Project Number:	**215219**
Project Acronym:	**SOA4All**
Project Title:	**Service Oriented Architectures for All**
Instrument:	**Integrated Project**
Thematic Priority:	**Information and Communication Technologies**

D11.3.1 Report on the Potential Standardization of the USDL

Activity 4:	Exploitation and Impact
Work Package 11:	Standardization
Due Date:	28/02/2011
Submission Date:	08/03/2011
Start Date of Project:	01/03/2006
Duration of Project:	36 Months
Organization Responsible of Deliverable:	SAP
Revision:	1.0
Authors:	Patrick Un (SAP), Juergen Vogel (SAP), Reto Krummenacher (UIBK), Carlos Pedrinaci (OU), Jean-Pierre Lorré (Petals)
Reviewers:	Barry Norton (KIT), Sandra Stinčić Clarke (BT)

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	17.01.2011	ToC	Patrick Un (SAP), Juergen Vogel (SAP)
0.2	20.01.2011	ToC and structure updated	Patrick Un
0.x	08.02.2011	Service Models (Section 3.2)	Reto Krummenacher (UIBK)
0.3	12.02.2011	Added further standardization sec.	Patrick Un
0.4	15.02.2011	Included Section Contribution and merged	Reto Krummenacher, Patrick Un
0.5	19.02.2011	Integration of contributions	Patrick Un , Carlos Pedrinaci (OU), Reto Krummenacher, Jean-Pierre Lorré (PETAL)
0.6	22.02.2011	Draft version for peer review	Patrick Un
0.7	01.03.2011	Incorporation Review BT	Reto Krummenacher
0.8	03.03.2011	Incorporation Review KIT	Reto Krummenacher
1.0	04.03.2011	Final version	Juergen Vogel

Table of Contents

List of Figures

List of Tables

Glossary of Acronyms

Acronym	Definition
API	Application Programming Interface
BPEL	Business Process Execution Language
BPM	Business Process Modeling
BPMN	Business Process Modeling Notation
CMS	Content Management System
CRM	Customer-Relationship Management
D	Deliverable
DSB	Distributed Service Bus
EC	European Commission
EJB	Enterprise Java Beans
EMF	Eclipse Modeling Framework
EP	Enterprise Portal
ERP	Enterprise Resource Planning
ES	Enterprise Service
ESR	Enterprise Service Repository
ESB	Enterprise Service Bus
EU	European Union
EUD	End User Development
FOAF	Friend of a Friend
FTP	File Transfer Protocol
GDT	Global Data Type
GUI	Graphical User Interface
HCM	Human Capital Management
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
ISO	International Organization for Standardization
IT	Information Technology
M	Month
MSM	Minimal Service Model
NLP	Natural Language Processing
OCL	Object Constraint Language
OWL	Web Ontology Language

POSM	Procedure-oriented Service Model
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	RDF Schema
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SaaS	Software as a Service
SAWSDL	Semantic Annotations for WSDL
SCM	Supply Chain Management
SD	Standard Deviation
SEE	Semantic Execution Environment
SEI	Service Endpoint Interface
SLA	Service Level Agreement
SME	Small and Medium Enterprise
SML	Service Modeling Language
SOA	Service-Oriented Architecture
SOA4All	Service-Oriented Architectures for All
SoaML	SOA Modeling Language
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SRM	Supplier Relationship Management
TCO	Total Costs of Ownership
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USDL	Unified Service Description Language
W3C	World Wide Web Consortium
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
XG	Incubator Group
XI	Exchange Infrastructure
XML	Extensible Markup Language

Executive Summary

The Unified Service Description Language (USDL) is a general purpose service description language for various types of services ranging from professional to electronic services. The USDL aims at a 'holistic' service description, which serves the needs of different stakeholders over the entire service lifecycle (provisioning, discovery, consumption, composition, and trading). In addition to technical properties of service (such as the service interface for a Web service), the USDL puts a special focus on business aspects such as ownership and provisioning, release stages and dependencies in a service network, composition and bundling, pricing and legal aspects. The USDL has been developed by SAP and other partners based on results from different public funded research projects, including SOA4All. In this deliverable, we first give an overview of the main concepts and elements of the USDL. We then discuss how the USDL can be aligned with the semantic service description model of SOA4All, WSMO-Lite. As a main result, we argue that the USDL should follow the linked data and link services approach of SOA4All when it comes to describing technical Web services, and we give recommendations for reusing existing ontologies for describing certain service properties. We also report on first results to implement SOA governance for USDL services. Lastly, we describe the current activities of the W3C USDL incubator group for standardizing the USDL.

1. Introduction

The Unified Service Description Language (USDL) is a relatively new initiative by SAP that has been started in 2009 with the aim to create a fundamental service description model that can serve as the basis for an open Internet of Services.[1] The Unified Service Description Language (USDL) is a general purpose language to describe various types of services ranging from professional to electronic services. It aims at a 'holistic' service description, which serves the needs of different stakeholders over the entire service lifecycle (provisioning, discovery, consumption, composition, and trading). In addition to technical properties of a service (such as the service interface for a Web service), the USDL puts a special focus on business aspects such as ownership and provisioning, release stages and dependencies in a service network, composition and bundling, pricing and legal aspects. It proposes a consolidated foundation for service-based systems enabling different roles to participate in diverse aspects of provisioning in service networks. The USDL follows a top-down approach and gives a concrete specification for all possible service properties.

In contrast, WSMO-Lite has been developed in SOA4All for providing lightweight semantic service descriptions following a bottom-up approach (see D3.4.2): WSMO-Lite defines a minimum service model to enrich existing service interface descriptions of technical Web services (WSDL or RESTful) with semantic annotations. The concrete domain ontologies for the semantic annotations are not specified and can be chosen freely.

1.1 Purpose and Structure of the Document

This deliverable takes the viewpoint that the USDL represents a potential service annotation and service provisioning mechanism that can be further harmonized with the existing work on semantic service annotation and semantic service technologies. Furthermore, USDL is viewed as a positive complementary effort to the SOA4All technologies to further describe important aspects of business requirements. In particular business-related metadata, which is becoming more important, alongside functional service metadata, in the real-world service application use cases. Standardization is only possible when there is a clear definition of the prevailing dominance or inevitable need. This deliverable documents the current state and development of USDL, and enumerates a series of comparable or other competitive efforts with similar goals. We also give recommendations for future versions of the USDL.

SAP has initialized a W3C Incubator Group (XG) to coordinate and guide further development of USDL and manage the pre-standardization efforts. SOA4All partners have participated in this effort in form of workshops and cooperation within the XG. In this deliverable, a comparison of the USDL and SOA4All service models is given to analyze potential alignments and/or reuse of concepts. SOA4All will continue to leverage standardization to generate long-term impact in this cooperative manner.

In the next chapter, we describe the USDL and provide an overview of the core and its most important modules. Subsequently, we show how USDL can optimize its potential as a Web standard, and how, in collaboration with SOA4All technologies, it could further flourish (Chapter 3). In Chapter 4, we briefly discuss how SOA Governance could be taken into account for the USDL, and we also present a first Governance support tool that has been developed by EBM for this investigation. In Chapter 5, the current USDL standardization effort is described, and a brief outline of its roadmap and goals is given.

[1] http://www.internet-of-services.com/

2. The Unified Service Description Language (USDL)

So far, research on Service Oriented Architectures (SOA) has concentrated on software applications, constructed by composing and configuring eServices. SOA and Web services have mainly served as technological solutions that enable enterprise functionality to be made available to users as shared and re-usable services on a network. Traditional metadata that describes services intended for these application integration purposes is based on specification languages that were developed in combination with the early service-based system architectures (SOA and Web services). These languages, e.g., WSDL, target the description of technical characteristics of services. However, a key aspect has been discarded, namely the fact that the main goal of an eService is to provide access to a business service, which may need a radically different description than the eService [12]. For an „Internet of Services" that integrates the service world of large enterprises, SMEs, and end users, enabling them to engage as peers (i.e., service consumers as well as service providers) within a network, it is not sufficient to describe the functional and non-functional technical aspects of eServices. Instead, an enhanced service representation of generic (business) services is needed that captures and aligns business, operational and technical characteristics.

With the move to the Internet of Services a new way for service discovery and invocation will emerge. The Internet will be used as a medium for offering and selling services, i.e. treating services as tradable goods [13]. In the Internet of Services, service platforms will enable automatic service discovery, provide a unique service description, service composition and negotiation, as well as QoS-based service level agreements and access rights handling. In order to provide an enhanced service description, which meets the requirements of such a Internet of Services, a conceptual model to capture business-related service data is required. Recently, SAP together with several partners introduced a novel service description language for that purpose, the so-called Unified Service Description Language (USDL). The USDL is based on the insights gained in several publicly-funded research projects (incl. SOA4All) as well as on SAP's general knowledge of business services and processes in different industries and application areas. The first version of the USDL was presented at the Future Internet Assembly (FIA) in Stockholm in 2009. Also, a public Web site is available at http://www.internet-of-services.com/.

The USDL aims at a 'holistic' service description, which serves the needs of different stakeholders over the entire service lifecycle (provisioning, discovery, consumption, composition, and trading of services). In addition to technical properties of service (such as the service interface for a Web service), the USDL puts a special focus on business aspects such as ownership and provisioning, release stages and dependencies in a service network, composition and bundling, pricing and legal aspects.

The USDL builds on models for describing business and technical services, and creates a unified description of related research efforts. The purely business description of services has been driven by research on the E3Service ontology, PAS 1018, and the taxonomy of non-functional properties of services identified by O'Sullivan [14]. From the technical side, the most significant proposals to describe services that have influenced USDL include WSDL, WSMO, and OWL-S. It should be pointed out that USDL is not meant to replace other specifications in the technical service stacks, but aims to complement them by adding essential business information required for the interaction between service consumer and service provider (possibly involving additional roles such as service hoster, service broker etc.). On the other side, the USDL was not designed to target automated services only but is generic enough to be used for the description of manual services that have no technical implementation. The general design principle was to create a unifying entry point into the overall set of service metadata, which in the end comprises several artifacts in different formats.

In SOA4All, we first started to investigate the USDL in WP7 for describing the non-functional business properties of services to be used in service discovery and service composition (see D7.6, D7.5, and D7.7). For instance, price information for each service can be used to select services for a SOA4All process (service composition) such that the price of the overall process is minimized (see D7.6). In SOA4All, services are semantic Web services specified in WSMO-Lite. WSMO-Lite allows specifying non-functional properties for each operation of the service. Thus, we defined a price model for each service following the USDL specification, serialized it in the form of WSML, and attached it to the WSDL service interface description via SA-WSDL following the WSMO-Lite specification.

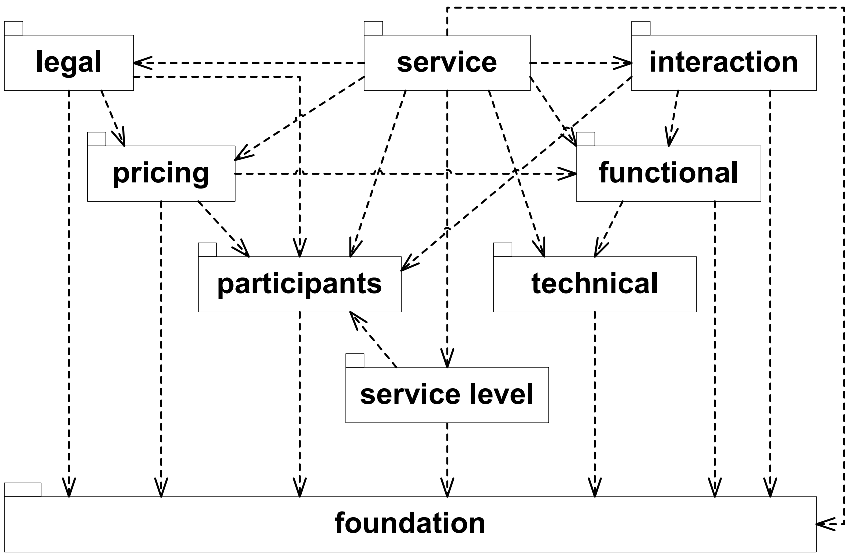


Figure 1: UML package diagram of USDL.

