Unified Service Description Language 3.0 (USDL) Overview

March 17, 2011

Abstract. Services are increasingly being exposed and delivered outside company firewalls and on the Internet through the rise of on-demand applications, cloud computing and business networks. As the forms of services being exposed become more sophisticated, the conventional approaches for describing and accessing them, largely through technical interface descriptions, pose limitations. These approaches are concerned with technical access, leaving open the way consumers understand details of business operations, pricing, legal and other implications of using services. In addition to consumers requesting services, there are emerging third-party intermediaries who can deliver and further monetize services in the form of brokers, channel partners and cloud providers. This raises questions about how such details can be understood without full reliance on service providers. The Unified Service Description Language (USDL) is proposed to describe various types of services ranging from professional to electronic services. It aims at a holistic service description putting 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 among others, in addition to technical aspects. It proposes a consolidated foundation for service-based systems enabling different roles to participate in diverse aspects of provisioning in service networks.

Table of Contents

Acknowledgements

The information in this document provides an overview of a publicly released specification of the Unified Service Description Language (USDL) for dissemination and further exploitation through the Internet of Services Community in accordance with the terms set forth below. This document does not represent any prior commitment for standardization or implementation of any portion of this specification by SAP.

The information in this document was developed through the following publicly co-funded research projects THESEUS/TEXO, Australian Smart Services CRC, Premium Services, SLA@SOI.

THESEUS/TEXO is research project funded by the German Federal Ministry for Economics and Technology.

Premium Services is research project funded by the German Federal Ministry for Education and Research.

Australian Smart Services CRC is research and development partnership funded by the private sector and governments under the Australian Government's Cooperative Research Centre program.

SLA@SOI is a research project funded by the European Commission under the 7th Framework Programme.

The contributing authors are: Alistair Barros (SAP), Uwe Kylau (SAP), Daniel Oberle (SAP)

Terms of Use Agreement

IMPORTANT- PLEASE READ CAREFULLY: THIS TERMS OF USE AGREEMENT ("AGREEMENT") IS A BINDING AGREEMENT BETWEEN SAP AG, A GERMAN COMPANY WITH OFFICES AT DIETMAR-HOPP-ALLEE 16, 69190 WALLDORF, GERMANY ("SAP"), AND YOU, BEING EITHER AN INDIVIDUAL OR SINGLE LEGAL ENTITY ("YOU" OR "YOUR") REGARDING YOUR USE OF THIS DOCUMENT AND ANY INFORMATION CONTAINED THEREIN ("MATERIAL"):

BY DOWNLOADING, COPYING, OR OTHERWISE USING THE MATERIAL, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT DOWNLOAD, COPY, OR USE THE MATERIAL.

IF YOU ARE DOWNLOADING, COPYING, OR USING THE MATERIAL ON BEHALF OF YOUR EMPLOYER OR A S A CONSULTANT OR AGENT OF A THIRD PARTY (COLLECTIVELY "YOUR COMPANY"), YOU REPRESENT AND WARRANT THAT YOU HAVE THE AUTHORITY TO ACT ON BEHALF OF AND BIND YOUR COMPANY TO THE TERMS OF THIS AGREEMENT AND ANY REFERENCE TO YOU OR YOUR SHALL ALSO INCLUDE YOUR COMPANY.

1. "SAP ENTITY/ENTITIES" shall mean SAP's affiliates and its subsidiaries, defined as corporations or other entities of which SAP owns, either directly or indirectly, more than fifty percent (50%) of the stock or other equity interests.

   "SAP SOFTWARE" shall mean the software products of SAP and/or SAP ENTITIES marketed and licensed by SAP and/or SAP ENTITIES.

   "INTELLECTUAL PROPERTY RIGHTS" means patents of any type, design rights, utility models or other similar invention rights, copyrights, trademarks, service marks, trade secret or confidentiality rights, and any other intangible property rights including applications for any of the foregoing, in any country, arising under statutory or common law or by contract and whether or not perfected, now existing or hereafter filed, issued, or acquired.

2. SAP grants YOU a nonexclusive, royalty-free, fully paid up, worldwide right to use, copy, display, perform, transmit, translate and distribute the MATERIAL provided hereunder. This shall include the right to reproduce, adapt, modify and to create derivative works of the MATERIAL and to make, have made, offer to sell, sell, lease, or otherwise distribute any product, and to practice any method, embodying such MATERIAL (including the right to sublicense any of the foregoing rights).

