SLA Object and SLA Process Modelling using WSLA and BPM Notations

Towards defining a Generic SLA Orchestrator Framework

Bukhary Ikhwan Ismail, Nurliyana Muty, Mohammad Fairus Khalid, Ong Hong Hoe
Advanced Computing Lab
MIMOS Berhad
Kuala Lumpur, Malaysia
ikhwan.ismail@mimos.my, nurliyana.muty@mimos.my, fairus.khalid@mimos.my, hh.ong@mimos.my

***Abstract*— SLA monitoring and enforcement ensure service dependability. In an ever challenging market, service providers strive to compete in the IT business by offering new innovative services. Fast to market and reliable Quality of Service goes hand in hand in determining the market leader. To address these challenges, a flexible and comprehensive tool is required to automate the provisioning of SLA and Service Level Management processes. Based on our investigation there is no single framework, which is flexible enough to orchestrate SLA provisioning for multiple services. To address this problem, we first describe the SLA object concepts and redefine the SLA management processes. It introduces an extended WSLA model and tiering mechanism, to promote modularity and reusability of the SLA model. We propose to model the SLA management process using the Business Process Management Notation to simplify and reduce the planning, design, and implementation effort in defining the SLA offerings. Both of these models act as guidelines to attain a generic SLA framework towards any service adaptation.**

Keywords-Service Level Agreement; Service Level Management; WSLA; Business Process Management; Dependable Service

I. Introduction

Service is a means of delivering value or functions, which the provider offers, to the subscribers. Business applications, software functions and infrastructural services are some examples. Subscribers require a certain level of service delivery guarantee and the Provider needs to ensure the dependability aspect of service fulfilment. In IT Service Management (ITSM), Service Level Agreement (SLA) provides a well-known standard in ensuring service delivery guarantee. The SLA captures the requirements and expectations of service guarantee where both parties agree. The SLA contract formalizes the requirements of; and not limited to Quality of Service (QoS) parameters; responsibilities of both parties; warranties or actions to be taken; compensations; guarantee coverage and service limitations or exclusion clauses [1].

Service Level Management (SLM) is a discipline of proactive methodology and procedures used to ensure adequate levels of service are delivered in accordance with business priorities at an acceptable cost [2]. Most SLA management strategy considers two common phases; (1) the *negotiation of the contract* – the formalisation of objectives, action guarantee and violations and (2) *the monitoring of its fulfilment in real-time* –the proof of service delivery [3].

In this paper, we analyse several prominent research works in the areas of service delivery governance from the provider's perspective. We conclude that 1) *Most SLA frameworks implement on a specific service or specific service domain.* Interpretation of QoS attributes such as availability, reliability, *or* performance is unique to the service domain. For example, storage availability versus web services availability differs greatly. It affects the way the SLA is calculated and action logic to perform. To adapt the same framework for new implementation would be impossible. 2) *The SLA implementation is embedded within service infrastructure* - For example, the SLA rules or logics are buried implicitly in the application code [4]. It would be impossible to utilize the same framework for new implementation. 3) *The dynamics of SLM* - Not all of the process activities e.g., SLA negotiation, template definition and compensation to name a few, are required in order to deliver service level guarantees. Each research project usually defines a fix set of SLM process while in actual implementation, not all of the processes are required. To adapt the same predefined sets of SLA management process for another service is unsuitable.

There are four main contributions of this paper. 1) A survey of multiple SLA research projects, which deduced the importance of SLA concepts, objects and SLM, processes surrounding the SLA management and the governance of service delivery guarantees. 2) A proposed new form of SLA and SLM process meta-model to illustrate the interactions and usage of SLA data objects with the SLM process activities. 3) An extension of existing Web Service Level Agreement (WSLA) model to support additional SLM processes that enable the separation of the *SLA service information*, the *logics* within the SLA, the *implementation* to enforce the SLA and *runtime* information of the model. Lastly, 4) an introduction of a modelling technique using Business Process Management to model the SLM process.

This paper is organized as follows. Section II summarizes the SLM processes. Section III, discusses the requirements and analysis towards the importance of generic SLA framework. Section IV, presents related works, Section V, discusses the SLA concepts and SLM meta-model. Section VI and VII describes the SLA Object and Process Model respectively. We conclude our work in the last section.

II. State of the art service level management survey

In order to define a generic SLM framework for service delivery guarantee, it is imperative to deduce and capture important phases and process activities of SLM. Here, we present our survey derived from multiple related SLA management research works. The investigation consists of 24 SLA projects survey by the EU commission, 2 known SLA frameworks; WSLA & WS-Agreement, 2 ITSM standards; ITILv3 and CoBiT Delivery & Support, 10 individual SLA projects and 3 related SLA products in order to broaden our investigation and the applicability of SLM. Full reference list for each process activity is in TABLE I. We identified and generalized the process activities, which from our point of view, is adequately generic and sufficient for any service adaptation.

There are 4 main stages in the SLM lifecycle; 1*) Requirement Specification* – the requirement stage for service and SLA input; 2) *Instantiation and Management* – negotiation of SLA and instantiation of service and SLA; 3) *Enforcement – to* monitor and assure QoS during service runtime*;* and 4) *Conclusion* –handles the closure of service and any reimbursement of credits due to breaches in the SLA contract. There are similar works in defining these stages. In [5] defines the 5 stages while [6] shows 4 stages where these stages are categorized by its processes.

A. Requirement Specification Stage

1) SLA Template Definition - is the process of requirement capturing of the SLA contract where Service Level Objective (SLO), guaranteed state, compensation, exclusion clauses of service are defined. In ITSM practices, it includes the requirements of Operating Level Agreement (OLA), defining the service support organization structure, support contract and Underpinning Contract (UC) [7]. Output of this activity is to translate the SLA document into a machine readable format e.g., XML, ontology or rules.