An overview of USDL, in the form of a UML package diagram is reproduced from [3] in Figure 1. The Ecore meta-modeling specification of the Eclipse Modeling Framework (EMF) was chosen, which is an implementation of the OMG EMOF specification developed by the Eclipse community. In turn, EMOF features abstract semantics in terms of the OMG Object Constraint Language (OCL) responding to the requirement.

In regards to the requirement for language expressivity, USDL has to capture the universe of discourse. USDL applies the technology of domain-specific languages. This fact already sets USDL apart from many related approaches such as WSMO or OWL-S which only prescribe relatively small core schemata and leave the modeling of the universe of discourse concepts (such as a generic schema for defining a price model or licenses) to the user. W3C SML, SAWSDL, and SA-REST are also designed to be agnostic to any specific service description schema.

2.1 USDL Modules

USDL is split into several packages (according to UML terminology), as shown in Figure 1, following the principle of modularity. Each package represents one module and contains one class model. The Foundation Module supports common parts of the remaining modules as a consistent continuation of modularization. As such, all other modules depend on the Foundation Module meaning they reference one or more of its elements; dependencies between modules are depicted as dashed lines.

Table 1 : Overview of USDL modules

Name	Description
Foundation Module	Captures concepts that are common among several aspects, e.g., concepts of naming and identification, or concepts that are completely independent of “service,” e.g., organizations or persons.
Service Level Module	Captures concepts concerned with guarantees regarding quality of service operation, which are claimed/requested by different actors involved in the provisioning, delivery and consumption of a service.
Participants Module	Captures concepts related to the actors that participate in the provisioning, delivery and consumption of services, e.g., provider, intermediary, stakeholder and consumer.
Pricing Module	Captures concepts that explicate the pricing structure of a service, e.g., price plan, price component and price level.
Legal Module	Captures licenses and copy rights according to German law. A version for US jurisdiction is in development. The module will eventually also capture general terms and conditions.
Service Module	Captures central service concepts, e.g., service and service bundle, and their relation to other service description aspects.
Interaction Module	Captures concepts that outline the sequence(s) of interactions between a consumer and a service (respectively the actors involved in delivery) – necessary to successfully complete service execution.
Functional Module	Captures concepts that describe the functionality offered as a service, e.g., function, parameter and fault.
Technical Module	Captures concepts that describe available means to access a service, e.g., interface and access protocols.

2.1.1 Modules’ Design

In the following the main constituent modules of USDL are described. More detailed descriptions of the modules and their features are available at http://www.internet-of-services.com/index.php?id=382.

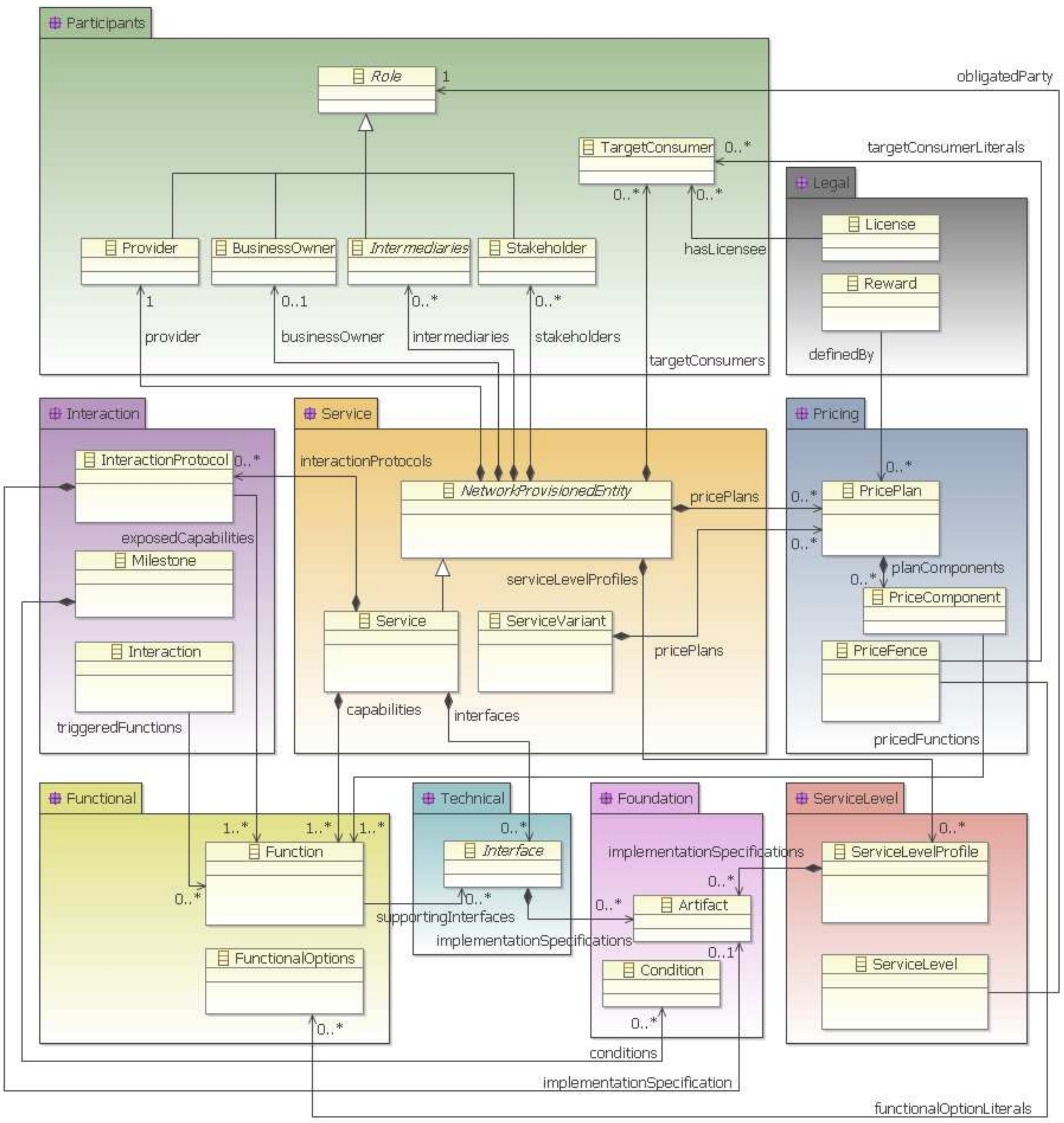


Figure 2: USDL class diagrams and interrelationships.

2.1.2 Foundation Module

Purpose: The Foundation Module encompasses the common parts of the USDL modules as a consistent continuation of the principle of modularity. Because of its basic character, all other modules depend on the Foundation Module and reference one or more of its elements. The Foundation Module also serves as the container for concepts that logically cannot be assigned to any other USDL module, as well as concepts that by themselves do not depend on the central concept of **Service**.

Content: Among the most fundamental and universal elements, which are independent of, but relevant to service description, are Time and Location. They are used in the context of service description, for instance, to express temporal and geographical availability; i.e., the time when and location where a service can be requested and delivered.

2.1.3 Service Module

Purpose: The Service Module can be regarded as the center of the USDL module collection. The module links all aspects of service description distributed across other USDL modules. Existing and separate USDL descriptions can be combined or bundled by establishing links. Similar to establishing connections to existing USDL descriptions for compositions and bundles, arbitrary dependencies can be modeled.

Content: **Service**, **ServiceBundle**, and **CompositeService** are the main concepts. Central information about the main concepts can be captured such as publication time, classifications, certifications, additional documentation, release stage, exposed resources, etc. The main concepts can be linked to price plans, service levels, licenses, stakeholders, intermediaries, etc., in the corresponding modules. In addition, **Dependency** relations can be established between the main concepts and resources. Beside these, there is the concept of a **ServiceVariant** which is a combination of options offered as a pre-packaged version of the service. Another central notion is the concept of an **AbstractService** which is used to represent classes of services, i.e., groups of services that comply with a number of predefined description properties. Services structured to operate in particular functional or organizational structure contexts can be abstracted and carried through into new deployments where they can be concretely instantiated.

2.1.4 Pricing Module

Purpose: The fundamental challenge of the Pricing Module is the modeling of the segmentation rules within price structures; i.e., rules determining when and how different consumers are charged different prices.

Content: For the model to be flexible and comprehensive enough to deal with the pricing complexity of today's service market, the cascading backbone of the Pricing Module is made up of three basic elements in a strict hierarchical structure: **PricePlans**, **PriceComponents**, and **PriceLevels**. This allows the efficient modeling of scenarios with alternative price plans that may be assigned to an offered service or bundle. Each plan may be made up of multiple price components and each component may have varying charges, either by specifying different levels or by adjusting them by means of premiums and discounts. All elements may then be constrained by segmenting conditions detailed in price fences,[2] i.e., the criteria a customer must meet or the service limitations he/she needs to accept to qualify for a certain price.

2.1.5 Legal Module

Purpose: The Legal Module offers a novel approach to represent copyrights in a formal way.

Content: The Legal Module introduces classes such as **License**, **UsageRight**, **Restriction**, or **Requirements**, and interlinks them with classes of the Service Module, namely, **Service** itself, **ServiceProvider**, **TargetConsumer**, or **PricePlan**.

[2] This terminology originates in "*The Strategy and tactics of Pricing*" **Error! Reference source not found.**, which, on top of being the recommended textbook for pricing courses at prestigious faculties such as MIT-Sloan , Berkley-Columbia and Yale, is considered an authoritative source across all industries.

2.1.6 Service Level Module

Purpose: Service Level Agreements (SLAs) are a common way to formally specify exact functional and non-functional conditions under which services are delivered.

Content: The Service Level Module introduces classes such as **ServiceLevelProfile,** which represents a set of service level specifications that are combined into one profile and that are offered, negotiated, or agreed with as a whole. Different profiles can be used to specify different options of how **ServiceLevels** may be specified and grouped. Further classes are **ServiceLevelAttribute**, **GuaranteedState**, **GuaranteedAction**, etc., which are interrelated and linked to Service in the Service Module and Role in the Participants Module.

2.1.7 Participants Module

Purpose: The provisioning, trade, delivery, and consumption of services or service bundles through service networks are all part of a process that potentially involves a multitude of parties or actors.

Content: The Participants Module introduces means for capturing information about individual roles. Consequently, the module adds classes, such as **BusinessOwner**, **Provider**, **Intermediary**, **Stakeholder**, and **TargetConsumer**.

2.1.8 Functional and Technical Modules

Purpose: A service description should express what it is that a service will achieve for the beneficiaries involved (e.g., customers), i.e., its value proposition. In order to enable both the description of human and automated services, the Functional and Technical Modules capture such service functionality in a conceptual way. Conceptual in this context means independent of the ways to technically access functionality (the how part). It is important to distinguish between these two concepts: one is the subject of the service itself and the other is the service’s interface. The reason is that a single service may be available completely or in parts via several interfaces

Content: In case of IT services, the module is able to refer to the interface descriptions. Similarly to WSDL, the Functional Module introduces classes including **Interface**, which represents a technical interface containing interface elements, e.g., operations or parameters that can be referenced explicitly. A technical interface can be “annotated” by a conceptual description of the interface, the operations, and parameters. The annotation class is called **Function**. A Function is used to capture an informal description of what the service does, i.e., its functionality, and can be recursively structured. Each Function may feature one or more input and output **Parameters**, as well as one or more **Faults**. The latter is used to capture information about exceptions that may occur when a function is performed.