   SAP reserves the right to modify, change or discontinue the MATERIAL without notice at any time.

   YOU must not remove, overprint or deface any notice of copyright, trademark, logo, legend, or other notice of ownership from any originals or copies of the MATERIAL accessed hereunder. YOU agree to comply with the terms of the SAP Copyright Policy and those terms found by clicking on Copyright/Trademark at the web page http://www.internet-of-services.com.

   FOR AVOIDANCE OF DOUBT, NOTHING IN THIS AGREEMENT SHALL BE DEEMED TO

   (I) TO ASSUME OR PROVIDE FOR THE TRANSFER OF OWNERSHIP OF ANY INTELLECTUAL PROPERTY RIGHTS. ALL INTELLECTUAL PROPERTY RIGHTS INCLUDED, WITHOUT LIMITATION, COPYRIGHT IN ANY MATERIAL PROVIDED HEREUNDER, SHALL VEST IN AND AT ALL TIMES REMAIN VESTED IN THE ORIGINATOR OF THAT INTELLECTUAL PROPERTY RIGHT.

   (II) GIVE YOU THE RIGHT TO MODIFY, COPY, DISTRIBUTE, TRANSMIT, DISPLAY, PERFORM, REPRODUCE, PUBLISH, LICENSE, CREATE DERIVATIVE WORKS FROM,

TRANSFER; OR SELL ANY SAP SOFTWARE OR OTHER PRODUCT FOR ANY REASON UNLESS OTHERWISE PERMITTED BY SAP.

3. Any MATERIAL made available hereunder is provided to YOU "as is". SAP does not guarantee or warrant any features or qualities of the MATERIAL or give any undertaking with regard to any other quality. No warranty or undertaking shall be implied by YOU from any published MATERIAL except to the extent SAP has expressly confirmed such warranty or undertaking in writing. Warranties are validly given only with the express written confirmation of SAPs management.

   SAP does not represent or endorse the accuracy or reliability of any MATERIAL provided hereunder. SAP shall not be liable for damages caused by the use of the MATERIAL, unless such damages have been caused by SAPs willful misconduct.

   TO THE EXTENT ALLOWABLE BY APPLICABLE LAW, SAP AND ITS AFFILIATES, SUBSIDIARIES, OFFICERS, EMPLOYEES, AGENTS, PARTNERS, AND LICENSORS ARE NOT LIABLE TO YOU FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE, CONSEQUENTIAL, OR EXEMPLARY DAMAGES, INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS, REVENUE, GOODWILL, USE, DATA, OR OTHER INTANGIBLE LOSSES (EVEN IF SAP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES), HOWEVER CAUSED, WHETHER IN CONTRACT, TORT, OR OTHERWISE, ARISING OUT OF OR RESULTING FROM: (i) THE USE OR THE INABILITY TO USE THE MATERIAL; (ii) THE COST OF PROCUREMENT OF SUBSTITUTE GOODS AND SERVICES ARISING OUT OF YOUR USE OR INABILITY TO USE THE MATERIAL; OR (iii) ANY OTHER MATTER RELATING TO THE MATERIAL PROVIDED HEREUNDER. NOTWITHSTANDING ANYTHING TO THE CONTRARY HEREIN, THESE LIMITATIONS SHALL NOT APPLY IN CASE OF INTENT BY SAP AND IN CASE OF SAPS STATUTORY LIABILITY FOR PERSONAL INJURY AND DEFECTIVE PRODUCTS.

4. This AGREEMENT represents the complete and full agreement. No verbal side-agreements exist. Any changes to this AGREEMENT must be made in writing. This applies also to the revocation of the requirements for the written form.
5. This AGREEMENT shall be governed by the laws of Germany. The sole place of jurisdiction for all disputes arising directly in connection with this AGREEMENT shall be Karlsruhe, Germany.

