Standard Monitor Design for SLA Parameters in SOA

Alawi Abdullah ahmed Al-sagaf
Dayang Norhayati Abang Jawawi
Universiti Teknologi Malaysia

ABSTRACT
The current trend in modeling and designing IT systems is by using service-oriented approach that follows a new paradigm called Service-oriented Architecture. SOA is a new paradigm that manages the execution of the service's instance which is not fully under the control of the client or service requestor but under the third party or provider. Service-level Agreement as means of specifying measurement parameters for performance (QoS), became extremely an important aspect in SOA framework due to the nature of cross-organizational services (i.e. outsourced email service). This is can be seen through standard SLA languages have been emerging recently to formalize SLA in order to become a machine-readable SLA instead of classical telecommunication's SLA that uses natural languages. WSLA is an XML-based language that used to create machine-readable SLAs. However, there is still a gap on designing monitors in standard and generally standard way of doing instrumentation process. This paper proposes standard vocabulary for monitor design helps communicate the problem and encourage automation. A strong relationship has been defined between SEI 6-element Framework and modern SLA languages like WSLA. The result of comparison between the two metamodels has presented which is a contribution to monitor design.

General Terms
Standard; Monitor; SLA parameters.

Keywords
SOA; SLA; performance; SEI6; WSLA.

1. INTRODUCTION
There is no doubt that IT infrastructure is growing due to business requirements which led to existence of complex and dynamic IT infrastructure. This situation in modern enterprise highlight the necessity of having a mechanism to monitor the quality of the service which provided by that infrastructure. For example, dashboard toolset that add values to enterprise business process by enabling user to monitor, detect and correct the infrastructure [1]. Therefore without these mechanisms it became critical to achieve business goals (performance constraint).

The function of these automatic monitoring mechanisms became necessary by maintaining it within more complex systems (think of Amzon business). An effective practical engineering approach is needed to improve the quality of monitor's design.

The current trend in modeling and designing of service-oriented systems follows a new paradigm called Service-Oriented Architecture (SOA) [2] [3]. In this approach the functionality of the system is assigned to loosely coupled services where integration between heterogonous systems is possible, thereby reuse increased agility to adapt to changing business requirements. Service Level Agreement (SLA) is an essential artefact defines the obligation between service provider and service consumer in which services and the level of quality are specified. SLAs have been used in IT organizations and departments for many years. The standard of SLAs in SOA framework are still new but recently it became extremely important due to the high demand on services in SOA systems that cross over the organizational boundaries and a lot of third-party service providers [4]. Therefore, it is required to measure and ensure quality of service from both service provider and service consumer prospective. For example, an online storage web service offered by Amazon Web Services, and an exchange server provider hosting customers emails (i.e. Microsoft live outlook). In both examples performance are critical QoS must be verified by the end users or third parties at provisioning time.

This due to many factors, If we look carefully at current service-based systems, services are able to communicate because they are independent of technology. Also service is allowed to grow dynamically. In this case a service provider could enhance the quality of functionality provided by their systems. For instance they could increase the resources available to the service. This causes a variation in the service's non-functional properties. For instance, optimization could improve a non-functional (i.e. performance) property while may lead to violation of the SLA obligations. For instance, in the Amazon S3 SLA the availability becomes lower than 95%.

The monitor do instrumentation by measuring performance of service instances to be compared with the expected results (SLA parameters) so this step requires considerable processing and human effort before they can be related to the SLA terms.

It is further complicated by the fact that a service provided may be the result of a composition of services (i.e. cross organization business process), so it's not entirely under the control of the provider organization.

The new trend towards SLA is to be in standard form and machine-readable by formalizing it .This direction is new as we can see a few standards exists with these properties in the literature [12]. For example IBM's WSLA framework and the WS-Agreement specification. In contrast there was no standard monitor process and terms have been established. For example WSLA standard does not show any details about the monitor process only a few elements. This means every client (many times third party) is free to configure and select components for this process. In addition, There is a lack of automation which makes this task more tedious (think of like Amazon provider with hundreds of thousands services). Unfortunately low level processing is need and the benefit of machine-readable SLA is still not utilized. The work in this paper is a step toward this direction.