2.1.9 Interaction Module

Purpose: The Interaction Module captures knowledge about the capabilities provided by a service and the means of communication employed to access them. In addition, it is necessary to also know in what order individual functions have to be performed or how to interact with the service, respectively the actors performing it.

Content: The main class **InteractionProtocol**[3] defines an order among the **Functions** defined in the Functional Module including the grouping into **Phases** with **Milestones**.

[3] Alternative terms are, for example, business protocol, conversation protocol, or public service view.

Phases consist of one or more Interactions. Each **Interaction** can be linked to an involved Role (in the Participants Module) and a triggered Function (in the Functional Module).

3. USDL and SOA4All

3.1 USDL for the Web

USDL is perhaps the most comprehensive specification around services which has been produced so far. What is most remarkable about USDL is that it is not yet another WS-* technology or another process language. It is instead primarily focused on capturing the main aspects that are relevant for offering services, that is, the actual business activities, in a digital form. This includes, for instance, pricing models, legal aspects, the notion of service bundles, the relationship between services and resources, etc., creating a "master data model for services" as the USDL specification calls it. The magnitude of USDL also carries as a consequence the fact that it is a large specification which will require significant work for its adoption. This is one of the factors that has motivated the creation of the USDL Incubator Group within the World-Wide Web Consortium (W3C). Given its scale, the Web appears as the most appealing environment for deploying and exploiting USDL but it is also most challenging since it is essentially unregulated (apart from the mediation of the W3C).

Looking at the possibilities and means for standardizing as well as promoting and supporting the uptake of USDL on the Web thus necessarily requires an alignment with the essential characteristics of the Web, as well as the application of previous experience in standardization activities and use of these specifications. Then, based on the characteristics of USDL which is a large domain model as opposed to a new technical specification for handling a very specific activity, we turn to Linked Data as the initiative that has had the highest impact in this kind of activity as related to semantic technologies.

3.1.1 Web Architecture and Linked Data

The W3C defines the Web as "an information space in which the items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers (URI)".[4] The Web architecture is based on three main principles, namely the *identification* of resources, enabling the interaction between agents (software or humans) via well-defined *protocols*, and *formats* that govern the representation of data and resources transmitted. The identification of resources on the Web aims to provide a global naming able to promote and support network effects. URIs are the basis for this on the Web and the main rule governing this naming scheme is that different resources should have different identifiers. Communication between networked agents is supported by a range of Web-related protocols including HTTP, FTP, SOAP, and SMTP, to name just a few. Finally, there has been a considerable number of specifications for representing resources, including (X)HTML, RDF/XML, CSS, etc. Although the Web architecture allows for the deployment of new data formats, creating and using new formats is considerably expensive and designers are therefore encouraged to reuse existing ones as much as possible.

These principles have effectively governed the Web and have still maintained the ability to extend it to cope with new kinds of resources, or to enable more complex activities to be carried out. A good example is the work carried out on the Semantic Web towards providing machine interpretable semantic descriptions of resources, which paves the way for the development of more intelligent agents. Most relevant is the use of RDFS and OWL to define domain-specific conceptualizations which are anchored on pre-existing Web standards but still allow the modeling of all sorts of domains in an effective and extensible manner. The focus of these efforts has, however, been essentially on creating ontologies or vocabularies,

[4] http://www.w3.org/TR/webarch/

and on manipulating RDF data described in these terms, leaving somewhat aside the aspects which are more related to exposing semantic data on the Web.

Recently, the so-called Linked Data principles[5] were presented as a means towards creating a Web of Data better suited for machine processing. These principles establish that one should:

1. Use URIs as names for things;
2. Use HTTP URIs so that people can look up those names;
3. Provide useful information, using the standards (RDF*, SPARQL) when someone looks up a URI;
4. Include links to other URIs so that they can discover more things.

Since these principles were proposed we have witnessed an outstanding growth in terms of data and vocabularies allowing people to freely expose and interlink large quantities of heterogeneous data. In fact, for raw data that can effectively be modeled in RDF, Linked Data principles are nowadays considered as the best means for publishing it on the Web.

Currently, in projects outside of SOA4All, USDL service descriptions are modeled as eCore data models and can therefore be serialized in XML. However, given that the main goal is to turn USDL into a global scheme for describing, exposing, and trading services on a large scale, Linked Data principles appear to be most appropriate for publishing USDL data on the Web.

Out of the four principles outlined above, the second and third principles pose essentially technical restrictions on the infrastructure exposing data and shall thus not be discussed in this document, which is concerned with the conceptual modeling of USDL. Finally, the fourth principle has a considerable impact on how resources should be modeled so as to support them being interlinked with existing resources. This last principle essentially impels vocabulary creators to reuse existing ones in as much as possible, as promoted by the Web principles covered earlier. It will also drive the preprocessing of data to interlink it with external resources prior to publication. In the next sections we therefore focus on existing vocabularies and how they could be reused to model USDL and USDL data.

3.1.2 USDL and Linked Data

USDL is composed of a number of modules, as introduced above. In particular, USDL 3.0 milestone 4 (which we use throughout) contains a Foundation Module, and seven other modules, namely, Service, Participants, Functional, Pricing, Interaction, Legal and Service Level. Creating an RDFS version of USDL requires, by definition, capturing the information covered by all these modules. This deliverable limits itself to the Foundation first and to the Service and Functional modules in the next section. These last two modules are the most relevant ones for SOA4All. Through this exercise it shall be seen not only that existing vocabularies and SOA4All models cover these for the most part, but it shall also show that modeling USDL in this manner has a number of benefits like the support for additional capabilities (e.g., temporal reasoning) and the compatibility with existing datasets.

The Foundation module provides the basis for the other ones. In a nutshell this module covers aspects such as Time, Locations, Resources and Agents. Additionally it includes some other elements that are necessary for capturing all the information USDL requires like units, etc.

[5] http://www.w3.org/DesignIssues/LinkedData.html

3.1.2.1 Generic Elements

IdentifiableElement is the super type of all elements in USDL and it used for ensuring elements can be uniquely identified. This notion is directly catered for by RDF where every node (except literals and blank nodes) has its own unique URI.

ElementDescription is used almost everywhere to capture aspects such as the *name*, the *description*, the *concept* and the *language* of the element being characterised. Most of these aspects are directly supported by RDF(S) using for instance *rdfs:label* (one could also use in addition *skos:altLabel* and *skos:prefLabel* from SKOS[6]), *rdfs:comment*, *dcterms:description* from Dublin Core[7] and the internationalisation support in RDF (e.g., *xml:lang*). The term *concept* is therein used for pointing to classes in external ontologies, which is precisely what *sawsdl:modelReference* is for, see SAWSDL[8]. In other cases, when domain-specific vertical extensions can be used, one could directly use RDF concepts and *rdf:type*.

ElementDescription also uses keywords. Essentially the role of this attribute is to support tagging elements. Although tagging, as opposed to annotating using existing ontologies, is somewhat open ended (i.e., one may use any term one thinks of), there has been a significant amount of work on creating vocabularies used to express that a certain resource has been tagged (not for fixing the actual tags that can be used). The interested reader is referred to [5] and Common Tag[9] for more information on some of the main possibilities. Choosing one over the other, at the moment, is somewhat a matter of taste since there seems not to be a de-facto standard. For our purposes having a simple tagging mechanism should be sufficient. Thus, Newman's Tag Ontology[10], and Common Tag are pretty good options. The latter, however, is supported by some companies like Yahoo! which may make more sense for USDL for its potential impact commercially.

Artifact allows pointing to additional metadata of different kinds including mimeType, URI, copyright. Dublin Core deals precisely with this. Notably among others it includes attributes such as *dcterms:type*, *dcterms:FileFormat*, *dcterms:identifier*, *dcterms:rights*, *dcterms:creator*. *Artifacts* have *ArtifactTypes* which could be modeled in several ways. SKOS could be used to define knowledge organisation systems, or alternatively one could use formal ontologies. In the latter case a similar approach to that used in WSMO-Lite for Functional Classifications[11] could be used by introducing a concept *ArtifactClassificationRoot* that any artifact classification should instantiate with its root concept. USDL includes a set of predefined types which we could reuse to model easily the categorizations using both approaches.

Resource is a generic concept to represent classes of real-world objects. This notion seems to be used in USDL as if it were a role played by certain entities. Among its properties, there are a number of attributes identified which were previously dealt with such as name, type and descriptions. Again USDL points here to the notion of classifications which could be approached in the same way as *ArtifactType*. Concerning *ResourceTypes*, USDL defines *SoftwareResource*, *HumanResource*, and *PhysicalResource* as the possible kinds. These

[6] http://www.w3.org/TR/skos-reference

[7] http://www.dublincore.org/schemas/rdfs/

[8] http://www.w3.org/TR/sawsdl/

[9] http://www.commontag.org/

[10] http://www.holygoat.co.uk/projects/tags/

[11] http://www.w3.org/Submission/WSMO-Lite/

concepts are all defined in other vocabularies, e.g., *dcterms:Software*, *foaf:Person*, and *dcterms:PhysicalObject;* see FOAF.[12] Again, we could structure the possible types of Resources in a formal ontology or using SKOS.

3.1.2.2 Time Representation

USDL includes quite a few concepts for handling time including *Time* (an abstract class), *TimeInstant*, *TimeInterval*, *AbsolutePointInTime*, *RelativePointInTime*. Representation of, and reasoning over, temporal models is a topic that has received considerable attention in the literature. Indeed, Semantic Web researchers encountered this issue early in the lifetime of that community, and there has already been quite some work on representing time. In particular, perhaps the most famous is OWL Time, which is hosted by W3C[13]. The Time Ontology defines temporal entities and temporal relationships inspired by James Allen's interval temporal algebra [4]. It therefore identifies Instants, defines Intervals on the basis of beginning and end Instants and includes the typical temporal relationships between Instants and between Intervals. Additionally, USDL provides the notion of DurationInterval which allows to convey statements like "during the next 3 years, starting from today". This notion is directly supported by OWL Time. Finally, OWL Time imports the Timezone ontology which covers the notion of timezone and provides some basic geographical modeling capabilities for (political) regions. [14]

On the basis of OWL Time, temporal manipulation would preserve USDL’s expressivity, in this regard, and further support for reasoning about intervals and instants could be provided based on an implementation of Allen's interval temporal algebra easily for reasoning about intervals and instants.

There still remain, however, some minor issues which would require further attention:

1. The most fine-grained value in OWL Time are seconds which may not be enough for automated settings;
2. *RecurrentTime* is not supported;
3. *TimePattern* is not supported – this notion is currently underspecified in USDL and therefore disregarded here;
4. *RelativePointInTime* is not directly supported although it could be specified relatively easily.

3.1.2.3 Location

One core aspect of USDL Foundation module concerns location related entities and relationships. In particular USDL includes the supertype for all location related entities, namely *Location*, and the elements *PhysicalLocation*, *GeographicalPoint*, *PhysicalAddress*, *AdministrativeArea* & *AdministrativeAreaType*, *Polygon*, *Area*, *VirtualAddress* & *VirtualAddressType*, *MessagingAddress* and *VirtualRegion*.

