A Semantic Model for Enhancing Network Services Management and Auditing

Carlos Rodrigues[1], Paulo Carvalho[1,3], Luis M. Álvarez-Sabucedo[2], and Solange Rito Lima[1]

[1] University of Minho, Department of Informatics, 4710-057 Braga, Portugal
[2] University of Vigo, Dept. of Telematics, Vigo, Spain
[3] e-mail: pmc@di.uminho.pt

Abstract. The road toward ubiquity, heterogeneity and virtualization of network services and resources urges for a formal and systematic approach to network management tasks. In particular, the semantic characterization and modeling of services provided to users assume an essential role in fostering autonomic service management, service negotiation and auditing.
This paper is centered on the definition of an ontology for multiservice IP networks which intends to address multiple service management goals, namely: (i) to foster client and service provider interoperability; (ii) to manage network service contracts, facilitating the dynamic negotiation between clients and ISPs; (iii) to access and query SLA/SLSs data on an individual or aggregated basis to assist service provisioning in the network; and (iv) to sustain service monitoring and auditing. In order to take full advantage of the proposed semantic model, a service model API is provided to allow service management platforms to access the ontological contents. This ontological development also takes advantage of SWRL to discover new knowledge, enriching the possibilities of systems described using this support.

Key words: Ontology; Network Service Management; SLA/SLS; Semantics; Multiservice IP Networks

1 Introduction

The evolution of the Internet as a convergent communication infrastructure supporting a wide variety of applications and services poses new challenges and needs to network management, which has to be more focused on managing services instead of network equipment. This approach requires the capability of viewing the network as a large distributed system, offering an encompassing set of services to users.

Commonly, the type of service, its Quality of Service (QoS) requirements and other technical and administrative issues are settled between customers and Internet Service Providers (ISPs) through the establishment of Service Level Agreements (SLA). The technological component of this agreement is defined through Service Level Specifications (SLS). SLSs provide a valuable guidance to service deployment of network infrastructures and monitoring of contracts' compliance. Attending to the ever growing number of home and business customers, contracted services and network heterogeneity, the implementation and management of network services are very demanding tasks for ISPs. Besides the inherent complexity, this process may lead to inefficient policy implementation and poor resource management.

In fact, under the current variety of available services, e.g. IP telephony, 3-play or 4-play solutions, the interaction between service providers and end customers is rigid and rather limitative regarding service negotiation and auditing tasks. For instance, from a user point-of-view, the possibility of a short-term upgrade on access bandwidth to the Internet or a tight quality control of the subscribed service would be of undeniable relevance.

From a service provider perspective, providing this sort of facilities, would clearly improve the level of service being offered, increasing competitiveness and resource management efficiency. These aspects are impelling ISPs to pursue autonomic solutions for service negotiation, configuration and management.

Although several proposals exist in the literature toward achieving dynamic service negotiation and management [1–4], the lack of a strong formal ground in addressing these tasks is evident and overcoming it is essential [5]. A formal specification of network services management semantics is required as the building blocks to create reasoning mechanisms to allow developing self-managed ISPs. By using a knowledge-based formal framework and an inference engine capable of reasoning over concepts, relations and changes of state, it is possible to create a more flexible and robust ground for specifying and implementing autonomic and adaptive management tasks.

As a contribution in this context, this work proposes an ontology specification in the domain of multiservice networks, which formally specifies the contractual and technical contents of SLAs, the network service management processes and their orchestration, promoting service autonomic management and configuration. This model provides support for a Service Management Platform that facilitates client and service provider interoperability, and service contracts management, including service data querying by the provider and, at some levels, by the client. This is enabled through a developed ServiceModel API, which allows the applicational use of the proposed ontology. The multiservice network semantic model is developed in Web Ontology Language (OWL), assisted by the Protégé-OWL tool. The use of Semantic Web technologies enhances service management modeling expansiveness and reusability.

This paper is structured as follows: research work on ontologies related to service definition and QoS is debated in Section 2; the developed model and its main modules are presented in Section 3; the way semantics is applied based on the developed API is discussed in Section 4; examples of practical use of the proposed model are provided in Section 5; the conclusions and future work are included in Section 6.

2 Related work

Several research studies focusing on ontologies for network services support and QoS are found within the research community. While part of the ontologies focuses on Web Services (WS) QoS requirements, other concentrate on SLA/SLSs support.