2. SERVICE LEVEL AGREEMENT
Service level agreement (SLA) is a prediction agreement between two parts consumer and provider [11], which lead to a description of the relationship between them that should involve the level of service(s) provided by the provider [12]. This information usually is statistical information which can tell about different Nonfunctional requirements depending on

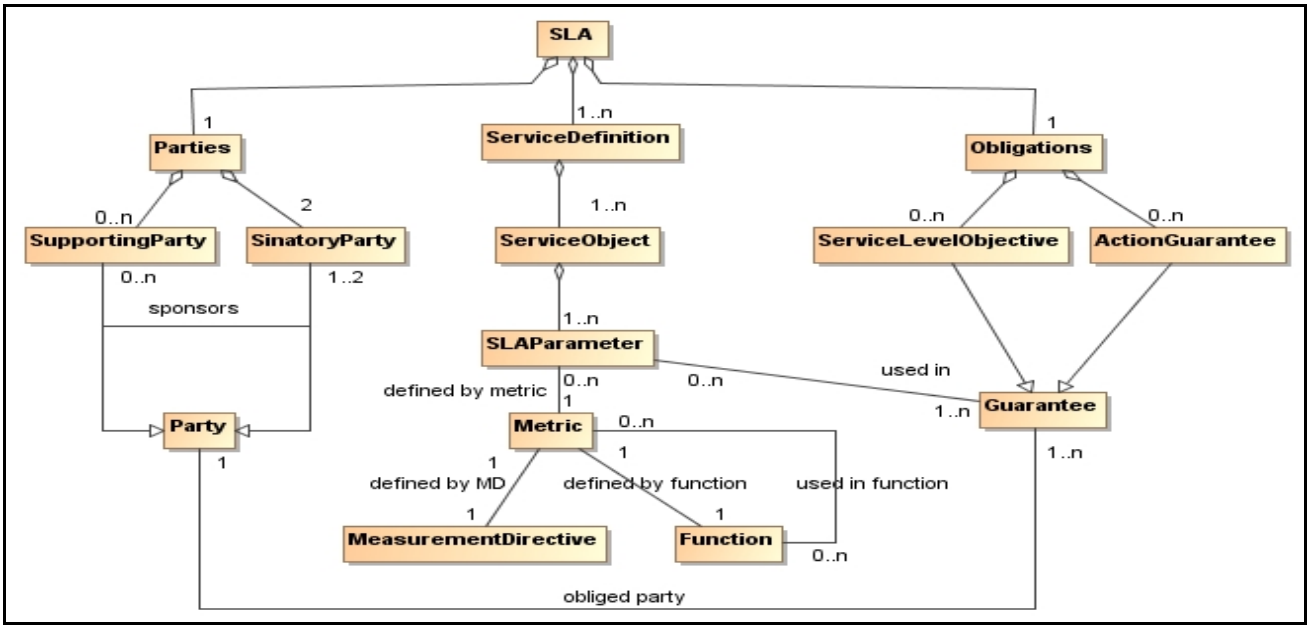

Fig 1: Overview of main WSLA concepts [9]

the application used. For example performance, reliability, security, etc. This document accurately describes the details of services and levels that serve as a legally binding between the service provider and consumer of the service.

There are different scales for SLA in IT can be categorized as in following Edward [11]:

a) Basic: is intended to justify technical support operation. It is in form of a single level of service agreement. Often requires manual data collection for Metrics establishment. This would facilitate management and reporting.
b) Medium: it is intended to reduce the cost while increasing service levels in a long term. There is sometimes introduction of multi-level quality based on the cost of the service. A comprehensive reporting to IT stakeholders needs automatic data collection .
c) Advanced: in this case resources allocation with an extension facility due to business evolving is dynamically managed.