Currently, perhaps the most reused vocabulary for geographic-related aspects is W3C Basic Geo Vocabulary[15] which allows capturing *GeographicalPoints* on the basis of their *latitude*, *longitude* and *altitude* according to the WGS84 coordinate system. In addition to this vocabulary, there is a comprehensive suite of vocabularies devised by Ordnance Survey as

[12] http://xmlns.com/foaf/spec/

[13] http://www.w3.org/TR/owl-time/

[14] http://www.w3.org/2006/timezone

[15] http://www.w3.org/2003/01/geo/wgs84_pos

part of the Data.gov.uk[16] initiative for the public release of a large quantity of governmental data in the UK[17]. Notably, they have devised:

- the Spatial Relations vocabulary for basic spatial relationships (e.g., contains, touches);[18]
- the Administrative geography and civil voting area ontology covering administrative divisions in the UK (e.g., region, county); [19]
- the Geometry ontology for describing abstract geometries (rather limited, this may be work still in progress); [20]
- the Postcode ontology which covers the modeling of UK postcodes including aspects such as Postcode Area and Postcode District. [21]

Although some of these vocabularies are somewhat specific to the UK, they could be reused, extended or adapted to deal with other countries. The level of detail to be captured should be minimized in as much as possible to avoid highly detailed cross-country definitions which may not be that beneficial for the adoption of USDL.

In addition to this effort, the W3C Geo Incubator Group devoted also some effort which is worth highlighting here.[22] This group produced a study of existing approaches as well as a simple and reusable vocabulary.[23] This vocabulary is perhaps a better option for capturing the basic geometry than the one from Ordnance Survey.

Finally, it is worth noting the ontology produced by geonames.org.[24] This ontology contains some of the relations previously indicated but perhaps the most relevant aspect is that it comes together with a large knowledge base of locations and services for accessing this data. Given that these would mostly be used at the instance level to create links or reuse data, we should not discuss further these aspects herein.

The vocabularies listed here could cover to a great extent the location specific requirements for the module. We still require, however, supporting the notion of *Address* and *Virtual Address* which we will cover next.

3.1.2.4 Addresses

In addition to generic and administrative geographic aspects, USDL provides support for capturing addresses, both physical (i.e., postal addresses), and virtual ones (e.g., email). This notion has largely been addressed by the vCard vocabulary[25] recently submitted to W3C. Although the vocabulary does not explicitly distinguish between physical and virtual addresses, it does include the main communication means (e.g., telephone, email, postal), and if necessary we could easily include this distinction. Although quite detailed, this

[16] http://data.gov.uk/

[17] http://www.ordnancesurvey.co.uk/

[18] http://data.ordnancesurvey.co.uk/ontology/spatialrelations/

[19] http://data.ordnancesurvey.co.uk/ontology/admingeo/

[20] http://data.ordnancesurvey.co.uk/ontology/geometry/

[21] http://data.ordnancesurvey.co.uk/ontology/postcode/

[22] http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/

[23] http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/

[24] http://www.geonames.org/ontology/documentation.html

[25] http://www.w3.org/Submission/vcard-rdf/

vocabulary does not cover some of the types such as *URL*, and *IPv4-v6*, nor does it cover the notion of *VirtualRegions* (e.g., *URI Templates*). These aspects should be added to the vocabulary, and the notion of *URI Template* possibly reused from the hRESTS extension to the Minimal Service Model.[26]

3.1.2.5 Agents

The concept *Agent* represents all entities that can take active part in the provisioning of a service. This term appears in a number of vocabularies, notably *dcterms:Agent*, and *foaf:Agent*. The notion of *Agent* is also intended to capture organisations which are covered in other vocabularies, one of which is GoodRelations;[27] e.g., *gr:BusinessEntity*. Agents may be classified (see previous cases on how to approach this), and they may have *Certifications*. There exist ontological approaches for supporting the modeling of quality certifications such as ISO 9000, see for instance [8] part of the work on TOVE. This would, however, add complexity which may not be necessary. Perhaps within other modules this becomes more relevant and we may need to revisit this issue.

3.1.2.6 Handling of Units

Finally, USDL includes some support for handling units. The support included is, however, limited to basic units and no knowledge is explicitly captured about the relationships between different units for the same dimension, different dimensions, and the implications from a processing perspective. Indeed, this limits to a significant extent what the information about units can be used for. Currently, there are no established vocabularies for handling units. GoodRelations includes a very limited support through the notion of *gr:QuantitativeValue* and *gr:UnitPriceSpecification*. Additionally there are other approaches like the work in progress on the Units of Measure ontology started within the Ontolog forum, the work on EngMath by Tom Gruber [7], or the work derived from EngMath presented briefly in [6] which covers the notions of physical quantities, international system of units, prefixes, dimension, and additional machinery for automating the manipulation of quantities taking into account their dimensions and units. On the basis of the approaches mentioned above, the handling of units within an RDFS model for USDL could easily be more advanced than it currently is, on the basis of the XML Schema, and it would better support the use of USDL on a world-wide basis where different systems of units are often applied.

3.2 USDL and the SOA4All Service Models

The service models in SOA4All are based on the conceptual model of the Web Service Modeling Ontology (WSMO). Although, the project has taken WSMO as starting point, the focus was on developing more lightweight and bottom-up models to emphasize the Web aspect of Web service infrastructures (cf. previous section) without, however, omitting the conceptual foundations provided by the WSMO framework and the WSML language stack. In this section a *compare and contrast* analysis of USDL is provided and the more lightweight models that emerged from SOA4All, although on a conceptual level various pointers to WSMO will be given. In fact, conceptually, USDL has significantly more in common with WSMO than with the SOA4All technologies such as WSMO-Lite and the RPC-based minimal service model that are linked to existing services descriptions via SAWSDL or MicroWSMO. USDL, as WSMO did, takes a top-down approach and provides a service description that

[26] http://cms-wg.sti2.org/minimal-service-model/

[27] http://www.heppnetz.de/ontologies/goodrelations/v1

only points back to the original service description via the 'implementationArtifact' element in the functional description of the technical interface.[28] On the other hand, In SOA4All, semantic service descriptions are directly integrated into the WSDL or HTML description of a service via the semantic linkage offered by SAWSDL and MicroWSMO; hence offering a bottom-up approach.

Further contrast can be drawn from SOA4All's focus is on the modeling of Web services' capabilities/functionality and the service interfaces, while USDL also describes services from a business perspective. As discussed later, some of these business aspects are in SOA4All reflected in so-called non-functional properties. Still, USDL yields a much more comprehensive model in this respect. In the context of USDL, the elements relevant in SOA4All are mostly referred to as the technical perspective of a service description.

While compositional aspects were highlighted in WSMO through the specification of choreographies and orchestrations, SOA4All took these aspects out of the much simpler service models and composition is guided by LPML process models (see D6.3.3). Similarly, at the level of a service network, USDL does not distinguish atomic services from those that are internally composed. In that sense, both the service models of SOA4All and USDL focus on services as exposures of functionality or provider of some value, and on how this value is accessed and made profitable within a service network. SOA4All, also however, provides via LPML a model of the composition; such processes are not modeled by USDL. A second distinction comes from USDL's insistence that composite services be made up only of component services from other providers; in SOA4All a service provider can certainly use LPML to provide composed versions of their own services. On the other hand, USDL distinguishes service compositions from service bundles; these latter are not treated in SOA4All models. Bundles are aggregations that are distinct from compositions, as services in a bundle are not composed in a functional way, but are merely grouped for the classification according to certain business related dimensions and business purposes such as selling.[29] Composite services on the other hand are built with the intent to create new functional value. In SOA4All, there is no such thing as service bundles, nor is there any use for such a concept in regards to the intended application of the service description models. In the long run, considering the situation beyond the SOA4All project lifetime, such concepts of 'higher-level' aggregation of services could be interesting to augment the functionality currently offered by SOA4All discovery components and service registries.

In the remainder of this section, USDL is firstly compared to the approach taken in SOA4All for describing the functional aspects of services and their interfaces. The aim is to present how far USDL matches "standard" Semantic Web service description models and to see how USDL could be aligned with the solutions that are proposed by SOA4All. In a second section, a look is taken at the non-functional aspects. As USDL explicitly does not only target Web services, they have various dedicated modules that aim at models for describing non-technical elements of services such as legal aspect and pricing.

3.2.1 Functional Aspects of Service Descriptions

To recall, the core elements of the SOA4All service models are the concepts *FunctionalClassificationRoot*, *Condition*, *Effect*, as well as *NonfunctionalParameter*, which is subject to the next section, that are specified in the W3C member submission WSMO-Lite,

[28] This claim obviously only applies for services that have an original description, such as a WSDL or HTML file. As SOA4All only considers such services, in the context of this deliverable the claim of USDL being a top-down approach thus applies; in the context of non-technical business services the argumentation needed however to be reconsidered.

[29] In the context of e^3value such aggregations are also referred to as networked value constellation 0.

and the structural information such as service and operation that are defined by the minimal service model, also known as the procedure-oriented service model (MSM/POSM). These are hence the primary elements on which this *compare and contrast* analysis is based.

Although concepts such as functional classification are specified in the USDL modules foundation and service, the model aspects which are most relevant for the comparison are given in the functional module that subsumes the other two. The USDL functional module captures both conceptual functionality on the one side and the technical realization that exposes this functionality on the other (if applicable).

The USDL element *Capability* is used to capture an informal description of the service functionality in terms of a course of action that is offered to consumers. Hence, in USDL any service has to have at least one capability – otherwise it cannot be considered a service. USDL describes capabilities, i.e., the actions, with inputs, outputs, faults, pre- and post-conditions plus further information on affected resources and concrete operations that realize the capability. In WSMO, a service has exactly one capability that is given by preconditions and effects. The same accounts for WSMO-Lite for which conditions can however be linked to both services and individual operations. Conceptually USDL and the SOA4All approach thus largely correspond.

More concretely, operations in SOA4All map to technical operations in USDL. Technical operations are the realization of the actions that constitute a capability. While at the level of WSMO-Lite inputs and outputs are only implicitly taken into account via formal expressions that specify the conditions, the MSM/POSM constructs allow for explicitly annotating input/output messages and faults; technically, instances of the 'Message' concept can be annotated using the SAWSDL *modelReference* predicate. USDL has concrete counterparts to the MSM/POSM message concept that are termed 'TechnicalParameter' and 'TechnicalFault', both given in the foundation module. While in SOA4All messages can be arbitrarily concrete; e.g., typed or even linked to more complex logical expressions and conditions, in USDL their attributes are predefined and limited to name, type and cardinality restrictions. Note that in USDL the concept 'Action' is defined to capture an informal description of individual steps/operations of a service. Actions are, in the case of technical Web services realized, as stated above, by Technical Operations and are exposed by one or more 'TechnicalInterface'.[30] In the sense of SOA4All, every technical interface results in one service, and the USDL elements 'Action' and 'TechnicalOperation' become one and the same. SOA4All is indeed only interested in services that have a technical interface in the USDL conceptual model.

In summary it shows that from a functional point of view, the approach taken by USDL is very much compatible with the one promoted by SOA4All: input/output messages, faults and conditions. In regards to the realization of the service descriptions the differences remain: USDL relies on XML and provides a top-down description of services, while SOA4All promotes RDF and has schemas expressed mainly using RDFS. Still, the proposal by USDL to use arbitrary expression languages for the formalization of conditions could be easily aligned with SOA4All, where conditions and effects, and also non-functional properties, are given by logical expressions in some formal language such as WSML or RIF; with the latter now preferred as it is a W3C recommendation. Table 2 provides a tabular summary of the alignment between the main elements that USDL and SOA4All have in common with respect to the description of functional aspects.[31]

[30] Technical interfaces are offered by services with a software implementation. Services may offer several technical interfaces realized in different technologies; e.g. WSDL and REST.

[31] The prefixes wsmo-lite, posm, rdfs, and sawsdl in Table 2 refer to http://www.wsmo.org/ns/wsmo-lite#, http://www.wsmo.org/ns/posm/0.1#, http://www.w3.org/2000/01/rdf-schema#, and http://www.w3.org/ns/sawsdl#, respectively.

