Journal of Universal Computer Science, vol. 20, no. 3 (2014), 366-384
submitted: 24/3/13, accepted: 4/2/14, appeared: 1/3/14 © J.UCS

Generating an Excerpt of a Service Level Agreement from a Formal Definition of Non-Functional Aspects Using OWL

Mariam Rady
(Johannes Kepler University Linz, Austria
m.rady@cdcc.faw.jku.at)

Abstract: If we take a look at current cloud computing services, the only quality guarantee they provide are vague Service Level Agreements(SLA). In this paper we modelled some non-functional aspects in an ontology and used this ontology as a knowledge base to generate an excerpt from a service contract. We concentrate in this excerpt on availability as it is one of the most discussed attributes in current Service Level Agreements.
Key Words: QoS, SLA, Contracting, Non-Functional Aspects, OWL, Ontology
Category: C.4, H.3.5, K.6.4, K.5.m

1 Introduction

Service Level Agreements (SLAs) define assertions of a service provider, that the service he is offering meets a certain guaranteed IT-Level and business-process level service parameters. In addition, the service provider should guarantee measures to be taken in the case these assertions were violated[Ludwig et al., 2003]. Current SLAs are vague and do not offer solid guarantees for customers, on which they can rely. If we take a look at how availability is promised in SLAs, it is usually presented as a percentage or rate value describing the total Uptime to the total service agreement time. In general, this is how the percentage is calculated:

$$\frac{TotalTime - DownTime}{TotalTime}$$

For storage services the error rate in a reference interval i is calculated as follows:

$$errorRate_i = \frac{\#FailedRequests_i}{\#TotalRequests_i},\ i >= 0$$

$$MonthlyUptimeRate = 100\% - \frac{\sum_{i=1}^{n} errorRate_i}{n}$$

It is the customer's responsibility to send claims of the downtime incidents, and the service provider checks if the claims are true. If the promised uptime percentage or rate is not met, then the customer gets a compensation in form of service credit[Amazon, 2008].

This is however, not a reliable guarantee for Quality of Service (QoS). In order for the customer to be able to rely on the Quality of the Service, quality attributes need to be well-described in SLAs. Their descriptions should be semantically complete and have a logical structure, in order to allow reasoning about and processing the information in the agreements. In this work, we define some non-functional aspects formally using the Web Ontology Language OWL. Our focus is on the automatic or semi-automatic generation of the cloud service agreements using ontologies. We start in section 2 with the background information about Description Logic and OWL, the formal language used to define this model. In section 4 we present the actual model, the different concepts and the relationships between them. Section 5 describes how we used different probability distribution functions to express service availability in the contract. Then in section 6 we present how we query our model to generate the excerpt of the contract. And finally in section 7 we conclude our work.

2 Background Information

2.1 Description Logic and OWL

Description logics(DLs) are a family of formal knowledge representation languages and it is of particular importance in providing a logical formalism for ontologies and the Semantic Web. A DL models concepts, roles and individuals and their relationships. There are varieties of DL families. We are concerned with $\mathcal{SHOIN}^{(\mathcal{D})}$ and $\mathcal{SROIQ}^{(\mathcal{D})}$ on which OWL 1.0 and OWL 2.0 are based, respectively[Krötzsch et al., 2012][Baader, 2003]. For the different constructs in both languages there are equivalent constructs in OWL.

The goal of the Semantic Web is to have machines understand Web content possibly without human interaction, by facilitating automated processing of descriptions on the Web. Annotations on the Semantic Web express links between information resources on the Web and connect information resources to formal terminologies, these connective structures are called ontologies. An ontology is a formal explicit specification of a shared conceptualization[Gruber, 1993]. It allows machine understanding of information through the links between the information resources and the terms in the ontology[Fensel et al., 2007].

Resource Description Framework (RDF), RDF-Schema and DAML+OIL were the first DL inspired languages to be integrated into the Semantic Web. W3C later updated the RDF recommendation and provided a formal semantic for RDF. W3C then defined the Web Ontology Language OWL to be compatible with RDF. W3C defined a family of three languages ranging from level of expressiveness (correspondence to DL) to degree of compatibility with RDF, these are OWL DL, OWL Full and OWL Lite[Baader, 2003].

In general, DL models concepts (classes), roles (properties), individuals and their relationships. We created an ontology to define various non-functional aspects, these are presented in section 4.

2.2 Literature Review

In this section we investigate existing models for SLAs. These are trying to find a general way to express all the quality aspects. Our efforts lie in trying to model the different quality aspects into depth, taking into account their individual meanings as well as putting in mind that the representation of the quality aspect should be monitorable, validatable and verifiable. The first model that we look at was developed by the SLA@SOI project. The SLA model in this project is concerned with modelling the physical structure of the document leaving out the intentional aspects of an agreement. The QoS term monitoring is in a later stage. The SLAs are then translated into operational monitoring specifications. Only on this level are the intentional aspects of the contract tackled through special engines for this purpose[Wieder, 2011].

