3rd IFIP/IEEE Workshop on Management of the Future Internet

OSLAM: Towards Ontology-based SLA Management for IPTV Services

Sin-seok Seo, Arum Kwon, and Joon-Myung Kang
Department of Computer Science and Engineering
Pohang University of Science and Technology
(POSTECH)
Pohang, Korea
Email: {sesise, arumk, eliot}@postech.ac.kr

James Won-Ki Hong
Division of IT Convergence Engineering
Pohang University of Science and Technology
(POSTECH)
Pohang, Korea
Email: jwkhong@postech.ac.kr

***Abstract*—IPTV is emerging as a new service that can be very lucrative for telecommunication service providers. In addition, work is progressing to make IPTV an entertainment platform that can replace traditional TV for customers. Guaranteeing service quality is one of the most important factors to make IPTV services successful. For this, an IPTV service provider and a customer make a contract that uses specific quality indicators in a Service Level Agreement (SLA). To efficiently manage and guarantee SLAs, a method is required that can manage all hierarchical performance indicators from raw device level performance indicators to key quality indicators. In this paper, we analyze various IPTV performance indicators from various standard organizations and suggest an IPTV performance indicator hierarchy by extending the DEN-ng information model. Then, we propose an architecture that uses an ontology and Semantic Web Rule Language to manage SLAs, and more specifically, to detect SLA violations. The proposed architecture is implemented and tested using a simple scenario. The test results showed a possibility that the architecture can be used for detecting SLA violations.**

***Index Terms*—QoS, SLA, Network Management, Ontology, SWRL, IPTV**

I. Introduction

Internet Protocol Television (IPTV) will be an important application for next-generation networks, and will provide exciting new multimedia services for service providers, such as Live TV, Internet on TV, Video on Demand (VoD), multi-channel SD/HD, and T-Commerce (purchasing products using an IPTV service) [1].

As shown in Fig. 1, customers using IPTV services enter into a Service Level Agreement (SLA) with a service provider. The SLA defines the set of services and their performance levels that the customer will receive. Customers often negotiate SLAs to determine costs and penalties based on achieved performance levels. Customers want consistent performance without any SLA violation. This paper presents an SLA management architecture to detect SLA violations based on performance indicators (PIs) from an IPTV service platform (Fig. 1). We define PI as a term that indicates all types of performance related metrics including Device Performance Indicators (DPIs), Key Performance Indicators (KPIs), and Key Quality Indicators (KQIs). Detailed explanations about these PIs are given in Section II and III.

This work was partly supported by the IT R&D program of MKE/KEIT [KI003594 , Novel Study on Highly Manageable Network and Service Architecture for New Generation] and WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-2010-000-10100-0).

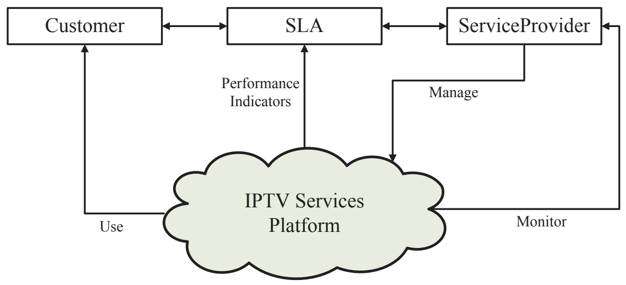

Fig. 1. General SLA Management Architecture

It is important to gather raw data for measuring end-to-end performance. This is usually done using one or more agents in the IPTV Set Top Box (STB) as well as in the network [2]. However, to efficiently manage and guarantee the SLAs, we need a method to manage all hierarchical PIs. The set of PIs to be managed range from raw device data, such as that obtained from Management Information Bases (MIBs), to KQIs, which represent important end-to-end quality metrics.

In this paper, we propose Ontology-based SLA Management (OSLAM), which is an approach to efficiently manage SLAs for IPTV services using ontologies in conjunction with the Semantic Web Rule Language (SWRL) [3]. First, we analyze and classify various IPTV PIs. Then, we define an IPTV PI hierarchy by extending the current DEN-ng information model [4], [5]. Next, we describe our OSLAM architecture by defining ontological models for representing PIs and SLAs, and specify SWRL rules for inferring hidden relationships among PIs and SLAs, for calculating PIs from other PIs, and for detecting SLA violations. We show the possibility of the OSLAM architecture using a simple scenario, which is implemented by Protégé [6] and Jess [7]. OSLAM aims not only to provide services to customers with good quality performance, but also to offer the chance to prevent SLA violations to service providers.

