SLA Foundation Template Library: Reusable-Component Repository for SLA

LUO Tie
National Lab of Switching Tech. and Telecom Networks
Beijing University of Posts and Telecommunications
Beijing, P.R. China
luotie@glorisoft.com

MENG Luoming
National Lab of Switching Tech. and Telecom Networks
Beijing University of Posts and Telecommunications
Beijing, P.R. China
lmmeng@bupt.edu.cn

Abstract – **Although aiming at facilitating service level management, the SLA is unfortunately not so promising for contracting SLAs is retarded by some challenges. This paper proposes to build a SLA Foundation Template Library (FTL) to overcome the obstacles from the perspective of SLA templates; The FTL provides a comprehensive set of well-defined and reusable components for rapid SLA development with higher quality. Furthermore, with the Object-Oriented approach and Unified Modeling Language applied, the FTL also achieves extensibility and flexibility as well as legible representation.**

Keywords – SLA; SLA Template; FTL; OO; UML

I. INTRODUCTION

As competition is spreading pervasively into every corner of the opening-up telecommunication market, as well as customers are becoming increasingly concerned about quality of service, service providers (SPs) have been focusing more and more seriously on service provision. Responding to the demand, the advent of Service Level Agreements (SLAs) for various services has enabled SPs to differentiate themselves from their competitors by demonstrating confidence in the quality of their services by providing a real-world, measurable SLA.

SLA is defined as a formal negotiated agreement, namely a contract (or part of one), which exists between the Service Provider and the Customer, designed to create a common understanding about services, priorities, responsibilities, etc. [1]. It specifies, usually in measurable terms, what services the SP will furnish and what penalties will assess if the SP fails to meet the established goals.

In the real world, however, problems are still besieging SPs and customers. First, since commonly recognized references are not available, SPs have to establish SLAs from scratch, potentially creating more workload hence dramatically slowing down the development of SLAs. Second, different SPs contract SLAs using a variety of perplexing jargons, frequently driving customers into confusion and causing misunderstanding among SPs. Third, deficient contracts with missing SLA parameters are not unusual, leading to reputation impairment, economic loss, or even lawsuits.

Much research on SLA has been done and focused mainly on four topics: (1) SLA management frameworks [1,3,4]; (2) SLA in-service monitoring and reporting [5,6,7,8]; (3) Quality of Service (QoS) control supporting SLA management [9,10]; (4) SLA parameter definition [2,12]. The research is important in a sense, but little systematic and effective work was devoted to solving the problems presented above.

This paper proposes to build a SLA Foundation Template Library (FTL) to address the issue. Like a repository storing a multitude of reusable components, the FTL provides a comprehensive set of well-defined foundation templates for developing SLAs not only rapidly but also with higher quality. By virtue of the Object-Oriented (OO) approach, it achieves extensibility and flexibility. Meanwhile, it offers us a legible graphical representation using Unified Modeling Language (UML),

Section II gives background information in order to explain our motivation of dealing with SLA-related problems from the perspective of SLA templates. Section III presents the main body of the solution. Then a case study is given in section IV to clarify how to apply the FTL to reality. Finally in section V, concluding remarks are made on the issue being discussed.

II. SLA TEMPLATE AND SLA

Our focusing upon SLA templates to address the problems harassing SLAs is closely associated to their relationship. As is shown in Fig. 1, a SLA references one or more SLA templates, each of which is dedicated to defining service level objectives for a commercial offer. Thus the production of a SLA template is an intrinsic part of service development, and SLA template defines the principal part of a SLA. Once a customer purchases an instance of a service, the template is used as the blueprint to form the actual SLA as part of the service contract between the customer and the SP.

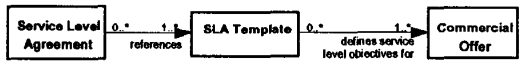


Figure 1. Relationship between SLA and SLA Template (adapted from [1])

A SLA template is a definition of a particular grade of service which can be offered with a SLA. The functionality of the template is to capture a set of service level objectives for a service along with details of consequences for not meeting the specified objectives, and potentially any exclusion conditions. Fig. 2 illustrates the components of a SLA template.