SLA can make it easier to know the real demands from provider prospective as well as keeping provider reputation.

There are number of benefits measuring against SLA such as summarized by Edward [11]:

a) To be able to do continuous quality improvement process by measuring against key performance.
b) A means by which you can specify conditions and penalties in case of not meeting expectations which improve trust.
c) An SLA provides a definition artifact for KPIs which supports tools selection, process definitions and skills (people, process and technology) for an organization.

The purpose of the service description is the clarification of four issues: What are the SLA parameters? To which service do they relate? How is SLA parameters measured or computed? How are the Metrics of a managed resource accessed? [7].

2.1 SLA Languages Elements

SLA has number of languages that can describe the main elements of the common components required in an SLA:

a) Parties describe Service consumers, providers, etc.
b) Validity period describe When SLA is active or expires.
c) Metrics describe Latency, response time.
d) Scope/exclusions describe Conditions under which to evaluate SLA.
e) Purpose describes high level statement.

Because there are two ends in this high-level contract we need to identify parties involved in the SLA contract such as service consumer and providers or third party.

The time frame during which the SLA is to be honored is defined by validity period. The necessary conditions under which SLA should be applied are specified as scope and exclusions (e.g. service applied during weekends).

There is a composite metric which is derived from atomic or other composite metrics. It includes the time of the request for service to the end of service (response time). For example Atomic metrics is a number of process invocations which are directly measured. While there are other possibility of representation for functional measurements such as average values and maximum, this is form composite metric. By using SLO the service provider must ensure that for example the system is working by 70% during a given time. A final component in an SLA is the penalty of violating the SLOs [13]. The new direction in specifying SLA is by formalizing it like web service level agreement (WSLA).

2.2 Web Service level Agreement (WSLA)

WSLA is standard machine readable language for SLA has been adopted by IBM in 2001 [7] as shown in simple abstract syntax (metamodel) in Figure 1 [9] .The concept of metamodel is important here because we need to study features and properties of the language. Hence the main concepts of language and their relationship will appear in a form of classes with relationships because we are using UML class model. This is although of the fact that the basic structure is xml-based.

XML is basically used to formally express the concepts of SLA. There are number of the main components on WSLA which are described in Figure 1 as the following three basic elements of SLA: service definition, obligation, and party. The parties and interfaces are for those should sign this agreement: A party is an organization, not a computational element. For example, a party that has the role of evaluating performance measures collected from services under the scope of the SLA may need to notify the other parties in case of measures are not matching certain thresholds. A definition of

the services and their operations is needed. This section sets out the definition of service and level of the relevant service. There is a contact Person as determined by the party for each measurement and the type of measurement need to be done in the following example we illustrate the process and the service and the type of measurement required. The snippet below shows a service named DemoService with an operation called GetQuote.

A parameter named Availability_UpTimeRatio is defined using a metric called UpTimeRatio, which in turn uses another metric StatusTimeSeries.

3. SEI 6-ELEMENT FRAMEWORK

The current trend in software engineering highlights the importance of architecture. The treatment of NFR in general is not just confined to the architectural design. It is a holistic approach contains all stages of the application software .It concerns with implementation and its details the same way as with the architecture. This is a contribution developed by SEI [14]. It is stemmed from two points as in the following.

Since the achievement of quality on architecture is critical, there should be a way to discriminate architectural elements from the point that it can be evaluated at the architecture level. There are many definitions and classifications for different NFR have been identified in the literature for decades, but the following are the limitations observed:

3.1 Non-operational definition

The definitions comes not concrete for example when we say the system is modifiable, it is with respect to one set of changes but not modifiable with respect to another.

3.2 Overlapping

If we take for example a system failure, it is an aspect of availability, but also an aspect of security, and so does usability. This means all three attribute communities would claim ownership of a system failure.

3.3 Multiply vocabulary

It means the same aspect described by different terms. For instance, the performance community has "events" arriving at a system, the security community has "attacks" arriving at a system, the availability community has "failures" of a system, and the usability community has "user input." All of these may actually refer to the same occurrence, but are described using different terms.