B. Instantiation & Management Stage

Here we define 4 processes:-

1) *Service & SLA Offering*– is the process which advertise the *service* e.g., Virtual Server; CPU core, memory, etc; and the *service SLA attributes* e.g., QoS, performance, availability or exclusion clauses prior to the service subscription process.

2) *Negotiation* - handles the negotiation process of SLA requirements between the provider (a system providing the service) and subscriber (a system or customer). The negotiation parameter is usually limited to the qualities, which the provider is able to satisfy. The output of this activity is an agreed SLA by both parties.

3) *Mapping and translation* – In an SLO, the service parameters to be guaranteed can be in high level description, which is close to business or application requirement language. This process bridges the gap between both. The process translates the *high-level metrics* i.e., *application response time* and maps it to the *low-level resource parameter*; i.e., `transaction per-second, transaction response time, etc.` Translation includes both *functional requirements* e.g., performance, capacity and *non-functional requirements* e.g., availability, redundancy, security to be translated and mapped.

4) *Service Provisioning* – creates the actual service instance and SLA in an enforced state.

C. Enforcement Stage

Enforcement is the most important stage to ensure service delivery guarantee and service dependability during service runtime [5][8][9]. There are 5 important processes:-

1) *Monitoring* - obtains the infrastructure or application performance metrics which acts as input for the SLA's *violation detection process.*

2) *Violation detection* –monitors reactively the SLO parameters for any potential violation breach.

3) *Violation prevention* – to detect a violation before it occurs where a proactive or predictive mechanism might be use.

4) *Violation corrective action*– corrective action which is triggered by a *violation detection* or *violation prevention* process in order to repair or reduce the onset of the violation.

5) *Violation Escalation* – to escalate *error, fault* or *failure* information to system administrator where manual corrective actions can be executed.

D. Conclusion Stage

This stage consists of 5 main processes:-

1) *Termination of service & SLA* – handles the 1) proper closure due to an end of subscription; or 2) termination of service caused by a breach in the service contract.

2) *Accounting & Billing* – handles the charging mechanism to the subscriber.

3) *Resolution* – to provide remedial action or compensation of breached SLA to the subscriber.

4) *SLA template archiving* – to store previous SLA documents and its related information for future references.

5) *SLA Review* – continous review of SLA and its performance for manual SLA management as defined in ITIL v3 [7] and CoBiT [10].

We have listed out necessary SLM processes, which are deem important for the adaptation of our generic SLM framework. We however, did not list out other processes, which are too unique for a specific service implementation or explain further the derivatives of each process i.e., monitoring; dynamic monitoring, scalable monitoring or negotiation; re-negotiation or auto-negotiation processes [8].

III. An analysis towards a generic SLA framework

To justify our argument in requiring a generic SLA framework, we provide concrete analysis in the next few sections which discusses the dynamics of the SLM management processes, SLA offering and deployment variations.

A. The Variation and Dynamics of the SLM Process

TABLE I. VARIATION OF SLM PROCESSES IN SLA IMPLEMENTATION

	EU Research	WS-Agreement	WSLA	CoBiT	ITIL v3	e-business	For SOA	A logic based	Improving Ent.	SLA@SOI	Multi-level	layered cloud	IRMOS	Cloud4SOA	OPTIMIS	WSRR	Uptime	ServiceNow
	[8]	[11]	[5]	[10]	[7]	[12]	[13]	[4]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]
Templ. Def.	X	X	X	X	X	X	X	X	X	X	X				X	X		
Offering											X							
Negotiation	X		X	X	X		X			X	X	X	X	X	X			
Map & Trans	X								X	X			X					
Serv. Provision	X	X	X	X	X			X		X	X		X	X	X			
Monitoring	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Vio. Detection	X		X			X		X		X			X	X		X	X	X
Vio. Prevention	X											X						
Vio. Corrective	X		X			X				X			X	X		X	X	X
Vio. Escalation				X	X									X			X	X
Termination			X															
Acc. & Billing	X																	
Resolution	X													X				
Archiving											X							
Review				X	X													

From the investigation done on the SLM management activities in Section II, we created a comparison matrix TABLE I, to show the variation of management process implemented in each project. For each project, we marked the processes that are being adapted. We concluded that not all of the process or activities are compulsory. For example, *negotiation*, *mapping and translation*, *accounting*, *billing*, *resolution and compensation* are some of the optional components. While *SLA template definition*, *monitoring* and *violation detection* can be considered as the *core* must-have component by deducing the total number of adaptation of the processes.

B. Types of SLA Offering

SLA offering is a service delivery guarantee commitment that the provider is willing to offer to the subscriber. The process takes place before service is instantiated. The design of the SLA offering is based on pricing strategy, customer segmentation profiling or other business factors. We conclude there are 4 types of SLA offering categories, which can affect overall implementation of the SLM process.

1) Common to all and *2) Template based SLA;* are widely used type of SLA contracts. Both types are considered as a non-negotiable contract, whereas the SLA for a particular service is fixed and applies to all or a particular segments of users. When the SLA breached, the provider will compensate by providing service credits into the customer's account or provides other forms of remedial compensation [24]. This type of SLA does not require any *negotiation* process or *mapping and translation* of monitoring metrics as the service delivery is static i.e., non-negotiable service guarantees are common to all users. It may however focus on *violation prevention* or *self-healing* capabilities in order to reduce the violation-breached effects.

3) Negotiable template based SLAs and *4) custom based SLAs* provide flexibility to the subscriber to tune the requirements to match the intended workload. Requirements example are cost, pricing, performance, availability or other attributes [16][17]. Both SLA categories require a *negotiation* process or its derivatives e.g., *auto-renegotiation* [18], *dynamic negotiation* [19][20] or *negotiation across multiple service layers* [15] i.e., between subscribers and service providers or between service providers and another service provider in a multi-level service deployment setup. It may require *mapping and translation* of high level metrics to low level metric [18][15] as the definition of SLA requirements are open for interpretation.