The remainder of this paper is organized as follows. Sec-

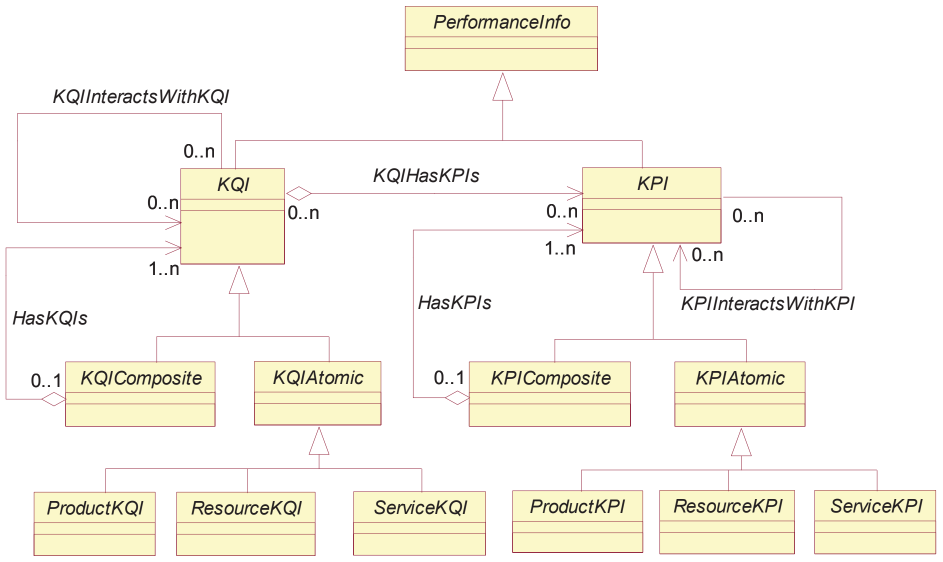

Fig. 2. Relationships among Performance Indicators of DEN-ng Model

tion II describes PIs, SLAs and SLA management as related work. We present our IPTV PI model in Section III. We then describe our OSLAM architecture in Section IV. Implementation and test of OSLAM are given in Section V. Finally, we draw our conclusions and discuss future work in Section VI.

II. Related Work

An SLA defines the level of service delivered to a customer in terms of contractual metrics. Typically, penalties can be given to the service provider in the case of non-compliance with an SLA [8]; similarly, bonuses can be provided for services that exceed appropriate metrics.

Service providers have used KPIs as metrics for reporting the performance of network-based services. The Tele Management Forum (TM Forum) defined KPIs as "indicators that provide measurements of a specific aspect of the performance of a service resource (network or non-network) or group of service resources of the same type" [8]. KPIs represent network performance; thus, they cannot completely represent end-to-end service quality.

The TM Forum proposed KQIs, which are "indicators that provide measurements of a specific aspect of the performance of the product, product components (services) or service elements, and draw their data from a number of sources including the KPIs" [8]. The TM Forum also defined a hierarchy among KPIs, KQIs, and SLAs. They asserted that network performance data consisting of KPIs, and KPIs are used for the calculation of Service KQIs. Service KQIs are used for calculating Product KQIs, and SLAs can be defined in terms of Product KQIs.

The definitions of KPI and KQI from the TM Forum are not enough to cover all PIs of IPTV services; the forum defined only two types of KQIs and did not define any subclasses of KPIs. We found that KPIs and KQIs can be divided into several groups. Therefore, we propose a performance indicator model that has a rich hierarchy in Section III.

Several standard organizations have defined their own PI classification criteria. The ITU-T classified IPTV related PIs into *Perceptual Quality*, *Video Stream Quality*, and *Transport Quality* [9]. The DSL Forum classification defined a three layer architecture: *Service Layer*, *Application Layer*, and *Transport Layer* [10]. The ATIS suggests four categories to classify IPTV PIs: *Transmission Quality*, *Media Stream Quality*, *Content Quality*, and *Transaction Quality* [11].

A model for managing these various IPTV PIs has been suggested [12]. The model can define KPIs, KQIs, and SLAs using eXtended Markup Language (XML) elements, and it is used for mapping from KPIs to KQIs and for checking for SLA violations. This work is very similar to our work; the difference is that we have used an ontology for defining PIs and SLA terms and, more importantly, their interrelationships. Moreover, we have used SWRL [3] rules for mapping between different PIs and for inferring the presence of SLA violations; this is more robust than using fixed rules to define SLA violations. In addition, we have incorporated DPIs as well as KPIs and KQIs, while [12] considered only KPIs and KQIs.