The solution presented for the first and the second is a quality attributes scenarios, a mechanism to characterize one quality from another. The solution provided to the third problem was analysis to extract common concepts and elements represent the specific quality characteristic. So it is like ontology for the quality attribute.

3.4 Quality Attributes Scenarios

The concept of scenario is borrowed to resolve the problem of overlapping between NFRs and to give operational framework. It consists of six elements that represent requirements for a given quality attribute. They are as following:

a) **Source of stimulus**: is an entity (a human, computer, or other actuator) who generates the stimulus.
b) **Stimulus:** the condition when arrives to the system will be considered.
c) **Artifact**: the part of the system that stimulated and may be the whole system.
d) **Environment**: The system can be in various operational modes, such as normal, or overload.
e) **Response**: The kind of activity the system should do after arrival of the stimulus.

3.5 Response measure

The response should be measurable in some fashion so that requirement should be tested.

The response measures are the time it takes to process the arriving events latency or a deadline by which the event must be processed or a characterization of the events that cannot be processed.

Performance is about timing. Events are interrupts, messages, requests from users, or the passage of time .Any event located in the system should have a response by the system. The complexity in performance comes from having multiple and different sources of events and patterns of access. The events can come from a user, system, or another system.

In the system Web-base financial system response means the number of completed transactions per seconds. While for the engine control system the responsiveness means variances in firing time. In each case the pattern of events corresponds to a pattern of responses would formulate a distinction. This distinction is the language that is built by the scenario of overall performance. Performance scenario begins by request for service arriving to the system. In order to meet that demand resources are consumed. System might be busy during the receiving of this event servicing anther request.

There are three types of patterns of events. Periodically such as every 10 seconds, and Stochastic arrival means that events arrive according to some probabilistic distribution. If however, does fall between the Periodic and Stochastic is called Sporadic. Note that it does not matter whether one user submits 20 requests in a period of time or whether two users each submit 10, but what is the point is the arrival pattern.

System response to stimulus can be distinguished by latency (time between the receipt of the request and response), deadlines, or number of transactions done per time (Throughput), or events that are not handled because the system is busy, and the data that was lost because the system was too busy.

Notice that this analytical model does not depend on whether the system is network or stand alone, nor does depend on a given system configuration or situation on the consumption of resources. From these considerations we can see an example for portions of the performance general scenario: “Users initiate 5,000 transactions per minute stochastically under normal operations, and these transactions are processed with an average latency of five seconds."

a) **Source of stimulus** The source of the stimulus is a collection of users. The stimuli arrive either from external or internal sources.
b) **Stimulus** The arrival pattern can be characterized as periodic, stochastic, or sporadic. The stimulus is the stochastic initiation of 5,000 transactions per minute.
c) **Artifact** The artifact is always the system's services, as it is in our example.
d) **Environment** In our example, the system is in normal mode.
e) **Response** The system must process the arriving events. This may cause a change in the system environment (e.g. from normal to overload mode). In our example, the transactions are processed.
f) **Response measure** The response measures are the time it takes to process the arriving events (latency or a deadline by which the event must be processed), the variation in this time (jitter), the number of events that can be processed within a particular time interval (throughput), or a characterization of the events that cannot be processed (miss rate, data loss). In our example, the transactions should be processed with an average latency of two seconds.

4. THE PROPOSED APPROACH

The need of mointors in SOA environments is becoming a first class concern for both service providers and consumers. Service provider need to know the resources alloacted to service(s) are sufficient or not while clients need to assure the agreed level of service that has invested on. Monitor is a kind of software arachitecture helps to know about that. Monitors are generally managed through framework. Several SLA frameworks have been proposed [17], but most of them involve lot of human intervention and technical expertise.

