

# A Reference Model for a Service Level Agreement

Citation for published version (APA):

Hofman, C., & Roubtsova, E. E. (2020). A Reference Model for a Service Level Agreement: In domain of Information Sharing Services. In B. Shishkov (Ed.), *Business Modeling and Software Design: 10th International Symposium, BMSD 2020, Berlin, Germany, July 6-8, 2020, Proceedings* (Vol. 391, pp. 55-68). Springer Nature Switzerland AG. Lecture Notes in Business Information Processing Vol. 391 [https://doi.org/10.1007/978-3-030-52306-0\\_4](https://doi.org/10.1007/978-3-030-52306-0_4)

**DOI:**

[10.1007/978-3-030-52306-0\\_4](https://doi.org/10.1007/978-3-030-52306-0_4)

**Document status and date:**

Published: 01/01/2020

**Document Version:**

Publisher's PDF, also known as Version of record

**Document license:**

Taverne

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

<https://www.ou.nl/taverne-agreement>

**Take down policy**

If you believe that this document breaches copyright please contact us at:

[pure-support@ou.nl](mailto:pure-support@ou.nl)

providing details and we will investigate your claim.


Downloaded from <https://research.ou.nl/> on date: 16 Jul. 2023

**Open Universiteit**  
[www.ou.nl](http://www.ou.nl)





# A Reference Model for a Service Level Agreement In Domain of Information Sharing Services

C. Hofman<sup>1</sup> and E. Roubtsova<sup>2</sup> 

<sup>1</sup> Graduated Master of Science Student of the Open University,  
Heerlen, The Netherlands  
cor.hofman@gmail.com

<sup>2</sup> Open University, Heerlen, The Netherlands  
ella.roubtsova@ou.nl

**Abstract.** Information sharing between government organizations is regulated by Service Level Agreements (SLA's). Design and implementation of an SLA demands involvement representatives of several organizations. They need to communicate with the same concepts and validate the requirements for the service and quality indicators. In order to support the design of an SLA and its monitoring, we propose related concept, goal and protocol reference models. The first conceptual model view is built using a literature review. The next model views include the details found by analysis of existing SLA's. The novelty of our models is that they compose an SLA from service level objectives (SLO's), explain the meaning of SLO's monitoring, support execution of an SLA and expose the monitoring logics.

**Keywords:** Service Level Agreement (SLA) · SLA modelling · SLA monitoring · Service Level Objective (SLO) · Goal model · Conceptual model · Executable protocol model

## 1 Introduction

In order to serve citizens, government organizations provide informational services to each other and rely on each other. For example, they share information about income of citizens. Government organizations formulate collaborative requirements for information sharing including timeliness and reliability. All these, mostly non-functional, requirements for the information sharing services are combined into a Service Level Agreement (SLA).

“A service-level agreement (SLA) sets the expectations between the service provider and the customer and describes the products or services to be delivered, the single point of contact for end-user problems, and the metrics by which the effectiveness of the process is monitored and approved” [3].

Representatives of different organizations should agree on an SLA. For communication, for understanding each other, the professionals need a shared conceptual model of the service and shared understating of its monitoring. As the

requirements in an SLA are mostly non-functional, their monitoring and validation is only possible with specially designed indicators assessing data collected from a running service process. Because an SLA-contract development process has so many points of attention from the partners of the contract and an SLA contains requirements for potential implementation automatic measurements, this process needs systematic modelling and a supportive system of a SLA life cycle. Supportive systems for SLA-contract development may share a reference model.

In this paper, we propose a reference model for a system that supports an SLA life cycle. It consists of three consistent views: a conceptual, a goal, and an executable protocol models. It supports an SLA preparation, agreement and monitoring. Our reference model contains points of changes and can be used for collecting the variable parts of SLA-contracts.

In order to guarantee methodological triangulation, we use more than one method [13] for gathering concepts and their relations and understanding of an SLA.

- Section 2 presents the results of a literature review used to built a first conceptual reference model of an SLA.
- Section 3 discusses the analysis of existing SLA documents and identification of different types of some concepts of the conceptual reference model.
- Section 4 validates the relations of the conceptual reference model by the strategic goal model of an SLA.
- Section 5 validates the relations of the conceptual reference model by the executable protocol model showing SLA development, acceptance and monitoring.
- Section 6 discusses the variation points of the reference models.
- Section 7 concludes the paper and proposes future work.