The authors of [13] proposed an autonomous architecture for QoE optimization, which is comprised of a *Monitor plane*, a *Knowledge plane*, an *Action plane*, and a *Knowledge base*. In this paper, ontology and SWRL are used for building a knowledge base similar to our work. However, [13] does not consider a hierarchy and mapping of DPIs, KPIs, and KQIs. Furthermore, it does not deal with the concept of an SLA.

III. IPTV Performance Indicators Model

In our previous work [4], [5], we modeled the hierarchy of KPIs and KQIs, as well as relationships between them, by extending the definition of KPIs and KQIs from the TM Forum and incorporating that definition into the DEN-ng information

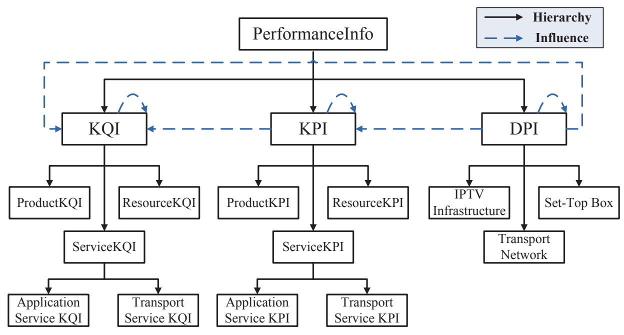

Fig. 3. Performance Indicator Hierarchy for IPTV Services

TABLE I
DEVICE PERFORMANCE INDICATORS (DPIS)

Premise	**Indicator**
IPTV Infrastructure	Server Utilization (CPU, Memory, Network) Server Rebooting Server Error ...
Transport Network	Number of Incoming Packets Number of Discarded Incoming Packets Number of Outgoing Packets Number of Discarded Outgoing Packets Bandwidth Queue Length ...
Set-Top Box (STB)	STB CPU Load STB Memory Utilization STB Rebooting STB Error STB Buffer Size ...

model (see Fig. 2). In this approach, we model KPIs and KQIs using a composite pattern [14], and it defines two types of KQI concepts: *KQIAtomic* and *KQIComposite*. *KQIAtomic* represents stand-alone KQIs. *KQIComposite* represents compound KQIs that are obtained by aggregating a set of KQIs (*KQIAtomic* and/or *KQIComposite*). *KQIAtomic* is divided into three sub-classes: *ProductKQI*, *ServiceKQI*, and *ResourceKQI*. The *ProductKQIs* represent the product qualities that are provided to end-users and are used for the contractual SLA between a service provider and a customer (e.g., availability of IPTV services). The *ServiceKQIs* represent the qualities of a specific service aspect. Therefore, multiple *ServiceKQIs* can be used for calculating a *ProductKQI*. The *ResourceKQIs* represent a specific aspect of the service that can be measured in a resource (e.g., channel zap time of IPTV services). Note that the TM Forum did not define the concept of a *ResourceKQI*. In addition, the TM Forum defined *ProductKQIs* and *ServiceKQIs* as independent concepts; this fails to take advantage of the similarities inherent in these concepts.

KPIs have a similar hierarchy, and also use the composite pattern to represent stand-alone and compound KPIs. This enables KPIs to be calculated from other KPIs, and KQIs to be calculated from other KQIs or KPIs.

TABLE II
KEY PERFORMANCE INDICATORS (KPIS)

ProductKPI	**ServiceKPI**		**ResourceKPI**
ProductKPI	Application Service KPI	Audio / Video KPI	Frame Impairment Level Encoding / Decoding Error Encoding / Decoding Delay Audio / Video Sync Loss Audio / Video Codec Resolution Bit Rate ...
		EPG KPI	EPG Control Delay EPG Error Rate EPG Font Size EPG Resolution ...
		Security KPI	Authorization Error Authentication Error Encryption Error Encryption Delay ...
	Transport Service KPI	IPTV Infra-structure KPI	Server Delay Cache Process Delay Server Error Ratio Maximum Users ...
		Trans-port Network KPI	Packet Loss Ratio Bandwidth Packet Delay Packet Jitter Packet Reordering Ratio Packet Loss Period Packet Loss Distance Packet Discard Rate ...
		STB KPI	STB Command Processing Delay STB Buffer Overflow STB Decoding Delay STB OS Boot Delay ...

We surveyed and analyzed IPTV PIs and their classifications from several standard organizations, including ITU-T FG IPTV [9], [15], DSL Forum [10], ATIS [11], TM Forum [16], [17], and ETSI [18]. As a result, we propose a PI hierarchy for IPTV services that encompasses key concepts from these standards by extending the DEN-ng PI model (Fig. 2), as shown in Fig. 3.