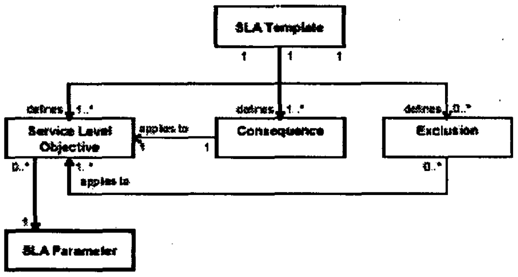


Figure 2. Components of a SLA Template (adapted from [1])

It is clear that three parts are defined in a SLA template:

- *Service Level Objective*: the core component representing the guaranteed grade of service to be offered. It is defined in terms of service metric, threshold values, and tolerances. Speaking more concretely, each objective stipulates a *SLA Parameter* associated with its

Supported by the China National Science Fund for Distinguished Young Scholars (60025104)
Supported by the Major Research Plan of National Natural Science Foundation of China (90104023, 90204002)
Supported by the General Project of National Natural Science Foundation of China (60202003)

threshold values.

- *Consequence*: describes what penalty will assess on the SP for failure to maintain and deliver the service as defined in the SLA. Violation of the stipulations may result in a variety of actions, including compensation, future discounts, rebates, etc.
- *Exclusion*: optionally specifies some conditions under which a violation should be considered as an exception and the penalty should be exempted.

The SLA Life Cycle introduced by TeleManagement Forum (TMF) also demonstrated the significance of SLA templates to a SLA. The life cycle is divided into five phases: product/service development, negotiation and sales, implementation, execution, and assessment. Preparing SLA templates is one of the main activities involved in the first phase, and the exit criterion of this phase is new service(s) with SLA templates.

All the above may have clarified our motivation of solving SLA-related problems via SLA templates. Suppose a set of well-defined templates, which can be reused directly or indirectly, meanwhile serving as an effective foundation for creation of new templates, are available, then contracting SLAs can be more smooth and produce better outcome. That is what triggers the birth of the SLA Foundation Template Library.

III. BUILDING SLA FOUNDATION TEMPLATE LIBRARY

With the OO concept built in, the process of building the SLA FTL bears much similarity to object modeling.

