Semantic Representation of Cloud Services: a Case Study for Openstack

Beniamino Di Martino, Giuseppina Cretella, Antonio Esposito and Graziella Carta

Department of Industrial and Information Engineering
Second University of Naples
Aversa, Italy

7th Internation Conference on Internet and Distributed Cloud Computing- IDCS'14

Cloud Computing: Opportunities…

- Reduced up-front investment and maintenance costs
  - In-house infrastructures substituted by Cloud resources
- Better use of existing hardware
  - Cloud infrastructure auto-scales according to requests
- “Pay as you Go” paradigm
- Enhanced resiliency and disaster recovery
  - Distributed and replicated data centers ensure data safety
- Strong competition can lead to better quality services

… and challenges

- High variety of offered services → Several interfaces
  - How to (automatically) choose the right provider?
  - Use of standard interfaces is still poor
  - Vendor lock-in problem
- Data formats may vary from provider to provider
- Clouds expose available functionalities via Web services
  - Operations' signatures vary
  - Parameters' semantics is the same

- Portability (to and across Cloud platforms)
- Interoperability (also hybrid scenarios)

Objectives

- Categorization of services to support the choice of the right solution
  - Functional and non-functional characteristics sould be considered
- Uniform description of Cloud services and resources' configurations
  - Compliance with standards
- Enable comparison and mapping among different provider's services
  - Simplify catalogues' navigation
- Use a single API call to access cloud resources on different provider platforms
  - API parameters need to be comparable too