Low-level DPIs have to be considered in order to manage all aspects of the PI hierarchy of IPTV services. Thus, we included them in our PI hierarchy; note that they are missing in existing work, such as [4] and [12]. DPIs represent the performance metrics that we can measure directly from devices, networks, and infrastructure for IPTV services. Low-level DPIs can influence other DPIs, as well as KPIs and KQIs. An IPTV services platform is comprised not only of a transport network, but also an IPTV infrastructure and an IPTV STB. Hence DPIs have them as sub-categories: *IPTV Infrastructure*, *Transport Network*, and *STB* (Fig. 3). Several examples of DPIs are given in Table I.

KPI has three sub-categories: *ProductKPI*, *SeviceKPI*, and

TABLE III
KEY QUALITY INDICATORS (KQIS)

ProductKQI	ServiceKQI		ResourceKQI
ProductKQI	Application Service KQI	Audio KQI	Consistent Loudness Level Noise Level Audio Distortion Volume ...
		Video KQI	Color Error Blurriness Edge Busyness Block Distortion Smearing Jerkiness Frame Freezing Frame Skipping ...
		A/V KQI	Lip Synchronization ...
		EPG KQI	EPG User-friendliness EPG Responsiveness ...
	Transport Service KQI	Responsive-ness KQI	Channel Zap Time STB Start Delay Content Play Delay ...
		Reliability KQI	Service Availability Out-of-Service Frequency Mean Time Between Failures Mean Time to Provisioning Mean Time to Repair ...

ResourceKPI as modeled in DEN-ng [4], [5]. *ServiceKPIs* are divided into two groups: *ApplicationServiceKPIs* and *TransportServiceKPIs*. The *ApplicationServiceKPIs* represent the performance metrics that are affected by the application aspects of IPTV services (e.g., Electronic Program Guide (EPG) control delay), and the *TransportServiceKPIs* represent the metrics that are affected by data transportation or processes (e.g., server delay). The *ApplicationServiceKPIs* are classified as *Audio/Video KPIs*, *EPG KPIs*, and *Security KPIs*, while the *TransportServiceKPIs* are classified as *IPTV Infrastructure KPIs*, *Transport Network KPIs*, and *STB KPIs*. We classified *ResourceKPIs* for IPTV services as well (Table II).

KQIs have three sub-categories and *ServiceKQIs* are divided into two groups: *ApplicationServiceKQIs* and *TransportServiceKQIs* similar to KPIs. The *ApplicationServiceKQI* includes *Audio KQIs*, *Video KQIs*, *A/V KQIs*, and *EPG KQIs*, while the *TransportServiceKQIs* include *Responsiveness KQIs* and *Reliability KQIs*. We classified *ResourceKQIs* for IPTV services as well (Table III).

IV. ONTOLOGY-BASED SLA MANAGEMENT

In this section, we describe our ontology-based SLA management approach, which is named OSLAM. First, we explain the overall architecture of OSLAM and suggest ontological models for representing SLAs and PIs. After that, we propose a method that uses SWRL [3] rules that can infer relationships among PIs and SLAs. A description of SWRL rules that calculate PIs from other PIs and detect SLA violations follows.

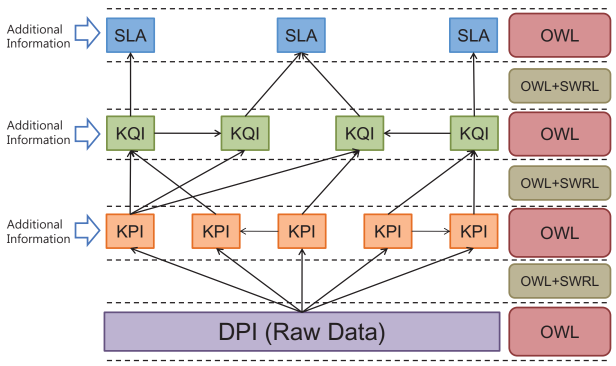

Fig. 4. Conceptual Diagram of OSLAM

A. OSLAM

The conceptual diagram of OSLAM is given in Fig. 4. OSLAM collects various PIs including DPIs, KPIs, and KQIs from devices and other additional information sources such as IPTV applications. Collected PIs are stored and maintained using W3C's Web Ontology Language (OWL) [19], [20]. OSLAM infers relationships among PIs using OWL and SWRL. For instance, if *DPI A* influences *KPI B*, and *KPI B* influences *KQI C*, then OSLAM infers that *DPI A* influences *KQI C*. In addition, OSLAM stores and maintains SLAs using OWL. Finally, OSLAM calculates PIs from other PIs and detects SLA violations by comparing the current value of PIs and the SLA using SWRL.