C. Service Deployment Variation; IaaS as a Case Study

IaaS provides infrastructural services such as compute, storage and network. Each IaaS Infrastructure deployment is unique and the deployment depends on the service functional requirements, cost, hardware, software or technology choices. It makes *monitoring* efforts; the core component of SLA system variable. We illustrate the complexity of IaaS deployment in TABLE II, which shows possible combinations of storage technology, transport, medium and backend options to host VM's virtual disk where multiple combinations can be constructed for a single deployment.

TABLE II. VIRTUAL MACHINE'S STORAGE DEPLOYMENT VARIATION

Disk	Storage Transport	Storage medium	Storage backend
File qcow2, raw, vmdk	Local	Local	Local Disk
	Over network	NFS, OCFS, GFS	JBOD, SAN
		CephFS,Gluster	Distributed Storage
Block	Over network	CEPH RBD	
	Local	LVM	JBOD, SAN
	Over network	iSCSI, AoE, FC	

We further emphasize the complexity with the example of a virtual disk deployment variation in Figure 1, where a VM can run in shared storage in *setup A* or local storage in *setup B.* Both setups require the VM Availability SLA parameter to be guaranteed but with different metrics to monitor in *Setup A*; needs to monitor host, switches, NFS storage as compared to *Setup B*, which runs on a local disk.

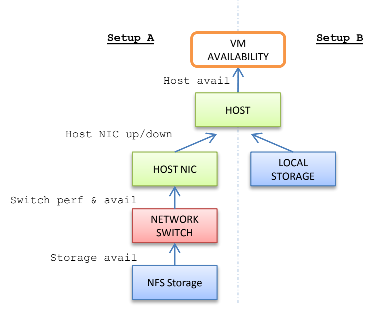

Figure 1. Deployment variation to host VM's Virtual Disk

The deployment setup will dictate the choice of performance metrics to be monitored and available corrective actions to be used. This will influence the SLA's *violation detection, violation prevention,* and *violation corrective action* components. In short, the service deployment will dictate the SLA *enforcement* process activities.

IaaS provides a good case study with multiple service types and multiple service deployment scenarios in order to test the concept of generic, flexible and compose-able SLA orchestrator framework.

Our investigation shows that SLM activities or processes are neither static nor absolute. It depends on multiple factors such as SLA offering types, business requirements or certain technical difficulties, which makes it harder to implement some of the components. A generic and flexible framework is needed in order to orchestrate the SLA offering by implementing all, or certain of the components towards service levels guarantee delivery.

IV. Related Work

Here, we describe two aspects of related work, the modelling of SLA and the modelling of SLM activities.

A. *Data Modelling*

WSLA is a framework which specifies SLAs for Web-Services implementation [25]. WS-Agreement – an OGF standard for the creation and specification of SLA [13]. SLAng model is the SLA for a spectrum of Internet Services and provides its own internal language specification [26] and lastly, a generic SLA semantic model for e-business outsourcing contract is taken as part of the survey [12]. All the above projects use UML and Object Constraint Logic (OCL) to model the SLA objects and its relationships and present it into XML schema format. In our view, this is the most suitable representation of SLA document.

Paschke, Dietrich & Kuhla [4], captures selected portion of the SLA agreement using rules and logical based Knowledge Representation concepts. Rules can be used in the function to evaluate conditions or violations in the expression for Action Guarantee. From our investigation, rule based only represents a portion of the SLA information. Service information relationships, between objects or descriptive information of the SLAs rely on other forms of representation.

Ontology based SLA model is another form of representation [14] . It captures business service performance requirements key indicators such as KPI (Key Performance Indicator and QKI (Quality Key Indicators) to define the SLA parameter. In our view, to support modularity and multiple service adaptation, domain specific knowledge should not be modelled within the same model. This hinders the generality of service implementation. For example, "availability" for software and hardware is defined differently.

Based on our literature review, we believe it is most suitable to use the UML object diagram and OCL to represent the SLA model. UML diagrams are a well-accepted software analysis and design tool. It is simple and captures a high level of information of the overall SLA document. It is sufficient to represent concepts and relationship information such as cardinality, aggregation and inheritance of those concepts.

B. *SLM Modelling*

There are several projects, which try to model the process flow of SLM using business process modelling. Correia & Abreu [27] proposed, a Model Driven Engineering (MDE) approach in modelling SLM activities for IT service *SLA specification* and *processes.* It uses the BPMN notation to create a process meta-model.

Corriea & Amaral [28] proposed a domain specific SLA Language Specification and Monitoring (SLALOM). It focuses on mapping of SLA to BPM notation for SLA implementation on ITIL standards. ServiceNow [23], an enterprise monitoring tool, uses a simplified form of BPMN notation for designing escalation, notification to user and scripting to automate tasks.

These are among several projects, which have the same SLM modelling objectives as ours. It provides an insight on the application and applicability of process modelling in the SLA management domain.

V. SLA Object & SLM Process Meta model

In this section, we present our deduction of SLA & SLM meta-models. We map each of SLA objects with the processes and activities of SLM to show its interactions. Figure 2 identifies the *SLA data object or concept*; presented in rectangular and *functional process;* represented in rounded rectangular. We show that, 1) *compulsory*; denoted as **(C)** – a must have activity or SLA object 2) *optional*; denoted as **(O)** - categorized as supporting activities or objects where the provider may adapt or drop.

Our SLA model extends the WSLA with additional concepts denoted as (N). In the existing WSLA model, there is no process, which uses the *exclusion* and *coverage clauses*. Both of these are being utilized by the *resolution* process. *Pricing Information* object provides information to calculate *billing and accounting*. We included the *violation* and *service action repair* object to support the *violation detection, prevention, corrective action* processes.