Generally there is no standard way of developing this framework so they are proprietary solutions so far. The innovaiton in this paper is about standarizing and raising abstraction level of monitor design by utilizing design concepts emerging from SEI-6 Element framework. In this view the SLA has been grasped as high level abstraction so monitor abstraction level is need that could implement SLA. This abstraction level establishes ontology of monitoring process. Model-based engineering is a trend and best practice toward complex systems engineering has been around for a decade. In this case models are used to derive the development process for example expressed by using BPMN or UML. This paper uses UML to show the model of new standard monitor which is not only afforded easy communication but help automate. Before that the basic elements of the SEI-Framewerok will be related to SLA in order show how SLA can fit monitor abstraction level.

5. THE RELATIONSHIP BETWEEN TWO METAMODELS

It turns out like in history of computing that a common practice for a language designer to choose one representation for publication of the language specification, but with the fact that an implementation might have a very different representation [10]. The formal structure is called abstract syntax (metamodel) and the representations concrete syntaxes. For instance BNF specification of C could be used to generate a compiler (implementation) accept only C source code. If the programming language (metamodel) is expressed in a standard metametamodel, much of the effort needed to develop the software environment can be reused (i.e. YACC) [10].

Due to that fact a metamodel for monitor design will be established in the next step after making comparisons between SEI-framework and machine readable language like WSLA.For best of our knowledge there was no effort in this space.The concept of stimulus could be seen as the event of collecting metrics. It is one of the essential step monitor should do. Events are the reason for stimulus but they are different kinds of events as classified by SEI-framework for example periodic, and stochastic. In SLA practice it is used to tell about one kind only (periodic) in general without this classification so it is an enhancement to WSLA specification. For instance in Figure 2 there is one metric (*UtilizationTimeSeries*) collected every 5 minutes as appear in Schedule tag.

```
<Metric name="UtilizationTimeSeries" type="TS" unit="">
  <Source>YMeasurement</Source>
  <Function xsi:type="TSConstructor" resultType="float">
      <Schedule>5minuteschedule</Schedule>
      <Metric>ProbedUtilization</Metric>
      <Window>12</Window>
   </Function>
</Metric>
```

Fig 2: Example of a periodic function for a metric

```
 <Operation name="GetQuote" xsi:type="WSDLSOAPOperationDescriptionType">

   <SLAParameter name="OverloadPercentage"
                 type="float"
                 unit="Percentage">
    <Metric>OverloadPercentageMetric</Metric>
<Communication>
  <Source>ACMEProvider</ Source >
  <Pull>ZAuditing</Pull>
  <Push>ZAuditing</Push>
</Communication>
  </SLAParameter>

  <SLAParameter name="TransactionRate"
                 type="float"
                 unit="transactions / hour">
   <Metric>Transactions</Metric>
   <Communication>
     <Source>ACMEProvider</Source>
     <Pull>ZAuditing</Pull>
     <Push>ZAuditing</Push>
   </Communication>
  </SLAParameter>
```

Fig 3: Example of metrics with its source for a service

In Figure 3 we have two metrics: overloadPercentageMetric and transactions. It shows the monitor is the source of stimulus (tag source) will collect these metrics (stimulus) from the system under monitoring. The name of this source is ACMEProvider.

It is obvious this generated relationship will add a classification value that helps monitor engine fixes the decision about the kind of metric. Different clients might be interested on different kinds at different time (different requirements). On other hand stimulus is a classifier for events.

An artifact concept is the part of functionality from the system that is stimulated (in this case the ACEMProvider). As presumed in this paper the SOA environment, functionalities are expressed in service-based concept. Therefore an artifact could be a direct representation like service interface or indirect like some representative (i.e. testcases). It is clear that SOA has a systematic way for representing the functionality (service concept) so it will help model it as artifact abstraction. This will enhances the SEI-framework itself although it's out of the paper scope.