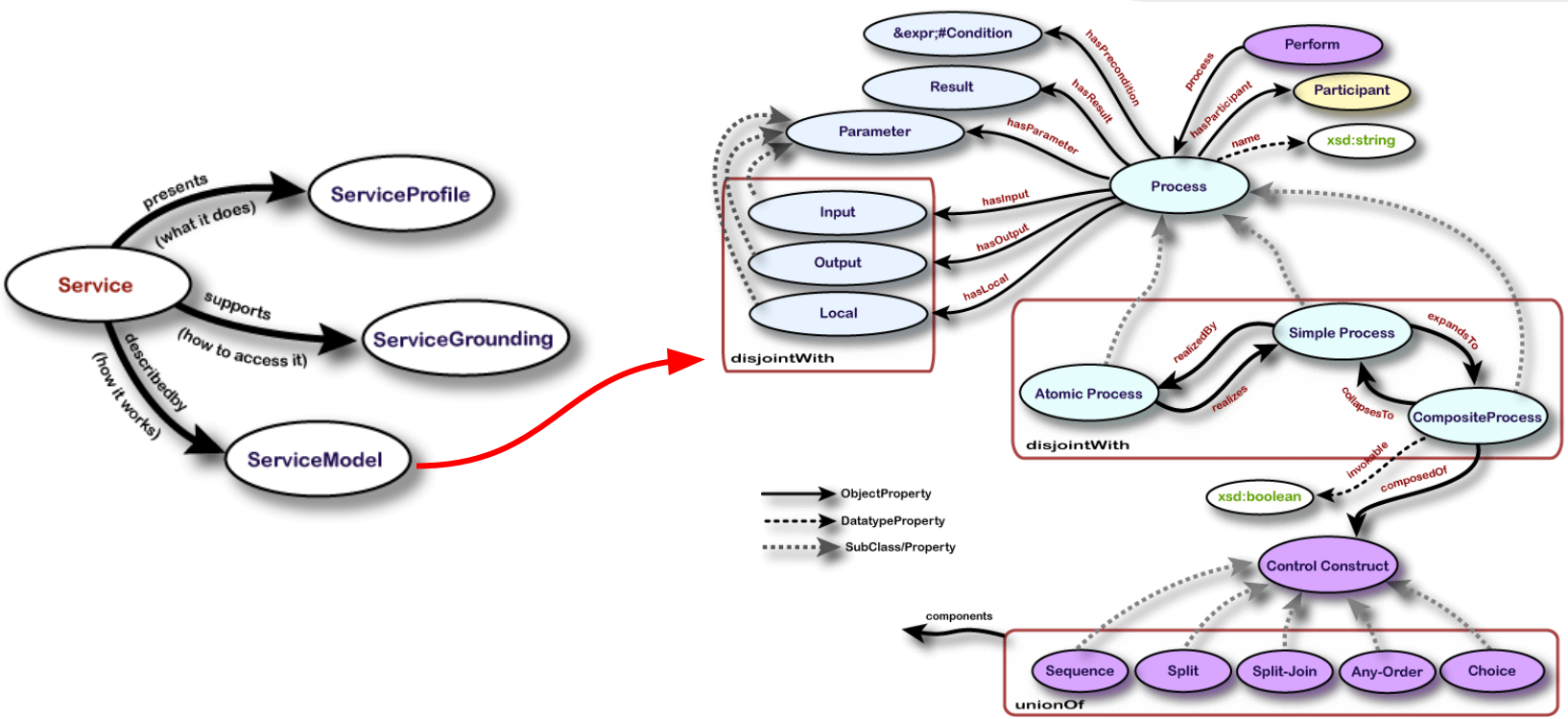
How Semantic can help

- Provide a machine readable format for services' and resources' descriptions
  - Using a standard formalism
- Automatic evaluation and navigation of available resources
- Uniform and agnostic description of both functional and non-functional aspects
- Overcome of the syntactic differences in API descriptions

Semantic technologies

- Web Ontology Language (OWL)
  - OWL is for processing information on the web
  - OWL was designed to be interpreted by computers, not read by people
  - OWL is written in XML
  - Ontology: description of things and their relationships
- OWL-S (OWL for services)
  - Ontology for the description of Semantic Web Services
  - Discovery, invocation, management and monitoring of services
  - Dynamic description of services
  - Management of services' orchestration
- SPARQL Protocol and RDF Query Language
  - Query Language for RDF databases (OWL is based on RDF)
  - Allows simple and fast interrogations on large RDF ontologies
  - Queries can be automatically written

OWL-S Model


Service
presents
(what it does)
ServiceProfile
supports
(how to access it)
ServiceGrounding
describedby
(how it works)
ServiceModel
&expr;#Condition
Result
Parameter
hasPrecondition
hasResult
hasParameter
Perform
process
Participant
hasParticipant
xsd:string
name
Process
Input
Output
Local
hasInput
hasOutput
hasLocal
disjointWith
Simple Process
Atomic Process
CompositeProcess
realizedBy
realizes
expandsTo
collapsesTo
disjointWith
invokable
xsd:boolean
composedOf
ObjectProperty
DatatypeProperty
SubClass/Property
Control Construct
components
Sequence
Split
Split-Join
Any-Order
Choice
unionOf

Openstack Services

Service included in the Provider Ontology (names refer to **Icehouse** release**)**