B. Ontological Model

We analyzed the characteristics of SLAs and PIs to develop ontological models for representing them and their relationships with OWL (Fig. 5 and 6). An SLA is a contract between a customer and a service provider over a specific service. Our SLA model contains both of these entities (Fig. 5). An SLA has constraints that a service provider has to guarantee. These constraints are stored in *Responsibility* objects, which contain *Threshold*, *CompareFunction*, and *PerformanceInfo*. For example, the SLA that has responsibility for

> "*Service Availability* should be maintained at over 99%",

has *Threshold* of '99%', *CompareFunction* of 'greater than', and *PerformanceInfo* of 'Service Availability'. The *PerformanceInfo* value is compared to the *Threshold* using the *CompareFunction*. If the responsibility condition is not satisfied, then the *Violated* value of the SLA is changed to 'true'. Our SLA model also contains *Penalty* objects to describe penalties that a service provider should incur when an SLA is violated.

The *PerformanceInfo* model is given in Fig. 6. The *PerformanceInfo* can be a member entity of a *Responsibility* to describe SLAs, and it influences or is influenced by other *PerformanceInfo* entities. For example, 'Queue Length', which is classified as a Transport Network DPI, influences 'Discarded

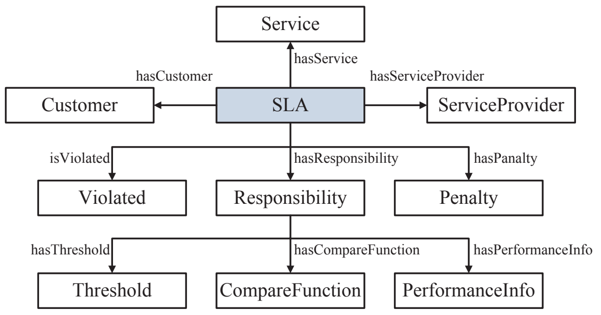

Fig. 5. Ontological Model for Representing SLA

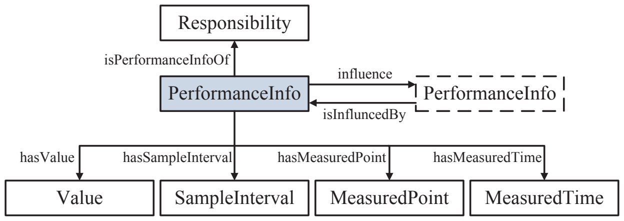

Fig. 6. Ontological Model for Representing PerformanceInfo

Incoming Packets' and 'Packet Loss Ratio', which are a DPI and a KPI respectively. The *value* of a *PerformanceInfo* can be directly obtained from a device or obtained by calculation using other *PerformanceInfos*. The obtained *Performance Info Value* is associated with *SampleInterval*, *MeasuredPoint*, and *MeasuredTime*.

C. Inferring Relationships among PIs and an SLA

We have suggested ontological models for representing SLAs and PIs (Fig. 5 and 6) in Section IV-B. These models include relationships between two different PIs and between an SLA and a PI, such as *hasPerformanceInfo*, *influence*, and *isInfluencedBy*. These relationships are not one-way, and some of them are transitive. For example, if *PerformanceInfo A influences PerformanceInfo B*, and *PerformanceInfo B influences PerformanceInfo C*, then we can say that *PerformanceInfo B isInfluencedBy PerformanceInfo A*, and *PerformanceInfo A influences PerformanceInfo C*. Manually finding and describing these hidden relationships for a huge number of PIs and SLAs requires considerable human resources and costs. Fortunately, we can find these unspecified relationships automatically by using SWRL rules. Three exemplary cases and corresponding SWRL rules follow.

1) Inferring Inverse Relationships between two PIs: The following SWRL rules complement the one-way *influence* relationship between two PIs by adding the other relationship.

- $PerformanceInfo(?x) \wedge PerformanceInfo(?y) \wedge influence(?x, ?y) \rightarrow isInfluencedBy(?y, ?x)$.
- $PerformanceInfo(?x) \wedge PerformanceInfo(?y) \wedge isInfluencedBy(?x, ?y) \rightarrow influenced(?y, ?x)$.

2) Inferring Transitive Relationships among three PIs: The following SWRL rules allow us to infer the transitive *influence* relationship among three hierarchically related PIs.

- $PerformanceInfo(?x) \wedge PerformanceInfo(?y) \wedge PerformanceInfo(?z) \wedge influence(?x, ?y) \wedge influence(?y, ?z) \rightarrow influence(?x, ?z)$.
- $PerformanceInfo(?x) \wedge PerformanceInfo(?y) \wedge PerformanceInfo(?z) \wedge isInfluencedBy(?x, ?y) \wedge isInfluenceBy(?y, ?z) \rightarrow isInfluencedBy(?x, ?z)$.