In the implementation of the SLA system, it is common to combine the business process flow, business logic and data model in single implementation. For example, the SLA contract rules are buried implicitly in the application code [4]. It is therefore hard to maintain the SLA when a new requirement is introduced and may require extensive re-implementation efforts. We opted the concept of addressing system complexity [29] by adapting common techniques such as abstraction and modularity.

VI. SLA Document – Data Modelling

SLA modelling is the process of synthesizing information within the SLA document and translates it into a model for the consumption of *Information Systems*. From a high-level perspective, modelling captures the service definition, objectives and guaranteed actions.

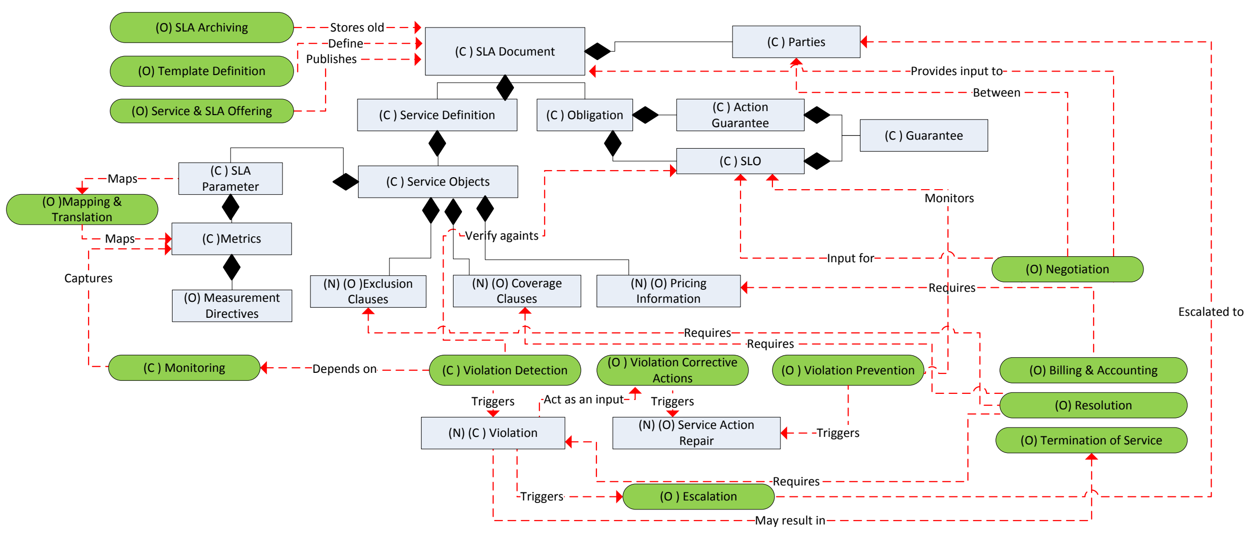

Figure 2. SLA Objects and SLM Process Meta-model

To model the SLA document requires domain experts in depth understanding of the ITSM practices. Thus instead of creating a new model, we opted an existing one. We chose an SLA model that is generic enough to be able to describe multiple services implementation and a model that is non-specific to any service implementation. We opted the WSLA as our case study.

The WSLA describe an SLA framework for web-service implementation. It is generic enough to be adapted by multiple service categories, such as service management, networks or business application. The data model captures the crucial attributes, which is used to measure and monitor the QoS parameters, violation detection, repair actions logics, and escalation mechanism to authorized parties [25]. The WSLA framework provides a layout of a SLA document schema and provides multiple custom type object definitions. The WSLA consists of 3 major parts: 1) *Service Definition* – captures the service definition, the SLA parameter to guarantee and metrics to monitor 2) *Obligation* – captures the SLO and action guarantees towards the state or the violation, and 3) *Parties involved* - between the signatory or supporting party that supports SLM.

To use and implement the WSLA introduces a new set of challenges in the creation, modification and management of the SLA template. WSLA does not separate the 1) *service objects definition description and relationship*; with 2) *rules, logics and algorithm* and 3) *runtime* and *implementation information*. Thus, the ***reusability*** – using the *parts-of* the SLA model and ***portability*** – to be able to export or import parts-of the SLA model is not possible. For example, in WSLA SLAParameter it captures the “how to measure”, “how to aggregate” into a *compositeMetrics* and includes, which “party” is responsible to provide the metric value. From our point of view, this information is only unique for a particular service implementation and mixing the modelling information with the implementation information will hinder reusability & portability.

Based on these challenges we segmentize the SLA model and decouple with clear natural separation boundary by adapting the Separation of Concern, a design technique to achieve modularity as being implemented in [4][15][16][30]. Modularity promotes reusability and portability parts-of the SLA model i.e., by allowing the provider to reuse or imports parts of the SLA model for new service implementation. This is to satisfy and support multiple implementation of the service based on the challenges defined in *Section I*.

We suggest 4 separations of the SLA model:-

1) **WHAT-IS** *the SLA-Model* ***(SLA-M)*** *– information about the* `ServiceDefinition`*,* `ServiceObject`*,* `SLAParameter` *and its relationship;*

2) **WHAT-IF** the SLA-Logics **(SLA-L)** – the logics and rules of `SLO`, `actionGuarantee`, and *monitoring calculation*. It may consist of `measurementDirectives` expression which calculates composite metrics i.e., high-level metrics such as availability may consist of multiple low-level metrics aggregation. Both can be defined using a combination of `function` and `expression`;

3) **HOW-TO** the SLA Implementation (**SLA-I**) – the service implementation information such as *Obligation-action*, monitoring GET `metricURI` endpoints, action SET `actionURI` endpoints together with;

4) **THE RUNTIME** **(SLA-R)** - the SLA information, which is populated during service and SLA runtime. For example, signatory parties information, and possibly the QoS parameters.

