Evolution and overview of Linked USDL

Conference or Workshop Item

Jorge Cardoso and Carlos Pedrinaci, Evolution and Overview of Linked USDL. 6th International Conference Exploring Services Science, IESS 2015, Porto, Portugal, February 4-6, 2015, LNBIP, Vol. 201, Novoa, Henriqueta, Dragoicea, Monica (Eds.), 2015.

Evolution and Overview of Linked USDL

Jorge Cardoso[1,2] and Carlos Pedrinaci[3]

[1] Technical University of Dresden, Dresden, Germany
[2] CISUC, Department of Informatics Engineering
University of Coimbra, Coimbra, Portugal
jcardoso@dei.uc.pt
[3] Knowledge Media Institute, The Open University,
Milton Keynes, UK
c.pedrinaci@open.ac.uk

Abstract. For more than 10 years, research on service descriptions has mainly studied software-based services and provided languages such as WSDL, OWL-S, WSMO for SOAP, and hREST for REST. Nonetheless, recent developments from service management (e.g., ITIL and COBIT) and cloud computing (e.g. Software-as-a-Service) have brought new requirements to service descriptions languages: the need to also model business services and account for the multi-faceted nature of services. Business-orientation, co-creation, pricing, legal aspects, and security issues are all elements which must also be part of service descriptions. While ontologies such as e^{3}service and e^{3}value provided a first modeling attempt to capture a business perspective, concerns on how to contract services and the agreements entailed by a contract also need to be taken into account. This has for the most part been disregarded by the e^3 family of ontologies. In this paper, we review the evolution and provide an overview of Linked USDL, a comprehensive language which provides a (multi-faceted) description to enable the commercialization of (business and technical) services over the web.

Key words: Linked USDL, service description, service management.

1 USDL overview

Linked USDL (Unified Service Description Language)[15] was developed for describing business and software services using computer-readable and computer-understandable specifications to make them tradable on the web/Internet [6]. Linked USDL takes the form of a reference vocabulary which is an approach used in many fields to facilitate the exchange of data and integration of information systems. For example, online social networks rely on FOAF[1] to describe people and relationships; computer systems use WSDL[2] to describe distributed software-based services; GoodRelations[3] is used to mainly describe products; and

[1] http://www.foaf-project.org/
[2] http://www.w3.org/TR/wsdl
[3] http://purl.org/goodrelations/v1

business-to-business systems use ebXML[4] to describe transactions, orders, and invoices. Adding to these existing standards, Linked USDL describes services in a comprehensive way by providing a business or commercial description around services. Therefore, Linked USDL is seen has one of the foundational technologies for setting up emerging infrastructures for the Future Internet, web service ecosystems, and the Internet of Services.

The objective of this paper is to provide a retrospective on the development of the service description language Linked USDL; the various technologies that have been used to support the evolving models; the current models and documentation available; and the projects that are using and evaluating Linked USDL. This will enable to ease future developments on the field of the web of services by providing an important overview to reduce ramp-up time.

This paper is organized as follows. Section 2 describes the evolution of USDL which started in 2007. Section 3 discusses our findings and the experience gained from modeling services over the years. Section 4 describes the various modules that have already been developed (e.g., core and pricing) and the ones that are in development. It also discusses the benefits of Linked USDL and the standardization efforts that were carried out in the past. Section 5 provides an example of how to describe a service using Linked USDL. We have chosen to model `last.fm` since it is a good example of an emerging type of services which are part of the so-called Web API economy. ProgrammableWeb, the world's leading directory of Internet-based APIs, shows a dramatic growth of APIs since 2005 and has as of July 2014 more than 11.500 entries. Section 6 describes the related work and the alternatives to using Linked USDL. Finally, in Section 7 the conclusions are presented and discussed.

2 Evolution of USDL

The initial main driving organization behind the first two versions of USDL was SAP Research. Developments were carried in conjunction with other research partners such as Siemens, Forschungszentrum Informatik (FZI), and Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO). Funding for the research was part of the Theseus project and supported by the German Federal Ministry of Education and Research (BMBF).

The two initial versions of USDL (versions 1.0 and 2.0) started to be developed in 2007 and were ready in 2009. They were built using XML Schema. Later, in 2011, based on the experience gained from the first developments, a W3C Incubator group[5] was created and USDL was extended leading to version 3.0. This version was built using the Ecore metamodel and the Eclipse Modeling Framework (EMF) to define UML modules for capturing the "master data" of a service. It included extensions for pricing, legal, functional, participants, interactions, and SLA aspects. The extensions resulted from the use of USDL in several European academic and industrial projects (e.g., RESERVOIR, SLA@SOI, and SOA4ALL).