## 2 Literature Review for Building the First Conceptual Model of an SLA

The presented literature review has been aimed to identify the concepts used for description of an SLA and relations of these concepts. We discuss the concepts and relations named in literature. We cover the works on SLA reported in journals “Decision support systems”, “Future Generation Computer Systems” and “Performance evaluation” by Elsevier, “Journal of Network and Systems Management” and “Distributed and Parallel Databases” by Springer, IEEE Software journal and the Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering. In the following text, the concepts and their relations found in literature are presented in *italic* and shown in Fig. 1.

The definitions of an *SLA* found in literature can be classified as external and internal.

External definitions specify the essential condition for an *SLA* to exist. External definitions see an *SLA* as a technical contract that *legally binds* two *parties* being *providers* and *consumers* [1, 5].

Internal definitions define the content of an *SLA* in form of *producer requirements* and *consumer requirements* [11], explicitly distinguishing between a *Service*, its *functional* and *non-functional requirements*, where the latter are expressed as *Service Level Objectives (SLO's)*.

The functional part of a *Service* (Fig. 1) consists of a *Consumer process (request)* and a *Producer process (reply)* [9,16]. The consumer sends one or more key values identifying the data of interest. The producer then sends a reply containing the requested data or an indication that the data is not available. The *non-functional requirements* of the *Consumer* and *Provider* are grouped in *Service Level Objectives SLO's* [5]. Each *SLO* is refined to a set of *Quality Of Service (QoS)* with an associated *QoS Level constraining the QoS* [4]. For example, a consumer can demand a maximum response time, where a producer might want to limit the maximum number of service requests per time interval to a maximum (i.e. maximum throughput). Both of these *SLO's* define performance constraints that are included into an *SLA*. For monitoring of quality, the performance of the real service should be compared with these constraints.

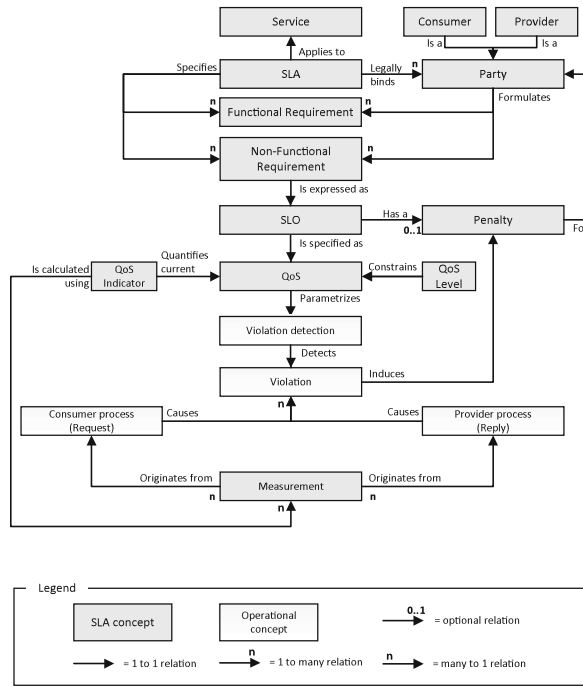


Fig. 1. Concepts defining an SLA in literature

The inability to meet specified *SLO's* are the reasons for *Penalties*, which can be *induced* on *Consumer* and *Provider* if a *Violation* is caused [10]. *Penalties* are seen as an essential part of an *SLA*, since they add a stimuli avoiding extra

costs if an *SLO* is not met [11]. A *violation* can also have a resolution, used to neutralize the *violation*. A violation can be sent to “the single point of contact for end-user problems” [3]. The resolution is not included into our conceptual model as it often demands investigation and such help desk issues are out of the scope of an SLA life cycle.

For monitoring of an *SLA* on a *producer process* and a *consumer process*, *QoS indicators* are specified. The literature does not differentiate the indicators. It is mentioned that some process specific *measurements* are needed to calculate the indicators, which can originate from producer and consumer process characteristics [10]. *Violation detection* is conceptually based on rules, which are decision functions, parameterized with *QoS Indicators* and *QoS levels* for the consumer and producer related *QoS*. It signals the current existence of *Violations*.

Each *QoS indicator* corresponds to the *QoS* specified for an *SLO* and uses specified *QoS level* acceptable for the consumer and the producer.