Executive Summary

The Unified Service Description Language 3.0 (hereafter USDL) is a platform-neutral language for describing services. It has been consolidated from SAP Research projects concerning services-related research and is intended as an enabler for wide leverage of services on the Internet. SAP Research believes that with the rise of commoditized, on-demand services, the stage is set for the acceleration of and access to services on an Internet scale. USDL coherently conveys the information about a service without requiring practitioners to make mental correlations across different service documents. The *unification* of information is schematically depicted in Figure 1. USDL brings together business, operational, and technical information in one coherent language. In order to achieve this, USDL defines normative UML class models and a corresponding serialization in XML Schema for capturing the "master data" of services. That includes modules, i.e., class models for pricing, legal, functional, participants, interaction, or service level aspects. Detailed specifications of each module can be found at www.internet-of-services.com. In addition, a W3C Incubator group has been established to drive standardization (cf. http://www.w3.org/2005/Incubator/usdl/).

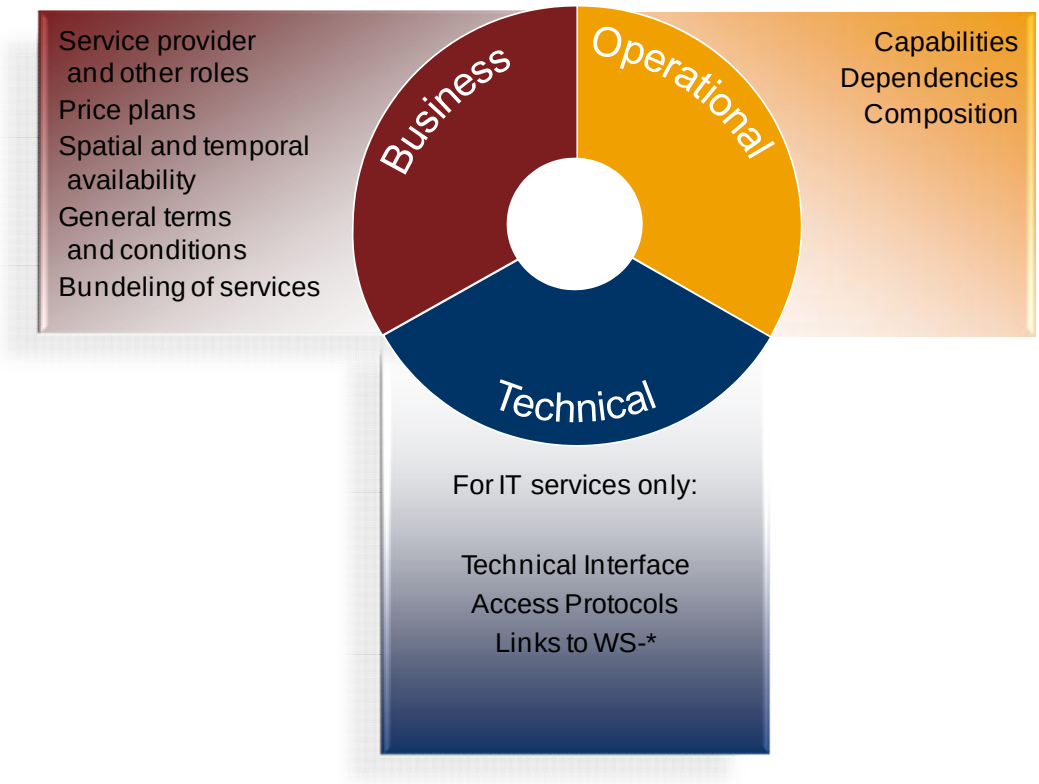

Figure 1 : USDL unifies business, operational, and technical aspects

USDL is built in a collaborative and interdisciplinary way. This means that about a dozen researchers at SAP Research, spread across different locations, contribute to the model by bringing in their expertise from different backgrounds (computer scientists, incl. security and SLA experts, business economists, legal scientists, etc.). A central governance body manages the modeling by coordinating the different contributions. The modeling is done in the context of several publicly funded research projects under the Internet of Services theme, where services from various domains including cloud computing, service marketplaces and business networks, have been investigated for access, repurposing, and trading in large settings. The projects include: the TEXO project within the

THESEUS1 research program initiated by the German Federal Ministry of Economy and Technology, projects funded by the German Federal Ministry of Education and Research projects (e.g., Premium Services), EU DG INFSO projects (e.g., FAST, RESERVOIR, MASTER, ServFace, SHAPE, SLA@SOI, SOA4ALL), and the Australian Smart Services CRC. A wide variety of use cases and perspectives were investigated across the different projects to develop service descriptions for forms of services and an alignment of business and technical aspects not available through previous R&D and standards efforts.