Another way to model SLAs is using the SLA Language SLAng proposed by [Lamanna et al., 2003]. The model presented in [Lamanna et al., 2003] defines two abstraction levels for compiling SLA agreements. It differentiates between vertical and horizontal SLAs. Vertical SLAs are concerned with governing the service level of the underlying infrastructure, while the horizontal SLAs are between parties providing services on the same level. SLAng is an XML language for capturing SLAs. The SLA structure includes three main concepts; namely an end-point description of the contractors, contractual statements and Quality of Service (QoS) descriptions and associated measures. However, in their definition they do not go into depth in the definition of QoS aspects.

WSLA is a framework for SLA establishment and monitoring of SLAs. The contract has three sections Parties, Service Description and Obligations. The WSLA Language is XML-based. SLA parameters are specified with their measures. However, exact measures for different quality aspects are not the main focus of this work[Keller and Ludwig, 2003]. The SLA models reviewed in this section do not thoroughly describe individual QoS aspects, they either mention some of their attributes or adopt the notion that each quality aspect should have measures, but do not further discuss these measures. Our work is not only concerned with the physical structure of the document, or only stating QoS aspects and their different measures, however, it is also concerned with the intentions of the SLAs. We used the basic physical structure that was defined in previous work, namely stating the parties that are included in the contract, the different promises or commitments that the service is offering as well as some general information about the service itself. As we think these are the basic cornerstones that should be there in an SLA, according to previous research done. And we extended that to formally define intentional aspects of some non-functional terms in the SLA. Meaning that we do not only state non-functional aspects with their measures in the SLAs, but we also define what service availability means in the context of cloud computing.

3 Cloud Service Interaction Model

The first step in our approach was to figure out an abstract way to represent the interaction between the user and the cloud service, that would allow us to model service availability in the SLA in a way that it can be monitored. The idea is to model the cloud service as a set of requests submitted to the cloud by the user. And the user receives to each of these requests a certain response. Each request/response should be monitored and compared to a probability distribution function as described in section 5. Figure 1 shows the Cloud Service Interaction Model.

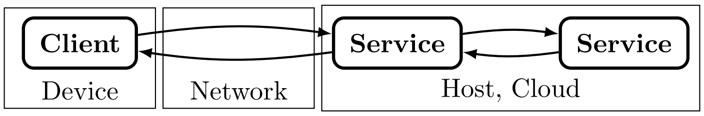

Figure 1: Cloud Interaction Model

Each Client can interact with a cloud service via a network. Each cloud service can be composed of a set of services. And the functionalities of the various services can be described using the different requests. The availability of these services can be monitored by observing the different requests and modelling its availability using probability distribution functions.

4 Representation of SLA using an ontology

In this section we will describe the basic structure and the main concepts that define the SLA. We used protégé (an open source ontology editor and knowledge-base framework) to construct the ontology. In protégé classes are equivalent to concepts. When defining an ontology, every class is a subclass of the class `Thing`. For the different concepts and roles we have drawn parts of the ontology, to facilitate the understanding of the reader. Concepts are represented using rectangles, datatypes are represented using ellipses, dashed lines show the relationship between the concepts and their subconcepts and the thick lines are representing datatype properties and object properties.

4.1 Structure of the SLA Document

The concepts that form the SLA document are `Parties`, `Commitments` and `Information` . These are defined in sections 4.3, 4.4 and 4.5, respectively. The structure of the SLA Document is shown in Figure 2.

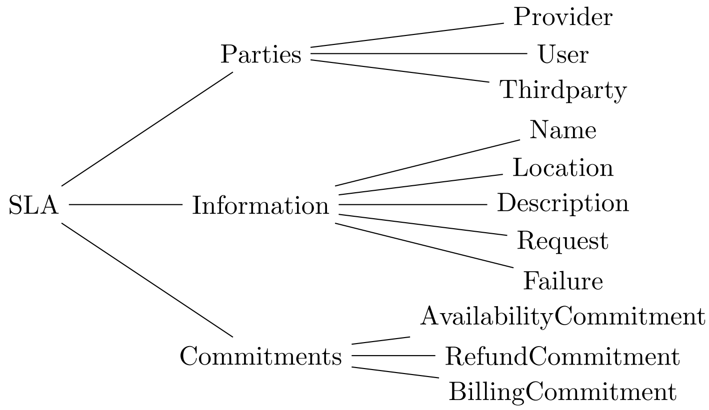

Figure 2: Basic Structure of the SLA Document

`Parties` define everyone who is involved in the SLA. `Information` describes the functionalities of the service, it lists all the possible requests to the service and the different failures. And `Commitments` are the non-functional aspects that are guaranteed by the SLAs.

4.2 Helpers-TemporalInformation

When thinking about contracts, one of the first things that comes to mind is the need to express temporal information. Using temporal information, we can express the start date and time of the contract and the end date and time, as well as when maintenance of the system takes place. OWL 2.0 has a built-in datatype called `datetime`, that specifies a certain instance of time. But we needed to extend the temporal information to cover also relative time. This is why we defined the following two concepts `Duration` and `Repetition`. This part of the ontology is shown in Figure 3.