3) Inferring Inverse Relationships between an SLA and a PI: The following SWRL rules complement the one-way *hasPerformanceInfo* relationship between a Responsibility of an SLA and a PI by adding the other relationship.

- $Responsibility(?x) \wedge PerformanceInfo(?y) \wedge hasPerformanceInfo(?x, ?y) \rightarrow isPerformanceInfoOf(?y, ?x)$.
- $PerformanceInfo(?x) \wedge Responsibility(?y) \wedge isPerformanceInfoOf(?x, ?y) \rightarrow hasPerformanceInfo(?y, ?x)$.

D. Calculating PIs from other PIs

In addition to inferring incomplete relationships among PIs or between SLAs and PIs, we can calculate the value of a PI from other PIs using SWRL rules. For example, we can calculate the 'Packet Discard Rate' with the following equation:

- $PacketDiscardRate = (DiscardedInputPacket + DiscaredOutputPacket) / (InputPacket + OutputPacket)$.

The above equation can be transformed into the following SWRL rule:

- $InputPacket(?ip) \wedge OutputPacket(?op) \wedge DiscaredInputPacket(?dip) \wedge DiscaredOutputPacket(?dop) \wedge PacketDiscardRate(?pdr) \wedge hasValue(?ip, ?ipValue) \wedge hasValue(?op, ?opValue) \wedge hasValue(?dip, ?dipValue) \wedge hasValue(?dop, ?dopValue) \wedge swrlb : add(?diopValue, ?dipValue, ?dopValue) \wedge swrlb : add(?iopValue, ?ipValue, ?opValue) \wedge swrlb : divide(?result, ?diopValue, ?iopValue) \rightarrow hasValue(?pdr, ?result)$.

We can easily define and add more SWRL rules to calculate more PIs by exploiting the OSLAM SLA and PI ontology.

E. Detecting SLA Violations

We can check whether an SLA is satisfied or not by using SWRL rules. *Responsibility*, which is a major element of our SLA model, consists of the *Threshold*, *CompareFunction*, and *PerformanceInfo*. We can detect SLA violations by comparing *Threshold* and *PerformanceInfo* using the *CompareFunction*. SWRL rules that detect SLA violations with a 'greater than or equal to' *CompareFunction* follow as an exemplary instance:

- $SLA(?sla) \wedge hasResponsibility(?sla, ?res) \wedge hasPerformanceInfo(?res, ?indicator) \wedge hasThreshold(?sla, ?threshold) \wedge hasCompareFunction(?res, ?operator) \wedge$

$$
\begin{aligned}
&hasValue(?indicator, ?value) \wedge \\
&swrlb : equal(?operator, ”gt”) \wedge \\
&swrlb : lessThan(?value, ?threshold) \\
&\rightarrow isViolated(?sla, ”true”),
\end{aligned}
$$

- $SLA(?sla) \wedge hasResponsibility(?sla, ?res) \wedge$ $hasPerformanceInfo(?res, ?indicator) \wedge$ $hasThreshold(?sla, ?threshold) \wedge$ $hasCompareFunction(?res, ?operator) \wedge$ $hasValue(?indicator, ?value) \wedge$ $swrlb : equal(?operator, ”gt”) \wedge$ $swrlb : greaterThanOrEqual(?value, ?threshold)$ $\rightarrow isViolated(?sla, ”false”)$.

Other SWRL rules that detect SLA violations with a different *CompareFunction* can be defined similar to the above SWRL rules.

V. Implementation and Tests of OSLAM

This section describes how we implemented and tested OSLAM. A discussion about OSLAM and its test results follow.

A. Implementation

We implemented OSLAM using Protégé [6] and Jess [7] as shown in Fig. 7. Protégé is a free open-source platform that can be used for building ontologies using OWL, and Jess is a rule engine that has the ability to reason using knowledge that is supplied in the form of declarative rules. Jess can be integrated with Protégé by adding the Jess library into the following folder.

- *Protege_Folder*/plugins/edu.stanford.smi.protegex.owl/

To implement OSLAM, we built *PerformanceInfo* and *SLA* OWL classes based on the ontological models that are described in Section IV-B. After that, we added DPI, KPI, and KQI OWL classes shown in Fig. 3, Table I, II, and III. Finally, we added SWRL rules that are used for 1) inferring hidden relationships among PIs or between an SLA and a PI, 2) calculating PIs from other PIs, and 3) detecting SLA violations. A detailed explanation and several examples about these SWRL rules are described in Section IV-C, D, and E.