QoSOnt [6] is an OWL ontology that centers on comparative QoS metrics and requirements definition. Although this ontology supplies the correct semantics for matchmaking, this was never demonstrated due to datatype limitations in OWL. To overcome this limitation, a pure XML based solution was used, losing all of the virtues of OWL [7]. The DAML-QoS [8] is a QoS metrics ontology for WS developed in DAML+O. The ontology is divided in three layers: QoSProfile Layer, QoS Property Definition Layer and QoS Metrics Layer. In [9] a new Service Level Objective (SLO) concept, metrics' monitoring and statistical calculation semantics are presented. MOQ [10] is another proposal of a QoS semantics model for WS, but it is not exactly an ontology. It only specifies axioms and does not present a taxonomy structure or a dictionary of concepts. MonONTO [11] ontology aims at creating a knowledge base to support a client recommendation system. The ontology serves as a support to a decision recommendation tool by providing high-level information to the user about the compliance of the network facing the service level demands. In [12], an ontology aiming at the automation of network services management and mapping of services requirements into the network is proposed. The ontology is viewed in three perspectives: (i) the network service classification; (ii) the service level specification; and (iii) the deployment of network services. A group of generic ontologies to provide a framework for building SLAs is presented in [13]. In this context, the Unit Ontology contains all the comparable elements of an SLA, with the intention of supporting the creation

of any type of measurable unit. It also allows the definition of unit supported comparators and the creation of comparison operations. The other examples of available ontologies are: the Temporal Ontology for temporal occurrences such as events and intervals; The Network Units Ontology for units related to telecommunications networks; and the SLA Ontology for basic SLA specification. Therefore, rather than a QoS ontology, a set of reusable ontologies is proposed for providing support for other QoS semantic model implementations. The OWL-based ontology NetQoSOnt [14] intends to be the support of a reasoning tool for service requirements matchmaking. It promotes the definition of SLSs containing quality parameters belonging to the following levels: the Quality of Experience, the Quality in the Application Level, the Quality in the Network Level and the Quality in the Link Level.

In the proposals discussed above, the lack of an unified and encompassing approach for semantic modeling of services and corresponding contracts in a multiservice environment is clear. In fact, most of the proposals are more focused on specification of network services metrics than on integrated service management. In the present work, a holistic model for modeling multiservice networks is provided paying special attention to the characterization and auditing of services quality. This ontology focuses on service contracts to assist network services' implementation by specifying how the defined contract elements are deployed in the network infrastructure, a feature not considered in the reviewed works. Although the proposed model is still evolving, its modular structure and the usage of Semantic Web technologies leaves room to model expansion and integration with other proposals.

3 Multiservice Network Ontology

The proposed model is divided in two main modules: the service management module and the network module. As illustrated in Figure 1, these modules are organized as a layered structure where the upper layer has a dependency relation with the lower layer. This structure mimics real scenarios where the management component is, indeed, above the physical network. This formal representation of a network is expressed in formal terms using the support of OWL, following the principles from Methontology [15].

The network module, as stated above, acts as the base layer. It includes concepts of network node, network interface and network equipment configuration elements related to the implementation of contracted services in the network. The management module covers the domain network service management related to service contracts, including service monitoring rules. This module uses several elements of the network module. Services are categorized by relating them to a type of SLS [16, 17]. According to recommendations from [18], ITU Y.1541 and 3GPP standards, current service types include: real-time services, multimedia services, data services, and default traffic service.

Another important component of the proposed service model regards to multiservice monitoring (see Figure 1). This implies the definition of the main monitoring issues to include in the multiservice ontology to assist auditing of Internet services both from an ISP and customer perspective. To service providers, it will also allow a tight control of services, network resources and related configuration procedures.

On top of the Multiservice Ontology, a complete ServiceModel API offers to a Service Management Platform the access to the ontological contents. Without detailing the construction of the ontology at this point, it is relevant to highlight the identification of competence questions. These are the first and the last step in this methodology and fulfill the need to establish the requirements and the outcomes of the ontology itself, i.e., which questions the ontology will be able to answer. In the present case, the definition of an ontology for multiservice IP networks intends to address multiple service-oriented goals. Possible competence questions include:

(i) *from a customer perspective*: Which type of service packs are available for subscription? Which service parameters are customizable? Which is the available bandwidth at a specific access point, for a particular service? Is the contracted service being delivery within the negotiated QoS?