From our conceptual model (Fig. 1), an SLA can be defined as a *legal binding* between a *producer* and a *consumer* specifying agreed service functionality (*functional requirements*) and *non-functional requirements* split into *service level objectives (SLO's)*. Each SLO indicates an aspect of quality of *service (QoS)*, has specified *quality level (Qos level)* and corresponding *QoS indicators*. Each pair of *QoS indicator* and (*Qos level*) is used to calculate a *penalty*.

## 2.1 Choice of the Document Analysis as the Next Research Method

The conceptual model (Fig. 1) has been constructed on the basis of the literature analysis. All the papers on an SLA modelling remain abstract about SLO's [2, 4, 5, 10] and this is reflected by our conceptual model. A close look at the found definition shows that the concept *Service Level Objectives (SLO's)* is not ready to be included into a supportive system for SLA specification and implementation. The types of SLO need to be discovered and the relations between the SLO types and QoS Indicators need to be identified. Also the functional requirements can be made more specific for the type of provided service. To find the types of SLO used in practice, we initiated a document analysis study.

## 3 Document Analysis of Existing SLA's

The document analysis has been fulfilled in the SLA's used by government organizations<sup>1</sup>. We have found the SLO's of three types:

1. SLO Volume per year with the Volume norm that should not be exceeded;
2. SLO Response Time in hours with
  - (a) Percentage of the on-time responses per year;
  - (b) Percentage of the late responses per year;
  - (c) Percentage of the too-late responses per year;

<sup>1</sup> We are not allowed to name the organizations, however we have the documents for revision.

### 3. SLO Data quality with the norm of data with faults;

Each of these norms-constraints has similar structure: a name, a norm and a period of measurement. An SLO presents an aspect of monitoring, i.e. the QoS-norm. For example, the “Response Time” is specified as one working day (8 h). It is accepted if the 92% of responses per year are on-time, 7% of late responses are within five days and 1% of too-late responses are within 10 days.

The SLA’s define also the data formats in requests and replies that can be controlled both on the provider and on the consumer sides.

The document analysis of existing SLA’s shows that an SLA is related with three aspects of service support: (1) security, (2) data controls, and (3) monitoring of Quality of Service (QoS).

The security deals with concept *Request*. Data controls are related to one pair of concepts *Request-Reply*. The security and data format controls are internal for the service producer. The consumer recognizes them as delays and delays are included into QoS of an SLA.

The monitoring of SLOs with QoS concerns both the producer and the consumer, it should be understood by both parties.

### 3.1 Choice of Goal and Protocol Modeling as the Next Research Method

The logic of QoS measurement cannot be exposed in a reference model. The logic of QoS measurement combines the strategic agreements, operations of measurement and monitoring and decision about penalties.

In order to combine the strategic agreements, operations of measurement and monitoring and decision about penalties, we define a pair of corresponding goal and executable protocol models. Both models use the concepts of the conceptual reference model of an SLA (Fig. 1). The choice of the modelling techniques is motivated by the observed resemblance between the monitoring KPI’s and SLO’s with QoS. The KPI’s have been already successfully modelled with pairs of semantically related goal and executable protocol models [12,14]. We use this experience for modelling of monitors for SLO’s with QoS.

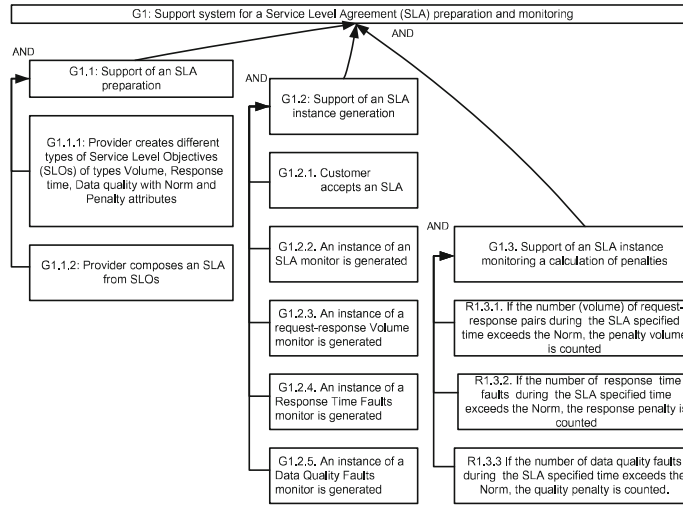
## 4 Goal Model of an SLA Life Cycle

Any service based business needs a supportive system for an SLA preparation and monitoring. This is the main goal of the goal model. *G1.Support system for a Service Level Agreement (SLA) preparation and monitoring.*

Now we present a description of the goal model depicted in Fig. 2.