The environment element in SEI-framework has a corresponding meaning in SLA which is load (frequency of service usage) is specified with some condition shows different provider behavior for example under normal case and abnormal cases (sudden congestion , etc).

```
<Obligations>

 <ServiceLevelObjective name="ConditionalSLOForTransactionRate">
   <Obliged>ACMEProvider</Obliged>
   <Validity>
      <Start>2001-11-30T14:00:00.000-05:00</Start>
      <End>2001-12-31T14:00:00.000-05:00</End>
   </Validity>
   <Expression>
     <Implies>
       <Expression>
         <Predicate xsi:type="Less">
           <SLAParameter>OverloadPercentage</SLAParameter>
           <Value>0.3</Value>      <!-- 30% -->
         </Predicate>
       </Expression>
       <Expression>
         <Predicate xsi:type="Greater">
           <SLAParameter>TransactionRate</SLAParameter>
           <Value>1000</Value>
         </Predicate>
        </Expression>
     </Implies>
   </Expression>
   <EvaluationEvent>NewValue</EvaluationEvent>
 </ServiceLevelObjective>
```

Fig 4: load specification in WSLA

Figure 4 shows us under normal situation overloadPecentage (in 70% of time) greater than 1000 transactions should be guaranteed while in abnormal case not less than 30%. In the following section the metamodel of monitor will be discussed. The response corresponds to the result of executing functions of metrics. This is which monitor needs to keep in order giving its judgment later. SLA does not involve these details because it's a monitor concern which is already found in SEI-element. This point shows rational behind SEI-Framework can fit into monitor design. Because there are different kind of responses to stimulus the response measurement is needed (latency, response time,etc). SLAs do have specification corresponding to it like in Figure 5.

```
<Metric name="ResponseTime" type="long" unit="milliseconds">
        <Source>ACMEProvider</Source>
       <Function xsi:type="Minus" resultType="double">
         <Operand>
       <Function xsi:type="TSSelect" resultType="long">
         <Operand>
<Metric>SumResponseTimeTimeSeries</Metric>
```

Figure 5: Example of response time as Response measurement

The metamodel in Figure 6 represents the proposed design for monitor of SLA elements. It consists of the SEI 6-elements as shown stimulus model element has four attributes SLAparameter, kind, unit, and type (see Figure 6). An instance of stimulus associates with instance of source (for each stimulus there is a source – see Figure 6). An instance of stimulus is also associate with an instance of artifact. Artifact is identified by ID and has content. Because the events that represent metrics need to be computed are in two kinds in the way they are collected we modeled them as publish-subscribe pattern which represents two scenarios pull and push (see Figure 6). Usually in SLA there are only these two kinds. Service Provider (or third party) may push or pull metric collections. An instance of response measurement is existed each time stimulus is generated that is why there is an association between them. A function is basic element in measurement which is one of two kinds: composite (i.e. average resonponse time) or atomic (i.e. response time). Functioncontext superclass has two subclasses: composite and primitive, is used to model this fact a function with its two kinds. An environment instance is associated with every instance of stimulus. Here the load specification will represent.