This document presents an overview of the USDL specification. Section 1 gives a general motivation of USDL by embedding it in the socio-economic situation of the services sector. Section 2 provides a survey of existing service description efforts. Section 3 elicits requirements for service description languages, which eventually guide the design of USDL presented in Sections 4 and 5. Finally, Section 6 gives an outlook on future developments and milestones.

[1] http://theseus-programm.de/en-US/home

1 Introduction

The service sector is by far the strongest economic growth driver in the world. As an example consider the Federal Republic of Germany, where about 70 % of the macroeconomic value of 2008 is generated by the service industry. As can be seen in Figure 2, jobs are only created in the services sector in Germany during recent years. A study commissioned by the German Federal Ministry of Economics and Technology concludes that there is growth potential in particular with regard to IT services. [67]

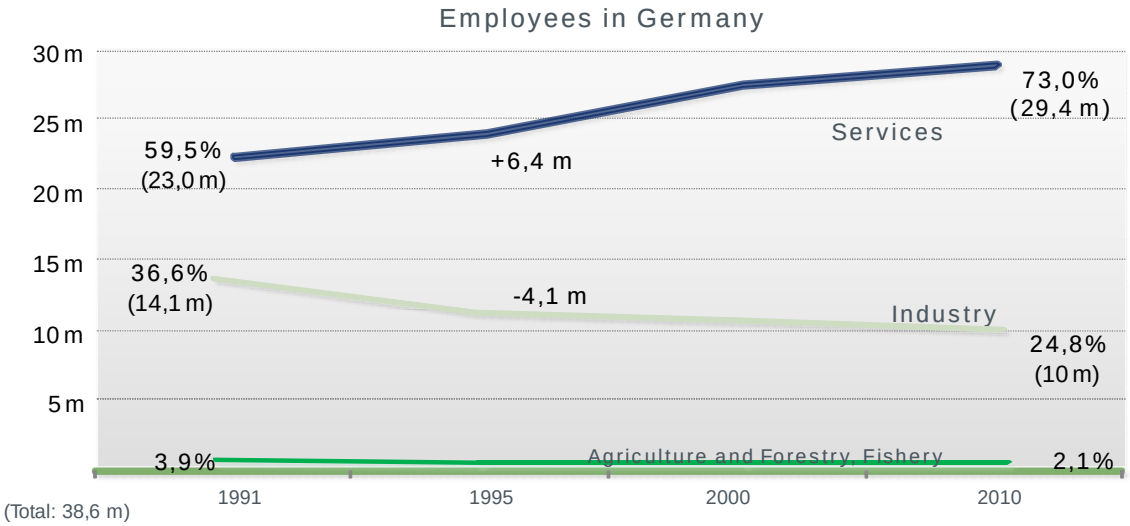

Figure 2 : Growth through services: Jobs created in Germany in the services sector as published by the German Federal Statistical Office.

This trend coincides with the ongoing industrialization of the tertiary sector in developed economies (i.e., the service sector). Figure 3 shows the three driving forces of industrialization, viz., automation, standardization, and specialization, which proved to be common across sectors.2 The Information and Communications Technology (ICT) serves as a *Automation* (cf. Figure 3) and transformation factor. Services that can be electronically consumed in wide settings, e.g., on the Web, have grown dramatically over the last few years. Beyond Web services and services available through the Web, new and disruptive models have emerged that are accelerating the ubiquity of services. Software-as-a-service, business process outsourcing, cloud computing and infrastructure-as-a-service, platform-as-a-service, service marketplaces, and service-centric business networks are a growing list of examples where services are commoditized, exposed and accessed beyond conventional boundaries. In addition to Web consumers, the reach extends to mainstream industries like transportation and logistics, banking and finances, public sector, and manufacturing, when one considers the following sorts of Web services now available: track-and-trace of shipments, tariff look-ups, health insurance comparisons, medical assistance, business formation and ERP hosting. Even if service delivery is physical, e.g., car repair, trading of such services can be facilitated by ICT.

[2] As an example, consider the automotive industry. In this particular case, automation started by introducing production lines as early as Henry Ford up to the increasing application of robots. In the beginning, each car manufacturer developed its own tires, radios, or electronic components. By standardization of such components and their interfaces, specialization became possible. That means, component suppliers emerged that specialize in the production of tires, radios, or electronic components sold to a multitude of car manufacturers.

