Linked USDL agreement: effectively sharing semantic service level agreements on the Web

How to cite:

© 2015 IEEE
Version: Accepted Manuscript

Link(s) to article on publisher’s website:
http://dx.doi.org/doi:10.1109/ICWS.2015.28

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.
Linked USDL Agreement: Effectively Sharing Semantic Service Level Agreements on the Web

José María García*, Carlos Pedrinaci†, Manuel Resinas*, Jorge Cardoso‡, Pablo Fernández* and Antonio Ruiz-Cortés*

*University of Seville, Spain
†The Open University, United Kingdom
‡CISUC/Department of Informatics Engineering, University of Coimbra, Portugal
Email: {josemgarcia, resinas, pablofm, aruiz}@us.es
Email: c.pedrinaci@open.ac.uk
Email: jcardoso@dei.uc.pt

Abstract—As the use of services available on the Web is becoming mainstream, contracts and legal aspects of the relationship between providers and consumers need to be formalized. However, current proposals to model service level agreements are mostly focused on technical aspects, do not explicitly provide semantics to agreement terms, and do not follow Web principles. These limitations prevent take-up, automatic processing, and effective sharing of agreements. Linked USDL Agreement is a Linked Data based semantic model to describe and share service agreements that extends Linked USDL, which offers a family of languages to describe various technical and business aspects of services. We followed a use case driven approach, evaluating the applicability of our proposal in a cloud computing scenario, and comparing its expressiveness with existing models. Finally, we show a concrete tool that helps to model and check the validity of agreements.

Keywords-service level agreements; semantic modelling; service trading; cloud services

I. INTRODUCTION

Despite the importance of services in developed economies and the widespread adoption of world-wide electronic commerce over the Web, most service trading is still essentially carried out via traditional and, often, manual means [1]. Searching for services, understanding their characteristics, or customizing a contract with service level guarantee are all activities generally carried out manually.

The vision towards a Web of services that would provide an economic fabric to complement existing brick and mortar services has led to the creation of conceptual models, and prototypes (see, eService [2], USDL [3], Linked USDL [1], and cloud computing management [4]). These contributions provide important building blocks in order to support the trading of services over the Web in an open, scalable, and automated manner.

Recently, USDL and notably its latest evolution, Linked USDL, have emerged as versatile general purpose means for capturing formal service descriptions covering aspects such as participants, distribution channels, interactions, and resources. Linked USDL has been devised as an extensible and modular family of ontologies providing convenient means for supporting the modelling, sharing, and processing of service descriptions openly over the Web. Thus far, however, Linked USDL provides no coverage for capturing agreement contracts between the parties engaged in a service transaction. Among these agreements, most relevant are service-level agreements (SLA) which define the level of a service (e.g., service reliability and availability) and corresponding actions in case of non-compliance such as compensations and liability issues (an example of a traditional paper SLA contract can be found here).

In this paper we present Linked USDL Agreement, an extension to the Linked USDL family of ontologies, which provides domain independent means for capturing SLAs. Our ontology provides the necessary means for capturing the semantics of those agreements in a way such that current heterogeneity issues within existing SLAs specifications are circumvented. At the same time, thanks to using Linked Data principles [5], Linked USDL Agreement constitutes a fundamental building block for the trading of services online, by empowering providers and customers alike to discover, interpret, reuse, and manage the SLAs involved in any service transaction. Compared to other alternatives [6], our proposal covers most of the SLA lifecycle activity, it natively embraces novel principles of the Web of Data as a means for sharing descriptions, and it is accompanied by tooling providing both validation and a reference implementation of essential SLAs analysis methods.

This paper is structured as follows. Sec. II provides an overview of the related work in the field of SLAs and introduces Linked USDL. Then, Sec. III enumerates our requirements and describes a motivation scenario used to drive the design of our solution. Sec. IV thoroughly describes the Linked USDL Agreement module. Sec. V evaluates our proposal, while Sec. VI showcases the implemented tooling. Finally, Sec. VII presents the conclusions and our future work.