3.1 Classification, Abstraction and Organization

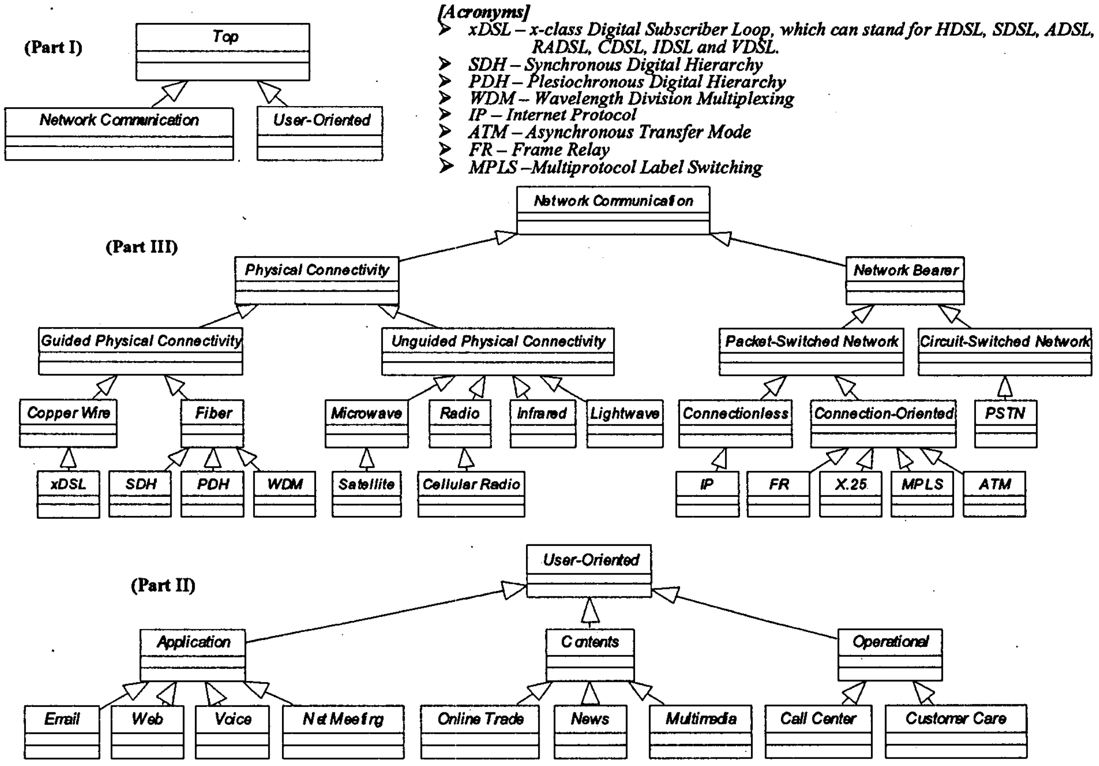
First of all, real-world services are classified into Service Elements (SEs) horizontally and vertically. A "service element" here is a building block of a service, possessing certain service characteristics and performing certain service functionality. "Horizontally" means layering a service in terms of the OSI Seven-Layer Reference Model. For example, an IP bearer network service may be supported by an ATM network over SDH connectivity with underlying WDM technology, thus we get four SEs: IP, ATM, SDH and WDM. "Vertically" means dividing a service bundle into individual entities.

Next, common service characteristics of these SEs are abstracted out and formed into new SEs. For instance, ATM, X.25 and MPLS are all connection-oriented networks, which in turn are packet-switched networks, thus Connection-Oriented SE and Packet-Switched SE can be abstracted out. Certainly, these SEs may be "abstract", i.e. no concrete equivalence exists in the real world.

Our intention of doing so is to maximize the reusability of Foundation Templates, for every SE will be directly mapped onto a foundation template in the library. By deriving from and combining together a subset of these foundation templates we can anticipate obtaining any SLA template being required.

The abstraction also determines a "is a" relationship among the foundation templates, which indicates an derivative connection and leads us to organize these templates onto a hierarchical inheritance tree. In addition, UML is adopted to denote this for legible representation.

Fig. 3 demonstrates the result. Limited by space, the whole tree is partitioned into three parts.



[Acronyms]
- *xDSL – x-class Digital Subscriber Loop, which can stand for HDSL, SDSL, ADSL, RADSL, CDSL, IDSL and VDSL.*
- *SDH – Synchronous Digital Hierarchy*
- *PDH – Plesiochronous Digital Hierarchy*
- *WDM – Wavelength Division Multiplexing*
- *IP – Internet Protocol*
- *ATM – Asynchronous Transfer Mode*
- *FR – Frame Relay*
- *MPLS – Multiprotocol Label Switching*

Figure 3. Hierarchical Overview of FTL (partitioned into 3 parts)

3.2 Encapsulation

Recall in Fig. 2 that a SLA template comprises three components: service level objectives, consequences, and exclusion conditions. The OO concept enlightened us to encapsulate them in a foundation template like attributes and operations in a class.

The service level objectives stipulate measurable *SLA Parameters* that a SP guarantees to its customers. Typically, SLA parameters include technical-related aspects such as network availability, or human-related aspects such as operator response. TMF had offered a SLA Parameter Framework shown in Fig. 4. (For more detail, refer to [1])

Service Perspective	**Parameter Category**		
	Technology -specific	**Service -specific**	**Technology/Service -independent**
Single User Instance (SAP-related)	e.g. physical interface details	e.g. service type or bundle	e.g. max. down-time of one event
Aggregated Requirements	e.g. monthly reported parameters	e.g. billing method (usage or time-based)	e.g. aggregate availability over all users