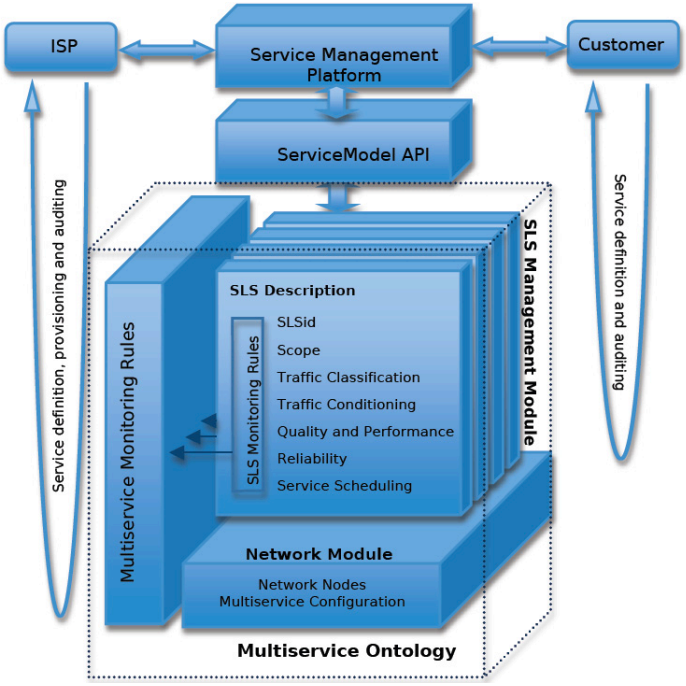

Fig. 1. Service model diagram

(ii) *from an ISP perspective*: At an aggregate level, which is the allocated bandwidth for a particular service type? Which are the negotiated parameters per SLS? Which are the configuration parameters on each interface of an edge network node and the available bandwidth per interface? Which services are supported between specific ingress and egress interfaces? Are the QoE/QoS requirements of a particular service being accomplished? On which network points are occurring QoS violations?

In the description of modules provided in the sections below, a top-down approach will be followed to allow a broad view of the multiservice ontology.

3.1 Management Module

The management module is where service contracts or SLAs are defined and managed. The first concept is the Client which identifies the customer part of the contract and stores all client information. A client is related to at least one SLA which represents a service contract. An SLA can have more than one SLS. The SLS structure, illustrated in Figure 2, follows the recommendations in [16, 17], and is briefly described below.

- *SLS Identification:* This field identifies the SLS for management purposes, being used by both provider and customer. It is composed of a unique SLS id parameter and a Service id parameter, allowing to identify multiple SLSs within the same service.
- *Scope:* The scope specifies the domain boundaries over which the service will be provided and managed, and where policies specified in a service contract are applied. Normally, SLSs are associated with unidirectional flows between at least one network entry point and at least one exit point. To cover bidirectionality, more than one SLS is associated with a service. The entry points and the exit points are expressed through ingress and egress interfaces, respectively (see Section 3.2). At least two `Interfaces` (ingress and egress) instances must be specified. The interface

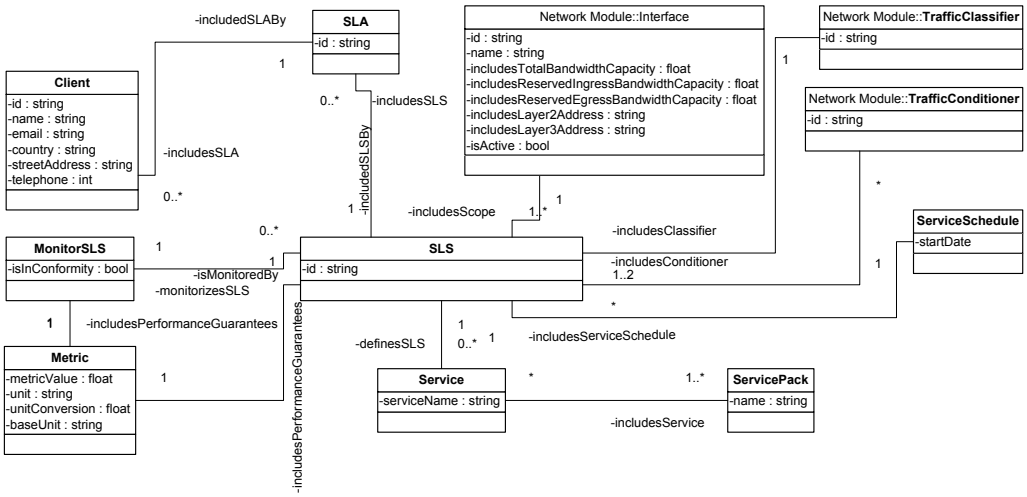

Fig. 2. SLS class diagram

identification must be unique and is not restricted to the IP address (the identification can be defined at other protocol layer).

– *Traffic Classifier:* The `Traffic Classifier` specifies how the negotiated service flows are identified for differentiated service treatment. Following Diffserv terminology, multifield (MF) classification and behavior aggregate (BA) classification are supported (see Section 3.2). Usually, BA classification takes place over previously marked traffic, e.g. in network core nodes or, in the case of SLSs, between ISPs. Two traffic classifiers can be specified, an ingress traffic classifier and optionally an egress one. The ingress/egress classifier is then applied to each ingress/egress interface within the scope of the SLS.

– *Traffic Conditioner:* This field specifies the policies and mechanisms applied to traffic flows in order to guarantee traffic conformance with the traffic profiles previously specified. Traffic conditioning occurs after traffic classification, so there is always a relation between the traffic classifier and the traffic conditioner specified within a SLS.