[4] `http://www.ebxml.org/`

[5] `http://www.w3.org/2005/Incubator/usdl/`

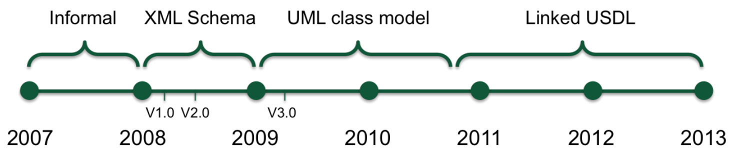


Fig. 1. The evolution of USDL and Linked USDL (2007-2013)

In 2012, version 4.0 was created and renamed to Linked USDL since its development followed Linked Data principles [4] due to the rather inflexible and close nature of previous technologies (e.g., XML, Ecore, and UML). Currently, Linked USDL is the version most often used to develop infrastructures and applications to manage services. The objective is to shift from a closed solution to a language which enabled the large scale, open, adaptable, and extensible description of services using a decentralized management. The use of Linked Data enabled USDL to inherit many distinctive aspects such as unique service addresses on the web via the use of URIs and the description data about services is published in a computer-readable and -understandable format.

The rationale behind the use of Linked Data is the requirement that USDL descriptions should be shared between interested parties and linked to other descriptions, standards, and vocabularies. Linked Data technologies are particularly suitable for supporting and promoting this level of web-scale interlinking. Globalization truly happens only when people, devices, processes, and services are all connected into a global network. As with its predecessors, Linked USDL was also conceived, explored, and evaluated in several research projects including FI-Ware (smart applications), FInest (logistics), and Value4Cloud (value-added cloud services).

Figure 1 provides an overview of the evolution of Linked USDL and Table 1 describes the various versions of the USDL language and their main characteristics.

Versions 1.0, 2.0, and 3.0 of USDL have been discontinued and are no longer supported. Linked USDL introduces many changes and follows a different philosophy when compared to its predecessors. While the core and pricing models for Linked USDL are finalized, work on developing and aligning several modules of the specification (e.g. legal, security, and service level) with various use cases (e.g., higher education, cloud computing, and human-based services) is still in progress. The modular approach followed, separating the different service aspects into independent vocabularies, has proved to be efficient, flexible, and easy to be processed by service delivery platforms. The following sections cover USDL version 4.0, i.e., Linked USDL, since it offers a larger spectrum of advantages to build a large-scale web of services.

3 Lessons Learned

The development of four versions of USDL to model human-driven and software-based services taught us many lessons about service modeling. Table 2 summarizes

Table 1. The main characteristics of USDL languages

Name	Ver.	Year	Characteristics
USDL [7]	1.0	2008	Introduces the notions of business, operational, and technical perspectives. The model is serialized with an XML Schema (in this paper it will be also referred to as USDL/XML).
USDL [6]	2.0	2009	The model is extended. It addresses the first set of concrete requirements for service description languages (e.g., extensibility, multiple views, and variability).
USDL [3]	3.0	2011	The model is divided into various modules, and Ecore and the Eclipse Modeling Framework (EMF) are used for modeling (also referred as USDL/ECore).
Linked USDL [15]	4.0	2013	The model is represented using Linked Data principles and RDFS. The core model is rebuilt to provide a simpler view on services and better coverage (also referred as Linked USDL/RDFS.)

our findings over the years and provides conclusions, which can be useful for future developments indispensable to create a web of services' infrastructure. We discuss the four most relevant findings: *model extensibility*, *data interoperability*, and *instance identification*.

Model extensibility [2] refers to the capability to create new service models as extensions/derivations from a base model. Data interoperability [8] is the ability of two or more systems to exchange service information and to use the information that has been exchanged. Data integration [12] refers to the capability to combine service data coming from different sources to provide a unified view of these data. Instance identification [14] is related to the creation of unique identifiers to identify service instances.

3.1 Model Extensibility

A web of services requires more than a one-size-fits-all model and, thus, the extensibility of service models was important. USDL/XML and USDL/Ecore provided a schema for service descriptions with a rather fixed structure. XML Schema does not provide an elegant mechanism to enable providers to add new elements to a data schema to better describe their services. In USDL/XML, the solution implemented consisted in using a place holder that captured a list of pairs attribute-value. Both attribute and value where elements of type string, which enabled providers to add new descriptive attributes to their services. Clearly, this type of approach is not practical either from a syntax or semantic perspective.