*G1* is refined by the following sub-goals:

- *G1.1. Support of an SLA preparation;*
- *G1.2. Support of a service with an SLA instance generation*
- *G1.3. Support of an SLA instance monitoring a calculation of penalties*



**Fig. 2.** Goal model of an SLA life cycle

*G1.1* is refined with sub-goals

- *G1.1.1. Provider creates different types of Service Level Objectives (SLOs) of types Volume, Response time, Data quality with Norm and Penalty attributes;*
- *G1.1.2. Provider composes an SLA from SLOs.*

*G1.2.* is refined with

- *G1.2.1. Customer accepts an SLA;*
- *G1.2.2. An instance of an SLA monitor is generated;*
- *G1.2.3. An instance of a request-response Volume monitor is generated;*
- *G1.2.4. An instance of a Response Time Faults monitor is generated;*
- *G1.2.5. An instance of a Data Quality Faults monitor is generated.*

*G1.3.* is refined with requirements for the monitoring

- *R1.3.1. If the number (volume) of request-response pairs during the SLA specified time exceeds the Norm, the penalty volume is counted;*
- *R1.3.2. If the number of response time faults during the SLA specified time exceeds the Norm, the response penalty is counted;*
- *R1.3.3. If the number of response time faults during the SLA specified time exceeds the Norm, the response penalty is counted.*

All concepts in requirements are countable and comparable, the norms can be built into the SLA monitors in correspondence with the SLA.

## 5 Protocol Model of an SLA

### 5.1 The Behaviour of the Information Sharing Service

The behaviour of the information sharing service is a request and the corresponding reply, where both the request and reply are specified with a data structure.

- Each request structure contains *request identifier*, *request time stamp*, *key field name for information search* and *name of requested data item*. For example, (*request identifier*, *day-month-year*, *identification number of a citizen*, *citizen related data (year-1)*).
- Each reply structure contains *reply identifier*, *initiated request identifier*, *request time stamp*, *reply time stamp*, *key field for information search from the request name of data item in the request* and *value of the requested data item*.
- The attributes of these data structures are used to measure the data quality, data volume and response time. Each field of a request and a reply has type quality borders and the quality checks are implemented in the service.
  - The number of faults of quality checks of replies indicates *Data Quality* of the service.
  - The number of requests in a given time period is called *Volume* of the service.
  - For each reply, the difference between the “reply time stamp” and the “request time stamp” is called *Response Time*.

### 5.2 What Is a Protocol Model?

The executable form of a protocol model is textual [15]. It specifies concepts as protocol machines presenting OBJECTS (concepts) and BEHAVIOURS (constraints) with their attributes, states, recognised events, transitions and callbacks for updates of attributes and derived states. Because a protocol model is an executable model of an information exchange service behaviour with data, it is a suitable model to illustrate and demonstrate the monitoring logic of quality indicators.

The graphical form of a protocol model is used for communication and model explanation. It illustrates the protocol machines (concepts and constraints), recognised events, states and transitions. It does not show the data structures of protocol machines and events.

A protocol machines are composed using the CSP composition for machines with data defined in [6, 8]. The CSP composition means that all protocol machines are synchronised, i.e. an event is accepted by a protocol model only if all protocol machines recognizing this event are in the state to accept it. The state of each protocol machine is a data structure. Any event in a protocol model is another data structure. The data from the event-instance is used to update the state of protocol machines accepting this event.