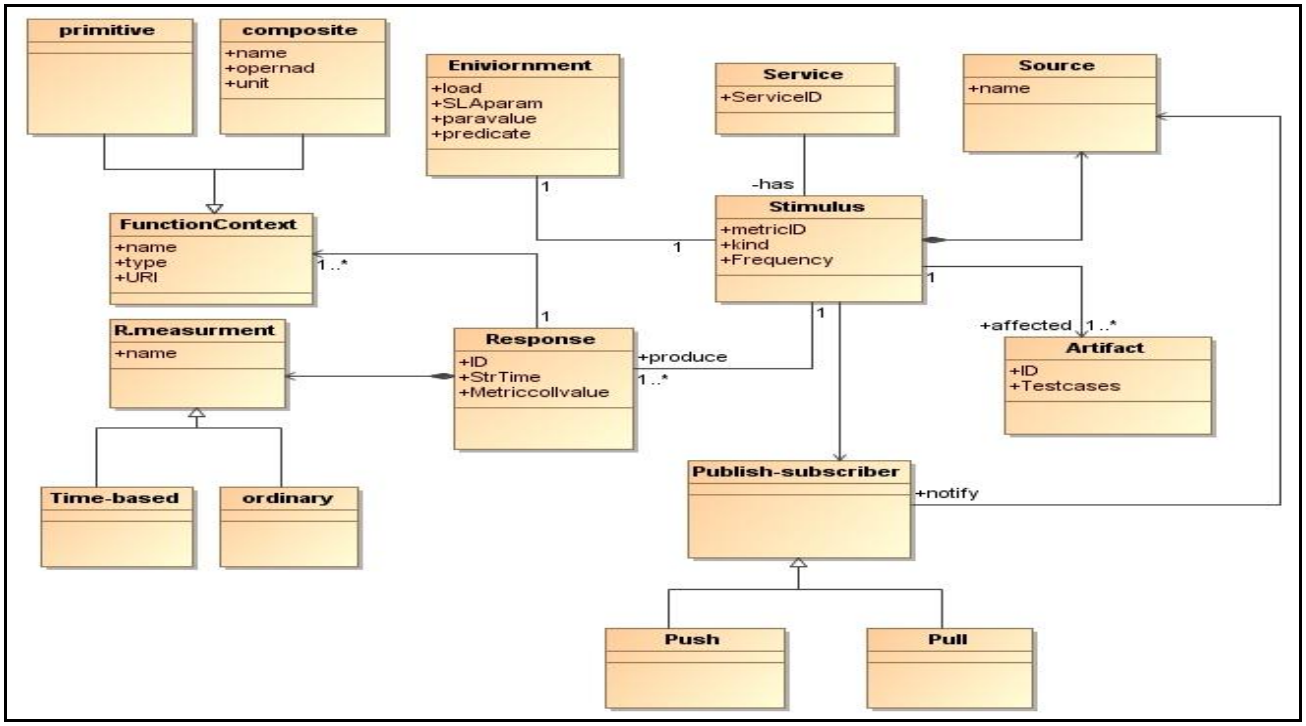

Fig.6 The monitor metamodel for SLA parameters

6. RELATED WORK

There is two directions can divide the literature in this space. The first direction is working to setup a language or descriptions of the elements of SLA. The trend is to formalizing SLA as already discussed in section 2.2. The common standard languages which our work based on one of them are WSLA and WS Agreement [9][5]. But these languages do not consider implementations details of the instrumentation process which the proposed framework is about. The other direction at this point only has a proprietary language for SLA such as [15]. However, the adoption of high level models became a famous phenomena and best practice as seen [15] in augmenting BPMN metamodel with that of the SLA life cycle. While Ajaya has highlighted mimizing manual efforts and technical expertise in so far SLAs engineering [16] by the proposed high level models. This paper is in line with those contributions because it bases on metamodel concept (UML is used).

The second basic direction in this area was description of monitor like in [8] the user working with specific and low

level implementation details. This work is focus on configuring components at each time new problem defined, but without benefits re-using these components with even variant values of SLA. Also in this research [8], there is no standard instrumentation process and design for monitor. The proposed approach is different by recognizing a common vocabulary of design for monitor like those of SEI-framework. Although there is a freedom in design left for clients such as Simon [8] and Ajaya [16], and [9] still more effort effort is needed. The point is in addition to reduce engineering efforts our approach has value added in encouraging automation and re-use of the same architecture with different SLA setups and environments.

7. CONCLUSION AND FUTURE WORK

Monitoring the level of service in SOA-based systems became a significant engineering activity because of increased number of outsourced services. SLA is a means whereby a monitor can be designed by establishing instrumentation process. So far attempts were in the space of standard SLA languages without focus on the monitor design itself. Many proprietary solutions were founded but there is no standard design vocabulary for monitors. SEI has established a strong framework for quality measures from architecture prospective.