An attribute-value approach can support extensibility (since any attribute can be added) but it is not suitable for interpretation (and integration) since attributes are just "strings" with no meaning attached. RDFS allows the same

Table 2. The main lessons learned

Lesson Learned	Description
Model Extensibility	Linked USDL/RDFS has the highest extensibility capabilities. USDL/Ecore and USDL/XML only provided extensibility via the creation of new attribute-value pairs.
Data Interoperability	Due to the adoption of XML by businesses, USDL/XML has the highest degree of data interoperability. Ecore, despite its XMI serialization, does not enjoy this popularity. RDFS has a growing base of adopters, which makes Linked USDL/RDFS part of a new generation of web specification languages. This movement corresponds to an interoperability through adoption.
Data Integration	USDL/XML and USDL/Ecore generate self-contained instances. In other words, the meaning or semantics of instance data is not shared across stakeholders via the model specification. It is given by the entity (e.g., provider) creating a service description instance. On the other hand, Linked USDL/RDFS enables to create instances which are integrated with external data managed and shared by linked data registries (e.g., `dbpedia`). Anyone can reuse preexisting instances/data defined by 3rd parties which enables data reuse (saving effort) but also enhances interoperability.
Instance Identification	While Linked USDL/RDFS relies on URIs to provide a simple way to create unique global identifiers for services, USDL/XML and USDL/Ecore did not provide such a decentralized and scalable mechanism.

kind of extensibility but every new attribute added comes with its own semantics ready to be interpreted by software.

3.2 Data Interoperability

While USDL/XML enables to archive a high degree of data interoperability, since XML has been adopted and used for many years by businesses and many tools exist, Ecore does not enjoy this popularity. While a serialization to XMI exists, the objective of Ecore is not to foster the exchange of data but to provide a metamodel to implement object-oriented programs. It can be used to model packages, classes, attributes, references, etc. to facilitate dynamic code generation. In other words, Ecore is suitable for the automated generation of code to develop applications that use the model (e.g., editors, service marketplaces, matchmakers, etc.). With respect to data interoperability, XML was more adequate than Ecore to support a web of services. When analyzing the difference of interoperability between USDL/XML and Linked USDL/RDFS, it is clear that XML has also a higher degree of data interoperability due to its dissemination and adoption level. Nonetheless, RDFS is gaining popularity for knowledge modeling and its adoption was strategic.

3.3 Data Integration

USDL/XML and USDL/Ecore specify a model that contains attributes (e.g., integers, longs, strings, or other structures) to which values (e.g., 232, 1.4, "ITIL Incident Management") are assigned. This means that the data associated with a service instance has only an established meaning to the entity providing the data. RDFS, on the other hand, enables to reuse data that already exists on the web in the form of Linked Data. For example, when creating a description for the service `ITIL_IM_service`, the company providing the service can be specified by assigning the unique URI `http://dbpedia.org/resource/Cloudera` maintained at `dbpedia.org`. This URI holds a lot of data interlinked to many other data sources. Thus, it is possible to know the number of employees, the address, the economic sector, etc. of the service provider.

Using USDL/XML and USDL/Ecore, the string "Cloudera" would be assigned to an attribute. Thus, its meaning would remain with the entity that provided it. As we converge to an interconnected world of data, the integration of data and services will be important. Thus, and when compared to USDL/XML and USDL/Ecore, Linked USDL/RDFS incorporates Linked Data mechanisms that are valuable to support data integration initiatives as part of a web of services.

3.4 Instance Identification

The result of describing a service with a model is an instance. For the global trading of services, each instance needs to be uniquely identified. USDL/XML and USDL/Ecore proposed to use universally unique identifiers (UUID) [14]. Problems associated with this approach include the need for a central management of identifiers and "hiding" service information, e.g., service name and provider, into a number via an encoding mechanism. On the other hand, Linked USDL/RDFS relies on URIs to provide a simple way to create unique global identifiers for services. Compared to UUID, Linked USDL URIs are more adequate to service distribution networks since they are managed locally by service providers. The same URI, which provides a global unique identifier for a service, also serves as endpoint to provide uniform data access to the service description. A Linked USDL URI can be used by, e.g., RDFS browsers, RDFS search engines, and web query agents looking for cloud service descriptions.

4 Linked USDL Family

Linked USDL[6,7] is segmented into modules that together form the Linked USDL family. The objective of this division is to reduce the overall complexity of service modeling by enabling providers to only use the modules needed. Currently, five modules exist but they have different maturity levels. The modules identified

[6] `http://www.linked-usdl.org/`
[7] `http://github.com/linked-usdl/`

with one star ($\star$) have been developed only as a proof of concept. The modules identified with two stars ($\star\star$) have passed the proof-of-concept stage and are being finalized. The modules identified with three stars ($\star\star\star$) are ready and have been validated.