The SLA-M can be considered as a SLA service catalogue, which defines (`ServiceDefinition`) and quality attributes to guarantee (`serviceObjects`) and SLA Parameter to monitor (`SLAParameter`) the provider guarantees. It is considered as a non-volatile SLA data model where it is rarely changed unless the provider can guarantee a new `serviceObject`. We explain further based on, extends the SLA-M `ServiceObject` type. `ServiceDefinition` is a Virtual Machine service, where the `serviceObjects` may consist of *VMAvailability*, *VMBootTime* and *VMNetworkPerformance*. For each of the `ServiceObject`, the provider has agreed to guarantee these qualities. By guarantying these qualities, we assume the providers have tested, able to collect the monitoring metrics and probably have a corrective action plan in hand to maintain the SLA for these qualities. If new `ServiceObject` is introduced, then a new sets of `SLAParameter`, `metrics`, `SLO` and `ActionGuarantee` needs to be laid out.

TABLE III. TIERING WSLA EXAMPLE

Type	**WSLA Elements**	**Examples**
SLA-R	**ServiceProvider**	
	Name	MIMOS cloud service
	Role	Provider
	Contact	Kuala Lumpur, MY
	Action	n/a
SLA-D	**ServiceDefinition**	
	Name	Virtual Machine Service
	Description	Virtualize infrastructure
SLA-D	**ServiceObject**	
	Name	VMAvailability
	Desc	Virtualization Availability
	Schedule	24X7 Coverage
	Trigger	On Service Hooks
	Constant	99.99%
SLA-D	**SLAParameter**	
	Name	VMAvailabilityUptimeRatio
	Unit	%
	Type	FLOAT
	Category	Availability
SLA-I	**Metric 1**	
	Name/Type/Unit/Function	VMStatus / INT / Boolean /N/A
SLA-I	**Metric 2**	
	Name/Type/Unit/	VMUptimeRatio / LONG / % /
SLA-L	Function	If(VMStatus==0) Then VMUptimeRatio-1%
SLA-R	SLO	
	Name	VMUptimeSLO
	ValidityStart	01:01:00 2014/06/01
	ValidityEnd	When Subscription Ends
	Expression	PREDICATE= GREATERTHAN
		SLAParameter >Name =
SLA-L		VMAvailabilityUptimeRatio, Value=0.95

The SLA-I is the require information in the *implementation* phases. For example, GET – to fetch the monitoring data information or service status, and SET – to react based on the violation status. All of the above depends on the deployment setup landscape. This information is rarely changed unless the same SLA-I model is applied to another service deployment setup. The SLA-I captures time or interval information such as `schedule` starts and stops, `period` e.g., *"SLO is valid only on working days"*, `interval` of monitoring data to be fetch or send notification response in intervals.

The SLA-L captures the rules, and logic. In WSLA, it can be in a form of evaluation `expression` or `function` in SLO, or ActionGuarantee *e.g., "live migrate the VM if underlying HW components fail"*. A sample SLA-L is SLO expression, for example *VMAvailabilityUptimeRatio >= 0.95* as depicted in table above. SLA-L can be depicted in 2 forms; 1) monitoring logics or violation detection; 2) violation prevention or violation correction action logic. The logic can change depending on the requirements, towards the service. For example, changing how the availability formula calculation or modify the action to perform once a violation is detected will affect the SLA-L.

SLA-R is the *SLA runtime* information during instantiation and enforcement stages. It records information captured during SLA instantiation and the enforcement lifecycle. In Negotiation process such as contractual information; parties involves in the SLA agreement, agreement date, and agreement validity period are captured into the SLA-R.

VII. SERVICE LEVEL MANAGEMENT – PROCESS MODELLING

Business Process Management (BPM) transforms real world business processes into a *process model representation*. Several standards are available, for example BPMN, EEML, Flowchart, BPEL and IDEF3. Business Process Model & Notation (BPMN) is one of the most widely accepted process modelling standards. It creates a standardized bridge between business process design and process implementation. To design a process, Business Process Diagram (BPD) provides a flowcharting technique to create graphical models of the process called workflow. The same workflow, with enough customization and coding, can be executed by any BPM System (BPMS) e.g., Activiti, Bonita, jBPM. Thus, it reduces the gap between analysis, design and implementation of the system.

Due to the nature of dynamics of SLM process varieties, we need an approach, which could address the design, analysis and implementation challenges. We propose to use BPMN, and utilize the BPD to model the SLM. To model, we need to identify the SLA concept, process flow and interactions of system component, required state, triggering events etc. It will act as a guideline for implementing the SLM into BPMN.

In delivering the SLA, SLM activities manage the SLA states throughout *creation*, *instantiation*, *enforcement* and *termination*. The provider needs to orchestrate the SLM activities. Here, we try to explain those processes and see its applicability throughout the state of service and SLA (TABLE IV).