B. Tests

To show the possibility of OSLAM, we assumed a simple scenario: a customer and a service provider agreed on the SLA that is described in Table IV. Three types of *PerformanceInfos* including ‘Actual Frame Rate’, ‘Channel Zap Time’, and ‘Service Availability’ are considered, and each of them has their own *CompareFunction* and *Threshold*. *PerformanceInfo Values* are filled manually, because we do not have a real IPTV services platform. We defined SWRL rules, which are similar to those we described in Section IV-E, and executed them using the Jess rule engine to figure out whether or not the SLA has been satisfied. The results showed that ‘Actual Frame Rate’ and ‘Service Availability’ SLAs are violated, and ‘Channel Zap Time’ SLA is satisfied (Table IV).

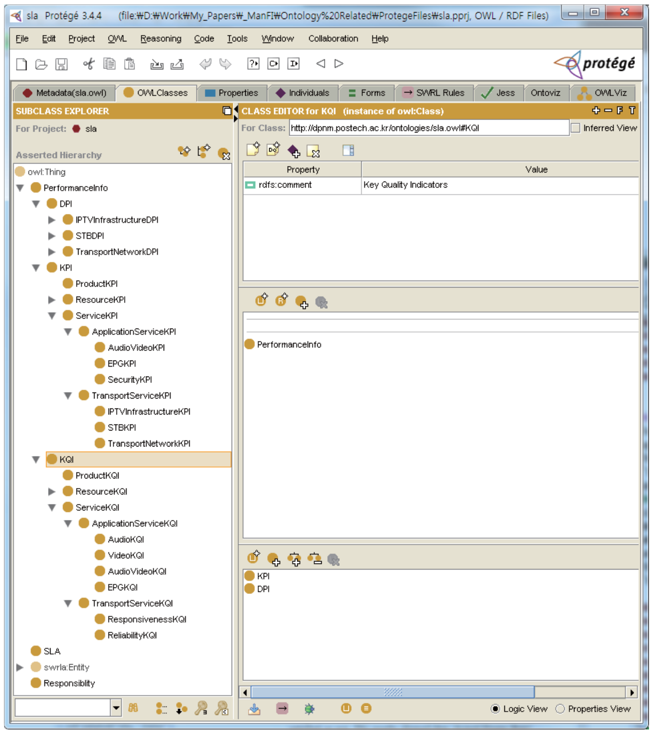

Fig. 7. Implementation of OLSAM using Protégé and Jess

TABLE IV
SLA Responsibilities of OSLAM Test Scenario and Violation Detection Results

Responsibility			PI	Violated
PerformanceInfo	**CompareFunction**	**Threshold**	**Value**	(result)
Actual Frame Rate	$>=$	24 fps	23 fps	**true**
Channel Zap Time	$<$	300 ms	290 ms	**false**
Service Availability	$>$	99%	98%	**true**

C. Discussion

We showed that the proposed OSLAM is an applicable framework for inferring hidden relationships among PIs and between a PI and an SLA, for calculating PIs from other PIs, and for detecting SLA violations using a simple scenario. However, there are several limitations and open areas:

- We analyzed almost every IPTV PI [9]–[11], [15]–[18] for constructing our IPTV PI hierarchy (Fig. 3, Table I, II, and III). However, it needs to be proven whether it reflects the actual customer and service provider’s criteria as they conceive them while using and providing IPTV services.
- We will extend our investigation into ways of calculating a PI from other PIs. One example of calculating a PI is given in Section IV-D, but many other SWRL rules have to be integrated with OSLAM, in order to cover all aspects of the PI hierarchy.
- The validation of OSLAM is done using a simple scenario, and related PI values are input manually, since we do not have a real IPTV services platform. We are planning to integrate OSLAM into a real IPTV service

platform for our future work to fortify the validation.

- Once an SLA violation occurs, then a service provider may have to pay some penalty to the corresponding customer. Hence, predicting SLA violations is more appropriate than just detecting them. This is left for our future work.
- We need to find a method that automatically changes configurations of devices in order to guarantee contracts between a customer and a service provider when OSLAM detects SLA violations. Achieving this is very hard and is our ultimate research goal.

Despite the above mentioned limitations, OSLAM has some advantages compared to existing SLA management work including [12]. Firstly, OSLAM considers DPIs while the others do not do so; in [12], the authors only considered KPIs and KQIs. Moreover, OSLAM is very flexible and easy to extend due to its use of ontologies. Finally, SWRL rules, which are used for reasoning in OSLAM, can be defined and modified dynamically without affecting other aspects of the code.