  An unlimited number of `TrafficConditioner` instances can be specified. As in the traffic classifier property, the conditioners are divided into ingress and egress depending on their role. The ingress/egress conditioner is articulated with the ingress/egress classifier on each interface defined in the SLS scope as an ingress/egress QoS policy. This property is not mandatory.

– *Performance Guarantees:* The Performance Guarantee fields specify the guarantees of service quality and performance provided by the ISP. Four quality metrics are considered: delay, jitter, bandwidth and packet loss, expressed through instances of the `Bandwidth`, `Delay`, `Jitter` and `PacketLoss` `Metric` subclasses. The definition of at least one instance of these `Metric` subclasses is mandatory, except on the Default Service type of SLS. Whenever there is a performance guarantee specification, a traffic conditioning action must also be specified. Delay and jitter are usually specified by their maximum allowed value or by a pair consisting of a maximum upper bound and a quantile. Packet loss (edge-to-edge) is represented by the ratio between the packet loss detected at the egress node and the number of packets sent at ingress node. Instead of quantitative, quality and performance parameters can also be specified in a qualitative manner.

– *Reliability:* The Reliability is usually specified by the mean downtime (MDT) and by the maximum allowed time to repair (TTR). The no compliance of the negotiated parameters may result in a penalty for the ISP.
– *Service Schedule:* The Service Schedule defines the time period of service availability associated with an SLS. While a start date is always specified, an end date is only specified in case of a reservation, `ReservedServiceSchedule`, in which the client requests the service during a specific period of time. In the default case, `StandardServiceSchedule`, only the service start date is specified, i.e., the contract must be explicitly terminated by the client.
– *Monitoring:* Monitoring refers to SLS' performance parameters monitoring and reporting. For that purpose, a measurement period, a reporting date and a threshold notification are specified. Other parameters such as the maximum outage time, total number of outage occurrences, reporting rules and reporting destination may be specified.
– *Type of Service:* The type of service is described by the `Service` class. This class allows the definition of services offered by the ISP to customers from a business-oriented perspective. Offered services are described through a set of qualitative metrics. The mapping from a qualitative service description to a quantitative service specification is assured by the ISP. The `Service` class allows to relate the SLS with a specific instance of service offering. It also helps establishing SLS templates on an application level. Services can be offered as a package (e.g. triple or quadruple play services) through the `ServicePack` class.

3.2 Network Module

At present, an ISP is represented as a cloud network, where only edge (ingress and egress) nodes are visible. The abstract representation of domain internal nodes and inherent internal service configuration mechanisms are left for future work. Therefore, instead of representing configuration elements at per-hop level, the model is focused on a per-domain level. In this module (see Figure 3) there are three key elements:

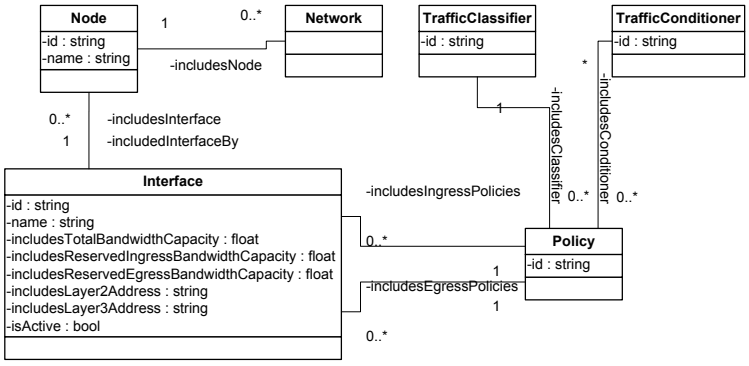

Fig. 3. Interface class diagram

– *Node:* The `Node` class represents a network node (on the current model, corresponds to a domain border node). It is related to a set of `Interface` class instances.
– *Interface:* The `Interface` class represents ingress and egress points of the ISP domain. Specifically it allows the mapping of external network interfaces or entry/exit points of ISP border nodes. The interface supports a two-way traffic flow. It is possible to attach layer 2 and layer 3 addresses to the interface concept in order to relate it to

a real network interface in the ISP domain. Each interface has a total bandwidth capacity and a reserved bandwidth capacity specified dynamically for ingress traffic and egress traffic. For QoS purposes, it is possible to specify a set of QoS policies. In this case, a QoS policy is a relation between a traffic classifier instance and a set of traffic conditioner instances applied to traffic classified by the former. A QoS policy can be an ingress policy or an egress policy.
The `Interface` class, as illustrated in Figure 3, is defined by an identifier, link and network layer addresses and total bandwidth capacity both downstream and upstream. It includes two counters for ingress and egress reserved bandwidth of all contracted services applied to this interface. Each instance can be related to a set of QoS policies applied on incoming and outgoing traffic. A boolean value is also defined for interface state indication.