II. RELATED WORK

USDL [3] is, to date, perhaps the most comprehensive approach for supporting the description of services for automated processing, with the aim of covering services

1http://www.slatemplate.com/ServiceLevelAgreementTemplate.pdf
description, interfaces, pricing models, SLAs, and related legal issues. Despite its comprehensive support, USDL underestimated the need for such a model to be widely open, highly flexible and extensible, and yet simple in nature [1]. To cater for these limitations Linked USDL set out to provide an all encompassing model for describing services, inspired by USDL but following a simpler, more extensible, open and Web centric solution [1].

Linked USDL is the latest evolution of USDL building upon the results and experience gained with USDL combined with prior research on Semantic Web Services, business ontologies, and Linked Data to better promote trading at Web scale [1]. Linked USDL is a family of Web vocabularies predicated upon two fundamental principles: i) the adoption of Linked Data [5] for representing and exposing the descriptions of services and related relevant entities, e.g., the companies involved; and ii) the use of formal ontology representation languages, albeit lightweight to retain scalability, as a means to capture the semantics of services and related entities. Linked USDL Core extends widely used vocabularies, such as GoodRelations [7], with the fundamental means for representing services, offerings, the involvement of business entities, as well as the communication channels allowing business entities to trade and deliver services.

While Linked USDL Core provides essential descriptive capabilities for managing services, given the wide range of aspects that are relevant to service trading, it enables the creation of extensions allowing users to increase the capabilities of the model as need arises. The management of SLAs is one of those aspects for which a specific extension is necessary. Researchers have faced the need for such an extension and have done preliminary work towards transforming Linked USDL Business Policies to WS-Agreement [8]. This transformation is based on a subset of the general WS-Agreement model extended with ad-hoc constructors which does not, however, cover the compensation elements introduced in Linked USDL Agreement. Marquezan et al. [9] also extend Linked USDL with a Transport and Logistics SLA Vocabulary. Unlike our proposal, this extension is domain-specific, and is therefore essentially targetted at modelling transport and logistics SLAs. Furthermore, it does not support expressing common characteristics of many SLAs such as penalties. Both proposals highlight nonetheless the clear need for a domain independent SLA extension to Linked USDL.

Outside USDL, several languages or models to specify SLAs have also been defined in the literature (cf. a comparative analysis in [6]); amongst them, WSLA [10] and WS–Agreement [11], introduced in 2001 and 2005 by IBM and the Global Grid Forum, respectively, represent the most prominent approaches within industry. Specifically, the latter, which was an evolution of the former, was developed as a specification framework that provides extensibility mechanisms to create fully-fledged SLA languages (cf. [12]). Despite the variety of approaches, most are predicated upon the existence of an underlying WSDL description which, with the advent of Web APIs, is often nonexistent. Furthermore, those approaches are essentially focussed on software based services which, although important, only represent a minimal share of the services market leaving many other service activities (e.g., insurance, eLearning, etc) with a poor coverage and support (cf. Sec. III-A).

III. REQUIREMENTS AND USE CASE

We have identified a set of requirements that are reflected in a motivating scenario in the cloud computing services domain, following a use case driven approach and using competency questions obtained from the scenario analysis.

A. Requirements on Modelling Service Level Agreements

Recent technological development in the field of services, e.g., cloud services and Web APIs, have substantially changed the face of computational services and, hence, their SLAs. Thus, there is the need to revisit the field of SLAs to determine if specifications still fulfill current requirements, which are enumerated in the following.

1) Shared Meaning of Content: Effective trading requires service providers and customers to speak the same “language”. Descriptions need therefore be based on an agreed upon format or schema (shared meaning of schema), and be expressed in mutually understandable terms and concepts (shared meaning of content). Previous SLA languages, such as WSLA and WS-Agreement, only address the first requirement. Linked Data on the other hand was purposely proposed to cover the publication, discovery, and interpretation of both schemas and content in a machine understandable form over the Web. For example, the vocabularies itil:{processes, roles, glossary}² organise more than 600 terms related to IT services, which can be used to unambiguously share the semantics of contracts’ content. Sec. IV explains how Linked Data can be used by SLA specifications.