Figure 4. TMF's Framework for Parameter Categories

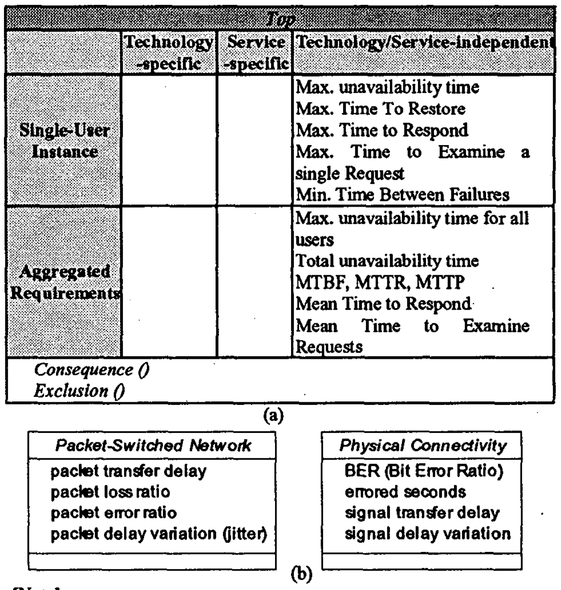
This framework is valuable in that it helps extract SLA parameters of a service and lower the possibility of missing parameters by partitioning a whole set into clearly defined categories. So it is adopted for attributes encapsulation.

With the functional behaviors abstracted into algorithms, the other two components, consequence and exclusion conditions, are encapsulated in two operations respectively. The pseudo code is expressed below.

```
PROCEDURE Consequence ()
BEGIN
    Initialize an empty table PPMT
    //PPMT stands for Parameter-Penalty Mapping Table;
    // each entry in this table is a parameter associated with
    // the corresponding penalty in case of violation
    for each parameter p defined in the template do
      if p is beyond its threshold
        then
        begin
            Create a new entry in the table
            Fill the entry with p and the corresponding
            penalty
            //penalties are determined by specific
            requirement
        end
    repeat
    return PPMT
END

PROCEDURE Exclusion (PPMT)
BEGIN
    for each entry e of the table PPMT do
        for each exclusion condition c do
        //conditions are determined by specific requirement
            if c matches the parameter in e
                and c matches actual facts
            then delete e from PPMT
        repeat
    repeat
END
```

Thus all the three components are encapsulated in a foundation template and samples are depicted in Fig. 5. To indicate, we made an alteration on UML class notation by integrating it with TMF's parameter framework.



Top			
	Technology -specific	Service -specific	Technology/Service-Independent
Single-User Instance			Max. unavailability time Max. Time To Restore Max. Time to Respond Max. Time to Examine a single Request Min. Time Between Failures
Aggregated Requirements			Max. unavailability time for all users Total unavailability time MTBF, MTTR, MTTP Mean Time to Respond Mean Time to Examine Requests
Consequence () *Exclusion ()*			

(a)

Packet-Switched Network	*Physical Connectivity*
packet transfer delay	BER (Bit Error Ratio)
packet loss ratio	errored seconds
packet error ratio	signal transfer delay
packet delay variation (jitter)	signal delay variation

(b)

[Note]

- *Part (a) gives a full representation of a foundation template, including an additional slashed frame serving as annotation only.*
- *Part (b) gives a brief representation of two foundation templates, only showing their technology-specific parameters with TMF's framework omitted. (This brief representative style will be used later in Fig. 7)*
- *"Top" template is the root of the FTL (as shown in Fig. 3); it is independent of any technology or service hence has no technology- or service-specific parameters.*

[Names and Terms]

- *Network Bearer: a type of telecommunication service that provides the capability for the transmission of signals between user-network interfaces (ITU-T Rec. I.112).*
- *Packet: a general term referring to any protocol data unit (PDU), including datagram, frame, cell, etc.*

[Acronyms]

- *MTBF - Mean Time Between Failures*
- *MTTR - Mean Time To Repair*
- *MTTP - Mean Time to Provision*

Figure 5. Samples of SLA Foundation Templates

Pay attention to an important distinction between foundation templates and actual templates. An actual template defines a grade of service using specific threshold values of parameters; a foundation template, contrarily, encapsulates parameters without specifying any value. For one thing, this is because the values depend largely on actual situation hence cannot be pre-determined. For another, leaving the parameters unassociated with concrete values increases reusability of the foundation templates because different SPs can define required thresholds of their own. By the way, this distinction also composes partially what the term "foundation" conveys.