4.2.1 Duration

This concept is used to describe a certain time duration or time interval. We define it in equation (1) using DL.

$$\begin{aligned} \mathtt{Duration} \sqsubseteq\ & \mathtt{TemporalInformation} \\ & \sqcap \exists \mathtt{hasDuration.Datatype(double)} \\ \mathtt{Duration} \equiv\ & \{\mathtt{Minutes}\} \sqcup \{\mathtt{Hours}\} \sqcup \{\mathtt{Days}\} \sqcup \\ & \{\mathtt{Weeks}\} \sqcup \{\mathtt{Months}\} \sqcup \{\mathtt{Years}\} \end{aligned} \tag{1}$$

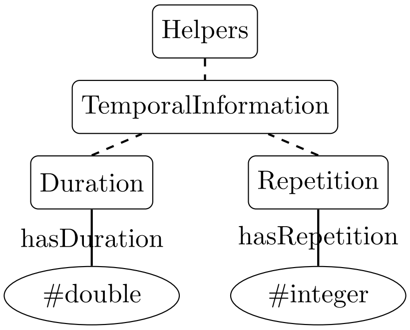

Figure 3: Temporal Information

4.2.2 Repetition

This concept has an integer that is representing multiples for expressing repetition (e.g. biweekly). It has the following individuals and is used to express if any event is repeated, e.g. Maintenance will take place on a certain date and will be repeated every two weeks.

$$\begin{aligned} \texttt{Repetition} &\sqsubseteq \texttt{TemporalInformation} \\ &\sqcap \exists \texttt{hasRepetition}.\texttt{Datatype}(\texttt{integer}) \\ \texttt{Repetition} &\equiv \{\texttt{Daily}\} \sqcup \{\texttt{Weekly}\} \sqcup \{\texttt{Monthly}\} \sqcup \{\texttt{Yearly}\} \end{aligned} \tag{2}$$

4.3 Parties

In this section we represent all the parties that are part of the service contract. This part of the ontology is shown in Figure 4. Each party has a name and a URI. Each SLA should have at least 2 parties involved. `Parties` can be either a `Provider`, a `User` or a `ThirdParty`.

$$\begin{aligned} \texttt{Parties} &\sqsubseteq \exists \texttt{hasName}.\texttt{Datatype}(\texttt{string}) \\ &\sqcap \exists \texttt{hasURI}.\texttt{Datatype}(\texttt{anyURI}) \end{aligned} \tag{3}$$

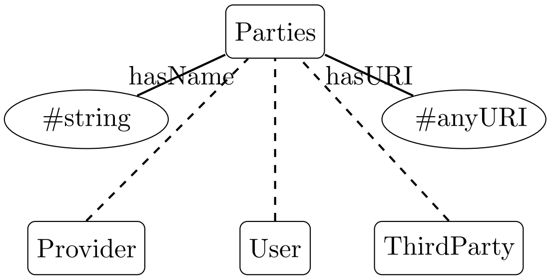

Figure 4: Parties

4.4 Commitments

In this section we define the different concepts for `Commitments` that are offered in the SLA. As we are discussing in this work availability as one of the main SLA commitments, the first concept that is defined is `AvailabilityCommitment`. We also define `RefundCommitments` and `BillingCommitments` of the SLA. The ontology can be then extended to cover other commitments, by adding an attribute `X` as a concept `XCommitments`.

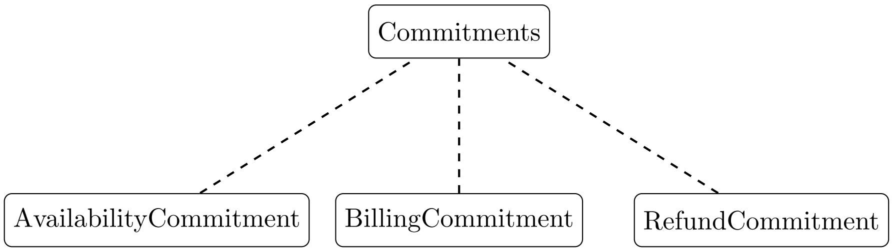

Figure 5: SLA Commitments

4.4.1 AvailabilityCommitment

It is the set of all commitments related to the attribute availability and is a subclass of `Commitments`. This part of the ontology is shown in Figures 6, 7 and 8.

$$\texttt{AvailabilityCommitment} \sqsubseteq \texttt{Commitments} \tag{4}$$

The subconcepts of `AvailabilityCommitment` are: `CommitmentValidity`, `MaintenanceTime`, `ProbabilityDistribution` and `MonitoringWindow`.