This paper proposes a standard vocabulary for monitor design stemmed from that framework by highlighting a strong relationship between SEI framework and SLA languages. Due to this an implementation metamodel suitable for monitors design has been developed. This step is expected to encourage automation using recent model-based engineering technology like MDA as well as getting the value of raising the abstraction level (i.e. less engineering effort, re-using ,etc). This paper has shown a logical conclusion to the progress of formalizing SLA in SOA such as WSLA in order to measure like performance QoS. A standard architecture for monitor is founded. However, this paper contributed to the possibility of automating the measurement of SLA performance which reduces the engineering effort. More importantly the monitor has standard design vocabulary for measuring performance parameters helps easy communicating ad different problems under SLA engineering.

8. REFERENCES

[1] SWEBOK2004(downloaded from www.swebok.or).

[2] E. Thomas, "SOA Principles of Service Design", Book Soa: principles of service design First,Prentice Hall Press Upper Saddle River, NJ, USA ©2007, ISBN: 9780132344821.

[3] M. Nicolai, Josuttis., "SOA in Practice" 2007.

[4] B.Philip, A .Lewis, "Service Level Agreements in Service-Oriented Architecture Environments" 2008.

[5] B.Antonia, Guglielmo A. De, F. Lars, P. Andrea, "Model-Based Generation of Testbeds for Web Services, Institute for Computing and Information Sciences", (ICIS),Radboud University Nijmegen – The Netherlandslf@cs.ru.nl, Department of Mathematics and Computer Science University of Camerino – Italy, andrea.polini@unicam.it. 2008

[6] M. Ed, William, A., Sriram, B., David, Carney., John, Morley., Patrick, P., Soumya, Simanta., Testing in Service-Oriented Environments, Software Engineering Institute, http://www.sei.cmu.edu, 2010.

[7] A. Keller, H. Ludwig, (2003), The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, IBM Research Division, T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598. E-Mail: falex,hludwigg@us.ibm.com.

[8] E. Simon,M. Graham,Toward reusable SLA monitoring capabilities, School of Computing Science, Newcastle University, U.K. 2011.

[9] IBM, Heiko Ludwig, IBM T.J. Watson Research Center Alexander Keller, Asit Dan, Richard P. King, Richard Franck, IBM Software Group, Version: 1.0, 2003.

[10] Colomb, R.M., Raymond, K., Hart, L., Emery, P., Welty, C., Xie, G.T., Kendall, E.The Object Management Group Ontology Definition Metamodel In: Calero, C., Ruiz, F., Piattini, M. (eds.). Ontologies for Software Engineering and Software Technology. Berlin Heidelberg: Springer. 217-247. 2006.

[11] Edward, W., (2002), Sun Professional Services Sun BluePrints™ OnLine – April 2002 Service Level Agreement in the Data Center, www.sun.com/blueprints.

[12] Di Modica, G., Orazio, T., Lorenzo, V., (2009) Dynamic SLAs management in service oriented environments.

[13] Vinod, M., Hans, J., Tony, C., Allen, C., Phil, C.,(2009), SLA-Driven Business Process Management in SOA, Paper appears in CASCON 2009, Richmond Hill, Ontario, Canada.

[14] Bass, L., Clements, P., Kazman, R., (2003), Software architecture in practice, Addison Wesley.

[15] Vasco, A., Anacleto, C., Fernando, B., (2011), SLALOM: a Language for SLA Specification and Monitoring, 1 CITI/FCT/UNL, 2829-516 Caparica, Portugal 2 IPS/EST, 2910-761 Setúbal, Portugal 3 DCTI, ISCTE-IUL, 1649-026 Lisboa, Portugal.

[16] Ajaya, K., Manas, R.,(2011), Modeling and Monitoring SLA for Service Based Systems, ISWSA'11 April 18-20, 2011, Amman, Jordan, Copyright c 2011 ACM 978-1-4503-0474-0/04/2011 ...$10.00.

[17] Domenico, B., Walter, B., Mauro, D., (2009), Automated Performance Assessment for Service-Oriented Middleware, USI-INF Technical Report, on site at NASA Ames Research Center.