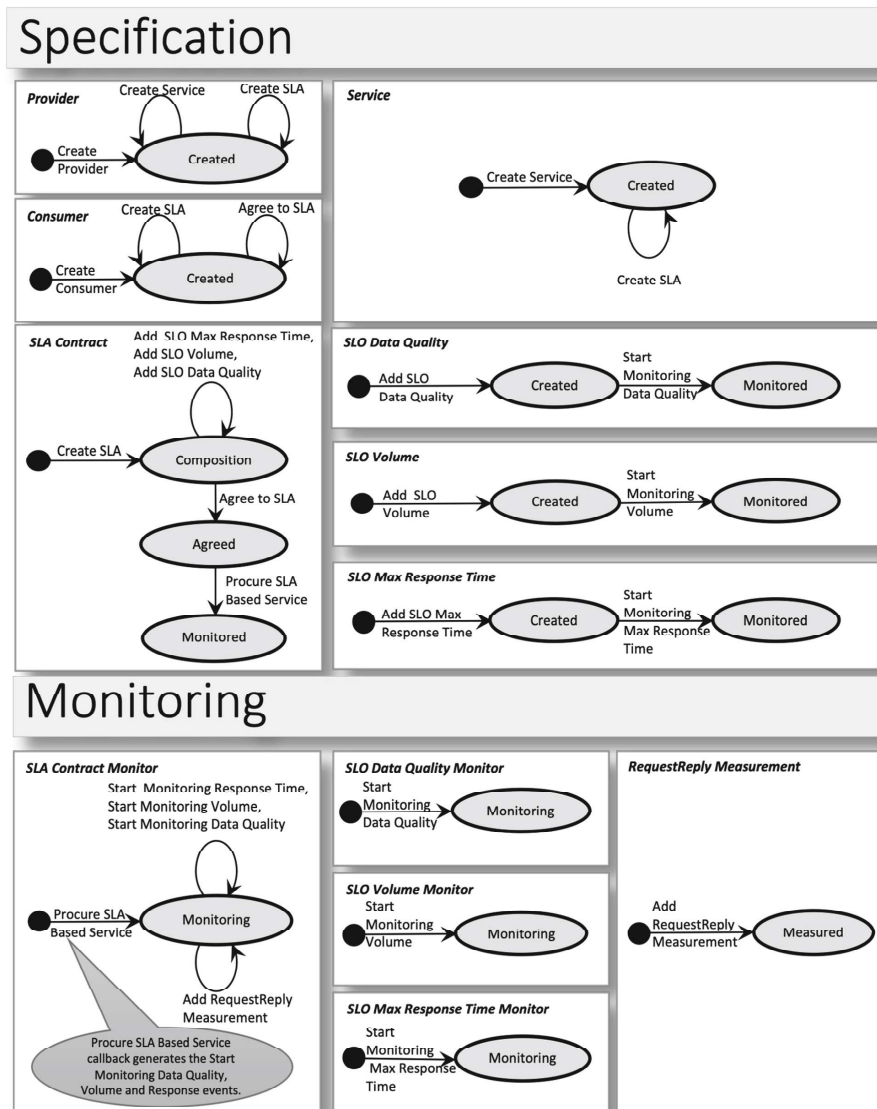


Fig. 3. Protocol model of SLA life cycle

### 5.3 Protocol Model of an SLA

In Fig. 3, the reader can see the graphical presentation of protocol machines *Provider*, *Consumer* and *Service* corresponding the concepts of the conceptual model (Fig. 1) and presenting behaviour of objects of those concepts. All these concepts should be in state *Created* to enable event *Create SLA* contract.

The protocol machine *SLA Contract* in state *Composition* allows one to *Add SLO Volume*, *Add SLO Max Response Time* and *Add SLO Data Quality*. Each SLO is a protocol machine that can be in the state “does not exist” (depicted as a small black circle), in state *Created* or in state *Monitored*.

When a *Provider* and a *Consumer* accept event *Agree to SLA*, the *SLA contract* goes to state *Agreed*. State *Agreed* reflects the agreement and fixes the *SLA Contract* preventing further SLO’s to be added. In the state *Agreed*, event *Procure SLA Based Service* is enabled. This event transits the *SLA Contract* to state *Monitored*.

An *SLA Contract Monitor* is automatically created by the event *Procure SLA Based Service*. The *SLA Contract Monitor* delegates SLO specific monitoring to several SLO specific monitors. This event also triggers submitting event instances *Start Monitoring Data Quality*, *Start Monitoring Volume* and *Start Monitoring Max Response Time*. These events create the specialised SLO monitors. To simulate service utilisation, the object instances of *RequestReply Measurement* are created with events *Add RequestReply Measurement*.

*Providers* and *Customers* should implement monitors of SLO’s. We model the executable monitors of SLO’s to show them to *Producers* and *Customers* before actual implementation. This helps to prevent unexpected penalties and misunderstandings. Each SLO is monitored by successively calculating an *Indicator Value*, the occurrence of a *Violation* and a *Payable Penalty*.