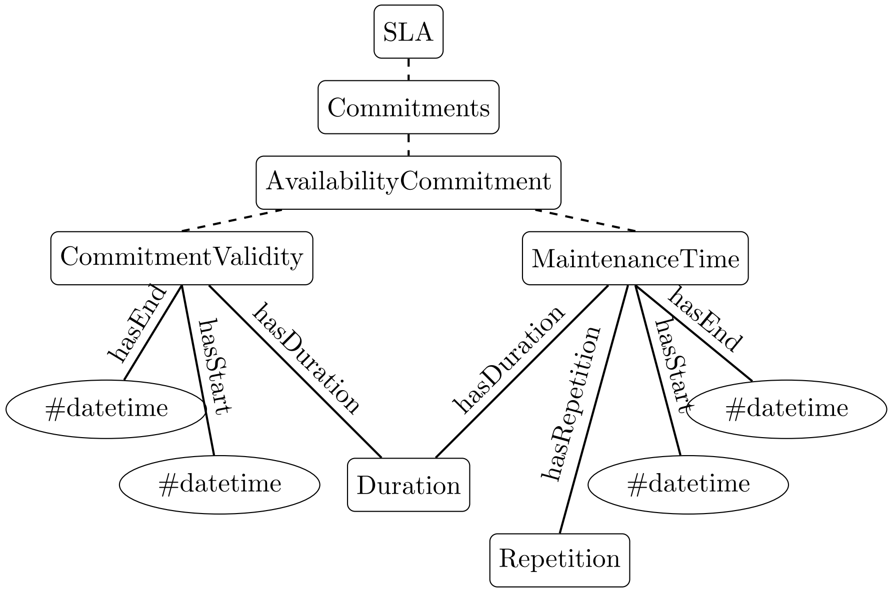

Figure 6: AvailabilityCommitment-CommitmentValidity and MaintenanceTime

`CommitmentValidity` is one of the subconcepts of `AvailabilityCommitment` and is defined to have a datatype property `hasStart` and `hasEnd` to the datatype `datetime`.

$$\begin{aligned}\texttt{CommitmentValidity} \sqsubseteq & \texttt{AvailabilityCommitment} \\ & \sqcap((\exists \texttt{hasStart.Datatype}(\texttt{datetime}) \sqcap \\ & \exists \texttt{hasEnd.Datatype}(\texttt{datetime})) \\ & \sqcup (\exists\ \texttt{hasStart.Datatype}(\texttt{datetime}) \sqcap \\ & \exists \texttt{hasDuration.Duration}))\end{aligned} \tag{5}$$

`MaintenanceTime` is another commitment that needs to be agreed on. It decides when maintenance will take place, because when maintenance takes place, the service might be unavailable for some time. `MaintenanceTime` is a subconcept of `AvailabilityCommitment`. `MaintenanceTime` is defining a start time and an end time for the maintenance. It has a role `hasStart` and `hasEnd` to the datatype `datetime`. In addition it has a relationship to the concept Repetition using the role `hasRepetition` defining how maintenance is scheduled. For relative time we define a start date and time and the duration of the maintenance.

$$\begin{aligned}
\texttt{MaintenanceTime} \sqsubseteq & \texttt{AvailabilityCommitment} \\
\texttt{MaintenanceTime} \sqsubseteq & (\exists \texttt{hasStart.Datatype}(\texttt{datetime}) \sqcap \\
& \exists \texttt{hasEnd.Datatype}(\texttt{datetime}) \\
& \sqcap \exists \texttt{hasRepetition.Repetition}) \\
& \sqcup (\exists \texttt{hasStart.Datatype}(\texttt{datetime}) \sqcap \\
& \exists \texttt{hasDuration.Duration} \\
& \sqcap \exists \texttt{hasRepetition.Repetition})
\end{aligned} \tag{6}$$

`ProbabilityDistribution` is a subconcept of `AvailabilityCommitment` and is defined as follows:

$$\begin{aligned}
\texttt{ProbabilityDistribution} \sqsubseteq & \texttt{AvailabilityCommitment} \\
\texttt{ProbabilityDistribution} \sqsubseteq & \exists \texttt{hasFormula.Datatype}(\texttt{string}) \\
\texttt{ProbabilityDistribution} \sqsubseteq & \exists \texttt{hasParameter.Datatype}(\texttt{string}) \\
\texttt{ProbabilityDistribution} \equiv & \{\texttt{ChiSquare}\} \sqcup \{\texttt{Gamma}\} \sqcup \\
& \{\texttt{Exponential}\} \sqcup \{\texttt{HyperExponential}\} \sqcup \\
& \{\texttt{Lognormal}\} \sqcup \{\texttt{Pareto}\} \sqcup \\
& \{\texttt{Normal}\} \sqcup \{\texttt{Weibull}\}
\end{aligned} \tag{7}$$

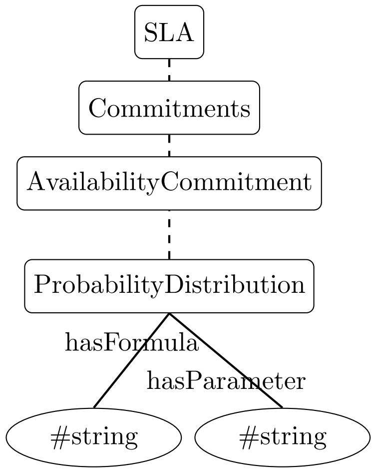

Figure 7: AvailabilityCommitment-ProbabilityDistribution

`MonitoringWindow` is the duration of time to which the availability commitment applies. `MonitoringWindow` is defined as

$$\begin{aligned} \texttt{MonitoringWindow} \sqsubseteq & \texttt{AvailabilityCommitment} \\ & \sqcap \exists \texttt{hasDuration}.\texttt{Duration} \end{aligned} \tag{8}$$