2) Open, Web-based Solution: To promote take-up and effectively share and process SLA descriptions online, the technological approach should be open to anybody to publish and exploit such descriptions, but also open to extensions to address unanticipated needs and scenarios. Our approach, as opposed to earlier proposals, embraces Web principles and technologies to provide a highly interoperable and scalable solution. Sec. V compares our solution to other SLA approaches.

3) SLA Lifecycle Automation: The negotiation and creation of SLAs is a key activity in the SLA lifecycle. Other activities include validity checking, conformance and monitoring, which seek to detect contract conflicts and breaches. Manually performing these activities is an expensive and error-prone process. To carry them out efficiently, it is necessary to feed SLAs specifications into automated software applications that can validate SLAs and find violations. Sec. VI demonstrates how this automation can be achieved.

²http://w3id.org/itil/\{processes, roles, glossary\}
B. Cloud Computing Services Use Case

The Cloud computing paradigm has emerged as a cost-effective and efficient form of on-demand provisioning of computing services. Businesses do not need to host a large number of resources to cope with their computing requirements, but can dynamically use external services that provide them, lowering operating and maintenance costs, as well as supporting a high scalability [13]. Cloud computing architectures are usually divided into four layers: hardware, infrastructure, platform and application layers. For each layer, several vendors provide related services depending on the users’ needs. For instance, the infrastructure layer can be offered as a service, the so-called *Infrastructure as a Service* (IaaS). These services offer infrastructural resources such as servers and virtual machines, including pre-loaded images or customizable ones. Some examples of IaaS providers are Amazon EC2⁴, Microsoft Azure⁵, and Google Cloud Platform⁶.

In this scenario, where different businesses interact using service-oriented architectures to outsource computing needs, the formalisation of the SLAs that govern business relationship is of outmost importance. Most SLAs are described in natural language at providers’ websites. For example, the following table shows a typical SLA for Amazon EC2 service commitment⁷. This information was written to be processed by humans and not by software.

<table>
<thead>
<tr>
<th>Monthly Uptime %</th>
<th>Service Credit %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 99.95% but equal to or greater than 99.0%</td>
<td>10%</td>
</tr>
<tr>
<td>Less than 99.0%</td>
<td>30%</td>
</tr>
</tbody>
</table>

Focusing on this particular use case and typical contents on other SLAs analysed, we can specify a series of competency questions to devise a semantic vocabulary [14] useful for the SLA lifecycle:

Q1: Which functionality and quality levels does a service provide?
Q2: Which service properties are guaranteed to have certain values?
Q3: Which compensation is obtained if the guaranteed value of a property is not provided?
Q4: Who is responsible for enforcing the guaranteed service level values?
Q5: Who is responsible for monitoring and computing the guaranteed values?
Q6: What is the assessment period during which a guarantee is provided?
Q7: How are service property values computed?

The design of our semantic model is driven by its ability to effectively answer these competency questions. In addition, an agreement model for services has to be designed for its exploitation on the Web, so that the associated SLA documents should be easily accessible and processable. Consequently, we impose additional requirements concerning scalability, ease of publication, use of existing standards and recommendations, and informed by major efforts on SLA specification frameworks, such as WS–Agreement [11].

IV. MODELLING SERVICE AGREEMENTS

Considering the identified requirements and driven by the competency questions discussed previously, we designed an agreement module integrated with the Linked USDL family of vocabularies, following the design decisions and architecture presented below. The Linked USDL Agreement module is publicly available in GitHub⁷, including the use cases presented throughout this paper.

To provide a shared meaning of SLAs, our model uses formal ontology representation languages to handle the structural and semantic heterogeneity of current SLAs. Therefore, as when designing Linked USDL [1], we have followed Linked Data principles [5], allowing our model to share and interlink data about service agreements on the Web.