Figure 3 : Driving forces for the industrialization of services

With the growth in number and sophistication of services widely available, the question turns to how effectively consumers can discover, understand and compare services – with relative independence and without full reliance on providers. Experience has so far shown that any attempt to describe and catalogue services faces a common obstacle: what is a service? Despite the widespread phenomenon of "service" in economic, political, business, communal and individual parts of our life, there are still many uncertainties and varying viewpoints regarding the development of a general conception of services – a conception that has to address the different notions and forms of services under consideration in different domains and applications.

Consequently, different communities established their own notion of service and came up with specialized description efforts. For example, the IT community identifies services most readily with software and the technical interface for accessing the service. Several platform neutral languages are introduced for capturing different aspects in Service-Oriented Architectures (SOA), one prominent example is collectively known as WS-*. The Semantic Web community introduced several ontology-based description efforts for such software services in order to automate the discovery and consumption of such services in SOA through techniques of reasoning. With the rise of on-demand applications, the notion of Software-as-a-Service (SaaS) has arisen, covering software applications or business process outsourcing. The emphasis of services here implies that the consumer gets the designated functionality he/she requested together with hosting through a pay-per-use model. Thus, the notion of a service in the realm of SaaS is not synonymous with the notion of service in SOA. Service providers need to disclose non-functional properties such as pricing or availability, carefully. A further dichotomy in the understanding of services is the distinction between business and software services. While some alignment with software services is made, the thrust of information captured about business services in commercial organizations is how they are delivered.

Standardization in the service sector (cf. Figure 3) considers several aspects including the description of services to address the issues mentioned above. Here, USDL comes into play as a normative and balanced unification of service information, which is required to further capitalize on the growth potential of the services industry. The unification should be machine-processable, consider technical and business aspects of a service as well as functional and non-functional attributes, and should provide both a blueprint and means for extensibility. The following quote underlines the importance of standardization in the service sector.

„Services make up an increasing part of the GNP of most developed economies. Similar to the area of physical products, standards are the basis and prerequisite of every further development. Therefore, standards will play a significant role also in the services industry. Standards are expected to drive the professionalization and industrialization of the service industry, to increase the transparency, and to lead to higher value services, and, thus, to contribute to the overall development of the service economy. “ [28]

The aspect of *Specialization* (cf. Figure 3) is twofold. First, services once targeted at a specific market, are repurposed to other markets in order to extend the reach. Second, there is a trend where certain participants in a business network specialize to play a specific role in the provisioning and delivery of services, that is, they act as intermediaries between other participants in the network. Such new specialized roles need to disclose and exchange, as well as comprehend business information about services (pricing, general terms and conditions, service-level agreements, etc.) in a standardized way. Consequently, the goal of USDL is to facilitate interoperability between such roles on the business level.

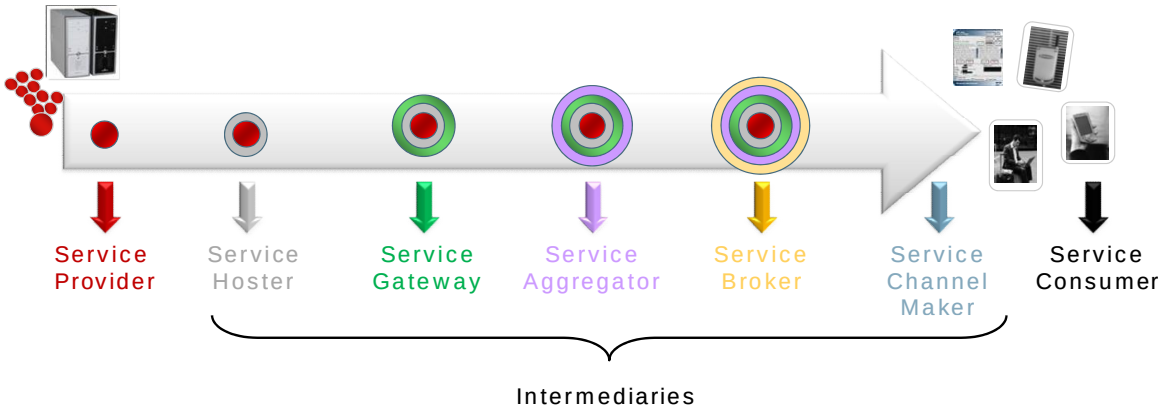

Figure 4 : Towards specialization in the service sector.