TABLE IV. SLA, SLM, SERVICE AND T-WSLA STATE AND ACTIVITIES

	t1	t2	t3	t4	t5	t6
Service State				Instantiation	Runtime	Terminate
Service Activities		Service Offering		Service deployment		Service termination
SLA & SLM State	SLA, SLM creation	SLM Instantiation		SLA Instantiation	SLA enforced	SLA & WF SLM terminated, SLA Archiving
SLM Activities	Template definition		SLA Negotiation process	SLA Deployment, Mapping & Translation	Evaluation & Enforcement, monitoring	Conclusion - Accounting, Settlement/Penalties/reward
T-WSLA State	Define SLA-D, SLA-I and SLA-L	Display the SLA-M info to the customer; Service Definition, SLAParameter and exclusion.	Capture the SLA-R Signatory information between parties		The SLA-L and SLA-I will be used throughout the service runtime state	

At `t0,` the assumption is made that the service is ready to serve the customer. In `t1`, the *creation* of SLA data model (SLA-D, SLA-I, SLA-L) and SLM process model is formalized by the provider. This is the design and creation state of T-WSLA Object & SLM process workflow. At `t2`, the service & SLA agreement is *offered* to the user and the SLM workflow starts in parallel. In `t3`, the SLA enters the negotiation process. The negotiation process is not compulsory and depends on the design of the SLM. Once negotiation completes, we populate contractual information into the SLA-R. In `t4`, the service and SLA is instantiated. In `t5`, the Service is *running* and the *SLA* is in *enforced mode* where continuous evaluation, enforcement and monitoring are executed. The SLA-Logic that includes the SLO, goals, metric calculation and SLA-I i.e., monitoring URI and action URI is continuously used in `t5`. At `T6`, when the service is terminated, the SLA & SLM activities will be stopped. The SLA conclusion processes like accounting, settlement, penalties or rewards may run in both `t5` and `t6` and not exclusively in `t6` since penalties can be compensated even during service runtime.

A. *Conceptual Modelling of SLA to BPMN*

TABLE V. CONCEPTUAL MAPPING OF SLA TO BPMN NOTATION

Concept	SLA Entity	BPMN
To show the data object transition between processes	<u>T-WSLA</u>	Data Object
To visualize signatory parties or communication between systems or components.	Role & Organization	Lanes & Pool
To express individual SLM process	negotiation, enforcement, service offering, conclusion	Sub-process; Process elements; tasks, gateway, events
To calculate composite metrics or SLA parameter	Monitoring metrics, SLA Parameters	
To express complex violation logics	Violation detection, prevention, action	
ActionGuarantee expresses a commitment to perform action if a given precondition is met.	Action Guarantee	
SLO expresses commitment to maintain a particular state of the service over a period of time.	SLO logics	Sub-process for complex representation of SLO and business rules
Any rules expressed in the SLA document.	SLA Exclusion, Termination clauses, penalties	Business rules or conditional Events
To escalate message, system signal to another party or system component.	Escalation events	Intermediate throwing or end (escalation, message, signal) events
To receive violation events.	Violation events	Start or intermediate catching (error, message, signal) events
Scheduling of SLA or Monitoring.	Schedule	Timer Event
SLA period or others.	Period	Timer Event
To depict interval i.e., monitoring collection intervals.	Intervals	Timer Event

There are 4 basic categories of notations, 1) <u>*flow object*</u> (*Event, Activity, Gateway*) –core objects types to represent the operation logics. 2) <u>*Connecting objects*</u> (*sequence, message, association flow*) – to show communication or interaction, 3) <u>*swimlanes*</u> (pools, lane) – to illustrate different functional capabilities or responsibilities. And 4) *artifacts (Data Objects, Group, Annotation)*- as a supplementary notation [31].

A conceptual mapping is discussed in TABLE V between the SLA [25] to BPMN 2.0 specification [32]. This provides a guideline on how to use the BPMN in modelling the SLM process is possibly incomplete or accurate and open for improvement. To illustrate further, we can express the SLA violation triggers from monitoring tool or workflow sub-process as `start` or `intermediate catching; error, message` or `signal events` or <u>escalation of events</u> to parties can be formalize using `intermediate throwing` or `end; escalation`, `message` or `signal events`. Any form of rules for example, SLA exclusion, and terminations can be put into complex `business rules` or simplified `conditional events` at process flow `gateways`.

B. *Example of IaaS Service into BPM Notation*

To illustrate BPMN notation, we present the VM availability *Service Object* into BPMN notation in Figure 3. It shows the *a) Enforcement stage* and b) *conclusion stage*; processes. This process model facilitates the SLA specification in the design phase of SLM activities as well as the interpretation of events during SLA monitoring.

BPMN, similar to BPEL deals with two parts of process modelling 1) the *abstract* – partially specified processes that are not intended to be executed, which hides some of the required operational details of a process. Abstract processes serve as a descriptive role with more than one possible use case, which includes the behaviour or events of the process i.e. the communication, function or states of processes. It acts as a management discipline where it was used originally for people-to-people communication through BPD modelling. The process may highly underspecify where the process is presented in high-level descriptions. 2) The *executable business process* is an actual behaviour of a participant in business interactions which is executed in a BPM-system (BPMS) such as Activiti or jBPM [33].

While this paper presented the SLM into the BPMN abstract process design model, it is imperative to mention that the implementation to deliver SLA will requires more programming effort in order for it to run in a BPMS workflow engine.

We have modelled the SLM and monitoring logics into BPMN, which can be executed on workflow engines. However, to reduce the traffic and monitoring overhead, it is best to push and translate the workflow into monitoring specific implementation through bash scripts or any available monitoring scripting available.