Besides, the two operations of "*Top*", *Consequence* and *Exclusion*, are not depicted in Fig. 5(b) because child nodes in the tree automatically inherited them. However, since they are applied to service level objectives bound with specific value of SLA parameters, and their concrete content depends on the reality, they are "virtual methods" in C++ jargon hence must be overridden to define an actual SLA template, following the algorithms presented earlier.

3.3 Application of FTL

From the starting point, the main goal of FTL is to provide an effective reference and foundation for contracting SLAs. The process of applying FTL to establishing an actual SLA template can be divided into four steps, like a production line shown in Fig. 6.

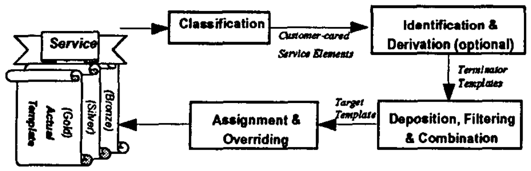


Figure 6 FTL's Production Line

(1) During the first step, classify the service to be provided into several Service Elements in accordance with the method described in section 3.1, but exclude the SEs whose quality is not cared by the customer.
(2) In the FTL, identify out the counterpart templates of these SEs. Try to locate the templates being closer to the leaf nodes of the tree for the lower tiers are more concrete and specific than the upper tiers owing to inheritance. Occasionally, the identified templates are abstract, or cannot completely cover the special characteristics of the SEs, then create new templates by deriving from them, and encapsulate the special attributes (SLA parameters) in these new templates obeying the method described in section 3.2.
(3) Treating the templates produced by identification or by derivation as "terminators", collects all encapsulated parameters along the path originating from node *Top* and ended with each terminator, and then put these parameters together into it. Following this "Deposition" of parameters; another activity called "Filtering" is performed to remove unnecessary parameters from each terminator according to the specific requirement. Then the resultant terminator*(s) are combined together for obtaining a single "target" template.
(4) Assign threshold value to each parameter in the target template for defining service level objectives, and override the two operations, *Consequence* and *Exclusion,* to declare penalties and exceptions. This task is suggested to be fulfilled in terms of differentiated levels, e.g., a superior one could be guaranteed higher service availability and higher amount of compensation in case of violation. In this way a series of actual SLA templates, perhaps constituted by a Gold Template, a Silver Template and a Bronze Template, will be available for the SLA to be provided with commercial offers.

Later we will give a case study to illustrate this production line.

3.4 Benefits

- The FTL enables rapid development of various SLAs by supplying a comprehensive set of reusable components, making a SP more competitive as service provision can be accomplished in a shorter time.
- The FTL facilitates the negotiation between SPs and customers since a SLA template provides a baseline for this activity and now it is easily accessible.
- By virtue of well-defined names and terms, the FTL minimizes misunderstanding and confusion among SPs and customers, meanwhile enhancing the clarity of SLAs.
- Thanks to the pre-encapsulated attributes, the FTL reduces the chance of missing parameters, improving the integrity of SLAs.
- Profiting from the OO methodology, the FTL merits itself with favorable extensibility and flexibility; new templates can be easily added to the library by derivation, as well as new attributes and operations can be easily inserted and shared by child nodes.
- Applying UML denotation, the graphical representation makes the FTL legible and readily understandable.
- Predictably, the FTL's production line can be partially automated by using Computer Aided Design (CAD) technology, thus more convenience can be achieved and productivity can be promoted.

