig. 3 shows NFV ontology constructed by this paper.

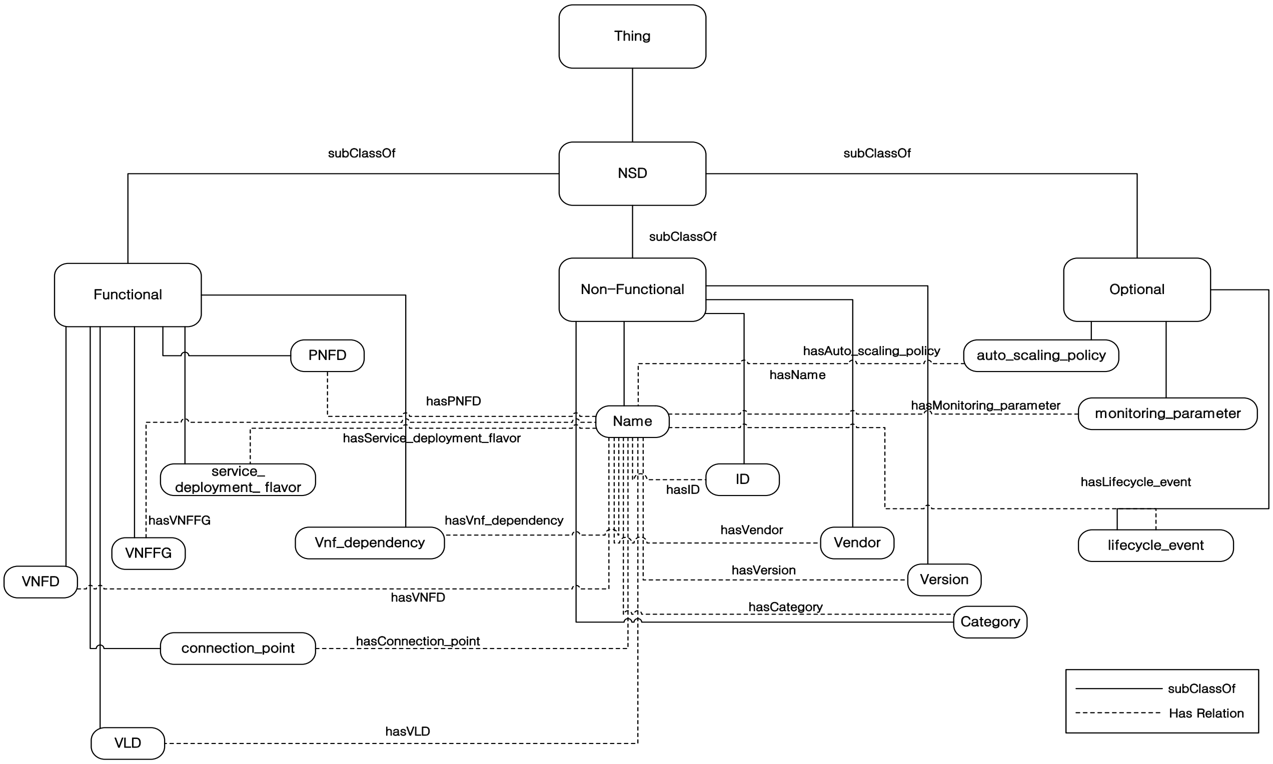


Figure 3. NFVO Ontology

As a language for the construction of the ontology, OWL As a language for the construction of the ontology, OWL was used. The OWL proposition describes facts about members of a class, and relationships between the members. Ontology, which consists of a set of such propositions, allows logical inference about facts not defined syntactically. The OWL ontology consists of **'classes'**, **'properties'**, and **'individuals'**.

In OWL, **owl:Thing** represents a set containing all individuals. All classes are the subclasses of **owl:Thing**. Therefore, in the NFV ontology, NSD, which is not a sub-concept of any other class, is defined as a subclass of **owl:Thing**. In the OWL ontology, a class is a set of individuals. OWL exactly expresses conditions for being a member of the class, by means of formalized description. Here, a hierarchical relationship of the superclass and the subclass between classes comes into being, and is called taxonomy. In the NFV ontology, blocks divided for annotation, i.e., **Functional**, **Non-Functional**, and **Optional** are defined as the subclasses of the NSD class, and their relevant nodes are also defined as sub-nodes of the VNFD class. Namely, the relationship of superclass and subclass implies that any belonging to the Functional class is a member of NSD and VNFD.