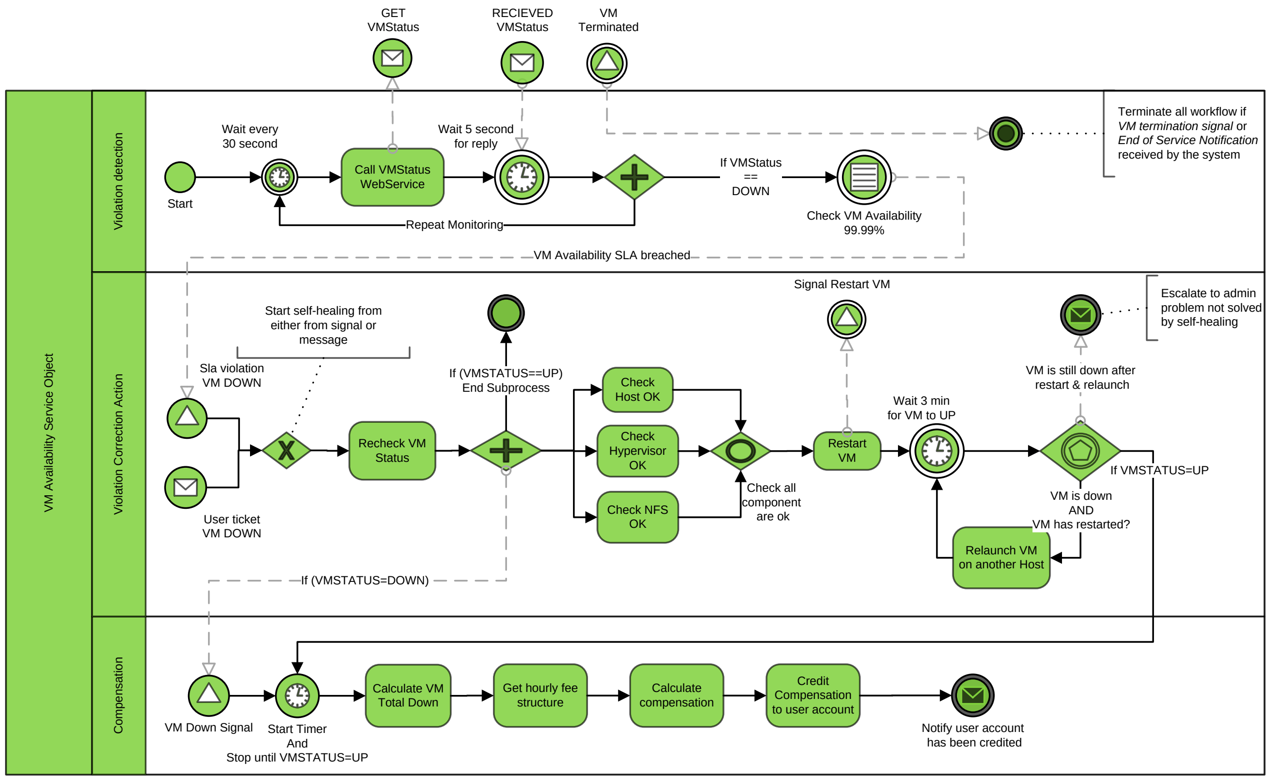
GET VMStatus
RECIEVED VMStatus
VM Terminated
VM Availability Service Object
Violation detection
Violation Correction Action
Compensation
Start
Wait every 30 second
Call VMStatus WebService
Wait 5 second for reply
Repeat Monitoring
If VMStatus == DOWN
Check VM Availability 99.99%
Terminate all workflow if VM termination signal or End of Service Notification received by the system
VM Availability SLA breached
Start self-healing from either from signal or message
Sla violation VM DOWN
User ticket VM DOWN
Recheck VM Status
If (VMSTATUS==UP) End Subprocess
Check Host OK
Check Hypervisor OK
Check NFS OK
Check all component are ok
Signal Restart VM
Restart VM
Wait 3 min for VM to UP
Escalate to admin problem not solved by self-healing
VM is still down after restart & relaunch
If VMSTATUS=UP
VM is down AND VM has restarted?
Relaunch VM on another Host
If (VMSTATUS=DOWN)
VM Down Signal
Start Timer And Stop until VMSTATUS=UP
Calculate VM Total Down
Get hourly fee structure
Calculate compensation
Credit Compensation to user account
Notify user account has been credited

Figure 3. Enforcement & compensation BPMN Notation

VIII. CONCLUSION & FUTURE WORK

In this paper, we have summarized relevant SLA objects and SLA management processes towards a generic SLA management framework. To design a good SLA orchestrator tool it is imperative to properly 1) represent the SLA document into a suitable SLA model and 2) SLM management process into a suitable process language notation. Based on our survey, we have derived the SLA and SLM meta-model to show the relationship, interaction and categorize those items as compulsory or optional. In designing the SLA offering and its management process, the framework should be flexible enough to incorporate or drop any of the process or object as the SLA offering is never a fixed process cycle.

To model the SLA, we opted for WSLA and extended its concepts in order to support new SLA process activities. We then proposed a tiered mechanism within the WSLA, which separates the information, logics, implementation and runtime information to promote reusability parts-of the model. We illustrate the usage with an example of an IaaS VM availability scenario.

To model the processes, we provide a concrete conceptual mapping of SLA objects and concepts to the notation of BPMN. In our opinion, the BPMN is the most suitable process modelling language in order to design and implement SLA. The BPMN captures the nature, variation of design and dynamics of the SLM processes. We believe that by correctly defining those two models it will act as a guideline in creating the SLA offering.

For future work, a proof of concept of the framework is required. The tool should be flexible in defining some or a combination of the SLA management process. IaaS would be a suitable test case as it is a complex and ever-changing technology.

REFERENCES

[1] J. Happe, W. Theilmann, A. Edmonds, and K. T. Kearney, “Service Level Agreements for Multi-Level SLA Management,” Service Level Agreements for Cloud Computing, P. Wieder, J. M. Butler, W. Theilmann, and R. Yahyapour, Eds. New York, NY: Springer New York, pp. 13–26, 2011.

[2] R. Sturm, W. Morris and M. Jander, “Foundations of Service Level Management.” Indianapolis: SAMS Publisher, p. 272, 2000.

[3] I. Rosenberg, A. Conguista, and R. Kuebert, “Management for Service Level Agreements,” Service Oriented Infrastructures And Cloud Service Platforms For The Enterprise, T. Dimitrakos, J. Martrat, and S. Wesner, Eds. Springer Berlin Heidelberg, pp. 103–124, 2010.

[4] A. Paschke, J. Dietrich, and K. Kuhla, “A logic based sla management framework,” in Semantic Web and Policy Workshop (SWPW) at 4th Semantic Web Conference (ISWC 2005), 2005.