– `usdl-core` ($\star\star\star$). The core module covers concepts central to a service description. It includes operational aspects, such as interaction points that occur during provisioning, and the description of the business entities involved.
– `usdl-price` ($\star\star\star$). The pricing module provides a range of concepts which are needed to adequately describe price structures in the service industry.
– `usdl-agreement` ($\star\star$). The service level module gathers functional and non-functional information on the quality of the service provided, e.g., availability, reliability, and response time.
– `usdl-sec` ($\star$). This module aims at describing the main security properties of a service. Service providers can use this specification to describe the security features of their services.
– `usdl-ipr` ($\star$). This module captures the usage rights of a service, which are often associated with the concept of copyright.

For example, the `usdl-agreement` module is being reconstructed with the objective to align it with the WS-Agreement specification. Customers and providers can use `usdl-agreement` to create service level agreements to, afterwards, monitor whether the actual service delivery complies with the service level agreed terms. In case of violations, penalties or compensations can be directly derived.

Linked USDL Core can be regarded as the center of the Linked USDL family since it ties together all aspects of service descriptions distributed across the USDL modules. Figure 2 shows the conceptual diagram of the core module. Classes are represented with an oval, while properties with an edge. Linked USDL Core has 12 classes and 13 properties (the reader is referred to [15] to understand the purpose of using external vocabularies such as `GoodRelations`, `SKOS`, and `MSM`).

Other modules are being developed as proofs of concept. For example, Linked Service System USDL (LSS USDL)[8] provides modeling constructs to capture the concepts of a service system. While Linked USDL looks into the external description of a service, i.e., a service is seen as a black box, LSS USDL looks inside the 'box' to describe its elements.

4.1 Standardization Efforts

Service standards are expected to drive the industrialization of the service market, to increase transparency and access, to lead to higher trading of services across countries, and to contribute to a new level of service innovation by aggregation or composition. Linked USDL fills the gap by proposing a specification language which enables the unified formalization of business and technical aspects.

[8] `https://w3id.org/lss-usdl/v2/`

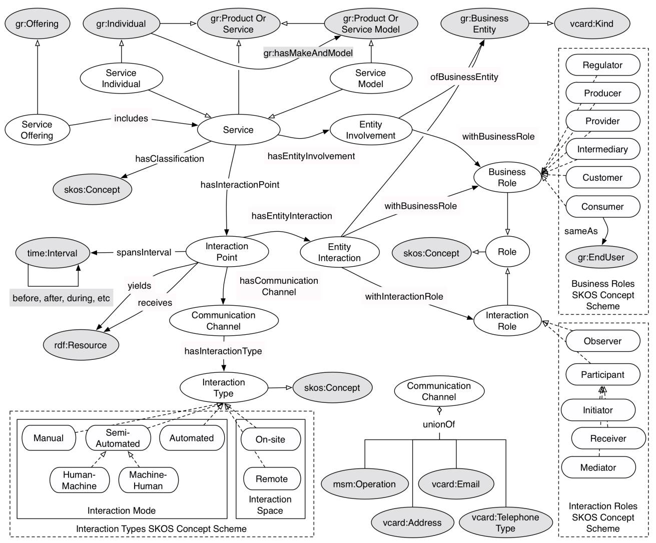


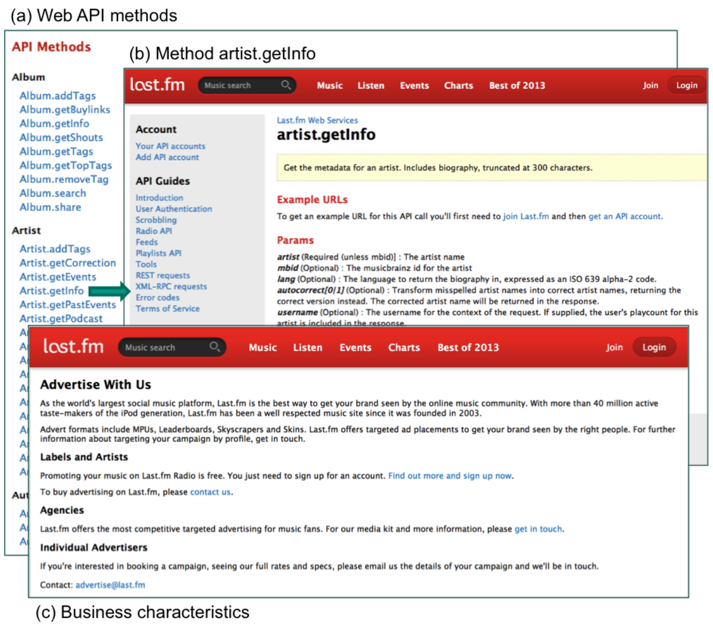
Fig. 2. Linked USDL Core schema [15]