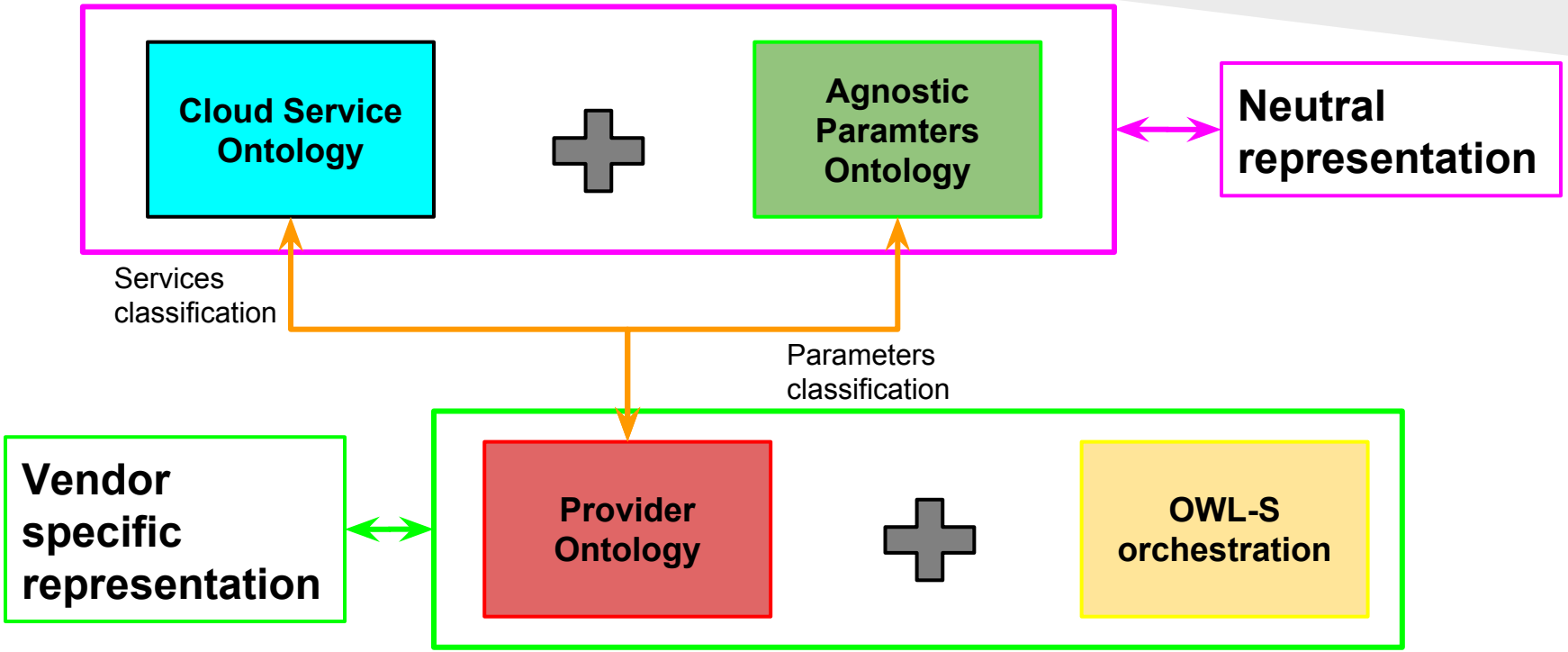
Core services:

- Compute (Nova)
- Object Storage (Swift)