The OWL properties consist of *Object Properties* and *Data-type Properties*. The *Object Property* connects an 'individual' and 'an individual', and the *Data-type Property* connects an individual to XML Schema: Datatypes or RDF Constant. That is, the *Object Property* describes the relationship between class factors, and the *Data-type Property* the type and value of data to be adopted by a class factor, respectively. For example, as shown in Figure 4, if the NSD class has a Functional class as its subclass and the Functional class is defined as having the '***hasVNFD'*** relationship with the VNFD class, the following relationship can be defined: "Since the Functional class is a sub-node of NSD due to the relationship of superclass and subclass and is defined as having the '***hasVNFD'*** relationship with the VNFD class, NSD has sub-information of the relevant VNFD class." Therefore, the NFV ontology of this paper defined the Functional, Non-Functional, and Optional classes as having the '***has'*** relationship with the classes of their respective sub-parameters.

IV. Ontology verification through ontology reasoning

This chapter verifies whether ontology was properly constructed, by verifying the ontology described in Chapter 3 through ontology reasoning.

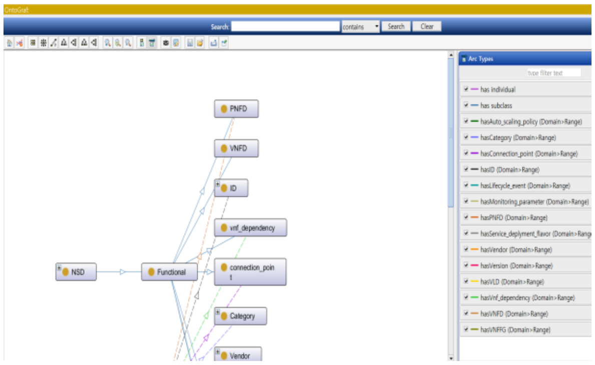
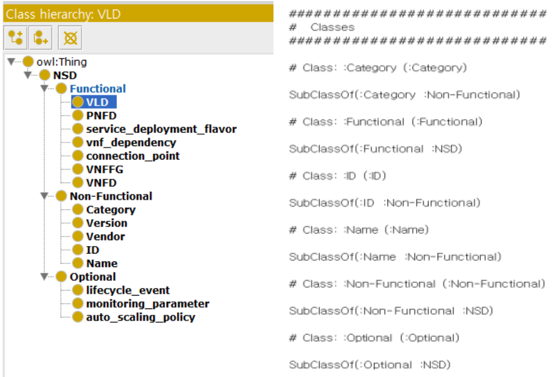


Figure 4. NFVO ontology (in Protégé)

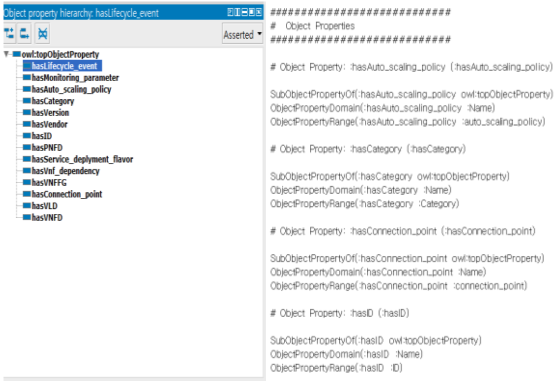
First, the ontology was constructed by using Protégé, an ontology building tool, as shown in Fig. 4. As seen in Fig. 4, it was confirmed that the ontology relationship described in Chapter 3 had been properly defined. Fig. 5 show the class structure and the owl code that were confirmed by **Protégé**[11]. Since this paper did not research methods for ontology reasoning, but used ontology reasoning to verify the ontology, the ontology relationship was verified with the simple SPARQL query.



```
Class hierarchy: VLD
owl:Thing
  NSD
    Functional
      VLD
      PNFD
      service_deployment_flavor
      vnf_dependency
      connection_point
      VNFFG
      VNFD
    Non-Functional
      Category
      Version
      Vendor
      ID
      Name
    Optional
      lifecycle_event
      monitoring_parameter
      auto_scaling_policy
```

```
##########################
#   Classes
##########################

# Class: :Category (:Category)

SubClassOf(:Category :Non-Functional)

# Class: :Functional (:Functional)

SubClassOf(:Functional :NSD)

# Class: :ID (:ID)

SubClassOf(:ID :Non-Functional)

# Class: :Name (:Name)

SubClassOf(:Name :Non-Functional)

# Class: :Non-Functional (:Non-Functional)

SubClassOf(:Non-Functional :NSD)

# Class: :Optional (:Optional)

SubClassOf(:Optional :NSD)
```