A W3C Unified Service Description Language Incubator Group was initiated by Attensity, DFKI, SAP, and Siemens on September 2010. The group concluded its activities on October 2011. The objectives were to investigate related standards and approaches; re-design USDL to include feedback, requirements, and related work, and define and implement reference test cases to validate USDL. The final outcome was a report and a reworked USDL specification: USDL V3.0 of Table 1.

While USDL did not reach to become a W3C standard after the Incubator Group concluded its activities, the working group agreed that creating a Linked Data version was one of the steps forward for the possible standardization and wider adoption. Linked USDL can evolve toward a language that can fill the gap existing in various fields requiring service modeling such as cloud computing. In fact, in 2012, a report requested by the German Federal Ministry of Economics and Technology [5] indicated that the potential contained in USDL to model services could be adapted to become an important contribution for cloud computing to describe cloud services.

5 Modeling Example

The objective of this modeling exercise is to describe part of the `Last.fm` service using Linked USDL. `Last.fm` is a music recommendation service which can be accessed using a browser or programmatically by accessing a Web API. Only part of the service will be described because showing the complete modeling would require a considerable space. Most of the information used for the modeling was retrieved from the web site `http://last.fm` and is shown in Figure 3. The description was written using the Turtle language[9].



artist.getInfo

Get the metadata for an artist. Includes biography, truncated at 300 characters.

Example URLs

To get an example URL for this API call you'll first need to join Last.fm and then get an API account.

Params

artist (Required (unless mbid)] : The artist name
mbid (Optional) : The musicbrainz id for the artist
lang (Optional) : The language to return the biography in, expressed as an ISO 639 alpha-2 code.
autocorrect[0|1] (Optional) : Transform misspelled artist names into correct artist names, returning the correct version instead. The corrected artist name will be returned in the response.
username (Optional) : The username for the context of the request. If supplied, the user's playcount for this artist is included in the response.

Advertise With Us

As the world's largest social music platform, Last.fm is the best way to get your brand seen by the online music community. With more than 40 million active taste-makers of the iPod generation, Last.fm has been a well respected music site since it was founded in 2003.

Advert formats include MPUs, Leaderboards, Skyscrapers and Skins. Last.fm offers targeted ad placements to get your brand seen by the right people. For further information about targeting your campaign by profile, get in touch.

Labels and Artists

Promoting your music on Last.fm Radio is free. You just need to sign up for an account. Find out more and sign up now.

To buy advertising on Last.fm, please contact us.

Agencies

Last.fm offers the most competitive targeted advertising for music fans. For our media kit and more information, please get in touch.

Individual Advertisers

If you're interested in booking a campaign, seeing our full rates and specs, please email us the details of your campaign and we'll be in touch.

Contact: advertise@last.fm

Fig. 3. LastFM web site description and Web API

The class `usdl:Service` provides the entry point for the description. As shown in Listing 1, the new service was named `service_SLastFM`. The specification also includes:

– Associating a service model with the service.
– Specifying the business entities participating during service provisioning.
– Enumerating the interaction points provided by the service.

The class `usdl:ServiceModel` is used to create groupings of services that share a number of characteristics. For example, a service model for the S-LastFM service can group services characterized for supplying online music services. In the same line of thought, the service "Vodafone unlimited internet service" may belong to the grouping "Internet provisioning service". The example from Listing 1 associates the service `service_SLastFM` with the grouping `onlineMusicServiceModel`[10](Line 4).

```
1 :service_SLastFM a usdl:Service ;
```

[9] Turtle – Terse RDF Triple Language (`http://www.w3.org/TR/turtle/`)
[10] The definition of the model `onlineMusicServiceModel` is not provided in this running example.

```
2     dcterms:description "A semantic recommendation service for music.";
3
4     usdl:hasServiceModel :onlineMusicServiceModel ;
5     usdl:hasEntityInvolvement [
6        a usdl:EntityInvolvement ;
7        usdl:ofBusinessEntity :be_SLastFM_Ltd ;
8        usdl:withBusinessRole usdl-br:provider
9     ];.
10    usdl:hasInteractionPoint :ip_Advertise ;
11    usdl:hasInteractionPoint :ip_Artist_GetInfo .
```

Listing 1. The S-LastFM service class

The class `usdl:EntityInvolvement` captures the `usdl:BusinessEntities` involved in the service delivery and the `usdl:Role` they play (lines 5-9). This enables specifying, for instance, that a given music service is provided by a certain company or that a third party is involved in the service delivery chain.