- Block Storage (Cinder)
- Networking (Neutron)

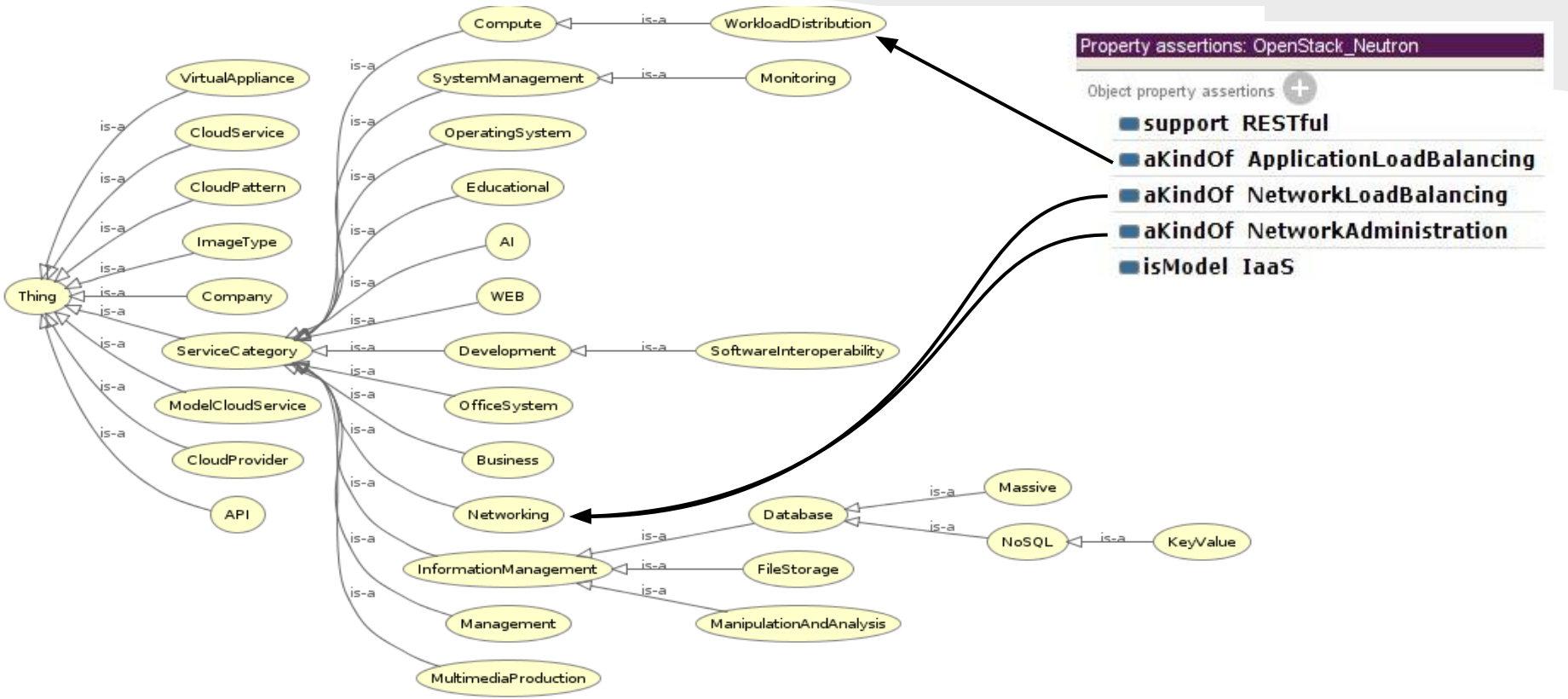
Shared services:

- Image Service (Glance)
- Identity Service (Keystone)
- Telemetry Service (Ceilometer)
- Orchestration Service (Heat)
- Database Service (Trove)

Semantic Representation of Services



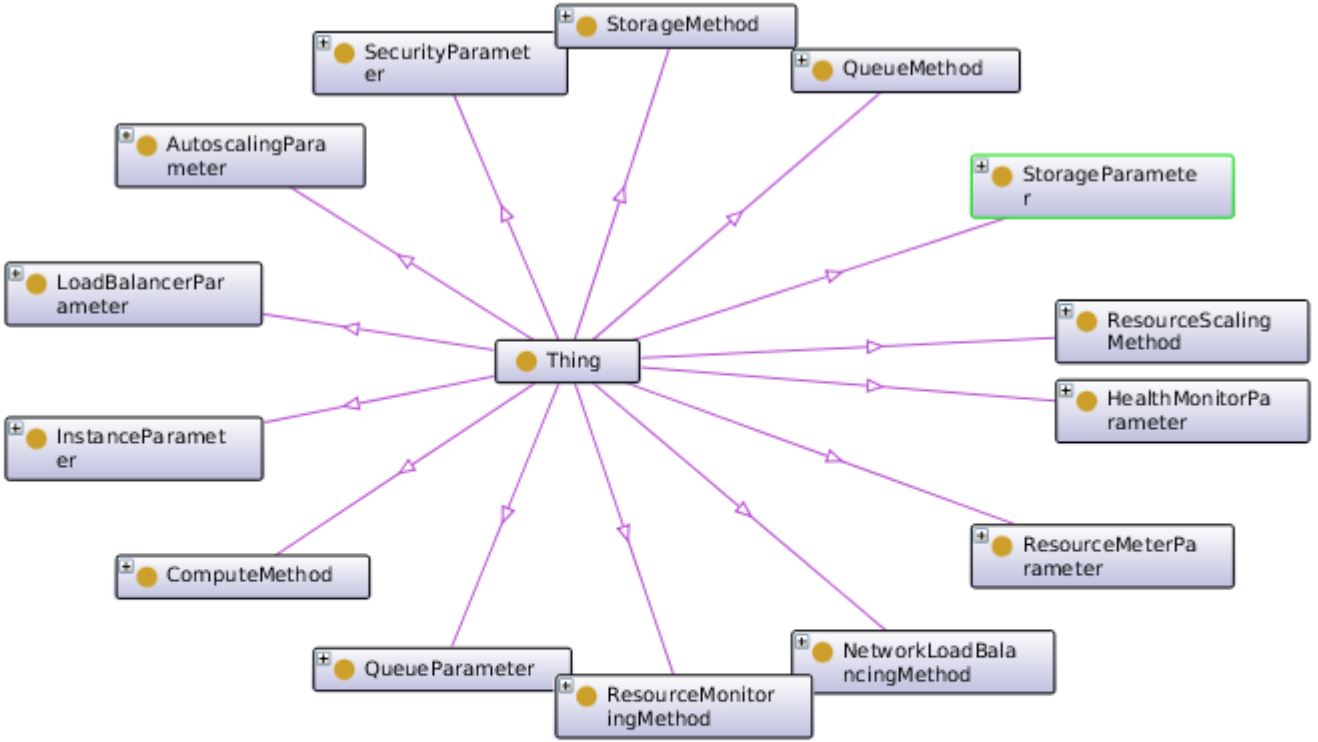
The Cloud Service Ontology



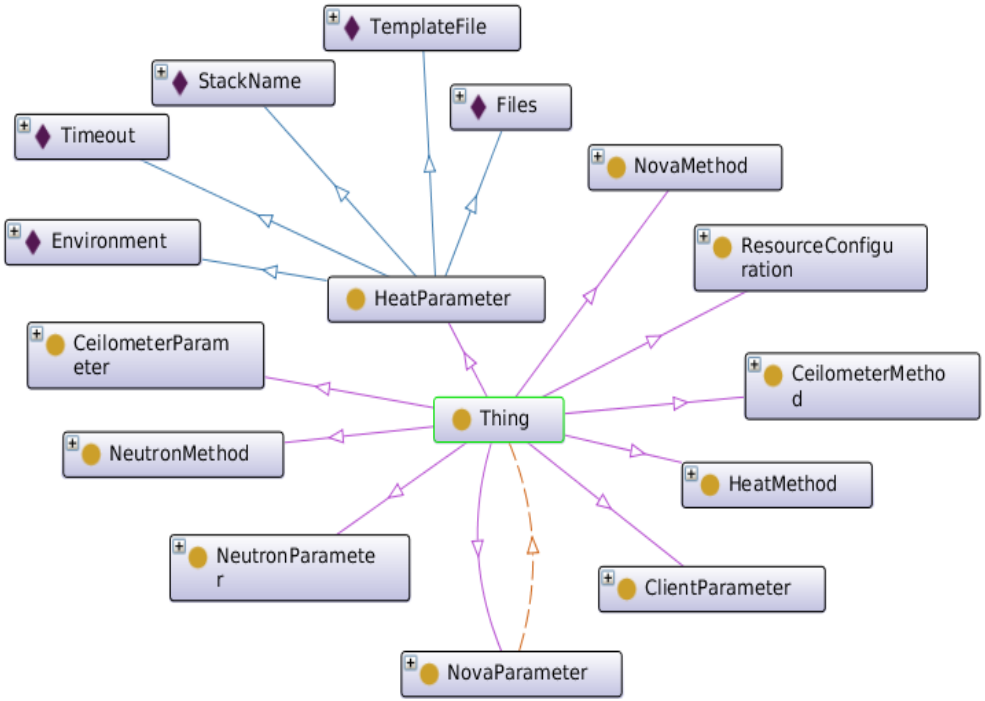
The Cloud Service Ontology

Property	Domain	Range	Meaning
aKindOf	Cloud Service/Virtual Appliance	Service Category	Specifies the category of a service or appliance
equivalence	Cloud Service/Virtual Appliance	Cloud Service/Virtual Appliance	Asses equivalence between services
hasVendor	Virtual Appliance	Company	Specifies the Company offering the appliance
hasVirtualizer	Virtual Appliance	Company	Specifies the virtualizer needed
isModel	Cloud Service	ModelCloudService	Specifies the service model
supports	Cloud Service	API	Specifies the API technology used
offeredBy	Cloud Service/Virtual Appliance	CloudProvider	Specifies the provider of the service or suporting the appliance