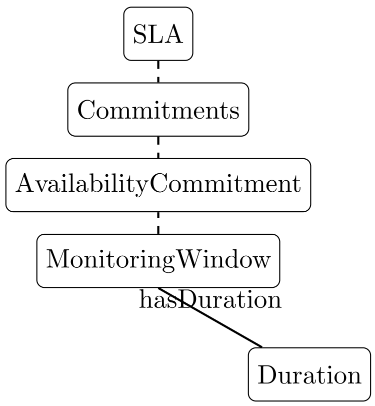

Figure 8: AvailabilityCommitment-MonitoringWindow

4.4.2 RefundCommitment

`RefundCommitment` are the terms for getting a compensation in case the SLA is not met. The `RefundCondition` is the condition for the customer to receive a refund. If the service provides less availability than promised, the customer is entitled to receive a `RefundPercentage`. `RefundCondition` and `RefundPercentage` are both defined to have a value of type double. This part of the ontology is shown in Figure 9.

$$\begin{aligned}\texttt{RefundCommitment} &\sqsubseteq \texttt{Commitments}\\ \texttt{RefundCondition} &\sqsubseteq \texttt{RefundCommitment}\\ &\sqcap \exists \texttt{hasCondition.Datatype}(\texttt{string})\\ &\sqcap = 1\ \texttt{hasRefundPercentage.RefundPercentage}\\ \texttt{RefundPercentage} &\sqsubseteq \texttt{RefundCommitment}\\ &\sqcap \exists \texttt{hasRefund.Datatype}(\texttt{double})\end{aligned} \tag{9}$$

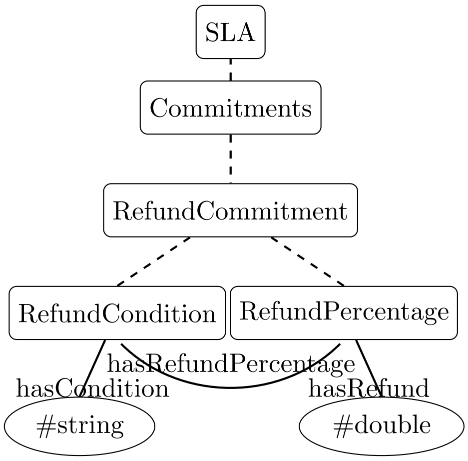

Figure 9: RefundCommitment

4.4.3 BillingCommitment

This concept represents the billing information. It has two subconcepts `Payment` and `Price`. `Payment` is a concept that defines where the payment is going to be made using a relation `hasURI` to the datatype `anyURI`. In addition it has a start date and time as well as a `Repetition` defining when the payment has to be made. The concept `Price` defines the price that has to be paid for the service. And the concept `Currency` defines the currency that is used to pay for the service. This part of the ontology is shown in Figure 10.

$$
\begin{aligned}
\texttt{BillingCommitment} \sqsubseteq & \texttt{Commitments} \\
\texttt{Payment} \sqsubseteq & \texttt{BillingCommitment} \\
\texttt{Payment} \sqsubseteq & \exists \texttt{hasURI.Datatype(anyURI)} \\
& \sqcap \exists \texttt{hasStart.Datatype(datetime)} \\
& \sqcap \exists \texttt{hasRepetition.Repetition} \\
\texttt{Price} \sqsubseteq & \texttt{BillingCommitment} \\
& \sqcap \exists \texttt{hasPrice.Datatype(double)} \\
& \sqcap \exists \texttt{hasCurrency.Currency} \\
\texttt{Currency} \sqsubseteq & \texttt{BillingCommitment} \\
\texttt{Currency} \equiv & \{\texttt{Dollar}\} \sqcup \{\texttt{Euro}\} \sqcup \{...\}
\end{aligned}
\tag{10}
$$

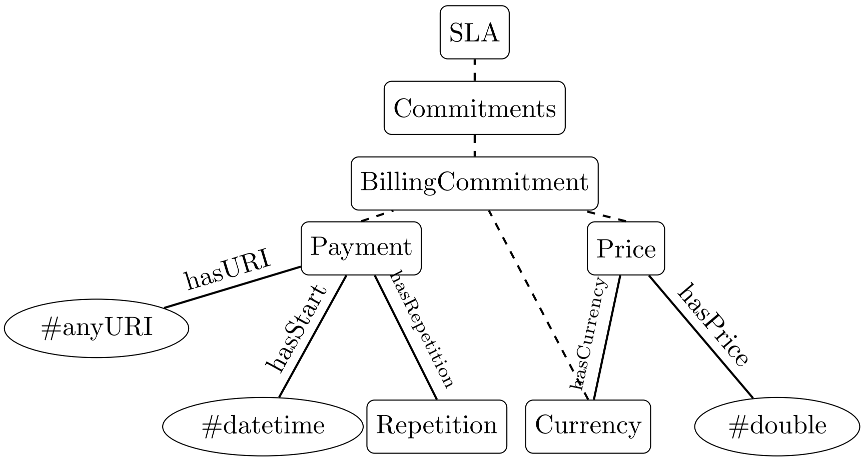

Figure 10: BillingCommitment

4.5 Information

`Information` is representing general information about the service and is a subtype of `SLA`. It includes different subconcepts. This part of the ontology is shown in Figure 11.