The executable textual form of the protocol model is available to be downloaded from [15], be executed in the Modelscope tool available online and the monitors are generated from the model. We show the three types of monitors and the callbacks code used by monitors to update data.

#### 5.4 Monitoring SLO Max Response Time

The values to monitor the *SLO Max Response Time* are shown in Fig. 4. It is monitoring the last measured response time, the maximum response time is 1 day and the penalty of 3,000,00, if violated.

The monitoring logic for the *SLO Max Response Time* is shown in the code fragment below. Function *SLA Contract Monitor.getIndicatorValue()* returns the *Response Time* attribute specified by the youngest *RequestReply Measurement* instance. Function *getViolation()* evaluates the relation *Response Time > Norm Value*. Function *getPayablePenalty()* calculates the *Payable Penalty*.

```
public class SLOMaxResponseTime extends Behaviour {
public int getPayablePenalty(){
// If no violation return no Penalty, i.e. zero
    if (! this.getBoolean("Violation")) return 0;
// Return penalty specified by SLO
    return this.getInstance("SLO").getCurrency("Penalty Value");
}
public boolean getViolation(){
// Get the last response time
    String duration = this.getString("Indicator Value");
// No last measurement, no violation
    if (duration == null) return false;
// Create classifier: > response norm and classify the current response time
```

Events

Add SLO Data Q ^  
 Add SLO Max Re  
 Add SLO Respon v

---

<b>SLO Name</b>	<input type="text" value="Max Response For C"/>	String
<b>Norm Value</b>	<input type="text" value="1"/>	String
<b>Norm Unit</b>	<input type="text" value="Last response time"/>	String
<b>Penalty Value</b>	<input style="text-align: right; border-right: 1px solid black; border-bottom: 1px solid black; border-left: 1px solid black; border-top: 1px solid black; width: 80px;" type="text" value="3000.00"/> x	Currency
<b>SLO Max Response Time</b>	<input type="text" value="(new SLO Max Response Time)"/>	SLO Max Response Time

**Fig. 4.** Adding the SLO Max Response. Generated from the protocol model [15] in Modelscope tool [7].

```

return new ClassifierDurationFromSLO( this.getInstance("SLO"), ">").classifies(duration);
}
public String getIndicatorValue(){
// Get the last request/reply measurement
    Instance measurement =
        this.getLastMeasurement(this.getInstance("SLA Contract Monitor"));
// Return the response time, formatted as: days (HH:mm:ss.sss)?
    if (measurement == null) return null;
    return measurement.getString("Response Time");
}}

```

## 5.5 Monitoring SLO Volume

Figure 5 shows a monitor for the *SLO Volume*. It evaluates the number of requests over the last 365 days. During this period a maximum of 12,500 data items may be requested. As long as

$$(Indicator\ Value \leq Norm\ Value) \text{ is true,}$$

no violation exists. If violated, an extra 100.00 is paid as a *Penalty* for every excess data item requested.

Function *getIndicatorValue()* calculates the cumulative number of data items requested during a period defined by the *Period Value* attribute of the SLO. This is calculated as the sum of the *RequestReply Measurement*, found in its *Volume* attribute. Only request volumes are cumulated that fall within the specified period. Function *getPayablePenalty()* identifies every excess data over the *Norm* and calculates *Payable Penalty*, proportional to the excess volume measured for the period.

```

public class SLOVolume Time extends Behaviour {
    public int getPayablePenalty() {
// If no violation, then no Penalty

```

Events

^  
 ^  
 v

<b>SLO Name</b>	<input type="text" value="Volume For C"/>	String
<b>Period Value</b>	<input type="text" value="365"/>	String
<b>Norm Criteria</b>	<input "="" type="text" value="&lt;="/>	String
<b>Norm Value</b>	<input type="text" value="12500"/>	Integer
<b>Norm Unit</b>	<input type="text" value="Data items"/>	String
<b>Penalty Value</b>	<input type="text" value="100.00"/>	Currency
<b>SLO Volume</b>	<input type="text" value="(new SLO Volume)"/>	SLO Volume