[5] A. Keller and H. Ludwig, “The WSLA framework: Specifying and monitoring service level agreements for web services,” J. Netw. Syst. Manag., vol. 11, no. 1, pp. 57–81, 2003.

[6] X. Liu, Y. Yang, D. Yuan, G. Zhang, W. Li, and D. Cao, “A Generic QoS Framework for Cloud Workflow Systems,” 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pp. 713–720, 2011,.

[7] OSIATIS S.A, "Service Level Management - Introduction and Objectives," 2014. [Online]. Available from: http://itil.osiatis.es. 06-Apr-2014

[8] D. Kyriazis, "Cloud Computing Service Level Agreements;Exploitation of Research Results," European Commision Directorate General Communications Networks, Content and Technology, Technical Report, 2013.

[9] V. C. Emeakaroha, I. Brandic, M. Maurer and S. Dustdar, "Low level metrics to high level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," The 2010 High Performance Computing and Simulation Conference (HPCS 2010), pp. 48–54, 2010.

[10] ISACA, "Information Systems Audit and Control Association." [Online]. Available from: http://www.isaca.org/. 23-Oct-2014

[11] "WSAG4J Programming Model." [Online]. Available from: http://wsag4j.sourceforge.net/site/server/programming_model.html. 06-Apr-2014

[12] C. Ward, M. J. Buco, R. N. Chang, and L. Z. Luan, "A Generic SLA Semantic Model for the Execution Management of e-Business Outsourcing Contracts," E-Commerce and Web Technologies, pp. 363–376, 2002.

[13] H. Ludwig, "WS-Agreement Concepts and Use–Agreement-Based Service-Oriented Architectures," IBM Research Division, Technical Report, 2006.

[14] A. Asosheh and P. Hajinazari, "Towards improving enterprise performance with Service Level Agreements," Telecommunications (IST), 2012 Sixth International Symposium, pp. 913–918, 2012.

[15] W. Theilmann, J. Lambea, F. Brosch, S. Guinea, P. Chronz, F. Torelli, J. Kennedy, M. Nolan, G. Zacco, G. Spanoudakis, M. Stopar and G. Armellin, "SLA@SOI Final Report," European Commission Information Society and Media, Technical Report, 2011.

[16] W. Theilmann, J. Happe, C. Kotsokalis, A. Edmonds, K. Kearney, and J. Lambea, "A reference architecture for multi-level sla management," Journal of Internet Engineering, vol. 4, no. 1, pp. 289–298, 2010.

[17] I. Haq, I. Brandic, and E. Schikuta, "SLA Validation in Layered Cloud Infrastructures," Economics of Grids, Clouds, Systems, and Services, R. Y. Philipp Wieder, Joe M. Butler, Wolfgang Theilmann, Ed. Springer Berlin Heidelberg, pp. 153–164, 2010.

[18] "IRMOS Project." [Online]. Available from: http://www.irmosproject.eu/. 23-Oct-2014

[19] "Cloud4SOA Project." [Online]. Available from: http://www.cloud4soa.eu/. 23-Oct-2014

[20] "OPTIMIS Project." [Online]. Available from: http://www.optimis-project.eu. 23-Oct-2014

[21] M. Smithson, "Applying SOA Governance Using WSRR, DataPower, and WS-Policy." [Online]. Available from: https://www-950.ibm.com/events/wwe/grp/grp004.nsf/vLookupPDFs/Applying SOA Governance Using WSRR, DataPower, and WS-Policy/$file/Applying SOA Governance Using WSRR, DataPower, and WS-Policy.pdf. 23-Oct-2014

[22] "Uptime Software." [Online]. Available from: http://www.uptimesoftware.com/. 20-Jun-2014

[23] Servicenow, "Service Level Agreements." [Online]. Available from: https://wiki.servicenow.com/index.php?title=Service_Level_Agreements_(SLA)_Plugin. 01-Jun-2014

[24] J. Meegan, G. Singh, S. Woodward, S. Venticinque, M. Rak, D. Harris, G. Murray, B. D. Martino, Y. L. Roux, J. McDonald, R. Kean, M. Edwards, D. Russell and G. Malekkos, "Practical Guide to Cloud Service Level Agreements version 1.0," Cloud Standard Consumer Council, Technical Whitepaper, 2012.

[25] H. Ludwig, A. Keller, A. Dan, R. P. King and R. Franck, "Web service level agreement (WSLA) language specification," IBM Corporation, Technical Report, 2003.

[26] J. Skene, D. D. Lamanna, and W. Emmerich, "Precise service level agreements," International Conference on Software Engineering (ICSE 2004), pp. 179 – 188, 2004.

[27] A. Correia and F. B. e Abreu, "Model-driven service level management," 4th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2010), pp. 85–88, 2010.

[28] A. Correia, F. B. e Abreu and V. Amaral "SLALOM: a Language for SLA Specification and Monitoring," Computing Research Repository (CoRR 2011), pp. 556–567, 2011.

[29] J. Saltzer and M. Kaashoek, "Principles of computer system design: an introduction." Morgan Kaufmann, 2009.

[30] A. Solberg, D. Simmonds, R. Reddy, S. Ghosh, and R. France, "Using Aspect Oriented Techniques to Support Separation of Concerns in Model Driven Development," 29th Annu. Int. Comput. Softw. Appl. Conf., vol. 1, pp. 121–126, 2005.

[31] S. A. White, "Introduction to BPMN," IBM Cooperation, Technical Paper, pp. 1–11.

[32] "BPMN 2.0 by Example-version 1.0," Object Management Group, Technical Paper, 2010.

[33] Acitiviti, "What is BPM." [Online]. Available from: http://activiti.org/faq.html#WhatIsBpm. 07-Mar-2014