$$Information \sqsubseteq SLA \tag{11}$$

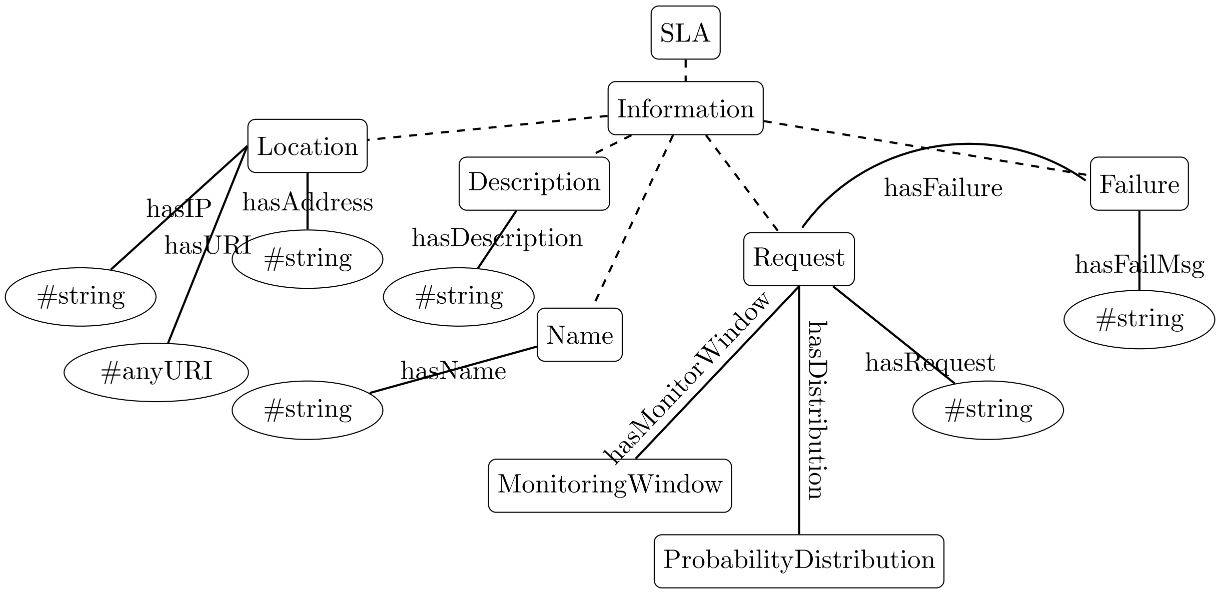

Figure 11: Information

4.5.1 Description

This is a string defining what the service is used for.

$$\begin{aligned}\texttt{Description} \sqsubseteq & \texttt{Information} \\ & \sqcap \exists \texttt{hasDescription.Datatype}(\texttt{string})\end{aligned} \tag{12}$$

4.5.2 Location

Any offered service should have a location. Not only the physical location to get information about jurisdiction but also the URL or IPAddress under which the service is available and accessible. Using local information we can have the information where the service can be requested from.

$$\begin{aligned}\texttt{Location} \sqsubseteq & \ \exists \texttt{hasAddress.Datatype}(\texttt{string}) \\ & \sqcap \exists \texttt{hasIP.Datatype}(\texttt{string}) \\ & \sqcap \exists \texttt{hasURI.Datatype}(\texttt{anyURI})\end{aligned} \tag{13}$$

4.5.3 Failure

This concept describes the different failures or errors the service can show.

$$\begin{aligned}\texttt{Failure} \sqsubseteq & \texttt{Information} \\ & \sqcap \exists \texttt{hasFailMsg.Datatype}(\texttt{string})\end{aligned} \tag{14}$$

4.5.4 Request

It describes the different requests that the service can make to different resources. The availability of a resource, can be modelled by a probability distribution function representing whether the resource was able to respond to the request made by the service user. This is how we used the cloud service interaction model, that was mentioned in section 3, in the ontology.

$$\begin{aligned}\texttt{Request} \sqsubseteq & \texttt{Information} \\ & \sqcap \exists \texttt{hasRequest.Datatype}(\texttt{string}) \\ & \sqcap \exists \texttt{hasFailure.Failure} \\ & \sqcap \exists \texttt{hasDistribution.ProbabilityDistribution} \\ & \sqcap \exists \texttt{hasMonitoringWindow.MonitoringWindow}\end{aligned} \tag{15}$$

4.5.5 Name

The `Name` is a concept that has a string which is the name of the service. We assume here that the service name is a unique identifier, but if the name will not be unique we can use another identifier, such as the service URL or a specific ID. This concept relates the service to the different concepts in the ontology: `MaintenanceTime , Request, Description, Parties, Payment, Refund, Location, Description, Price, MonitoringWindow` and `CommitmentValidity`. This part of the ontology is shown in Figure 12.