Table 2: Comparison of USDL and SOA4All

USDL functional module	SOA4All service models
TechnicalInterface	posm:Service
implementationArtifact Artifact	rdfs:isDefinedBy rdf:Resource
operations TechnicalOperation	posm:hasOperation posm:Operation
TechnicalOperation	posm:Operation
precondition TechnicalCondition	sawsdl:modelReference wmso-lite:Condition
postcondition TechnicalCondition	sawsdl:modelReference wsmo-lite:Effect
inputs / outputs Technical Paramter	posm:hasInput / hasOutput posm:Message
Infaults / outfaults Technical Fault	posm:hasInputFault / hasOutputFault posm:Message
TechnicalCondition	wsmo-lite:Condition / Effect
conditionExpression ExpressionElement	rdf:value xsd:string (LogicalExpression)

Only quickly mentioned above, the WSMO-Lite *FunctionalClassificationRoot* finds a correspondence in USDL at two levels.[32] In the foundation module, USDL includes a *Concept* element within the generic description class – *ElementDescription* for offering different information elements – that provides access to Semantic Web data coming from well-defined global ontologies, classification schemas and taxonomies. In this context *Concept* matches the USDL *Keyword* element that points to arbitrary non-ontology tags.

An even more concrete match between USDL and WSMO-Lite in this respect is given by the *Classification* class, which is bound to *Service* elements according to the service module of USDL. Although USDL does not explicitly refer to ontologies, as WSMO-Lite does, but rather to taxonomies in general, it shares the exact same role as the *FunctionalClassificationRoot class.*

Before concluding this section on functional aspects, an interesting access point for further alignment between SOA4All and USDL is highlighted. In the interaction module there is a concept *InteractionProtocol* defined which captures the externally observable behavior of a service in terms of atomic communication steps, i.e., its interactions. Conceptually, the interaction protocol matches a WSMO Choreography which, however, is not part of the lightweight service description models used and promoted in SOA4All. Although there is no overlap with existing standardization activities, various potential links to the work on process descriptions were pointed out and LPML that is conducted in WP6 of SOA4All. In USDL, so-called phases – a sequentially ordered set of interactions – and the linking of BPEL artifacts from the interaction protocol element are aspects that relate to process construction work in SOA4All. Processes in SOA4All are distinct from services, although they eventually could be exposed and annotated as simple Web services too. Still, comparing the interaction protocol related concepts to the work in SOA4All is out of the scope of this deliverable, as it exceeds the currently possible alignment in terms of standardization.

3.2.2 Non-Functional and Business Related Aspects

So far we concentrated on a comparison of the functional aspects of services which is the main focus of the SOA4All project. USDL on the other hand, emphasizes the business aspects of services too and concentrates mainly on so-called non-functional properties that

[32] FunctionalClassificationRoot is a class that is a root of a classification which also includes all the RDFS subclasses of the root class (the actual functional categories). A classification (taxonomy) of service functionalities can be used for functional description of a service.

in many cases explain how well a service fits the consumers' preferences. With its modules for pricing, participants, legal and service level aspects, USDL has a large specification share that falls in the area that is only marginally covered by SOA4All in context of selection and ranking. In this section, we thus only discuss the non-functional aspects of USDL that have some relationship to the SOA4All service models and that could profit from our approach and project results.

In USDL, a concept *Artifact* is introduced with the purpose of allowing for the inclusion of links to USDL-external service metadata, as well as arbitrary documents, files, or Web pages. In WSMO such information is provided via mostly Dublin Core-based annotations of services and service elements. A similar concept that is represented in WSMO as annotation is, for example, the USDL, *ContactProfile* that is used to collect physical and logical locations (virtual addresses) where persons or organizations can be contacted, e.g. postal address, telephone number, email address, etc. In USDL, further details in this respect are given through elements that are defined in the Participants Module. A Provider represents the entity that holds governance and operational responsibility for a service in terms of organizational structures and business aspects, as well as systems and other implementation artifacts. A *BusinessOwner*, on the other hand, represents an entity that shares some of the responsibilities of the provider regarding interaction between provider and consumer. Business owners can be understood as sales channels with custodial ownership of services.

Aspects such as policies, quality of service, pricing, or time and location-based service usage indications are non-functional descriptions, and define the incidental details specific to a service provider or to the service implementation or its running environment. In other words, non-functional properties describe how a service is delivered, while the functionality states primarily what is delivered and how it is accessed. For example, the functionality of a service is generally not affected by the price, even though the desirability may be. WSMO-Lite allows for arbitrary logical expressions to model non-functional properties and the associated rules. The RDF in **Error! Reference source not found.** depicts an example service annotation with an attached non-functional parameter that defines the cost of a service in relationship to the number of SMS messages ordered. USDL defines dedicated concepts such as time- or location-related elements and entire modules such as the Pricing Module or the Legal Module for this purpose.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix wsml: <http://www.wsmo.org/wsml/wsml-syntax#>.
@prefix sawsdl: <http://www.w3.org/ns/sawsdl#>.
@prefix wsl: <http://www.wsmo.org/ns/wsmo-lite#>.
@prefix posm: <http://www.wsmo.org/ns/posm/0.2#>.
@prefix sms: <http://www.example.com/SMS#>.

sms:a a posm:Service ;
        sawsdl:modelReference sms:UnitCost ;
        posm:hasOperation sms:sendBatchSMS ;
        rdfs:isDefinedBy <http://seekda.com/providers/cdyne.com/PhoneNotify> .