- *Traffic Classifier:* The `TrafficClassifier` class has two subclasses: MF and BA. The `BA` class instances, applied to previously marked traffic, only have one field, a relation with a `Mark` class instance. The `Mark` class contemplates all forms of aggregated traffic marking (such as DSCP, IPv6 FlowLabel, MPLS Exp, etc.). The `MF` class allows the definition of traffic classification rules with multiple fields. There are no constraints on the number of allowed fields and these are divided into: link, network and transport header fields. This means that several types of fields can be used: IPv4 and IPv6 addresses, IPv6 Flow Label, ATM VPI/VCI and MPLS Labels. The fields used in the classification rule are combined through a logic operator represented through the `LogicOperator` class instances `AND` and `OR`. For a more complex classifying rule definition, other `TrafficClassifier` class instances can be stated as fields, working as nested classification rules.
- *Traffic Conditioner:* The traffic conditioner is designed to measure traffic flows against a predetermined traffic profile and, depending on the type of conditioner, take a predefined action based on that measurement. Traffic conditioning is important to ensure that traffic flows enter the ISP network in conformance with the established service profile. It is also an important policy for handling packets according to their conformity level facing a certain traffic profile with the purpose of differentiating them in the network. According to their features, there are three `TrafficConditioner` subclasses: the `Marker`, `Policer` and `Shaper` classes. The policer usually takes an immediate action on packets according to their compliance against predefined traffic profile. A `Policer` class instance must have a set of traffic measurement parameters and at least two levels of actions defined. Three different policers are defined in the current model. The `TokenBucketPolicer` represents a single rate policer with two level actions (for in profile and out of profile traffic). The `SingleRateThreeColorMarker` and `TwoRateThreeColorMarker` are examples of policers with three levels of conformance actions. The `Shaper` is the only conditioner subclass where no immediate action is taken on traffic flows. Instead, all packets are buffered until traffic profile compliance is verified. The `Marker` class is a special type of conditioner which performs traffic marking and may be combined with other traffic conditioner elements.

3.3 Multiservice Monitoring Module

As illustrated in Figure 1, the aim of the monitoring system to develop is twofold:

(i) to monitor and control SLSs parameters in order to ensure that measured values are in conformance with the negotiated service quality levels. This auditing purpose involves a prior characterization of each service requirements, monitoring parameters and corresponding metrics, and the definition of appropriate measurement methodologies and tools to report multilevel QoS and performance metrics to users and system administrators;

(ii) to measure and control the usage of network resources. This includes the identification of network configuration aspects impacting on services performance, namely scheduling and queuing strategies on network nodes. In fact, monitoring network resources and

triggering traffic control mechanisms accordingly will allow to maintain consistent quality levels for the supported services and the fulfillment of the negotiated SLSs.

Another main concern of this task is to congregate users and ISPs perspectives regarding the description and control of services quality. This means that the perceived service quality for users (Quality of Experience - QoE), commonly expressed through subjective parameters, has to be identified and mapped into objective and quantifiable QoS parameters, able to be effectively controlled by network service providers. Therefore, the articulation of QoE and QoS, and the identification of appropriate measurement methodologies for evaluating and controlling service quality levels in both perspectives (users and ISPs) is a main concern to cover in the present module.

In this context, multiservice monitoring is expected to provide a clear identification and layering of all monitoring issues to include in the multiservice ontology to assist auditing and control of negotiated service levels through the proposed Service Management Platform.

3.4 The VoIP Service as Example

As mentioned before, a service provider may describe each provided service through a set of qualitative metric values, which are then mapped to quantitative values to assist, for instance, configuration and service control. For example, a VoIP type of service can be described as:

VoIP Service
Bandwidth: `Low_Bandwidth` = at least 1 Mbps
Delay: `VeryLow_Delay` = at most 100 ms
Jitter: `VeryLow_Jitter` = at most 50 ms
Packet Loss: `VeryLow_Loss` = at most 0.001 % of lost packets

This type of service description is used for SLS classification in accordance with the specified metrics. In other way, SLSs can be built based on the type of service description when required. An example of an SLS instance for the VoIP service is shown in Figure 4.