**Fig. 5.** Adding the SLO Volume. Generated from the protocol model [15] in Modelscope tool [7].

```

    if (! this.getBoolean("Violation")) return 0;
    // Get the associated SLO specification and
    // Calculate excess volume and proportional penalty
    Instance slo = this.getInstance("SLO");
    return (this.getInteger("Indicator Value") - slo.getInteger("Norm Value"))
    * slo.getCurrency("Penalty Value");
  }
  public int getIndicatorValue() {
    // Get only measurement for a specific SLA within the period and sum the measured volumes.
    return sum( this.getInstance("SLA Contract Monitor"),
    this.getInstance("SLO").getString("Period Value"),"Volume");
  }
  protected int sum(Instance slaMonitor, String period, String
  measurementObjectName, String measurementAttributeName) {
    // The every individual attribute value to the totalValue.
    int totalValue = 0;
    for (Instance measurement: this.getMeasurementsInPeriod(slaMonitor,
    period, measurementObjectName))
    totalValue += measurement.getInteger(measurementAttributeName);
    // Return the sum calculated in totalValue
    return totalValue;
  }
  //
  protected List<Instance> getMeasurementsInPeriod(Instance slaMonitor,
  String period, String measurementObjectName) {
    // For every measurement that matches the SLA Contract Monitor
    long periodStart = System.currentTimeMillis() - new Duration(period).getTime();
    List <Instance>measurements = new ArrayList<Instance>();
    for (Instance measurement:
    slaMonitor.selectByRef(measurementObjectName,"SLA Contract Monitor")){
    // If the measurement is within the period add it to the list
    if (periodStart <= AbstractMeasurement.getTimeMillis(
    measurement.getString("Time Stamp"))) measurements.add(measurement);
    }
    // Return the list of measurements for the SLA Contract monitor within the period
    return measurements;
  }
  }}

```

## 5.6 Monitoring SLO Data Quality

Figure 6 shows the example SLO Data Quality specification. A monitor instance for an SLO Data Quality is calculated over a period of 365 days. A maximum of 1% data quality issues are allowed for the data items delivered by the provider. The SLO is violated if the percentage of data quality issues exceeds 1% during the period. If violated, a Penalty of 1,000.00 has to be paid once.

The screenshot shows a web interface for adding an SLO Data Quality. At the top, there is a section titled "Events" with a dropdown menu containing three options: "Add SLO Data Q...", "Add SLO Max Re...", and "Add SLO Respon...". Below this is a form with the following fields:

- SLO Name:** "Data Quality For C" (String)
- Period Value:** "365" (String)
- Norm Criteria:** "<=" (String)
- Norm Value:** "1" (Integer)
- Norm Unit:** "% data quality issues" (String)
- Penalty Value:** "1000.00" (Currency)
- SLO Data Quality:** "(new SLO Data Quality)" (SLO Data Quality)

At the bottom of the form, there are two buttons: "Add SLO Data Quality" and "reset form".

**Fig. 6.** Adding the SLO Data Quality. Generated from the protocol model [15] in Modelscope tool [7].

Function *getIndicatorValue()* in this monitor calculates the percentage of data quality issues during the monitoring period. The data quality issues and the entire data population are cumulated a to calculate this percentage.

```
public class SLODataQuality extends Behaviour {
    public int getIndicatorValue() {
        // Get the related SLO specification, Get the SLA that is being monitored and
        // get the period that is monitored
        Instance slo = this.getInstance("SLO");
        Instance sla = monitor.getInstance("SLA Contract Monitor");
        String period = slo.getString("Period Value");
        // Get the measurements for the SLA within the period and
        // sum the value of attribute Data Quality Issues
        int value = sum(sla, slo.getString("Period Value"), "Data Quality Issues");
        // Sum the total volume of data items requested
        if (value == 0) return 0;
        int total = sum(sla, period, "Volume");
        // Calculate percentage of data quality issues
        return Math.round( 100f * (float)value / (float)total );
    }
}
```

## 6 SLO-Concept as a Model Variation Point

In Sect. 3 we mentioned our study of existing SLA's and their SLO's. Almost all SLO's fit in three types *SLO Response Time*, *SLO Data Quality* and *SLO Volume*. Some of them can be seen as composition of several SLO's of a given type. For example, we have found *SLO Classified Response Time*, which is composed from four *SLOs Response time*. Response times in this SLO are classified into four categories: *on time*, *late*, *too late*, and *far too late*. Response times of requests are collected over a period of time and a category are expressed as a percentage of responses. We have used this compositional SLO to validate our reference model. The reference model with this SLO remains the same, but the monitoring logic is specified for each category of response times.