IV. Case Study: Broadband Access Network Service Management

This case study is based on a project named Metar-BANS (Broadband Access Network Service management) that accomplishes broadband access network service management supporting SLA management. Currently the system is supervising three types of access networks: ADSL, Ethernet and HomePNA. In this paper, only the ADSL section is cited.

The two parties of the SLA are an ADSL Network Operator as the SP, and its customer who may be an individual end user or a business organization. In the former case, the customer may care single-user related SLA parameters only; in the latter case, however, aggregated metrics are important as well.

Following the product line, firstly the Operator classified the access service into two SEs: the *ADSL Connectivity* SE enabling the customer to connect to the core network, and the *IP Bearer* SE enabling the customer to send and receive traffic via Internet. It is necessary to point out that application such as Web and Email are out of the scope unless the SP is a ISP.

In the second step, two foundation templates in the FTL were identified out: *xDSL Connectivity* and *IP Bearer.* Since the required *ADSL Connectivity* SE had some special service characteristics not covered by *xDSL*, a new template named *ADSL Connectivity* was created by deriving from *xDSL Connectivity*. Fig. 7 depicts this extension of the FTL.

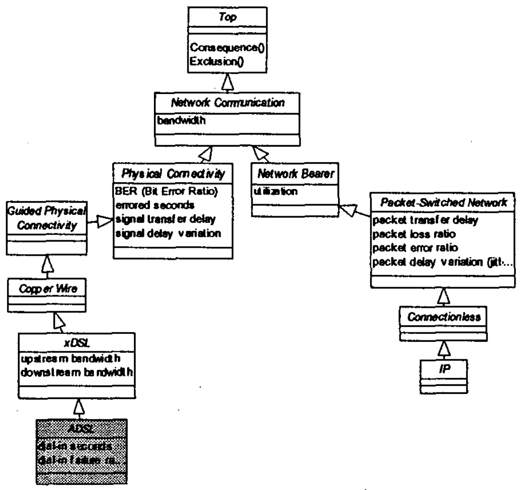


Figure 7. The 2nd Step of the Production Line: templates *xDSL* and *IP* were identified out while *ADSL* was created by derivation.

Thirdly, deposition and filtering were performed successively. Some parameters such as "signal transmission delay" and "signal delay variation" were filtered out for being not cared by the customer. Then the two "terminators*", *ADSL** and *IP**, were combined as a single *Target* template. See Fig. 8 for detail.

*ADSL**	
Max. BER	Max. unavailability time
Max. Errored Seconds	Max. Time To Restore
Max. & Min. Upstream Bandwidth	Max. Time to Respond
Max. & Min. Downstream Bandwidth	Max. Time to Examine a single Request
Max. Dial-in Seconds	Min. Time Between Failures
Max. Dial-in Failure Ratio	
Mean BER	Max. unavailability time for all users
Mean Errored Seconds	Total unavailability time
Mean Upstream Bandwidth	MTBF, MTTR, MTTP
Mean Downstream Bandwidth	Mean Time to Respond
Mean Dial-in Seconds	Mean Time to Examine Requests
Mean Dial-in Failure Ratio	
Consequence ()	
Exclusion ()	

*IP**		
Max. Packet Transfer Delay	Max. Utilization	Max. unavailability time
Max. Packet Loss Ratio		Max. Time To Restore
Max. Packet Error Ratio		Max. Time to Respond
Max. Packet Delay Variation		Max. Time to Examine a single Request
		Min. Time Between Failures
Mean Packet Transfer Delay	Mean Utilization	Max. unavailability time for all users
Mean Packet Loss Ratio		Total unavailability time
Mean Packet Error Ratio		MTBF, MTTR, MTTP
Mean Packet Delay Variation		Mean Time to Respond
		Mean Time to Examine Requests
Consequence ()		
Exclusion ()		