In Listing 1, the business entity is defined with the class `be_SLastFM_Ltd` and its role is defined as `usdl-br:provider`. Linked USDL provides a reference taxonomy of basic business roles that cover the most typical ones encountered during service modeling such as regulator, intermediary, producer, and consumer. The prefix `usdl-br` identifies the taxonomy `usdl-business-roles`[11] which defines the default roles available.

Listing 2[12] illustrates the description of the company providing the S-LastFM service and described with the class `be_SLastFM_Ltd`. The description include the ISIC (International Standard Industrial Classification of All Economic Activities) code for S-LastFM: 5920 – sound recording and music publishing activities. It also specified the NAICS (North American Industry Classification System) code, legal name, tax ID number, and country where the company is located.

```
:be_SLastFM_Ltd a gr:BusinessEntity ;
  foaf:homepage <http://Slast.fm/> ;
  foaf:logo <http://cdn.last.fm/flatness/badges/lastfm_red.gif> ;

  gr:hasISICv4 "5920"^^xsd:string ;
  gr:hasNAICS "512220"^^xsd:string ;
  gr:legalName "SLast.fm Ltd."^^xsd:string ;
  gr:taxID "830 2738 46"^^xsd:string ;

  vcard:hasAddress
     [ a vcard:Work ;
       vcard:country-name "UK"@en ] .
```

Listing 2. Description of the business entity providing the S-LastFM service

[11] `http://linked-usdl.org/ns/usdl-business-roles`

[12] The prefixes `:gr`, `:dcterms`, `:foaf`, and `:vcard` refer to relevant vocabularies such as GoodRelations and Dublin Core

The extract from Listing 1 also defines two interaction points `ip_Advertise` and `ip_Artist_GetInfo` for the service `service_SLastFM` . An interaction point (`usdl:InteractionPoint`) represents an actual step in performing the operations made available by a service. On a personal level, an interaction point can model that consumer and provider meet in person to exchange service parameters or resources involved in the service delivery (e.g., documents that are processed by the provider). On a technical level, this can translate into calling a web service operation. An interaction point can be initiated by the consumer or the provider.

Listing 3 describes the interaction point `ip_Advertise` which enables customers to book advertising campaigns and inquire about rates and specs. Interaction points define four main pieces of information:

– The communication channels that customers or applications can use to interact with a service.
– The entities that are involved during the interaction.
– The resources that are needed for an interaction.
– The resources that are generated from an interaction.

Communication channels are additionally characterized by their interaction type. Linked USDL provides two reference taxonomies covering the main modes (e.g., automated, semi-automated, and manual) and the interaction space (e.g., on-site and remote).

The specification describes how customers can ask for information to advertise a campaign with S-LastFM. This can be done by using traditional mail, a telephone, or email. All the communication channels require a manual (`usdl-it:manual`) and remote (`usdl-it:remote`) interaction. This means that humans, not software applications, will be involved in the interaction.

The example also indicates the role of the two entities that will interact (lines 29-39): both will be participants. This information is represented using the class `usdl:EntityInteraction` which links interaction points to business entity types (e.g., provider, intermediary, and consumer), and the role they play within the interaction (e.g., initiator, mediator, and receiver).

```
1  :ip_Advertise a usdl:InteractionPoint ;
2    dcterms:title "S-LastFM Advertisement"@en ;
3    dcterms:description "If you are interested in booking a campaign,
         seeing our full rates and specs, please send us the details of
         your campaign and we will be in touch."@en ;
4
5    usdl:hasCommunicationChannel [
6      a usdl:CommunicationChannel ;
7          vcard:country-name "UK";
8          vcard:locality "London";
9          vcard:postal-code "SE1 0NZ";
10         vcard:street-address "Last.fm Ltd., 5-11 Lavington Street" ;
11         usdl:hasInteractionType usdl-it:manual ;
12         usdl:hasInteractionType usdl-it:remote
13   ];
```

```
14
15  usdl:hasCommunicationChannel [
16    a usdl:CommunicationChannel ;
17        vcard:telephone "tel:+61755555555" ;
18        usdl:hasInteractionType usdl-it:manual ;
19        usdl:hasInteractionType usdl-it:remote
20  ];
21
22  usdl:hasCommunicationChannel [
23    a usdl:CommunicationChannel ;
24        vcard:hasEmail <mailto:advertise@slast.fm> ;
25        usdl:hasInteractionType usdl-it:manual ;
26        usdl:hasInteractionType usdl-it:remote
27  ];
28
29  usdl:hasEntityInteraction [
30    a usdl:EntityInteraction ;
31    usdl:withBusinessRole usdl-br:provider ;
32    usdl:withInteractionRole usdl-ir:participant
33  ];
34
35  usdl:hasEntityInteraction [
36    a usdl:EntityInteraction ;
37    usdl:withBusinessRole usdl-br:customer ;
38    usdl:withInteractionRole usdl-ir:participant
39  ];
40
41  usdl:receives dbpedia:Advertising ;
42  usdl:yields dbpedia:Contract .
```