We also have found one SLA that includes an *SLO Mean Time To Repair* used to monitor resolving service disturbances reported by the consumer to the single point of contact of the service. The concept *disturbance* is outside of our reference model. This SLO can be included into an SLA only if the service provider is able to repair disturbances.

## 7 Conclusion

The information exchange services are often used by non-technical businesses and they need a reference model for preparation, monitoring and reviewing. In this paper, we have presented a version of a reference model that shows the concepts, the goals and the executable protocol of an SLA monitoring. The reference concepts, goals and an executable model contribute to the understanding of the designed SLA.

The concepts and their relations have been found via a literature review. The concepts of Service Level Objectives (SLO's) have been refined using the document analysis of the existing SLA's in domain of information exchange services. In the domain of Information Sharing Services, three main types of SLO's (Volume, Response Time and Data Quality) have been identified. These types of SLO's are used to structure the process of SLA preparation and monitoring. The logic of SLO's indicators, violation and penalty calculation can be reused and composed for different SLO's.

Keeping an SLA alive is considered as one of the issues in organizations. The goal and protocol models can be used for demonstration of an SLA for the customers and providers both before their agreement, and during the service utilizing. The models show the logic of the SLO measures, indicators and penalty calculation. An executable protocol model transforms an SLA documentation into a part of management process and, therefore, contributes to active use of the SLA for reviews, assessment of targets and planning.

In the future work, our reference model, built for the information exchange services, can be validated in different service domains. The reference model can be also useful in context of help desk processes in organizations, as SLA's are often guaranteed by several departments and the customer expects the declared quality of service provided by several departments.

## References

1. Blake, M.B., Cummings, D.J., Bansal, A., Bansal, S.K.: Workflow composition of service level agreements for web services. *Decis. Support Syst.* **53**(1), 234–244 (2012)
2. Emeakaroha, V.C., et al.: Towards autonomic detection of SLA violations in Cloud infrastructures. *Future Gener. Comput. Syst.* **28**(7), 1017–1029 (2012)
3. Gartner Glossary: Gartner (2019). <https://www.gartner.com/en/information-technology/glossary/sla-service-level-agreement>
4. Keller, A., Ludwig, H.: The WSLA framework: specifying and monitoring service level agreements for web services. *J. Netw. Syst. Manage.* **11**(1), 57–81 (2003)
5. Leitner, P., Ferner, J., Hummer, W., Dustdar, S.: Data-driven and automated prediction of service level agreement violations in service compositions. *Distrib. Parallel Databases* **31**(3), 447–470 (2013)
6. McNeile, A., Roubtsova, E.: CSP parallel composition of aspect models. In: *AOM 2008*, pp. 13–18 (2008)
7. McNeile, A., Simons, N.: (2011). <http://www.metamaxim.com/>
8. McNeile, A.T., Simons, N.: State machines as mixins. *J. Obj. Technol.* **2**(6), 85–101 (2003)
9. Menascé, D.A., Ruan, H., Goma, H.: QoS management in service-oriented architectures. *Perform. Eval.* **64**(7–8), 646–663 (2007)
10. Paschke, A., Bichler, M.: Knowledge representation concepts for automated SLA management. *Decis. Support Syst.* **46**(1), 187–205 (2008)
11. Raimondi, F., Skene, J., Emmerich, W.: Efficient online monitoring of web-service SLAs. In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 170–180. ACM (2008)
12. Roubtsova, E.: *Interactive Modeling and Simulation in Business System Design*. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-319-15102-1>
13. Roubtsova, E.: Categories of research methods and types of research results illustrated with a continuous project. In: *21st International Conference on Enterprise Information Systems*, pp. 634–641 (2019)
14. Roubtsova, E., Michell, V.: KPIs and their properties defined with the EXTREME method. In: Shishkov, B. (ed.) *BMSD 2013. LNBIP*, vol. 173, pp. 128–149. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-319-06671-4\\_7](https://doi.org/10.1007/978-3-319-06671-4_7)
15. SLA protocol model: Protocol Modelling (2019). [https://newprotocolmodelling.weebly.com/uploads/2/8/7/6/28769871/sla4\\_20191010.zip](https://newprotocolmodelling.weebly.com/uploads/2/8/7/6/28769871/sla4_20191010.zip)
16. Zimmermann, O.: Architectural decisions as reusable design assets. *IEEE Softw.* **28**(1), 64–69 (2011)