Linked Data promotes reuse of existing models and datasets, facilitating the design of our model and compatibility with existing tools. Our approach ensures that the identified competency questions were answered successfully, allowing the complete description of our cloud computing use case, as discussed in Sec. V.

Fig. 1 presents our proposed agreement model. Essentially, an agreement consists of a set of terms that state the conditions that are guaranteed under the SLA, and which compensations are taken if a certain guarantee is violated. In the following we describe in detail the most important concepts of Linked USDL Agreements.

**AgreementTerm** represents a single term of an SLA, which could possibly have a *precondition* that restricts the situation when the term is enforced. All instances of this concept that are related to a concrete *service offering* describe the complete SLA provided with that offering. In particular, we differentiate two subtypes of terms that can appear in an agreement, namely *guarantees* and *compensations*.

**Guarantee** represents an agreement term of an SLA that specifically *guarantees* certain conditions over service properties. This concept is commonly called Service Level Objective (SLO) in other SLA models. An example of a *Guarantee* could capture that “Amazon guarantees that the monthly uptime of its EC2 service will be at least 99.95%”.

**Compensation** is a specialisation of an agreement term that represents an alternative term that will be guaranteed in case that the original *guarantee* term (associated with the compensation via the *hasCompensation* property) is not fulfilled, e.g. “a service credit of 10% will be entitled if the monthly uptime is less than 99.95% but equal to or greater than 99.0%”. Note that this example contains a *precondition* on the monthly uptime.

³http://aws.amazon.com/ec2
⁴http://azure.microsoft.com
⁵https://cloud.google.com
⁶http://aws.amazon.com/ec2/iaa/
⁷https://github.com/linked-usdl/usdl-agreement
AgreementCondition describes a particular constraint or axiom that can be checked within the terms of an SLA. These conditions usually refer to a concrete service property, constraining their possible values depending on the actual definition of the condition, e.g., “the monthly uptime will be at least 99.95%” part of the previous guarantee term example. Our vocabulary offers some pre-defined facilities for common axiom types, including concrete guaranteed values (as in the previous compensation example), maximums, minimums, and intervals. However, arbitrary axioms using domain-specific languages to describe conditions can be also included using rdf:value.

ServiceProperty is a convenience class that allows an agreement condition to refer to either a qualitative (e.g., region availability) or a quantitative (e.g., monthly uptime percentage) service property, as defined in GoodRelations vocabulary [7].

Metric defines how to measure a particular service property. It is usually defined by a mathematical expression that needs to be computed in order to monitor a concrete property. For example, Amazon EC2 SLA describes that “Monthly Uptime Percentage is calculated by subtracting from 100% the percentage of minutes during the month in which Amazon EC2 (...) was in the state of Region Unavailable.”

EntityLiability is an extension of the entity involvement concept in Linked USDL Core that enables capturing the liability role that an involved business entity has in a particular agreement term, i.e., its responsibility with respect to that term. For instance, the provider of a service can act as a guarantor of a particular guarantee, being responsible of the fulfillment of that guarantee, while the consumer can be considered to have a beneficiary role in a compensation since they will benefit from it.

Linked USDL Agreement provides a simple SKOS\(^8\) taxonomy of liability roles, including the basic Guarantor and Beneficiary already discussed, but can be easily extended depending on the use case. Following this philosophy we extended the reference business roles SKOS scheme defined in Linked USDL Core module to support the identification of the business entities responsible for evaluating conditions and providing metrics, since these roles are usually defined in SLAs.

Apart from the main concepts described previously, we rely on several external vocabularies following Linked Data principles. First and foremost, being an extension of Linked USDL, our model builds upon the main classes of Linked USDL Core. We show this relationship in Fig. 1, where a ServiceOffering is related

\(^8\)http://www.w3.org/2004/02/skos/core
to its corresponding agreement terms by the property hasAgreementTerm. Each guarantee term is itself linked to a particular Service included in the offering over which the conditions are guaranteed by means of guaranteedOver property.