sms:UnitCost a wsl:NonFunctionalParameter ;
       rdf:value
            "sms#cost(?order, 10) :- ?x[posm#hasMessagePart hasValue ?number] memberOf
             posm#InputMessage and ?number < 1000.
             sms#cost (?order, 5) :- ?x[posm#hasMessagePart hasValue ?number] memberOf
             posm#InputMessage and ?number >= 1000 and ?number < 2000.
             sms#cost(?order, 2) :- ?x[posm#hasMessagePart hasValue ?number] memberOf
             posm#InputMessage and ?number >= 2000."^^wsml:AxiomLiteral.
```

Figure 3: SOA4All service description with pricing information

In USDL, this has the effect that concepts for time and location were partially re-modeled, and do not uniquely nominate existing standards or directly reuse their representation. In SOA4All, such location- and time-related elements are not explicitly part of the service description models but are assumed to be taken over from public and commonly agreed ontologies such as the Basic Geo Vocabulary for WGS84 coordinates,[33] geonames for geo-locations,[34] or vcard for addresses.[35] With respect to time, SOA4All, as discussed in Section 3.1, suggest to use the W3C Time Ontology, which is considered a de facto standard for modeling time instances, intervals, time zones and dates.[36] Many more such vocabularies exist that could be used to link USDL elements to exist ontologies and datasets. With respect to licensing that would be covered by the Legal Module, the recommendation of this deliverable would be to consider the Dublin Core Metadata initiative with predicates such as 'rights' and 'license',[37] or the ontology of creative commons that defines concepts and relations to describe licenses, permissions and prohibitions.[38]

[33] http://www.w3.org/2003/01/geo/wgs84_pos#

[34] http://www.geonames.org/ontology/

[35] http://www.w3.org/Submission/vcard-rdf/

[36] http://www.w3.org/TR/owl-time/

[37] http://dublincore.org/

[38] http://creativecommons.org/ns#

4. SOA Governance and the USDL

4.1 SOA Governance

SOA Governance extends IT Governance by assigning decision rights, policies and measures around the services, processes, for the purpose of ensuring the success of SOA. Specifically, it focuses on the lifecycle of services, metadata and composite applications in an organization's service oriented architecture. According to [9], "*SOA Governance is a set of solutions policies and practices which enable companies to implement and manage an enterprise SOA"*. The lack of governance SOA can be a serious impediment to success and the most common reason for the failure of SOA projects [10].

While the specific focus of SOA Governance is on the development and use of services, effective SOA Governance must cover the people, processes and technologies involved in the entire service lifecycle. Some key activities that are often mentioned as being part of SOA Governance are as follows.

- *Managing the portfolio of services*: planning the development of new services and updating current services.
- *Controlling and Managing the service lifecycle*: according to [11], the major aspect in SOA governance is the management of whole phases of the service lifecycle.
- *Using policies to restrict behavior*: clear guidelines for assigning the right policies and involving the right stakeholders to in the right steps.
- *Monitoring the performance of services*: cause of service composition, the consequences of service downtime or underperformance can be severe.

4.1.1 SOA Governance Model

To facilitate the SOA governance adoption, it would represent a significant advance to agree a governance model that covers all SOA related policies, metadata management, processes, control and management of service lifecycle, and governance mechanisms that are required to monitor the SOA. Among existing languages to describe services, we quote the standard SoaML, CBDI SAE (which is used in the Petals Master tool as described in WP5) and USDL.

4.1.2 Service oriented Architecture Modeling Language (SoaML)

SoaML has been created by the Object Management Group (OMG) in September of 2006 and was adopted as a specification in November of 2008. This standard aims to provide a way for users to model basic concepts SOA applications such as consumers' requirements, providers' offerings and the interaction and agreements between them.

SoaML specifies a UML Profile and Metamodel for modeling and design of services.

- The SoaML Profile supports the range of modeling requirements for SOA and includes the specification of systems of service, individual service interfaces and service implementations. It defines a set of stereotypes to allow using SoaML in UML tool, namely Agent, Attachement, Capability, CollaborationUse, MessageType, Milestone, Participant, ParticipantArchitecture, Port, Property, RequestPoint, ServicesArchitecture, ServiceChannel, ServiceContract, ServiceInterface, and ServicePoint. Figure 4 below depicts the services package of the SoaML Profile.

- The *SoaML Metamodel* is based on the UML2 metamodel and provides minimal extensions to UML, only where absolutely necessary to accomplish the goals and requirements of service modeling. It defines a few concepts that serve to support different service modeling scenarios such as single service description or SOA modeling.

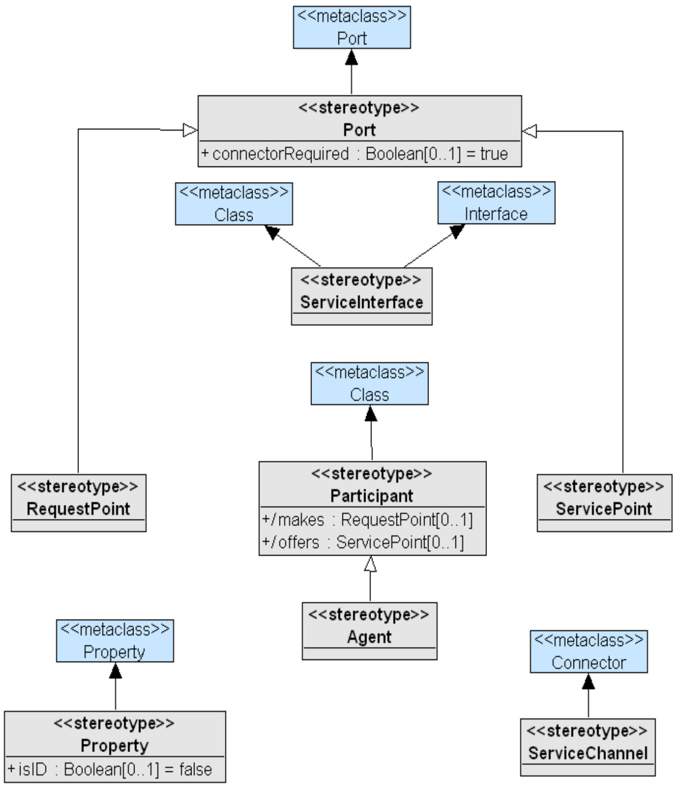

<<metaclass>> Port
<<stereotype>> Port
+connectorRequired : Boolean[0..1] = true
<<metaclass>> Class
<<metaclass>> Interface
<<stereotype>> ServiceInterface
<<metaclass>> Class
<<stereotype>> RequestPoint
<<stereotype>> Participant
+/makes : RequestPoint[0..1]
+/offers : ServicePoint[0..1]
<<stereotype>> ServicePoint
<<metaclass>> Property
<<stereotype>> Agent
<<metaclass>> Connector
<<stereotype>> Property
+isID : Boolean[0..1] = false
<<stereotype>> ServiceChannel

Figure 4: SoaML Profile: Services Package.

4.1.3 Comparison of USDL and SoaML

In studying the specification of SoaML,[39] one can notice that it differs from USDL in two ways.

The first one is that SoaML is much smaller than USDL. It includes references to its UML profile, around only twenty stereotypes and enumerations unlike USDL which contains more than one hundred. These difference is explained by the fact that SoaML is focused primary on the basic service modeling and design concepts, and reinforce it by using other OMG standards, such as Business Motivation Model (BMM), Business Process Definition Model (BPDM) and Business Process Modeling Notation (BPMN). As opposed to USDL which defines a way to describe services not only from a business view but also from an operational point of view and align this with the technical perspective. Moreover, SoaML covers less aspects of the overall service description and especially service lifecycle concepts than USDL.

Another reason for this difference is that USDL is made up of a set of modules *related* to each other to allow the reuse of concepts from all modules as opposed to SoaML where there is no real dependency between the different packages of its UML profile.

The second aspect that one can observe is that the approach used in SoaML for modeling

[39] http://www.omgwiki.org/SoaML/doku.php?id=specification

services is more technical than the one used in USDL. In SoaML, collaboration diagrams are used to model, for example, Service Contract or Services Architecture and interaction diagrams to describe the communication protocol between service interfaces for how properly use and implement a service. Conversely, USDL uses only class diagrams to model the different aspects of the global service description.

4.2 USDL for SOA Governance Tool

In order to offer to the SOA Governance tool such facilities as controlling SOA resources, managing service lifecycle, a new service model is designed. It provides, as shown in Figure 5, an administrative API enabling to synchronize with a service infrastructure (such as Petals ESB) that allows managing services description as WSDL, events as Topic Namespaces[40] and service level agreements as WS-Agreement.[41]

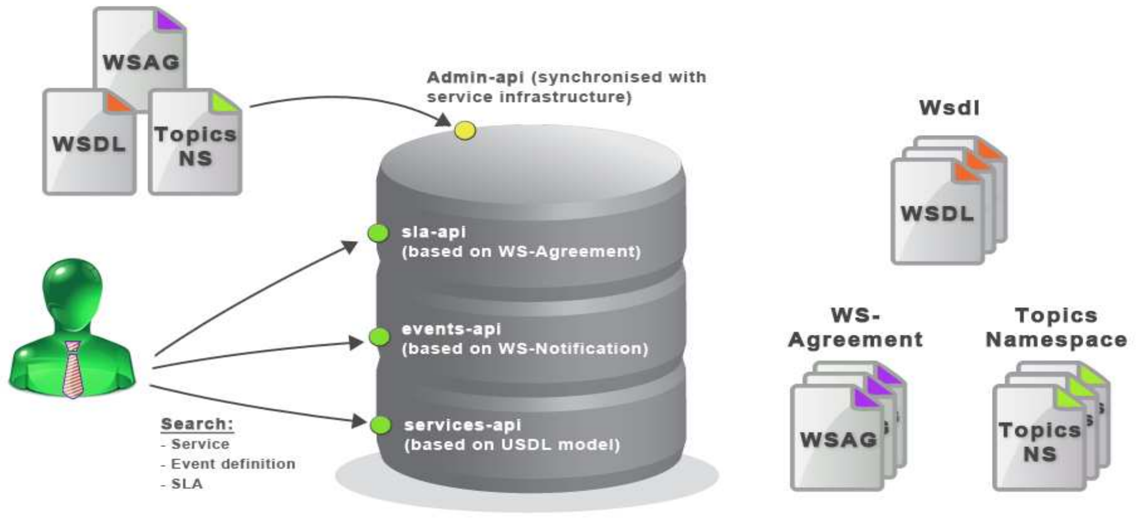
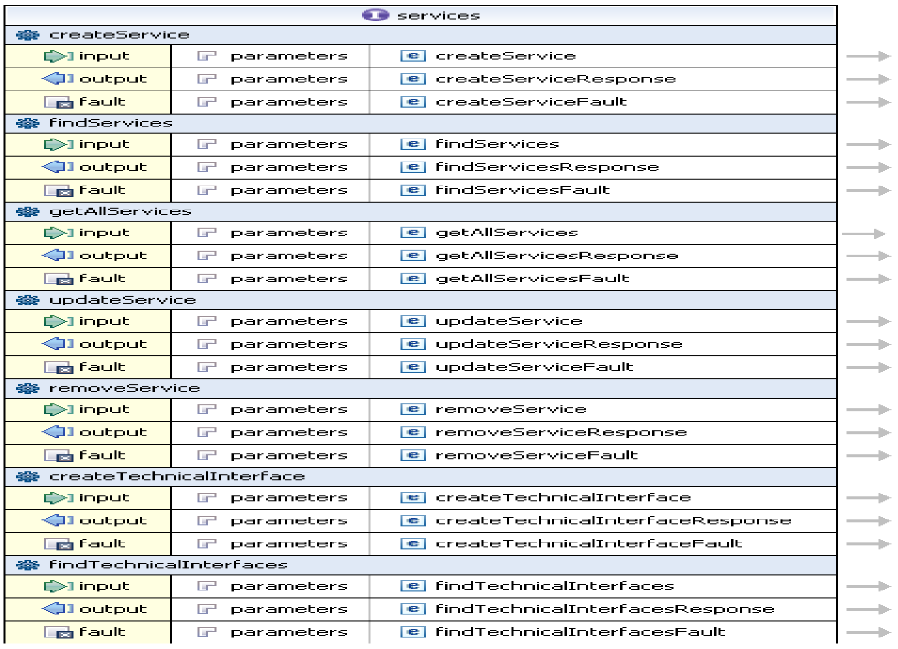

WSAG
WSDL
Topics NS
Admin-api (synchronised with service infrastructure)
sla-api (based on WS-Agreement)
events-api (based on WS-Notification)
services-api (based on USDL model)
Search:
- Service
- Event definition
- SLA
Wsdl
WSDL
WS-Agreement
WSAG
Topics Namespace
Topics NS

Figure 5: Governance tool.

For the previously presented reasons, USDL appears to be an interesting service model for the SOA Governance tool. In order to be able to manipulate knowledge about services, a web service API is defined to present CRUD (create-read-update-delete) operations as shown in Figure 6.

[40] http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

[41] http://www.ogf.org/documents/GFD.107.pdf



services		
createService		
input	parameters	createService
output	parameters	createServiceResponse
fault	parameters	createServiceFault
findServices		
input	parameters	findServices
output	parameters	findServicesResponse
fault	parameters	findServicesFault
getAllServices		
input	parameters	getAllServices
output	parameters	getAllServicesResponse
fault	parameters	getAllServicesFault
updateService		
input	parameters	updateService
output	parameters	updateServiceResponse
fault	parameters	updateServiceFault
removeService		
input	parameters	removeService
output	parameters	removeServiceResponse
fault	parameters	removeServiceFault
createTechnicalInterface		
input	parameters	createTechnicalInterface
output	parameters	createTechnicalInterfaceResponse
fault	parameters	createTechnicalInterfaceFault
findTechnicalInterfaces		
input	parameters	findTechnicalInterfaces
output	parameters	findTechnicalInterfacesResponse
fault	parameters	findTechnicalInterfacesFault

Figure 6: Services.wsdl.

The Services API uses an XSD schema that maps the USDL language according to its packages, its components and their dependencies. Table 3 describes an excerpt of the XSD schema that defines the important elements of the USDL Functional Module (cf. Section 2).

Table 3: Services-API XSD Schema.

Schema: http://www.petalslink.org/services-model/1.0	
Elements	Types
Action: ActionType	ActionConditionType: tns:BaseIDType
ActionCondition: ActionConditionType	ActionFaultType: tns:BaseIDType
ActionFault: ActionFaultType	ActionParameterType: tns:BaseIDType
ActionParameter: ActionParameterType	ActionType: tns:BaseIDType
Artifact: ArtifactType	ArtifactType: tns:BaseIDType
ExpressionElement: ExpressionElementType	BaseIDType
NetworkProvisionEntity: NetworkProvisionEntityType	ExpressionElementType: tns:BaseIDType
Resource: ResourceType	NetworkProvisionEntityType: tns:BaseIDType
Service: ServiceType	ResourceType: tns:BaseIDType
TechnicalCondition: TechnicalConditionType	ServiceType: tns:NetworkProvisionEntityType
TechnicalFault: TechnicalFaultType	TechnicalConditionType: tns:BaseIDType
TechnicalInterface: TechnicalInterfaceType	TechnicalFaultType : tns:BaseIDType
TechnicalOperation: TechnicalOperationType	TechnicalInterfaceType: tns:BaseIDType
TechnicalParameter: TechnicalParameterType	TechnicalOperationType: tns:BaseIDType
TechnicalProfile: TechnicalProfileType	TechnicalParameterType: tns:BaseIDType
TechnicalProtocol: TechnicalProtocolType	TechnicalProfileType: tns:BaseIDType
	TechnicalProtocolType: tns:BaseIDType
	ProtocolTypeType :string

As presented in Figure 7, the inheritance relationship, as defined in USDL, is respected, for example a *service* is derived from *NetworkNetworkEntity*. Moreover, the composition relationships and their multiplicities are respected by referencing the composed object (such as *TechnicalInterface*) in the composite (*Service*). Then, in order to respect the dependency links between elements (between *Service* and *TechnicalInterface as an example*), we have added the identifier of the service: "*idService*" as an element in *TechnicalInterface* Object.

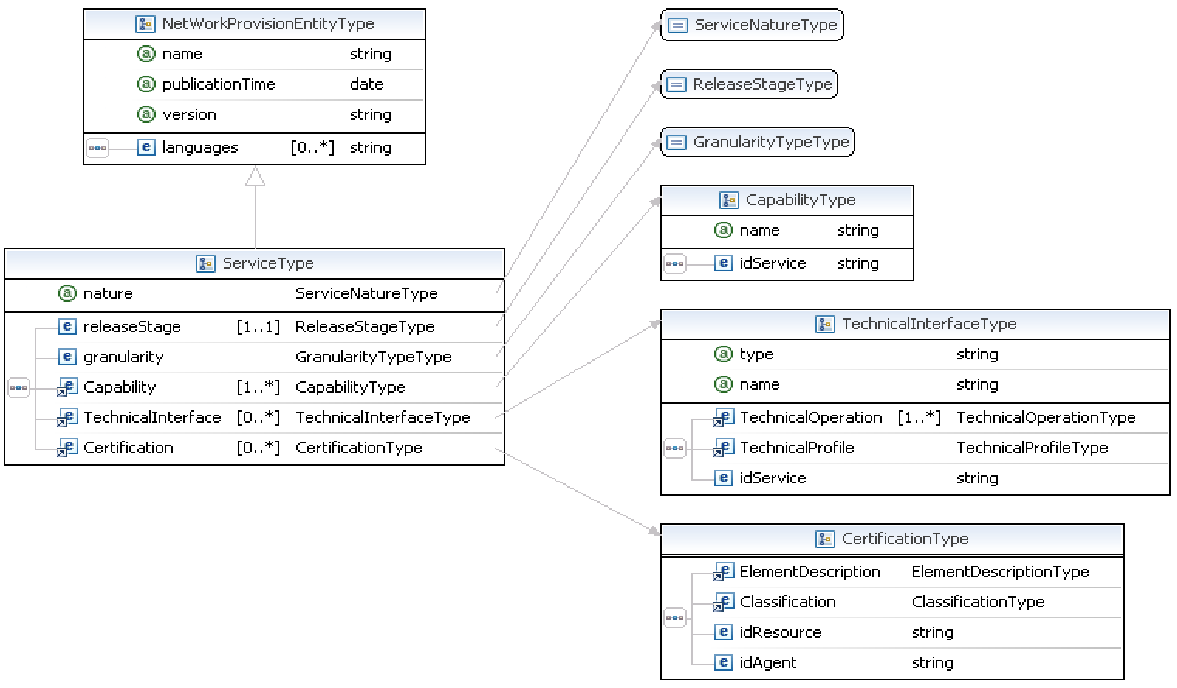

NetWorkProvisionEntityType
name string
publicationTime date
version string
languages [0..*] string
ServiceType
nature ServiceNatureType
releaseStage [1..1] ReleaseStageType
granularity GranularityTypeType
Capability [1..*] CapabilityType
TechnicalInterface [0..*] TechnicalInterfaceType
Certification [0..*] CertificationType
ServiceNatureType
ReleaseStageType
GranularityTypeType
CapabilityType
name string
idService string
TechnicalInterfaceType
type string
name string
TechnicalOperation [1..*] TechnicalOperationType
TechnicalProfile TechnicalProfileType
idService string
CertificationType
ElementDescription ElementDescriptionType
Classification ClassificationType
idResource string
idAgent string

Figure 7: XSD Schema: Service Element.

5. USDL Standardization

SAP and its partners follow the vision of a large-scale Internet of services that allows provisioning, trading, and consumption of services, and that is open for everyone who wants to participate. In this picture, the USDL is the base data model for the knowledge exchange during the different interactions among participants of the Internet of services. As such, there is a strong need for the USDL to be a widely used and accepted standard. For this purpose, it was decided to standardize the USDL via the World Wide Web Consortium (W3C)[42].