VI. Concluding Remarks

We have analyzed various IPTV PIs [9]–[11], [15]–[18] and suggested an IPTV PI hierarchy (Fig. 3, Table I, II, and III) by extending the DEN-ng information model [4], [5]. We have then proposed the OSLAM architecture, which is composed of 1) ontological models for representing PIs and SLAs and 2) SWRL rules that are used for inferring hidden relationships among PIs and SLAs, for calculating PIs from other PIs, and for detecting SLA violations. The proposed OSLAM architecture was implemented and tested using Protégé [6] and Jess [7]. The test results showed the possibility of OSLAM that it can be used for managing SLAs for IPTV services in an efficient and flexible way. In addition, OSLAM is not restricted only to IPTV services; it can be extended for managing various SLAs of other services including audio/video streaming, web browsing, and Internet games.

We are planning to improve OSLAM by addressing the points mentioned in Section V-C. We will extend OSLAM to cover other services that can be broadcast over an IPTV service platform, including Internet games. Finding a way to predict SLA violations is one of our major research goals for the future. Ultimately, we are hoping to provide a complete framework for managing SLAs by adding a method that automatically changes configurations of devices in order to always guarantee SLAs.

Acknowledgment

We would like to thank professor John Strassner of POSTECH for helping us with valuable comments and creative ideas.

References

[1] Y. Xiao, X. Du, J. Zhang, F. Hu, and S. Guizani, "Internet Protocol Television (IPTV): The killer application for the next-generation internet," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 126–134, Nov. 2007.

[2] Y. Ryu, E. Ko, H. Kang, G. Lee, and Y. Kim, "The web based sla system for customer quality assurance in providing IPTV services," in *Proc. 1st International Workshop on Broadband Convergence Networks (BcN '06)*, Vancouver, Canada, Apr. 7, 2006, pp. 1–10.

[3] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Member Submission 21, May 2004.

[4] J. Strassner. "Introduction to DEN-ng", tutorial for FP7 PanLab II project. [Online]. Available: http://www.autonomic-communication.org/

[5] A. Kwon, J.-M. Kang, S. Seo, S.-S. Kim, J. Y. Chung, J. Strassner, and J. W.-K. Hong, "The design of a quality of experience model for providing high quality multimedia services," in *Proc. 5th International Workshop on Modelling Autonomic Communication Environments (MACE '10)*, ser. Lecture Notes in Computer Science, vol. 6743, Niagara Falls, Canada, Oct. 28, 2006, pp. 24–36.

[6] The protégé ontology editor and knowledge acquisition system. [Online]. Available: http://protege.stanford.edu/

[7] Jess, the rule engine for the java platform. [Online]. Available: http://www.http://www.jessrules.com/

[8] *SLA Management Handbook: Volume 2 - Concepts and Principles*, TM Forum GB 917-2, Jul. 2005.

[9] *IPTV QoS/QoE metics*, ITU-T FG IPTV-C-0411, Jan. 2007.

[10] *Triple-play Services Quality of Experience (QoE) Requirements*, DSL Forum TR 126, Dec. 2006.

[11] *A framework for QoS metrics and measurements supporting IPTV services*, ATIS Std. 0 800 004, 2006.

[12] E. Toktar, G. Pujolle, E. Jamhour, M. C. Penna, and M. Fonseca, "An XML model for SLA definition with key indicators," in *Proc. 7th IEEE International Workshop on IP Operations and Management (IPOM '07)*, ser. Lecture Notes in Computer Science, vol. 4786. San José, USA: Springer, Oct. 31–Nov. 2, 2007, pp. 196–199.

[13] S. Latré, P. Simoens, B. D. Vleeschauwer, W. V. de Meerssche, F. D. Turck, B. Dhoedt, P. Demeester, S. V. D. Berghe, and E. G. de Lumley, "Design for a generic knowledge base for autonomic QoE optimization in multimedia access networks," in *Proc. 2nd IEEE Workshop on Autonomic Communications for Network Management (ACNM '08)*, Salvador, Brazil, Apr. 11, 2008, pp. 335–342.

[14] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns - Elements of Reusable Object-Oriented Software*. Boston, MA, USA: Addison-Wesly, 2000.

[15] *Quality of Experience Requirements for IPTV Services*, ITU-T REC. G.1080, Dec. 2008.

[16] *Best Practice: Video over IP SLA Management*, TM Forum GB 938, Jun. 2009.

[17] *Best Practice: Voice over IP SLA Management*, TM Forum GB 934, Jun. 2009.

[18] *Digital Video Broadcasting (DVB); Measurement guidelines for DVB systems*, ETSI TR 101 290, May 2001.

[19] *OWL Web Ontology Language*, W3C Recommendation 10, Feb. 2004.

[20] *OWL 2 Web Ontology Language*, W3C Recommendation 27, Oct. 2009.