Indirectly, we also use GoodRelations vocabulary [7] through its relationship with Linked USDL Core module. GoodRelations defines concepts related to commerce, such as business entities, products and services. We use the qualitative and quantitative service properties from GoodRelations when defining agreement conditions. Thus, an agreement condition may refer to a property that is a subproperty of gr:quantitativeProductOrServiceProperty or gr:qualitativeProductOrServiceProperty. Correspondingly, the values used in the condition definition through the hasValue property can be instances of gr:QuantitativeValue or gr:QualitativeValue.

The Time Ontology\footnote{http://www.w3.org/TR/owl-time} is also integrated to cover temporal properties relevant for the SLA, including the evaluation and measuring intervals of agreement conditions and metrics, respectively, as well as the validity period of SLA terms [15]. Regarding the metrics support for conditions and service properties, we do not restrict a particular vocabulary to be used, but we recommend the integration with QUDT\footnote{http://qudt.org/} for describing units of measurement, or SPIN\footnote{http://spinrdf.org/spin.html} for defining the metric expressions, for instance.

In addition to those vocabularies, we also make use of Dublin Core\footnote{http://purl.org/dc/terms/}, VANN\footnote{http://purl.org/vocab/vann/} and Friend of a Friend (FOAF)\footnote{http://xmlns.com/foaf/0.1/} to cover general purpose metadata about the vocabulary itself, such as creators, modification dates, and preferred namespace prefixes and URIs. Finally, as already discussed, we use Simple Knowledge Organization System (SKOS) vocabulary for creating the classification scheme for liability roles, similarly as other role schemes defined in Linked USDL Core module.

V. EVALUATION

In this section, we evaluate how well Linked USDL Agreement fulfills the requirements enumerated in Sec. III, validating our model using the introduced cloud computing scenario and additional real-world use cases. Furthermore, we discuss the SLA description coverage considering the framework proposed in [6].

A. Cloud Computing Service Agreement

The motivating example described in Sec. III-B served also a validation purpose in our work. Thus, we tested the suitability of our vocabulary to completely describe the SLA of the cloud computing provider Amazon EC2\footnote{http://www-3w.w3.org/Tr/owl-time}. We especially focus on verifying the extent to which Linked USDL Agreement vocabulary can describe the terms expressed in the Amazon EC2 SLA, while being able to answer the listed competency questions.

The Amazon EC2 SLA contains a series of definitions regarding the monthly uptime percentage and the extent to which it is guaranteed for all the infrastructure provided by Amazon EC2. If the SLA is violated, Amazon honours a service credit to the user. First, our vocabulary is able to associate the guarantees of an SLA to the description of the service offering which is governing, which in turn is associated with a particular service description that describes its functionality, answering Q1.

The definitions included in the SLA refer to properties of the Amazon EC2 service that are guaranteed. Therefore, we modelled them using the properties definition from GoodRelations, as described in Sec. IV. Q2 can thus be answered by querying the model about the properties that are referenced, using for example SPARQL as shown in Listing 1.

\begin{verbatim}
Listing 1. Obtaining service properties relevant to the agreement
1 SELECT ?prop WHERE {
2 :amazonEC2ServiceOffering :hasAgreementTerm ?term ,
3 ?term usdl-agreement:guarantees ?conditions .
4 ?conditions usdl-agreement:refersTo ?prop } 
\end{verbatim}

The main part of the agreement states the guaranteed value of the monthly uptime percentage. Listing 2 shows how we modelled that service commitment. First, the guarantee term refers to the concrete service included in the original service offering over which the guarantee is applied, as well as the liability of the different entities involved in the agreement (Amazon as a provider and an abstract customer in our example). This information about the different roles of the entities provides the answers to both Q4 and Q5.