$$
\begin{aligned}
\texttt{Name} \sqsubseteq & \texttt{Information} \\
& \sqcap = 1\texttt{hasValidity.CommitmentValidity} \\
& \sqcap \exists\texttt{hasMaintenance.MaintenanceTime} \\
& \sqcap \exists\texttt{hasRequest.Request} \\
& \sqcap \exists\texttt{hasDescription.Description} \\
& \sqcap \exists\texttt{hasLocation.Location} \\
& \sqcap = 1\texttt{hasPayment.Payment} \\
& \sqcap = 1\texttt{hasPrice.Price} \\
& \sqcap \exists\texttt{hasRefundCond.RefundCondition} \\
& \sqcap \exists\texttt{hasName.Datatype(string)} \\
& \sqcap = 2\ \texttt{hasParty.Parties} \\
& \sqcap \exists\texttt{isOfferedBy.Provider} \\
& \sqcap \exists\texttt{isOfferedTo.User} \\
& \sqcap \exists\texttt{hasEnvolved.ThirdParty}
\end{aligned}
\tag{16}
$$

5 Using Probability Distribution for Service Availability Description

Current SLAs present a probability or an error rate as a measure for availability. Different availability models have been developed as in [Wang and Trivedi, 2005] and [Xie et al., 2002], however, none of them managed to get through to the market SLAs. Our idea is to have a more concrete representation for availability of a cloud system in the service contracts. This representation is done using probability distributions. This will also allow the development of monitoring tools to check if what is promised in the contracts is being fulfilled. The representation of availability using probability distribution is done by gathering data sets for long periods of time for requests to different resources of the service using either graphical analysis or Goodness-of-fit tests[Nurmi et al., 2005]. It is

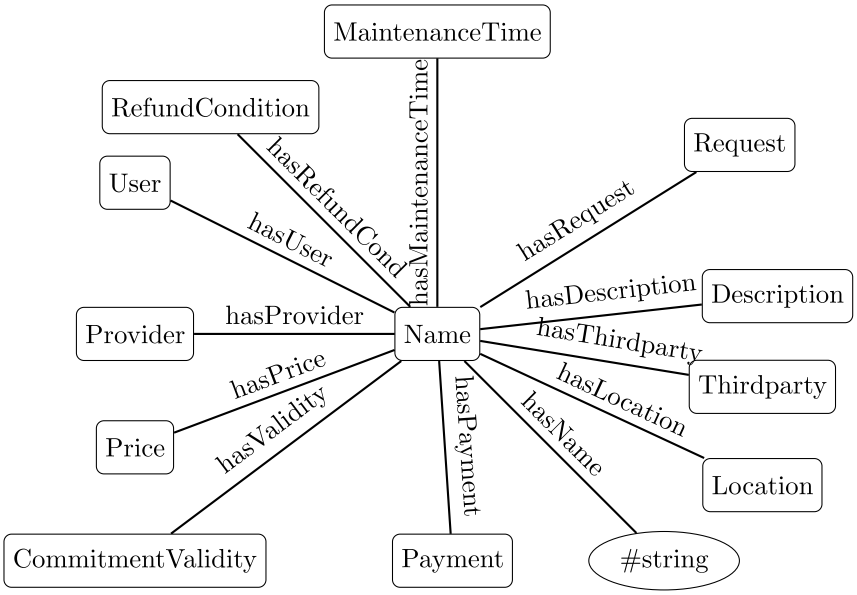

Figure 12: Name - Linking the different concepts together

observable that in different networks, to have a best-fit, different distribution functions are needed. The most common probability distributions that fit to availability data sets are Weibull, Hyperexponential, Gamma and Log-Normal [Javadi et al., 2011] [Wingstrom and Casanova, 2007] [Nurmi et al., 2005]. In our ontology however we included more distributions for different networks.

The concept `ProbabilityDistribution` is defined in equation (7). We chose to represent 7 different probability distribution functions as 7 different individuals under this concept. Each of these distributions has different parameters to it. The idea is to find out these parameters through graphical or goodness-of-fit test analysis and state them in the contract. That way availability becomes monitorable for service providers and for customers. For each of the in equation (7) stated probability distributions we have different parameters that shape the distribution and different formulas. These are stored in the ontology as a string. Table 5 shows each of the distributions and the different parameters needed to calculate it. After deciding which distribution is the best fit and the values of the different parameters, the distribution name and the parameters are stated "as-is" in the contract. And upon agreement these parameters should be checked every `MonitoringWindow` and if there are violations the customer will get a `RefundPercentage` back.

Distribution	Parameters
ChiSquare (v)	Degree of Freedom (v>0)
Gamma (α,β)	shape(α >0) and rate(β >0)
Exponential (λ)	scale inversly proportional to (λ >0)
HyperExponential ($\{\alpha_1, ..., \alpha_m\}$,$\{\lambda_1, ..., \lambda_m\}$)	m-phase hyperexponential distribution with phase probabilities α_i and ratesλ_i.
Lognormal (μ, σ)	shape($\sigma^2 > 0$) and log-scale($\mu \in R$)
Pareto (k,α)	minimum value(k>0) and shape(α >0)
Weibull (α,β)	shape ($\alpha > 0$) and scale ($\beta > 0$)

Table 1: Probability Distribution Functions and their Parameters

6 Generating an Excerpt of a Contract

In this section we will describe how from the previously described Service Contract Model we will generate an excerpt from the service contract. We used Protégé to create the ontology to represent non-functional aspects of a service. And then we used the Jena framework[Jena, 2011] to process the ontology. We first created an `OntologyModel` from the .owl file in the Jena Framework, which is basically an RDF Graph. We then used SPARQL to query this RDF graph to get an excerpt from a service contract. We relied on the fact that in OWL/RDF every statement has a subject, a predicate and an object and queried the ontology for the different statements that we think are important for the consumer.