Listing 3. An interaction point involving human interaction

Listing 3 shows that the interaction point receives (`usdl:receives`) and yields (`usdl:yields`) resources (lines 41-42). Receives is the input required and yields corresponds to the outcome yielded by an interaction point. The example shows that the interaction point `ip_Advertise` receives an `dbpedia:Advertising` and yields a `dbpedia:Contract`. Naturally, other computer-processable data sources such as `freebase.com` can be used.

While the previous example of an interaction point involved only human participants, the example from Listing 4 illustrates a fully automated interaction which does not require human intervention. Linked USDL covers the most widely used human-based communication channels (e.g., email, phone, and mail) by means of vCard (a standard for electronic contact details), and application-driven channels (e.g., SOAP and REST Web services) by relying on the Minimal Service Model (MSM).

```
:ip_Artist_GetInfo a usdl:InteractionPoint ;
  dcterms:title "Artist metadata"@en ;
  dcterms:description "Get the metadata for an artist. Includes
      biography, truncated at 300 characters."@en ;

  usdl:hasCommunicationChannel :ArtistGetInfo ;

  usdl:hasEntityInteraction :ei_provider ;
  usdl:hasEntityInteraction :ei_customer ;

  usdl:receives dbpedia:Artist ;
  usdl:receives dbpedia:Software_license_server ;
  usdl:yields dbpedia:Record_software .

:ei_provider a usdl:EntityInteraction ;
  usdl:withBusinessRole usdl-br:provider ;
  usdl:withInteractionRole usdl-ir:participant .

:ei_customer a usdl:EntityInteraction ;
    usdl:withBusinessRole usdl-br:consumer ;
    usdl:withInteractionRole usdl-ir:initiator ;
    usdl:withInteractionRole usdl-ir:receiver .
```

Listing 4. An interaction point for an application-driven interaction

The first interaction point `ip_Advertise` established a remote communication channel between the provider and the customer. The interaction is manual from both sides of the channel. Nonetheless, the interaction point `ip_Artist_GetInfo` shown in Listing 4 is different: it is automated. This means that in both sides of the communication channel, applications will be involved during service provisioning by exchanging data. This requires a well-defined programming interface which must be understood by applications.

A `usdl:ServiceOffering` is an offering made by a `gr:BusinessEntity` of one or more `usdl:Service` to customers. An offering usually associates a price, legal terms of use, and service level agreements with a service. In other words, it makes a service a tradable entity. Listing 5 illustrates an offering named `offering_SLastFM` for the service `service_SLastFM` (Lines 1 and 10). A service offering may have limited validity over geographical regions or time. The offering adds various pieces of information such as temporal validity, eligible regions, and accepted payment methods (Lines 2-9).

```
1  :offering_SLastFM a usdl:ServiceOffering ;
2     gr:validFrom "2014-01-17T09:30:10Z"^^xsd:dateTime ;
3     gr:eligibleRegions "DE"^^xsd:string, "US-CA"^^xsd:string ;
4     gr:acceptedPaymentMethods gr:VISA, gr:ByBankTransferInAdvance ;
5     gr:eligibleDuration [
6         a gr:QuantitativeValue ;
7         gr:hasValueInteger "1"^^xsd:int ;
8         gr:hasUnitOfMeasurement "MON"^^xsd:string
9     ] ;
10    usdl:includes :service_SLastFM ;
```

```
11
12    usdl:legal :legal_SLastFM ;
13    usdl:price :price_SLastFM .
```

Listing 5. A concrete offering of a service

Finally, the last part of the example indicates that the classes `legal_SLastFM` and `price_SLastFM` describe the legal aspects and the price of the S-LastFM service, respectively (lines 12-13).