\begin{verbatim}
Listing 2. Agreement terms
1 :ec2ServiceCommitment a usdl-agreement:Guarantee ;
2 usdl-agreement:guaranteedOver :amazonEC2M1LargeInstanceType ;
3 :monthlyInterval ;
4 :liab_customer , :liab_Amazon ;
5 :monthlyUptimePercentage a gr:QuantitativeValueFloat ;
6 usdl-agreement:guarantees ?conditions .
7 ?conditions usdl-agreement:MinGuaranteedValue ;
8 qudt:unit <http://qudt.org/vocab/unit#Percent> ;
9 usdl-agreement:hasEvaluationInterval :monthlyInterval ;
10 usdl-agreement:hasValue [ a gr:QuantitativeValueFloat ;
11 gr:hasValueFloat "99.95"^xsd:float ] ;
12 usdl-agreement:refersTo ?monthlyUptimePercentage .
13 :monthlyUptimePercentage a usdl-agreement:MinGuaranteedValue ;
14 usdl-agreement:hasValidityInterval :monthlyInterval ;
15 usdl-agreement:hasCompensation :ec2ServiceCredit30 , :ec2ServiceCredit10 ;
16 usdl-agreement:hasValidityInterval :monthlyInterval .
\end{verbatim}

Second, the guaranteed condition is defined as the minimum value that the \texttt{monthlyUptimePercentage} property has to provide. We also include in our description the definition of the metrics used to compute that property, relying on external vocabularies and tools to properly
answer Q7. Third, the time intervals where values are guaranteed or need to be monitored by involved parties are described using intervals modelled with the Time ontology, covering Q6.

Finally, compensation terms model alternative conditions that will take into place in the case that the guaranteed values are not fulfilled (Q3). In the particular case of Amazon EC2 compensations, the SLA defines two compensation levels depending on the final value of the monthly uptime percentage, as shown in Sec. III-B. We model them adding preconditions to the compensation terms.

B. Software as a Service Contracts

To make our evaluation comprehensive, we have also analyzed software as a service agreement contracts that are commonly used in the industry to establish SLAs. These paper-based contracts are often prepared by lawyers and require a case-by-case customization. The following illustrative and representative extract of a service agreement contract\(^\text{16}\) describes service level availability.

Contracts are often composed of two parts: 1) the agreement and 2) the exhibits. The agreement describes the general terms of the contract using natural language with no underlying structure. On the other hand, exhibits provide a structured description, still in natural language, about the specific terms of a contract\(^\text{17}\). The part that we have successfully modeled was “Exhibit A” of the contract under analysis. Nonetheless, we have identified that further research and solutions are still required to model the part of the contract agreement due to the complexity of the language used as shown in the following text box.

Alternatively, the agreement part of a contract needs to be simplified and also needs to be written using structured descriptions, as done with the exhibits, to enable an automated processing. This approach is being followed by major cloud computing providers, such as Amazon, Google, and Microsoft.

C. Linked USDL Agreement coverage evaluation

Finally, we evaluate the coverage of Linked USDL Agreement against the comparison framework proposed in [6]. This comparison framework comprehends 22 criteria grouped by the SLA lifecycle activity in which they are more relevant. These criteria were used to compare 14 SLA and Service Contract Languages. Table I summarises the criteria and shows the evaluation results of Linked USDL Agreement. It also depicts how many of the 9 SLA languages analysed fulfill each criteria.

Linked USDL Agreement fulfills 13 out of the 22 criteria. The formalism used to define Linked USDL Agreement are ontologies. Both functional and quality terms can be expressed in Linked USDL Agreement through Linked USDL Core’s ServiceOffering and through the Guarantee introduced by Linked USDL Agreement, respectively. The reusability of SLAs is native to the Linked USDL approach. Metric providers and metric schedule are modelled including the MetricProvider business role in an involved entity, and hasMeasuringInterval of Metric, respectively. The condition evaluator can be specified using the corresponding business role on the relevant involved entity. Qualifying conditions are expressed using the property hasPrecondition of AgreementTerm. The obliged party can be modelled for each AgreementTerm using liability roles via hasEntityLiability property. The assessment schedule of an SLO is specified with property hasEvaluationInterval. Validity periods are expressed by means of property hasValidityInterval for each AgreementTerm. Both penalties and rewards can be expressed at the level of SLOs using property hasCompensation of Guarantee. Finally, the validity period of the whole SLA can be expressed using the validThrough property of a ServiceOffering included in Linked USDL Core.