The Parameter Ontology

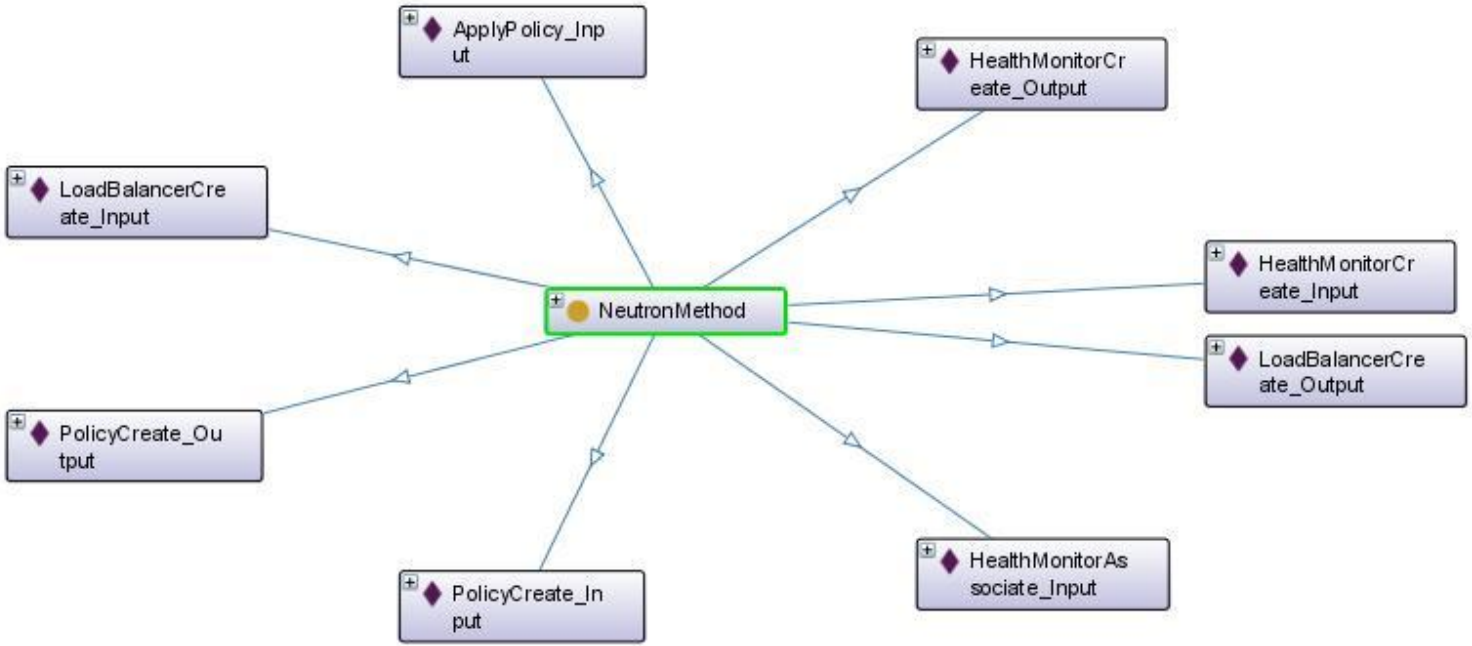


The Cloud Provider Ontology: Openstack

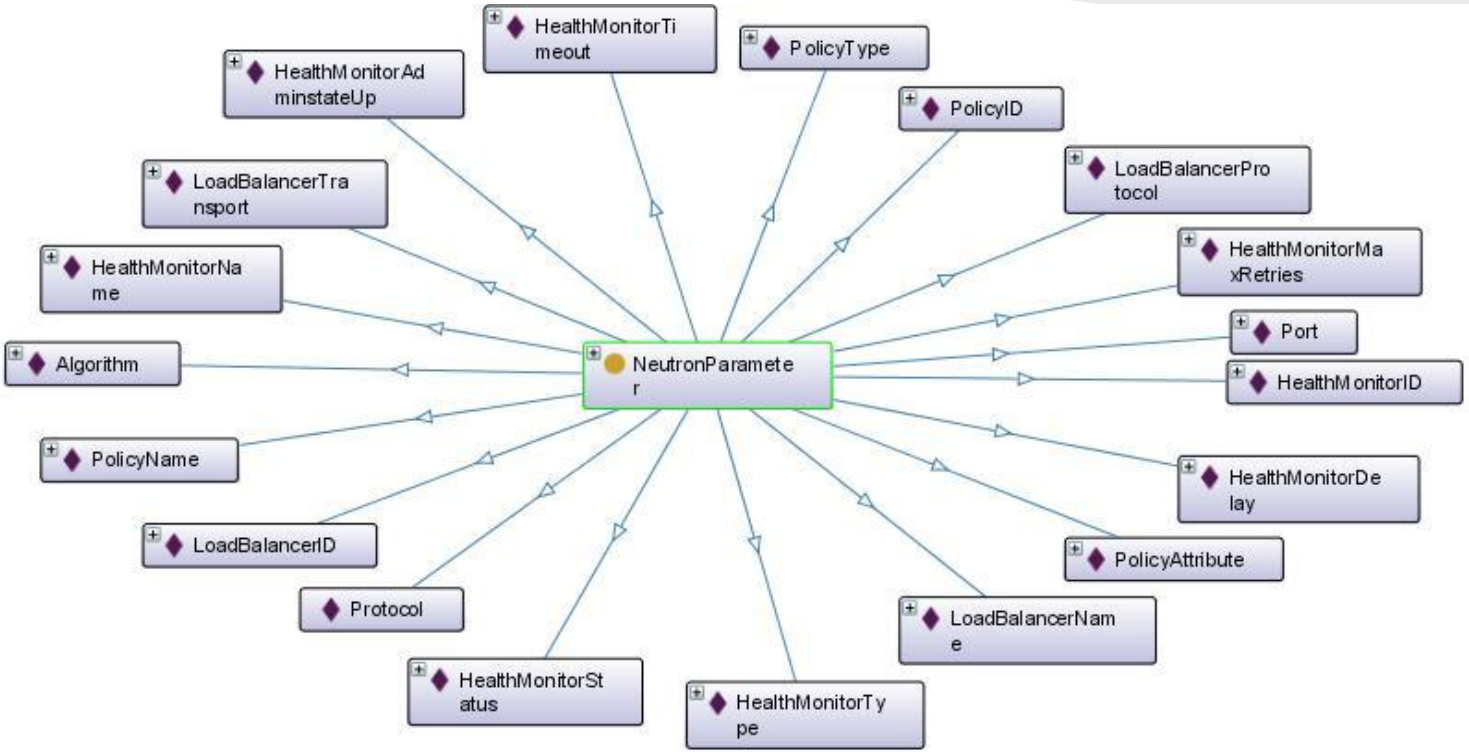


- **_Method** classes define operations exposed by a service
- **_Parameter** classes define input/output parameters
- **isInput/isOutput** properties connect a parameter to a method
- Each parameter instance is connected to the agnostic parameter ontology