Figure 5. NFVO ontology class (in Protégé)



```
Object property hierarchy: hasLifecycle_event
Asserted
owl:topObjectProperty
  hasLifecycle_event
  hasMonitoring_parameter
  hasAuto_scaling_policy
  hasCategory
  hasVersion
  hasVendor
  hasID
  hasPNFD
  hasService_deplyment_flavor
  hasVnf_dependency
  hasVNFFG
  hasConnection_point
  hasVLD
  hasVNFD
```

```
##########################
#   Object Properties
##########################

# Object Property: :hasAuto_scaling_policy (:hasAuto_scaling_policy)

SubObjectPropertyOf(:hasAuto_scaling_policy owl:topObjectProperty)
ObjectPropertyDomain(:hasAuto_scaling_policy :Name)
ObjectPropertyRange(:hasAuto_scaling_policy :auto_scaling_policy)

# Object Property: :hasCategory (:hasCategory)

SubObjectPropertyOf(:hasCategory owl:topObjectProperty)
ObjectPropertyDomain(:hasCategory :Name)
ObjectPropertyRange(:hasCategory :Category)

# Object Property: :hasConnection_point (:hasConnection_point)

SubObjectPropertyOf(:hasConnection_point owl:topObjectProperty)
ObjectPropertyDomain(:hasConnection_point :Name)
ObjectPropertyRange(:hasConnection_point :connection_point)

# Object Property: :hasID (:hasID)

SubObjectPropertyOf(:hasID owl:topObjectProperty)
ObjectPropertyDomain(:hasID :Name)
ObjectPropertyRange(:hasID :ID)
```

Figure 6. NFVO ObjectProperty (in Protégé)

First, assuming that there were the VNFDs of P-GW and S-GW in two network services called vEPC, it was checked whether vEPC came out as a relevant network service and S-GW came out as a relevant VNF when data was inserted into the ontology individual. As a result of making the query, when searching for P-GW, vEPC was returned as the relevant network service, and S-GW as the relevant VNF. And it was found that sub-information for each service was also retrieved together as well as the name.

V. Conclusion

In this paper, NFV service ontology was constructed, using semantic technology. Similar or associated network services and VNFs were easily discovered by defining as ontology the relationship between parameters that could not be defined by the existing TOSCA. This allows the easy discovery of similar VNFs without having to search for VNF one by one, when common carriers create a specific network service, or network service providers produce network service; and the discovery of similar network service prevents the duplicate production of the same service, which was very effective in terms of efficiency and re-usability. However, this paper constructed ontology only, but did not investigate reasoning technology for providing more precise and diverse information. Therefore, it is planned to proceed in the future with research on methods for the automatic mash-up of network services through providing information on similar services based on the ontology constructed in this paper.

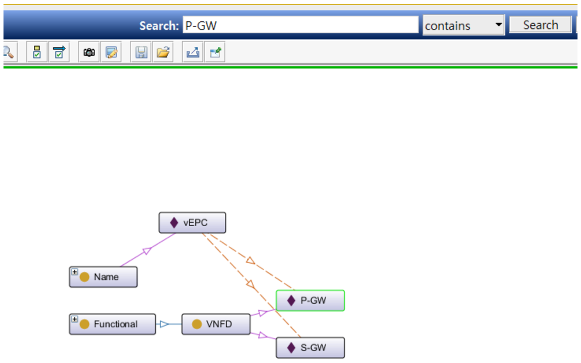


Figure 7. Ontology reasoning result

ACKNOWLEDGMENT

This research was supported by the IT R&D program of MSIP(Ministry of Science, ICT and Future Planning) / IITP(Institute for Information &amp; Communications Technology Promotion). [12221-14- 1001, Next Generation Network Computing Platform Testbed]

REFERENCES

[1] "ETSI", Network Functions Virtualisation White - Paper #1, 2012.

[2] Matt Rutkowski, TOSCA –An Open Standard for Cloud Application Portability, OASIS, 2014

[3] OASIS, "Topology and Orchestration Specification for Cloud Applications Version1.0 OASIS Standard ", 2013.

[4] OASIS, "TOSCA Simple Profile for Network Functions Virtualization (NFV) Version1.0", 2016

[5] H. Alesso, C. F. Smith, Developing Semantic Web Services, A K Peters/CRC Press, 2004.

[6] W3C Org, "Web Service Description Language (WSDL)", Http://www.w3.org/TR/WSDL, Accessed Feb, 2018

[7] L. A. Kamaruddin, J. Shen, G. Beydoun, "Evaluating usage of wsmo and owl-s in semantic web services", Proceedings of APCCM '12, pp. 53-58, 2012.

[8] W3C Org, "OWL Web Ontology Language", https://www.w3.org/OWL/, 2018, Accessed Feb, 2018

[9] N. F. Noy, D. L. McGuinness et al., Ontology development 101: A guide to creating your first ontology, 2001. Http://www.w3.org/TR/2004/REC-owl-features-20040210.

[10] ETSI ISG NFV, GSNFV-MAN 001 v0.6.2 (2014-07), "Network Function Virtualization (NFV) Management and Orchestration", 2014. 07

[11] Protégé website, https://protege.stanford.edu/, Accessed Feb, 2018