Concerning the remaining criteria, the main reason they have been left outside of Linked USDL Agreement is because they are not shared by most real-world SLAs we have found in our analyses. Specifically, composability is not supported because most SLAs are not for composite services, or rather, they are expressed for the resulting composition which is exposed as a single service. The same applies to the ability to express alternative service levels since most SLAs define just one service level\(^\text{18}\); the ability to express soft constraints since most SLOs are expressed as hard constraints; the two negotiation-related criteria since most SLAs are take-it-or-leave-it.

\(^{16}\)Obtained from http://assets-production.govstore.service.gov.uk/GS/1756/5.G5.1756.003/QD1/MasterSoftwareasaServiceAgreement2014.docx

\(^{17}\)A partial description in Linked USDL Agreement of the use case can be found at https://github.com/linked-usdl/usdl-agreement/tree/master/UseCases/SaaS

\(^{18}\)Note that different service levels can still be expressed in Linked USDL Agreement through different ServiceOfferings or by means of pre-conditions.
Table I

<table>
<thead>
<tr>
<th>Criteria Description</th>
<th>Evaluation</th>
<th>Proposals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formalism</td>
<td>Ontol.</td>
<td>Several</td>
</tr>
<tr>
<td>Coverage</td>
<td>[y], [y]</td>
<td>2</td>
</tr>
<tr>
<td>Reusability</td>
<td>yes</td>
<td>7, yes, 2 part.</td>
</tr>
<tr>
<td>Composability</td>
<td>no</td>
<td>1, good, 4, fair</td>
</tr>
<tr>
<td>Metric definition</td>
<td>no</td>
<td>5</td>
</tr>
<tr>
<td>Alternatives</td>
<td>no</td>
<td>7 impl.</td>
</tr>
<tr>
<td>Soft constraints</td>
<td>no</td>
<td>2</td>
</tr>
<tr>
<td>Matchmaking Metric</td>
<td>no</td>
<td>2</td>
</tr>
<tr>
<td>Meta-Negotiation</td>
<td>no</td>
<td>1, good, 2 fair</td>
</tr>
<tr>
<td>Negotiability</td>
<td>no</td>
<td>2 part</td>
</tr>
<tr>
<td>Metric Provider</td>
<td>yes</td>
<td>4</td>
</tr>
<tr>
<td>Metric Schedule</td>
<td>yes</td>
<td>4</td>
</tr>
<tr>
<td>Condition Evaluator</td>
<td>yes</td>
<td>2</td>
</tr>
<tr>
<td>Qualifying Condition</td>
<td>no</td>
<td>2</td>
</tr>
<tr>
<td>Obliged</td>
<td>yes</td>
<td>7</td>
</tr>
<tr>
<td>Assessment Schedule</td>
<td>yes</td>
<td>3</td>
</tr>
<tr>
<td>Validity Period</td>
<td>yes</td>
<td>4</td>
</tr>
<tr>
<td>Recovery Actions</td>
<td>no</td>
<td>4</td>
</tr>
<tr>
<td>Penalties</td>
<td>SLO</td>
<td>3 SLO, 2 SLO</td>
</tr>
<tr>
<td>Rewards</td>
<td>SLO</td>
<td>1 SLO, 2 SLO</td>
</tr>
<tr>
<td>Settlement Actions</td>
<td>no</td>
<td>2</td>
</tr>
<tr>
<td>SLA Validity Period</td>
<td>yes</td>
<td>5</td>
</tr>
</tbody>
</table>

offers without any possible negotiation; and the ability to express recovery actions and settlement actions since only penalties are usually defined in SLAs. Furthermore, these criteria are also those that are fulfilled by less proposals with no more than 2 different proposals fulfilling each of them except for recovery actions, which are supported by 4 proposals. This reinforces our belief that they are only useful in a very limited set of scenarios.