A simple SELECT statement is querying the ontology for various concepts. It starts by information about the service itself, the name, description, provider, user and third parties also involved in the contract. The concept third party is covering the case where the service provider is a cloud user of a service himself and is using a cloud service to offer a composed service. Using this concept we can recursively get all the third parties that are involved in the service offer, given they are also providing their service contract using an ontology. Then there is the information about the contract, the start- and end- date and time of its validity. The price and the currency of the service. When the Maintenance will take place and the pattern of the repetition of maintenance events. Then there is the information about availability, what is the probability distribution of the service availability, the formula used and the different parameters. `MonitoringWindow` represents the amount of time that is considered to calculate the probability distribution of availability. And then there is the information about the refund, when the previously mentioned service availability distribution is not met. A list of the different possible requests to the service as well as the different failure messages is also displayed. Last but not least the payment information of the service, when and where should the payment be done.

7 Conclusion and Limitations

In this paper we presented a comprehensive excerpt from a service contract. We started by formally defining different concepts that are needed in a service contract and defined one of the most discussed attributes in SLAs, which is availability along with other non-functional aspects. The formal definition of the different aspects in the contract was modelled in an ontology using OWL. This ontology was queried using SPARQL which is an acronym for SPARQL Protocol and RDF Query Language. The advantages of generating the contract this way is that we will have a knowledge base of the different concepts needed for a service contract and information can be added and removed very easily from this ontology. Another advantage is that every concept is referenced as a URL, allowing also references to contract terms of other service providers offered by third parties to be included, which will solve the problem of issuing contracts for composed services. Having this knowledge base will give the chance for users to add their own terms and negotiate the different contract conditions as well as bargain to get an optimal price. In addition it will allow us to have concrete measures to monitor what is promised by the service provider. These measures will allow the development of advanced monitoring tools that can help to make sure that the service is offered as promised.

The data that was used for testing the query is test data, which did not allow sufficient testing. For future work we are planning to investigate the possibility of having real data generated by cloud interaction monitors. Another aspect we will look at is developing a monitoring tool that will check the promised availability distribution against the actual availability of the service.

This work is describing a work in progress and we are currently working on extending, improving as well as verifying this model. In the future different concepts will extend to include more details, such as security considerations for billing, or security considerations of the service ...etc. There is a monitoring solution currently being developed to check compliance to the specified SLA and report any violations of the SLA using the model described in this paper.

References

[Amazon, 2008] Amazon, W. S. (2008). Amazon web services customer agreement. Online; accessed 29-August-2012.

[Baader, 2003] Baader, F. (2003). *The description logic handbook: theory, implementation, and applications.* Cambridge university press.

[Fensel et al., 2007] Fensel, D., Lausen, H., Bruijn, J., and Polleres, A. (2007). *Enabling semantic web services: the web service modeling ontology.* Springer.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199 – 220.

[Javadi et al., 2011] Javadi, B., Kondo, D., Vincent, J.-M., and Anderson, D. P. (2011). Discovering statistical models of availability in large distributed systems: An empirical study of seti@ home. *Parallel and Distributed Systems, IEEE Transactions on*, 22(11):1896–1903.

[Jena, 2011] Jena (2011). Apache jena project. Online; accessed 15-September-2012.

[Keller and Ludwig, 2003] Keller, A. and Ludwig, H. (2003). The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11:57–81.

[Krötzsch et al., 2012] Krötzsch, M., Simancik, F., and Horrocks, I. (2012). A description logic primer. *arXiv preprint arXiv:1201.4089.*

[Lamanna et al., 2003] Lamanna, D. D., Skene, J., and Emmerich, W. (2003). Slang: a language for service level agreements.

[Ludwig et al., 2003] Ludwig, H., Keller, A., Dan, A., King, R. P., and Franck, R. (2003). Web service level agreement wsla language specification. *IBM Corporation*, pages 815–824.

[Nurmi et al., 2005] Nurmi, D., Brevik, J., and Wolski, R. (2005). Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-Par 2005 Parallel Processing*, pages 432–441. Springer.

[Wang and Trivedi, 2005] Wang, D. and Trivedi, K. (2005). Modeling user-perceived service availability. In *Service Availability.* Springer Berlin Heidelberg.

[Wieder, 2011] Wieder, P. (2011). *Service level agreements for cloud computing.* Springer.

[Wingstrom and Casanova, 2007] Wingstrom, J. and Casanova, H. (2007). Statistical modeling of resource availability in desktop grids. Technical report, Technical Report ICS2007-11-01, Dept. of Information and Computer Sciences, University of Hawaii at Manoa.

[Xie et al., 2002] Xie, W., Sun, H., Cao, Y., and Trivedi, K. S. (2002). Modeling of online service availability perceived by web users. In *IEEE Global Telecommunications Conference (GLOBECOM 2002).*