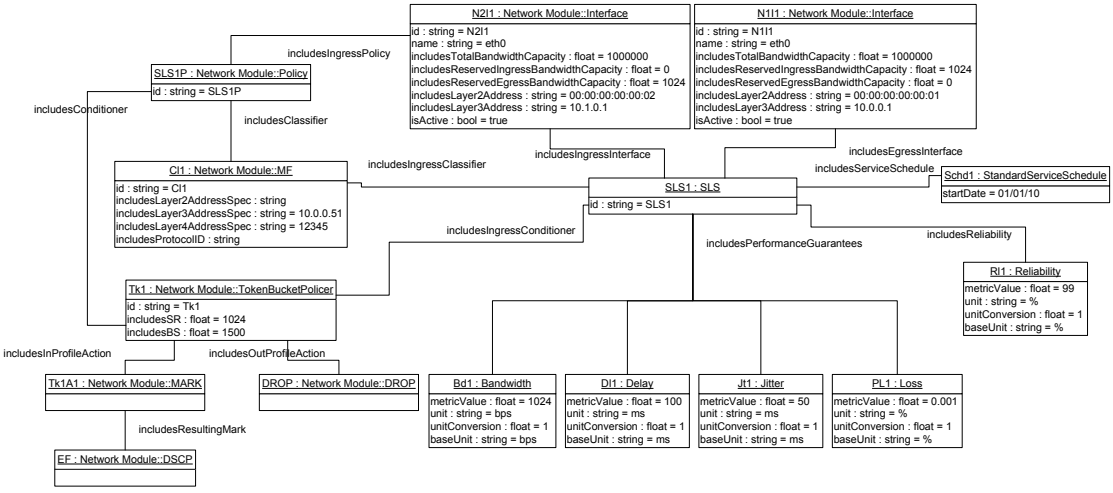

Fig. 4. SLS example diagram for VoIP service

When the SLS instance is set, the `TrafficClassifer` and `TrafficConditioner` specified lead to QoS policy instances. A relation is then established between each QoS

policy and network interfaces instances specified in the scope of the SLS (Figure 4). This policy information is useful for automating the deployment of QoS mechanisms in the ISP network infrastructure.

By establishing relations among all these entities, a change in one of them affects all other related entities. For example, a change in an SLS parameter is spread through all the corresponding SLS configurations in the network infrastructure.

4 Applying semantics

This section discusses how the presented model is converted into an ontological support. Thus, the characterization of the multiservice domain can be used in further software solutions ranging from web contents to complex software agents responsible for decision making. This ontology was developed according to the basis proposed by Methontology [15]. In this way, it is guaranteed its conformance with a set of methodological rules and the final product can be traced to its origin and reused in a simple and cost-effective manner.

The proposed ontology provides the main concepts and properties required to describe multiple services levels and corresponding quality in a network domain. For its implementation, according to the terminology proposed in Methontology, it was used Protégé to generate the OWL representation. This representation uses not only classes and properties, but it also includes restrictions on the values of the previous ones. Therefore, it is ensured the conformance of current contents and future pieces of information to the established parameters of the system.

Besides per-class restrictions, a set of general rules are defined for establishing new rule-based relations between individuals. These rules are expressed using SWRL and they are applied to check information in order to discover new possible instances and properties within the system. So far, there are defined rules for (i) validation of interfaces capacity included in a contract scope; (ii) compliance verification of monitored metrics in relation to service contract specifications; (iii) changing interfaces network status; (iv) qualitative classification of performance metrics; and (v) classification of SLSs according to the type of service. For example, the following rule states that if all SLS performance metrics have qualitative values matching a definition of a type of service then the SLS specifies a service of that type.

$$
\begin{aligned}
&SLS(?sls) \wedge Service(?service) \\
&\wedge\, includesBandwidth(?sls, ?bandwidth) \\
&\wedge\, includesBandwidthQualValue(?service, ?qualiBandwidth) \\
&\wedge\, includesDelay(?sls, ?delay) \\
&\wedge\, includesDelayQualValue(?service, ?qualiDelay) \\
&\wedge\, includesJitter(?sls, ?jitter) \\
&\wedge\, includesJitterQualValue(?service, ?qualiJitter) \\
&\wedge\, includesLossQualValue(?service, ?qualiLoss) \\
&\wedge\, includesPacketLoss(?sls, ?loss) \\
&\wedge\, includesQualitativeValue(?bandwidth, ?qualiBandwidth) \\
&\wedge\, includesQualitativeValue(?delay, ?qualiDelay) \\
&\wedge\, includesQualitativeValue(?jitter, ?qualiJitter) \\
&\wedge\, includesQualitativeValue(?loss, ?qualiLoss) \\
&\rightarrow definesSLS(?service, ?sls)
\end{aligned}
$$

Additional rules are defined for the above mentioned issues. Nevertheless, for other purposes, it is suggested to define rules at application level due to the complexity and limitations of SWRL [19] at using knowledge from different sources and involving advanced logical checks.

On the top of the provided ontology, it is developed a complete software API. This API, referred as the ServiceModel API, is implemented following the diagram presented in Figure 5.