Nevertheless, due to the nature of our modelling approach, new extensions to Linked USDL Agreement can be seamlessly integrated in order to provide these advanced features in scenarios that require them. For instance, one might design a negotiation-related extension that extends the ServiceOffering with information about the negotiation process and the Guarantee with information about its negotiability.

VI. TOOLING SUPPORT

Writing an SLA in Linked USDL Agreement (like in any other formal language) can be a challenging task. Since it is manual, there is a risk of errors that, depending on the complexity of the SLA, can be very high. Additionally, as SLAs represent rights and responsibilities of the stakeholders that could lead to compensations, they include sensitive statements that should be carefully designed and modelled. Conflicts amongst the terms of the SLAs, such as inconsistencies, represent a major drawback that should be avoided to assure specifications that would not lead to misunderstandings or unexpected situations. To address this drawback, we provide a tool\(^{19}\) for the definition and consistency checking of SLAs.

Fig. 2 shows our tool performing a validity check. The tool was developed within the context of the IDEAS framework that supports the creation of on-line environments for the usage and analysis of formal models by means of different language modules. The proposed Linked USDL Agreement tooling is based on an underlying analysis module that detects problems in SLA documents using constraint programming [12] by performing a validity check that includes detection of dead guarantees (in case a precondition can not be satisfied at any point) and inconsistent terms (when agreement conditions are contradictory). In addition to validity checking, the Linked USDL Agreement tool provides an analysis report answering the different competency questions presented in Sec. III by means of SPARQL queries.

Validity check is based on an analysis of the constraints defined in agreement conditions. Specifically, to reuse the constraint programming based technique presented in [12], we developed a transformation from Linked USDL Agreement to a WS–Agreement template that directly maps those constraints as follows: 1) Linked USDL Agree-

\(^{19}\)Available at http://www.isa.us.es/IDEAS/Linked_USDL_Agreement
ment guarantee terms are transformed into WS–Agreement guarantee terms, in which its guarantees, preconditions and compensations are transformed into SLOs, qualifying conditions and penalties or rewards in business value lists, respectively; 2) Linked USDL Agreement service properties referred by agreement conditions are transformed into WS–Agreement service properties as variables; 3) the properties used by the services and included in the service offerings of Linked USDL Agreement are transformed into properties in the service description terms of WS–Agreement and the concrete values assigned to those properties in services are transformed into creation constraints for the service description terms.

VII. CONCLUSIONS AND FUTURE WORK

Since existing specifications for creating agreements for services, such as WS–Agreements, WSLA, and SLA*, were developed to capture the technical aspects of Web services, we developed Linked USDL Agreement, an extension to the Linked USDL service description family, to capture business aspects, compensations and time constraints, among others. The new specification is to be used to establish and share agreements between customers and providers who seek to automatically perform service trading over the Web.

The evaluation of Linked USDL Agreement was two-fold. On the one hand, we evaluated its capabilities to model services such as EC2 made available by Amazon AWS. On the other hand, we showed how our proposal covers the SLA lifecycle compared to existing ones, focusing on actually used features in common SLAs. Furthermore, we discuss how the information captured by our model can be automatically used by tools to perform validity checking, for instance.

Future work requires to build a proof-of-concept prototype to illustrate how a service marketplace could automatically provision services to consumers with regards to their requirements and preferences [16] coping with heterogeneity issues, as well as to establish contracting using Linked USDL Agreement, and to automatically detect service level objectives’ violations, which would be reported to customers and trigger compensation actions.

ACKNOWLEDGMENTS

Authors would like to thank Juan Luis de la Fuente for the development of the tooling. This work has been partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&D programmes (grants P12-TIC-1867, TIN2012-32273, TIC-5906, IPT-2013-0890-3).

REFERENCES