6 Related Work

In the past, schemas have been explored to describe (web) services. For example, WSDL, a W3C standard, focused on describing technical aspects of web services such as interaction interface and protocols. Since WSDL was essentially a specification for the syntax to describe services it was insufficient, the accuracy of service search algorithms was inadequate, especially at a global scale. Therefore, there was research streams towards the semantic representation of web services. Service descriptions were annotated with semantics to improve not only search but also composability and integration. As a result, new description languages, such as OWL-S [13], Semantic Annotation for WSDL (i.e., SAWSDL) [11], and WSMO [16], were proposed. The research has only tackled the semantic enrichment of function-based services, such as WSDL and REST, by using domain knowledge describing mainly technical interfaces.

In fact, legal aspects, pricing models, and service levels are all elements which need to be explicitly described when dealing with cloud services. Therefore, efforts were redirected to the development of new languages to capture business and operational perspectives beside the technical one. USDL [6] and Linked USDL [15] are probably the most comprehensive attempts.

The most notable effort able to represent and reason about business models, services, and value networks is the e^3 family of ontologies which includes the e^{3}service and e^{3}value ontologies [1, 9]. This research has, however, not been much concerned with the computational and operational perspectives covering for instance the actual interaction with services. Likewise, the technical issues related to enabling a Web-scale deployment and adoption of these solutions were not core to this work. GoodRelations [10] (GR) on the contrary is a popular vocabulary for describing semantically products and offerings. Although GR originally aimed to support both services and products, it is mostly centred on products to the detriment of its coverage for modelling services, leaving aside for instance the coverage of modes of interaction, or the support for value chains.

7 Conclusion

Services and service systems, such as cloud services and digital government services, are showing increasing interests from both academia and industry.

Among the many aspects which still require to be studied, such as service innovation, design, analytics, optimization, and economics, service description is one of the most pressing and critical components since it is a keystone supporting a web of tradable services.

While several service description languages have been developed over the past 10 years to model software-based service descriptions, such as WSDL, OWL-S, SAWSDL, e^3service, and e^3value ontologies, a language that also covers business and interaction aspects is missing. This paper summarizes our efforts to create USDL and, more recently, Linked USDL, a family of languages providing a comprehensive view on services to be used by providers, brokers, and consumers when searching, evaluating, and selecting services.

References

1. H Akkermans, Z Baida, J Gordijn, N Peña, et al. Value Webs: Ontology-Based Bundling of Real-World Services. *IEEE Intelligent Systems*, 19(4):57–66, 2004.
2. Mikaël Barbero, Frédéric Jouault, Jeff Gray, and Jean Bézivin. A practical approach to model extension. In *MDA - Foundations and Applications*. Springer, 2007.
3. Alister Barros and Daniel Oberle. *Handbook of Service Description: USDL and Its Methods*. Springer, 2012.
4. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 4(2):1–22, 2009.
5. BMWi. The standardisation environment for cloud computing. Technical report, Germany Federal Ministry of Economics and Technology, Feb. 2012.
6. J. Cardoso, A. Barros, N. May, and U. Kylau. Towards a unified service description language for the internet of services: Requirements and first developments. In *IEEE International Conference on Services Computing (SCC)*, pages 602 –609, July 2010.
7. Jorge Cardoso, Matthias Winkler, and Konrad Voigt. A service description language for the internet of services. In *Int. Symp. on Services Science (ISSS'09)*, 2009.
8. Anne Geraci. *IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries*. IEEE Press, 1991.
9. Jaap Gordijn, Eric Yu, and Bas van der Raadt. e-service design using i* and e3value modeling. *IEEE Software*, 23:26–33, 2006.
10. Martin Hepp. GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In *Knowledge Engineering: Practice and Patterns*. Springer, 2008.
11. J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell. SAWSDL: Semantic Annotations for WSDL and XML Schema. *Internet Computing, IEEE*, 11(6):60 –67, 2007.
12. Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st symposium on principles of database systems*, pages 233–246. ACM, 2002.
13. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, et al. OWL-S: Semantic markup for web services. *W3C Member submission*, 22:2007–04, 2004.
14. N. Paskin. Toward unique identifiers. *Proceedings of the IEEE*, 87(7):1208–1227, July 1999.
15. Carlos Pedrinaci, Jorge Cardoso, and Torsten Leidig. Linked USDL: A vocabulary for web-scale service trading. In *11th Ext. Semantic Web Conference*, Greece, 2014.
16. Carlos Pedrinaci, John Domingue, and Amit Sheth. *Handbook on Semantic Web Technologies*, volume Semantic Web Applications, chapter Semantic Web Services. Springer, 2010.