The standardization process at the W3C is described in D11.2.1. In short, W3C standards, which are called recommendations, are developed by Working Groups of W3C Members in several iterations following the formal standardization process. In order to be successful, a working group needs to have strong members who are nonetheless able to reach a consensus on the proposed specification. Furthermore, the specification needs to be mature, and compatible reference implementations need to be provided. One possibility to prepare this heavyweight standardization process, are the so-called Incubator Groups offered by the W3C. The W3C states that „Incubator Groups foster rapid development, on a time scale of a year or less, of new Web-related concepts. Target concepts include innovative ideas for specifications, guidelines, and applications that are not (or not yet) clear candidates for development and more thorough scrutiny under the current W3C Recommendation Track" (http://www.w3.org/Consortium/activities#about) of the W3C Working Groups.

Therefore, in September 2010 the USDL Incubator Group (XG) was initiated by Attensity[43], DFKI[44], SAP[45], and Siemens[46], later joined by Open University. Its mission statement is given on http://www.w3.org/2005/Incubator/usdl/charter as follows: "The mission of the Unified Service Description Language (USDL) Incubator Group, part of the Incubator Activity, is to define a language for describing general and generic parts of technical and business services to allow services to become tradable and consumable. Technical services are considered electronic services based on WSDL, REST or other technical specifications. Business services are defined as business activities that are provided by a service provider to a service consumer to create value for the consumer. Thus, the business services are more general and comprise manual and technical services. This enables new business models in the field of service brokerage because services can automatically be offered, delivered, executed, and composed from services of different providers". The complete charter can be found in appendix A.

The USDL incubator group has the following three goals:

1. investigate related standards and approaches with respect to the compatibility and potential relation to the USDL,
2. define a formal specification of the USDL, reusing existing standards where appropriate,
3. define and implement reference test cases to validate the USDL.

Each of these goals is pursued in a separate work package, and a project plan with deliverables and milestones that can be found at the wiki of the Incubator Group[47] and is also

[42] http://www.w3.org/

[43] http://www.attensity.com/home/

[44] http://www.dfki.de/web

[45] http://www.sap.com

[46] http://www.siemens.com

[47] http://www.w3.org/2005/Incubator/usdl/wiki/

given in Table 4. As the first result of WP1, the document "D1 Report on landscapes of existing service description efforts" has been published at http://www.w3.org/2005/Incubator/usdl/wiki/D1. This report analyzes different service description approaches coming from SOA (WS*, SoaML etc), semantic Web services (OWL-S, WSMO, WSMO-Lite etc.), software as a service (SML), service networks (e3Service, SNN), service systems (OASIS Reference Architecture), and economics (DIN PAS 1018).

WP	Tasks	Deliverables
1	1. Relation to Semantic Web Services and Linked Open Data 2. Relation to UDDI 3. Relation to W3C standards 4. Relation to other Service Description Languages	D1: „Report on landscapes of existing service technologies"
2	1. Requirements on the standard 2. Target Standardization Body 3. Rework USDL and consider existing standards where applicable (WEBify)	D2.1: "Requirements on a standard specification" D2.2: "Formal specification of the language (USDL XG Version)"
3	1. Specification of reference test cases 2. Implementation of reference test cases	D3.1: "Specification of reference test cases" D3.2: "Implementation of reference test cases"
4	1. Project management 2. Communication	D4: „Report on communication and administration activities"

Table 4: USDL Incubator Group Work Plan

In February, the latest version 3.0 M5 of the USDL specification has been released[48] on http://internet-of-services.com/index.php?id=382.

Based on the results described in Chapter 3, the incubator group has also started a discussion on how USDL based service descriptions could be serialized and published following the principle of and in the form of linked data.

The final outcome of the USDL incubator group will be a report with an updated USDL specification and an assessment of how to proceed with USDL in September 2011.

[48] http://www.w3.org/2005/Incubator/usdl/wiki/2011-02-10_USDL_3.0_Specifications_M5_released

6. Conclusions

This deliverable presented and discussed different aspects concerning USDL and the potential of leveraging the USDL as a standard. For this purpose, we discussed the USDL from two angles:

- as a holistic service language for business services in the service marketplace on the Web; and
- in correlation with the SOA4All service concepts and service models.

A finding of the deliverable is that SOA4All can indeed harness USDL as a service description language to augment its potential to cover, among others, non-functional, business-oriented aspects of services and provide adequate description of them. Vice versa, the USDL could profit from SOA4All results mostly by a closer alignment of the USDL with existing Web and Linked Data standards:

- In regards to automated services; i.e., services with a software implementation, the USDL concepts of technical interface and operation and their attributes can be directly reflected through WSMO-Lite and MSM/POSM concepts and properties.
- Expression elements in USDL can be realized via WSML logical expressions or rather RIF rules which are standardized by W3C.
- Functional classification ontologies from SOA4All could be directly used for the classification of services in USDL, and USDL operations could be easily extended to allow for classifications too.

The deliverable moreover shortly presented the USDL in the light of SOA governance and the potentials in relating SOA governance concepts with the USDL service model. To further drive the efforts of standardization of USDL at W3C, a W3C Incubator Group that has been formed and joined by some SOA4All partners.

7. References

[1] S. de Kinderen and J. Gordijn: Reasoning About Substitute Choices And Preference Ordering in e-Services. Proc. 20th Int'l Conference on Advanced Information Systems Engineering, 2008.

[2] Meta Object Facility (MOF) 2.0 Core Specification. OMG Available Specification. http://spemarti.googlecode.com/files/MOF2.0.pdf

[3] Unified Service Description Language 3.0 Overview Alistair Barros, Anis Charfi, Daniel Oberle, et al. http://internet-of-services.de/fileadmin/IOS/user_upload/pdf/USDL-3.0-M4-module-service.pdf

[4] J. Allen. Maintaining knowledge about temporal intervals. Communications of the ACM (1983) vol. 26 (11) pp. 832-843

[5] H. L. Kim et al. Review and Alignment of Tag Ontologies for Semantically-Linked Data in Collaborative Tagging Spaces. Semantic Computing, 2008 IEEE International Conference on (2008) pp. 315 – 322

[6] C. Pedrinaci and J. Domingue. Ontology-based metrics computation for business process analysis. In: 4th International Workshop on Semantic Business Process Management (SBPM 2009), Workshop at ESWC 2009, 1 June 2009, Crete, Greece. (2009)

[7] T. R. Gruber and G. R. Olsen. An Ontology for Engineering Mathematics. In J. Doyle, P. Torasso, and E. Sandewall, editors, Fourth International Conference on Principles of Knowledge Representation and Reasoning, pages 258–269, Bonn, Germany, 1994. Morgan Kaufmann.

[8] H. M. Kim and M. S. Fox. Using enterprise reference models for automated ISO 9000 compliance evaluation. In Proceedings of the 35th Hawaii International Conference on Systems Science, 2002

[9] B. Brauer, S. Kline: "SOA Governance: A Key Ingredient of the Adaptive Enterprise". 2005. http://www.managementsoftware.hp.com/product/soa/swp/soa swp governance.pdf

[10] M. Afshar: "SOA Governance: Framework and Best Practices". 2007. http://www.oracle.com/technologies/soa/docs/oracle-soa-governance-best-practicies.pdf

[11] M. Niemann, J. Eckert, N. Repp, and R. Steinmetz: Towards a Generic Governance Model for Service-oriented Architectures. In Proceedings of the Fourteenth Americas Conference on Information Systems (AMCIS 2008), Toronto, ON, Canada, 2008.

[12] Di Nitto, Elisabetta and Karastoyanova, Dimka and Metzger, Andreas and Parkin, Michael and Pistore, Marco and Pohl, Klaus and Silvestri, Fabrizio and den Heuvel, Willem Jan. S-Cube: Addressing Multidisciplinary Research Challenges for the Internet of Services. Towards the Future Internet: A European Research Perspective. Amsterdam : IOS Press, 2009.

[13] Jorge Cardoso, Konrad Voigt, and Matthias Winkler. Service Engineering for the Internet of Services. Enterprise Information Systems, Lecture Notes in Business Information Processing (LNBIP). 2008.

[14] O'Sullivan, Justin. Towards a Precise Understanding of Service Properties. Queensland University of Technology. 2006.

Appendix A: Charter of W3C USDL Incubator Group[i]

The mission of the Unified Service Description Language (USDL) Incubator Group, part of the Incubator Activity, is to define a language for describing general and generic parts of technical and business services to allow services to become tradable and consumable. Technical services are considered electronic services based on WSDL, REST or other technical specifications. Business services are defined as business activities that are provided by a service provider to a service consumer to create value for the consumer. Thus, the business services are more general and comprise manual and technical services. This enables new business models in the field of service brokerage because services can automatically be offered, delivered, executed, and composed from services of different providers.

The language is usable for any purpose and implementation scenario of future business services on a general level. However, it will be extendable for industry-specific aspects. The specification aims at complementing the technical language stack by adding required business and operational information. The targeted groups for USDL are service providers, hosting providers, gateways, and service consumption channels. Industry-specific and general-purpose attributes of a service will be derived based on these use cases and their target groups. In the end, the initiating and contributing members will define a language that enables its users to model arbitrary services and to integrate with existing standards. The Unified Service Description Language Incubator Group will derive best practices and learning from testing cycles that will then be deployed in a number of use cases. These use cases serve as references and proof-of-concept of USDL.

End date	18 September 2011
Confidentiality	Proceedings are public
Initial Chairs	Daniel Oberle (until 23 Nov. 2010), Kay Kadner (starting 23 Nov. 2010)
Initiating Members	Attensity Europe GmbH (formerly named Empolis GmbH) German Research Center for Artificial Intelligence (DFKI GmbH) SAP AG Siemens AG
Usual Meeting Schedule	Teleconferences: Every two weeks Face-to-face: 2-3 per year

Scope

Motivation

Any organization that manufactures, promotes, sells and applies services in its business environment is urged to use the internet for its purposes. Hence, the internet could serve as an accelerator and stimulator to trade service regardless of the geographical and industrial surroundings of an organization. The Internet of Services creates easy access to services for automation, trading and consumption of services between industries, businesses and consumers. For this purpose, complex integration between applications on different levels (e.g., industries, agencies, portals) and the choreography of different business processes will be enabled.

What is missing to date?

Past work in various projects and a gap analysis delivered the following results. A service description language is missing, which emphasizes the business-related aspects when defining a service. Service description languages such as WSDL concentrate on the proposition of technical interfaces to get services exchanged. Stakeholders from small, medium and large sized businesses as well as representatives from various industries expressed the need of an open and accessible service description language that is able to describe their business needs based on the following aspects (modules)

- Core (general information about the service)
- Interaction (how the service can be consumed)
- Participant (roles of businesses and / individuals that become involved in the provision and consumption of a service)
- Functional (what business functions are provided by the service)
- Pricing (how much does the service cost)
- Legal (legal terms and conditions under which the service may be consumed)
- Service Level Agreements (levels of service provided, e.g. availability, response time, etc.)

To date, service description languages lack the inclusion of business and operational attributes to describe a service in its whole because they mostly target technical services instead of services as such. For instance, the description of a Web Service that provides weather data includes the technical interface but no information about how much the user has to pay for calling the service. Moreover, the use of services must not be limited to a specific application, interaction protocol, service type or deployment in certain technical infrastructures. Existing offers of pre-defined service descriptions are often bound to a dedicated service or application provider as for example Salesforce.com. Further initiatives such as APS Standard focus on specific application areas of services, software-as-a-service in this case.