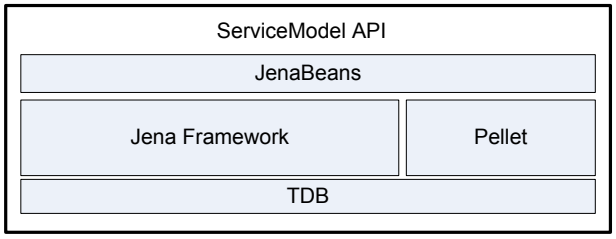

Fig. 5. ServiceModel API structure diagram

The Jena Framework [20] plays a major role in the developed software. It provides support for working with RDF and OWL based archives. The handling of OWL entities (classes, individuals and restrictions) is provided by the Jena Ontology API. Recall that the ontological content can be accessed from the local computer or from a remote server. The Pellet [21] engine is the reasoner used due to its SWRL [22] support.

Working on top of the Jena framework, the Jena beans API binds RDF resources (in this case, OWL classes) to Java beans simplifying the process of Java-to-RDF conversion. This feature enables users to work with individuals as Java objects.

The persistence of the knowledge is guaranteed by the TDB [23] technology, which is clearly simpler and more efficient than the SDB solution (uses SQL databases for storing RDF datasets). However, the API integration of an SDB solution is not totally abandoned.

The ServiceModel APl intents to assist future projects in several goals: (i) to foster client and service provider interoperability; (ii) to manage network service contracts, facilitating the dynamic negotiation between clients and ISPs; (iii) to access and query SLA/SLSs data on a individual or aggregated basis to assist service provisioning in the network; and (iv) to assist service monitoring and auditing. Therefore, this API, aimed to sustain further developments, supports in a straightforward manner for software developers the following features: (i) the insertion and removal of information on the Knowledge Base (creating/destroying individuals); (ii) the validation of the Knowledge Base information (classification and realization); (iii) establishment of more complex rules based relations (not possible through SWRL); (iv) Knowledge Base querying, implemented through SPARQL [24] and the ARQL Jena API; and (v) Knowledge Base persistence.

5 Practical Application

Once the semantic model for fully describing SLAs/SLSs is set, services can be provided on top of it. Semantics is not, in general, an end by itself. On the contrary, its use is motivated by achieving further goals. In the case of this proposal, it is generated a framework to boost interoperability and advanced data mining features. Bearing that in mind, services were derived as proof-of-concept. Firstly, it is suggested a RDFa support to introduce annotations on xhtml descriptions about SLAs and SLSs. Afterwards, it is suggested the use of a semantic engine to recover information from a repository of information regarding the formers.

RDFa [25] is a semantic technology that empowers users to include semantic annotations on XML (or xhtml) contents. These annotations are invisible for human user but easily recovered for software agents using GRDDL [26]. It is important to keep in mind that both technologies are official recommendations from W3C.

Taking as a basis the provided OWL model for describing the system, annotations can be included in xhtml describing SLAs and SLSs. For the sake of clarity, it is included the following example:

```
<p xmlns:sla="http://owl.det.uvigo.es/sls/">
  The provided connection under the interface
  <span property="sls:id">Service1</span>,
  provides a total BW of
  <span property="sls:includesTotalBandwithCapacity">
   20 MBps</span>.
</p>
```

As shown in this xhtml snippet, a network interface and some of its properties are described. This definition of capacities of the Network Module can be directly recovered using GRDDL. The use case expected for this functionality is related to the web pages of ISPs. Service providers, when offering their services, can include this information into their web pages. Users will be able to recover this information through software agents on their behalf, and include it into a data repository for further decisions.

Once these pieces of information are included in a Semantic Database, regardless of its origin, either from GRDDL extraction or from other sources, it is possible to get added-value services. Using SPARQL Queries [24], for example, it is possible to locate SLAs/SLSs fulfilling specific properties the user is interested in. The only requirement is to identify the graph matching the desired properties and implement the corresponding SPARQL query. Authors successfully tested this feature by means of the API provided. It is actually rather simple to deploy a software tool that looks for, for instance, the cheapest SLA in the market or the one offering the fastest network access, among other features.

6 Conclusions and Future Work

This paper has presented an innovative approach to the development of a semantic model in the domain of multiservice networks. This model formally specifies concepts related to service and SLS definition, network service management, configuration and auditing, creating the reasoning mechanisms to ground the development self-managed ISPs. Although being conceptually aligned with the differentiated service model, the solution is generic without being tied to a specific QoS paradigm.

The usefulness of the present semantic service modeling has been pointed out for multiple applications in the context of multiservice management. In particular, aspects such as dynamic service negotiation between service provides and end customers, and auditing of Internet services being provided may be strongly improved as consequence of using the proposed ontology.

Possibilities and features from this ontology are also presented to software developers by means of a ServiceModel API. The functionality within this library can be used for the above mentioned goals. Due to the modular schema for this software component, its inclusion on future projects constitutes a simple task that will provide a useful support in further developments.