Target		
Max. BER	Max. Utilization	Max. unavailability time
Max. Errored Seconds		Max. Time To Restore
Max. & Min. Upstream Bandwidth		Max. Time to Respond
Max. & Min. Downstream Bandwidth		Max. Time to Examine a single Request
Max. Dial-in Seconds		Min. Time Between Failures
Max. Dial-in Failure Ratio		
Max. Packet Transfer Delay		
Max. Packet Loss Ratio		
Max. Packet Error Ratio		
Max. Packet Delay Variation		
Mean BER	Mean Utilization	Max. unavailability time for all users
Mean Errored Seconds		Total unavailability time
Mean Upstream Bandwidth		MTBF, MTTR, MTTP
Mean Downstream Bandwidth		Mean Time to Respond
Mean Dial-in Seconds		Mean Time to Examine Requests
Mean Dial-in Failure Ratio		
Mean Packet Transfer Delay		
Mean Packet Loss Ratio		
Mean Packet Error Ratio		
Mean Packet Delay Variation		
Consequence ()		
Exclusion ()		

[Note] The slashed frame is omitted in this figure.

Figure 8. The 3rd Step of the Production Line: *ADSL** and *IP** templates were obtained via Deposition and Filtering; the *Target* template combined them together.

Finally in the *Target* template, differentiated levels of threshold values were assigned to the SLA parameters, as well as corresponding penalties and exceptions were declared by overriding the two operations, producing the actual SLA templates in three final forms: Basic, Pro, and VIP.

V. CONCLUSION AND FUTURE WORK

This paper proposes to build a SLA Foundation Template Library to solve problems harassing SLAs. Like a repository storing a multitude of reusable components, the FTL provides a comprehensive set of foundation templates for developing SLAs rapidly and with higher quality. Applying the OO approach and UML denotation, it achieves extensibility and flexibility as well as legible representation.

We have already presented a prototype of the FTL for illustration; work still remains to consummate the library by supplementing necessary foundation templates, complementing appropriate SLA parameters, and polishing names and terms. This could be done by different SPs, but standardization organizations are expected to participate in and synthesize the individual accomplishments into an industrial standard for better recognition and wider application.

REFERENCES

[1] TMF GB917v1.5, "SLA Management Handbook", Jun. 2001.

[2] TMF TMF701v2.0, "Performance Reporting Concepts & Definitions", Nov. 2001.

[3] E. Marilly, O. Martinot, H. Papini, and D. Goderis, "Service level agreements: a main challenge for next generation networks", 2002.

[4] E. Bouillet, D. Mitra and K. G. Ramakrishnan, "The structure and management of service level agreements in networks", IEEE *J. On Selected Areas In Communications*, vol. 20, no. 4, pp. 691-699, May 2002.

[5] M. C. Chan, Y. Lin, X. Wang, "A scalable monitoring approach for service level agreements validation", *ICNP 2000*, pp. 37-48.

[6] F. De Turck, S. Vanhastel, F. Vandermeulen, P. Demeester, "Design and implementation of a generic connection management and service level agreement monitoring platform supporting the virtual private network service", *Proceedings of the 7thIFIP/IEEE Int. Symp. on Integrated Network Management (IM2001)*, pp. 153-166.

[7] F. De Turck, S. Vanhastel, P. Backx, B. Duysburgh, P. Demeester, "Design of a generic architecture for service management and monitoring of service level agreements through distributed intelligent agents", *IN2001*, pp. 50-57

[8] E. C. Kim, J. G. Song, C. S. Hong, "An integrated CNM architecture for multi-layer networks with simple SLA monitoring and reporting mechanism", *NOMS 2000*, pp. 993 -994.

[9] J. Park, J. Baek, and J. W. Hong, "Management of service level agreements for multimedia Internet service using a utility model", *IEEE Comm. Mag.*, pp.100-106, May 2001.

[10] M. S. Dang, R. Garg, R. S. Randhawa, H. Saran, "A SLA framework for QoS provisioning and dynamic capacity allocation", *IWQoS 2002*, May 2002.

[11] A. S. Tanenbaum, *Computer Networks*, 3rd ed., Prentice Hall, 1996.

[12] ITU-T Rec. E.800, "Terms And Definitions Related To Quality Of Service And Network Performance Including Dependability", 1994.