Moreover, the study of business needs revealed the need to enable any stakeholder to describe, publish and consume the described service. There is a demand of an easy-to-use and consume service regardless of the position of the service provider or consumer in the service supply chain. The service needs to be configured, i.e., the service is allowed to mature, for instance from a regional to an international use. Other forms of allowing a service to mature relate to:

- The publication of services on distinct service marketplaces
- The discovery of services with similar attributes through search engines
- A direct consumption of a service via hosting functions: it will be taken into account within the specification analysis, if certain services require a download support. The user will get in direct interaction with the service provider or a service broker.
- Business processing contains a number of services that are attached to a business purpose as for example IT installation services covered in an ERP implementation project, repair services in a production environment or health care at-home services in combination with medical treatments. Thus, a service can be consumed via the Internet and be connected with further activities of a business process. Therefore, it should be executed via mash-up tools or process engines.
- Very often, the service is being bundled with other services (composite services) and being sold to the consumer as a service package

The description of business aspects of a service is closely related to the technical interface by which a service is made available. Since the description is based on well-known modeling languages like Eclipse Modeling Framework (EMF) or XSD, the description can be the outcome of a transformation from a higher-level language (like the Service oriented architecture Modeling Language (SoaML)). Additionally, USDL can be the basis for generating other technical artifacts like WSDL documents, price rules for business rule engines and similar artifacts. Our observations concluded that such a model driven approach would serve best the various stakeholders involved in the provision and consumption of services. However, the technical interface might also be directly included in USDL by some reference mechanism so that the transformation is obsolete. EMF and XSD are supported by various freely available tools like Eclipse and associated projects, which allows the meta models to be extended according to specific requirements. By using EMF for creating the USDL meta model, according editors can be created without much effort.

USDL in a nutshell

The Unified Service Description Language now aims at aligning business services by unifying them using a common description format. USDL is one of the final artifacts of the service engineering process and - as such - complements existing documents and standards for describing the service interface (e.g., WSDL). Herein, USDL composes activities and functions of these stakeholders together based on commonly shared or traded services. This is done regardless of organization's size, industry, or position in the service lifecycle. The USDL services can easily be created by extending concrete representations from pre-defined abstract services. Furthermore, USDL can also be made available with a minimal set of elements to allow for simple and fast description of new services. The benefits of USDL result from a context-independent design. USDL provides a frame that is then being filled by the context. The context results then from the business or user-driven purpose of providing and consuming services. Moreover, the proposition of USDL is driven by the overall need that services require openness to be spread and composability among applications, infrastructures and consumption channels. The Incubator Group will incorporate a staged approach that allows a versioned USDL development and delivery. Even more, the staged approach will deliver in a foreseeable timeframe the USDL artifacts. The USDL standard will be ready to use with least possible barriers. The stakeholders that are involved in the design, build and consumption of a USDL-based service are able to consider a versioning of the service and offer distinct versions for different regions for example. The versioning of USDL and the further above-described design approach will be incorporated in the work plan of the Incubator Group.

The work plan foresees the phases of assessing input (beyond the existing inputs from the described projects involved), designing, building and delivering. Any of these phases contains documentation, trial and feedback as well as quality assurance activities.

The Unified Service Description Language Incubator Group is proposed by members of the THESEUS research program, which provides experience in foundations and practice of the Internet of Services. The XG builds on substantial scientific work which has already been carried out within the THESEUS-program and European and Australian research projects such as SOA4ALL, Premium Services, Smart Services CRC, ACTOR, FAST, ITAIDE, MASTER, MODELPLEX, MORE, PICTURE, RESERVOIR, Secure SCM, ServFace, SHAPE, SLA@SOI, SoKNOS, VIRTEX, XtreemOS. In particular, the research initiatives already created iterations of the Unified Service Description Language in THESEUS / TEXO and SOA4ALL and further to-be-announced project contributions. Hence, the XG can be expected to produce tangible results within one year.

To sum up, the XG will concentrate on the specification and further development of USDL as an open standard. The XG members will target a broad dissemination of USDL, an ease of use by multiple stakeholders involved in service provision and consumption. Beyond the core focus of the specification work, the XG members will draft recommendations for possible tools and editorial support. In addition, the XG members will outline mapping scenarios to represent USDL in UML and how the meta model in UML will look like.

Success Criteria

The fulfillment of the targeted mission of the Incubator Group can be measured against the following two success criteria:

- Firstly, the Unified Service Description Language Incubator Group will report on the landscapes of existing service technologies and examine their compatibility and inter-relationships with respect to the USDL specification. The report will be based on a gap analysis that is being conducted beforehand in our research activities within THESEUS and further above-mentioned projects.
- Secondly, the Incubator Group efforts will target the development of a formal draft specification of the language. The specification covers the above-described modules. USDL and its modules will make use of existing standards for describing specific types of information wherever possible. A consequence to that is that USDL will be the glue between specifications and languages that bind them together and provide benefit through their composition. The concept of separating different aspects in different modules makes it easy to create new modules and link them to USDL in order to create extensions for further use cases or other industries. It is possible to create transformations, which provide mappings from existing description languages to and from USDL. First prototypes exist for the transformation from SoaML to USDL.

In addition, after having completed and delivered the two above-referenced success criteria, it is planned to conduct a proof-of-concept phase. This activity will be conducted based on available resources and the remaining timeframe. The proof-of-concept phase could encompass an implementation of four pre-selected reference cases that test the USDL specification from distinct viewpoints: service consumer, service provider concerning the provision of service marketplaces and service provider concerning the service engineering, as well as the service hosting.

Out of Scope

USDL is not meant to replace existing standards, e.g. for WebService-Descriptions (WS-*). Moreover, USDL will complement these with business related aspects and further allow integrating existing specifications for specific parts of the description. USDL does also not claim to be complete with respect to the required attributes and elements for describing each and every service. There will be the possibility and necessity to extend USDL by use case or industry-specific requirements (like insurance, banking). Furthermore, USDL is not intended to semantically describe existing Web Services or other services.

Deliverables

A report describing the work performed in the group. The report will provide:

- A state-of-the-art document that assesses the existing landscape of service description languages. It covers the identification of existing standards that feed technical aspects into the Unified Service Description Language; it delineates needed, existing and required parameters and architectural concepts of service description language to allow a generic use of a service in multiple technical and business environments. The document will point to the benefits of existing standards and the benefit of inclusion of the Unified Service Description Language.
- A formal draft specification for a Unified Service Description Language;

In case the existing timeline and resources are made available to conduct a proof-of-concept phase, the report will be extended by four reference cases described earlier.

Through consultation with potential users, technology and business partners as well as experts on service science and service engineering, the XG will investigate whether a sufficient degree of normalization can be achieved at this stage, and whether the resulting specification can add value to other markup languages, e.g. in the form of a specialized plug-in language. Accordingly, the group will propose to either terminate its activity after its lifetime or continue in the more formal Recommendation Track. The formal decision is left to the relevant W3C entities.

Dependencies and Liaisons

W3C Groups

Contexts, which may benefit from the Unified Service Description Language, can already be found in current W3C activities, such as the W3C Recommendations WSDL , SAWSDL and SML.

External Groups

The work of the group will also be coordinated with the activities of the Internet-of-Services Community. The activities of the OASIS TGs operating on so called WS-* have to be taken into account.

The work of the group will assess and use the integration potential of the Unified Service Description Language into existing service engineering concept and equivalents wherever possible. The deliverables will point out the correlations where feasible and necessary.

Participation

Members should be expected to introduce themselves and participate over the public list-serv. Members should attend teleconferences, and send regrets if unable to. While participation from all is welcome on the public list-serv, editors of XG deliverables will be W3C members or invited experts. The face-to-face meeting will be strongly recommended.

Communication

This group primarily conducts its work on the public mailing list public-xg-usdl@w3.org (archive) .The group's Member-only list is member-xg-usdl@w3.org (archive)

Information about the group (deliverables, participants, face-to-face meetings, teleconferences, etc.) is available from the Unified Service Description Language Incubator Group home page.

Decision Policy

As explained in the Process Document (section 3.3), this group will seek to make decisions when there is consensus. When the Chair puts a question and observes dissent, after due consideration of different opinions, the Chair should record a decision (possibly after a formal vote) and any objections, and move on.

When deciding a substantive technical issue, the Chair may put a question before the group. The Chair must only do so during a group meeting, and at least two-thirds of participants in Good Standing must be in attendance. When the Chair conducts a formal vote to reach a decision on a substantive technical issue, eligible voters may vote on a proposal one of three ways: for a proposal, against a proposal, or abstain. For the proposal to pass there must be more votes for the proposal than against. In case of a tie, the Chair will decide the outcome of the proposal.

This charter is written in accordance with Section 3.4, Votes of the W3C Process Document and includes no voting procedures beyond what the Process Document requires.

Patent Policy

This Incubator Group provides an opportunity to share perspectives on the topic addressed by this charter. W3C reminds Incubator Group participants of their obligation to comply with patent disclosure obligations as set out in Section 6 of the W3C Patent Policy. While the Incubator Group does not produce Recommendation-track documents, when Incubator Group participants review Recommendation-track specifications from Working Groups, the patent disclosure obligations do apply.

Incubator Groups have as a goal to produce work that can be implemented on a Royalty Free basis, as defined in the W3C Patent Policy.

For more information about disclosure obligations for this group, please see the W3C Patent Policy Implementation.

About this Charter

This charter for the Unified Service Description Language Incubator Group has been created according to the Incubator Group Procedures documentation. In the event of a conflict between this document or the provisions of any charter and the W3C Process, the W3C Process shall take precedence.

[i] http://www.w3.org/2005/Incubator/usdl/charter