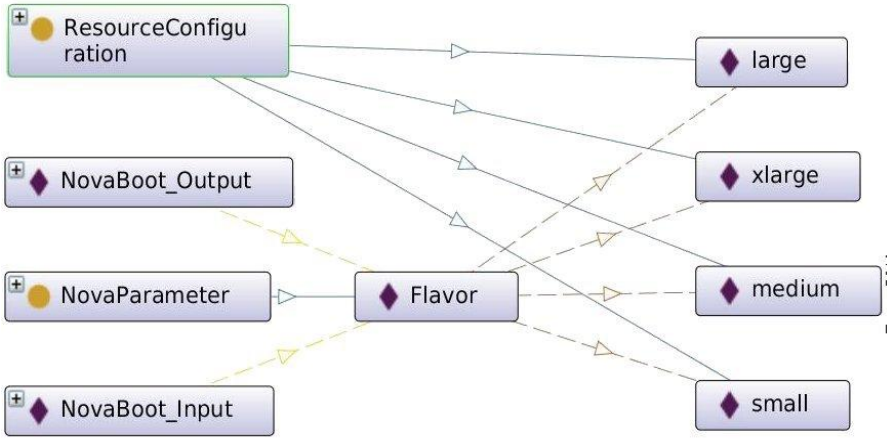
Neutron Method Class



Neutron Parameters

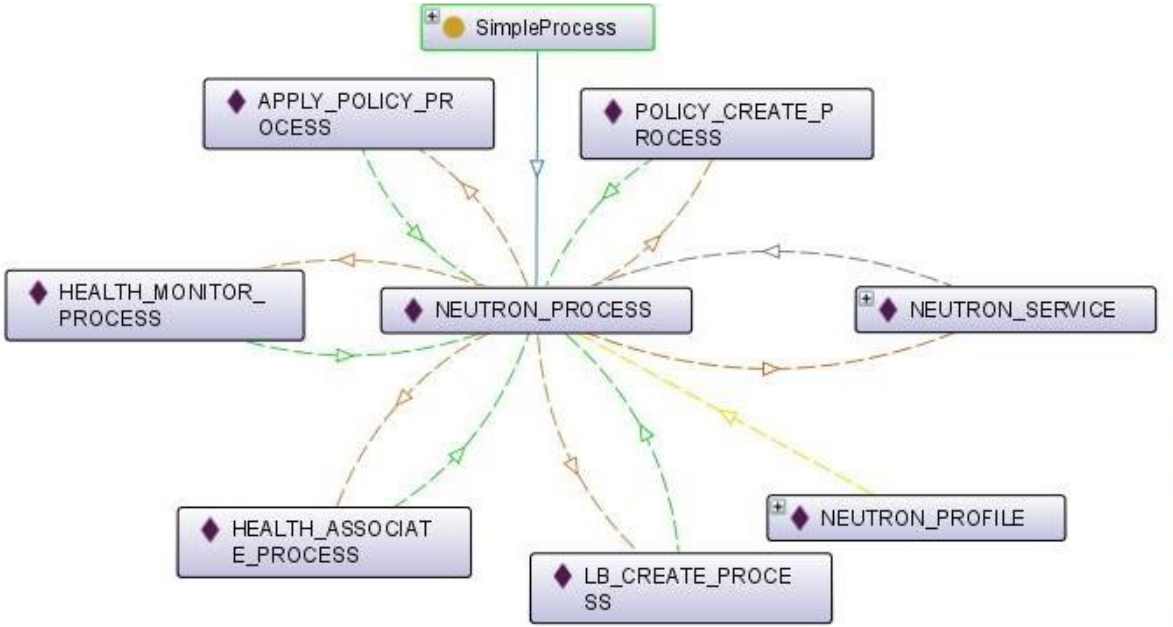


Resources Configurations



Flavor	Disk	vCPU	EphemeralStorage	Memory
tiny	1	1	0	512 MB
small	10	1	20	2048 MB
medium	10	2	40	4096 MB
large	10	4	80	8192 MB
xlarge	10	8	160	16384 MB

OWL-S Annotation



- **Neutron_Process** is a **SimpleProcess**
  - Directly connected to the **Neutron_Service** description
- Others are **Atomic processes**
  - Each Neutron Method is connected to an atomic process
- **hasOutput** and **hasInput** properties used for parameters

Querying the knowledge base\1

```
SELECT ?category ?equivalentService
WHERE {
        CSo:OpenStack_Neutron  this:aKindOf ?category.
         ?equivalentService CSo:aKindOf ?category}
```

Select all services falling in the same category of Openstack Neutron

Category	**EquivalentService**
NetworkAdministration	Azure_VirtualNetwork, BrocadeVyatta_vRouter, RedHat_JBossOperationNetwork,
NetworkLoadBalancing	Zeus_ExtensibleTrafficManager, Azure_TrafficManager, Amazon_ElasticLoadBalancer
ApplicationLoadBalancing	Azure_TrafficManager, Amazon_ElasticLoadBalancer Amazon_AppStream

Querying the knowledge base\2

```
SELECT ?resource vendor ? CPU ? Memory
WHERE { ? resource rdf:type pO:ResourceConfiguration.
        ?resource pO:hasVendor ?vendor.
        ?resource pO:vCPUs ?CPU.
        ?resource pO:Memory ?Memory .
        FILTER (?CPU>=3) }
```

Select resources configurations offering a minimum number of virtual CPUs

Resource Configuration	Vendor	vCPUs	Memory
xlarge	Openstack	8	16384 MB
large	Openstack	4	8192 MB
A3	Azure	4	7 GB
A4	Azure	8	14 GB
CP3	MyPlatform	3	5 GB

Conclusions and future works

- Needed a better categorization of services
  - Equivalence assessed on the base of methods exposed
- Automatic inference rules (SWRL?) to determine parameters and services equivalences
- Cloud Computing Patterns can be used to define a template for applications' orchestration
  - A semantic representation of them can be queried to retrieve information on applications' workflows

Thanks for your attention