References

1. D'Arienzo, M., Pescapè, A., Ventre, G.: Dynamic service management in heterogeneous networks. Journal of Network and System Management **12**(2) (2004)
2. Sarangan, V., Chen, J.: Comparative study of protocols for dynamic service negotiation in next generation internet. IEEE Communications Magazine **44**(3) (2006) 151–156
3. Cheng, Y., Leon-Garcia, A., Foster, I.: Toward an autonomic service management framework: A holistic vision of SOA, AON, and autonomic computing. IEEE Communications Magazine **46**(5) (May 2008) 138–146

4. Zaheer, F.E., Xiao, J., Boutaba, R.: Multi-provider service negotiation and contracting in network virtualization. In: IEEE NOMS'10. (2010) 471–478
5. Atkinson, E., Floyd, E.: IAB concerns and recommendations regarding internet research and evolution. RFC 3869, Internet Engineering Task Force (August 2004)
6. Dobson, G., Lock, R., Sommerville, I.: Qosont: a qos ontology for service-centric systems. In: EUROMICRO '05: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications, Washington, DC, USA, IEEE Computer Society (2005) 80–87
7. Dobson, G., Sanchez-Macian, A.: Towards unified qos/sla ontologies. In: SCW '06: Proceedings of the IEEE Services Computing Workshops, Washington, DC, USA, IEEE Computer Society (2006) 169–174
8. Zhou, C., Chia, L.T., Lee, B.S.: Daml-qos ontology for web services. In: IEEE International Conference on Web Services, Los Alamitos, CA, USA, IEEE Computer Society (2004)
9. Zhou, C., Chia, L.T., Lee, B.S.: Qos measurement issues with daml-qos ontology. In: IEEE International Conference on E-Business Engineering, Los Alamitos, CA, USA, IEEE Computer Society (2005) 395–403
10. Kim, H.M., Sengupta, A., Evermann, J.: Moq: Web services ontologies for qos and general quality evaluations. Int. Journal of Metadata, Semantics and Ontologies **2**(3) (2007) 195–200
11. Moraes, P., Sampaio, L., Monteiro, J., Portnoi, M.: Mononto: A domain ontology for network monitoring and recommendation for advanced internet applications users. In: Network Operations and Management Symposium Workshops, IEEE NOMS 2008. (April 2008) 116–123
12. Alípio, P., Neves, J., Carvalho, P.: An ontology for network services. In: International Conference on Computational Science (3). (2006) 240–243
13. Green, L.: Service level agreements: an ontological approach. In: ICEC '06: Proceedings of the 8th international conference on Electronic commerce, New York, NY, USA, ACM (2006) 185–194
14. Prudencio, A.C., Willrich, R., Diaz, M., Tazi, S.: Quality of service specifications: A semantic approach. IEEE International Symposium on Network Computing and Applications (2009) 219–226
15. Fernández-López, M., Gómez-Pérez, A, Juristo, N.: Methontology: From ontological art towards ontological engineering. Symposium on Ontological Art Towards Ontological Engineering of AAAI. (1997) 33–40
16. Morand, P., Boucadair, M., P. Levis, R.E., Asgari, H., Griffin, D., Griem, J., Spencer, J., Trimintzios, P., Howarth, M., Wang, N., Flegkas, P., Ho, K., Georgoulas, S., Pavlou, G., Georgatsos, P., Damilatis, T.: Mescal D1.3 - Final Specification of Protocols and Algorithms for Inter-domain SLS Management and Traffic Engineering for QoS-based IP Service Delivery and their Test Requirements . Mescal Project IST-2001-37961 (Jan 2005)
17. Diaconescu, A., Antonio, S., Esposito, M., Romano, S., Potts, M.: Cadenus D2.3 - Resource Management in SLA Networks. Cadenus Project IST-1999-11017 (May 2003)
18. Babiarz, J., Chan, K., Baker, F.: Configuration Guidelines for DiffServ Service Classes. RFC 4594 (Informational) (August 2006)
19. Zwaal, H., Hutschemaekers, M., Verheijen, M.: Manipulating context information with swrl. A-MUSE Deliverable D3.12 (2006)
20. McBride, B.: Jena: a semantic web toolkit. IEEE Internet Computing **6**(6) (2002) 55–59
21. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics **5**(2) (June 2007) 51–53
22. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, W3C (2004)
23. Owens, A., Seaborne, A., Gibbins, N., mc schraefel: Clustered tdb: A clustered triple store for jena. In: WWW 2009. (November 2008)
24. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C recommendation, W3C (January 2008)
25. Birbeck, M., Adida, B.: RDFa primer. W3C note, W3C (October 2008)
26. Connolly(Editor), D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL). W3C recommendation, W3C (January 2007)