In the context of the Internet of Services and Business Web activities spearheaded by SAP Research, specific intermediaries are introduced (cf. Figure 4). These intermediaries have to be understood as one possible set of architectural roles, with each role conceptualizing a certain behavior or mode of operation requiring a specific set of applications and tools. Parties act in one or several of these roles while operating in service networks.

The Service Hoster is an example for an intermediary that catalogues special types of services, namely infrastructure-as-a-service and platform-as-a-service offerings (commonly termed cloud-computing services). It also provides means to interface uniformly with the providers of these services, i.e. re-hosts services through cloud computing environments. Likewise, the Service Gateway is a specific intermediary that provides interoperability through cataloguing and interfacing with a choice of a 3rd Party B2B gateway, which provide services like message translation and store-forward processing. The Service Aggregator provides additional value by packaging and combining services. In addition, the Service Broker cares for central service publication, discovery, and ordering. Finally, the Service Channel Maker is positioned at the consumer end of the service provisioning chain where services are channeled into user environments and consumed.

In this document, we provide an overview of the USDL specification. In Section 2 provides a survey of existing service description efforts. Section 3 elicits requirements for service description languages, which eventually guide the design of USDL presented in Sections 4 and 5. Finally, Section 6 sketches an outlook on future developments and milestones.

2 Survey

A plethora of service description efforts exist that can be grouped into different strands summarized in Table 1 below. As outlined in the introduction, each strand has its own notion of a service and different representation needs for capturing service information. The individual efforts can be attributed to the following criteria: (i) whether the scope of the effort lies in capturing IT or business aspects of services or the whole service system, (ii) the purpose of the corresponding effort, e.g., enabling of normative data *exchange*, facilitation of software *engineering*, or acting as *reference* model, (iii) whether the effort is standardized.

Table 1 Survey of service description efforts

Effort	Scope	Purpose	Repre-sentation	Standardized
1. Service Oriented Architectures				
WS-*	IT	Exchange	XML	Various
UDDI	IT	Exchange	XML	OASIS
SoaML	IT	Engineering	UML	OMG
UPMS	IT	Engineering	UML	No
SOA-RM	IT	Reference	Informal	OASIS
SOA Ontology	IT/ Business	Reference	OWL	Open Group
WADL	IT	Exchange	XML	W3C
Core Ontology of Web Services	IT	Reference	OWL	No
2. Semantic Web Services				
OWL-S	IT	Automation	OWL	No
WSMO	IT	Automation	WSML	No
...				
SAWSDL	IT	Exchange	XML	W3C
SA-REST	IT	Exchange	XML	W3C
RO-SOA	IT	Reference	RDFS	OASIS
WSMO-Lite	IT	Exchange	XML	W3C
MicroWSMO	IT	Exchange	RDFS	No
Minimal Service Model	IT	Exchange	RDFS	No
3. Software-as-a-Service				
SML	IT	Exchange	XML	W3C
SaaS-DL	IT	Engineering	XML	No
4. Service Networks				
OBELIX	Business	Configuration	RDFS	No
e^{3}Service	Business	Configuration	RDFS	No
SNN	Business	Optimization	UML	No
5. Service System				
Alter	System	Reference	Informal	No
Reference Architecture Foundation of SOA	System	Reference	UML	OASIS
Service Design Model	System	Engineering	Ecore	No
Ontological Foundations of Service	System	Reference	FOL	No

Science				
TEXO Service Ontology	System/ Business	Reference	OWL	No
6. Economic				
DIN PAS 1018	Business	Reference	Informal	DIN
Emmrich	Business	Reference	UML	No
O'Sullivan	Business	Reference	ORM	No
Toma	Business	Automation	WSML	No

2.1 SOA Efforts

In the IT community, services are most readily indentified with software, and the technical interface for accessing the service. Therefore, the first strand of service description efforts is the field of *Service-oriented Architectures (SOA)*. SOA the architectural paradigm of the IT community thinking about IT assets as service components, i.e., functions in a large application are factorized in stand-alone services that can be accessed separately. Originally, several standards bodies specified several dozens of different aspects which are collectively known as WS-* (WSDL for interface descriptions, WS-Security, WS-Policy, WS-SLA for service level agreements etc.[3]).

Since one of the key components of a SOA is a service registry, the OASIS standards body introduced the concept of Universal Description, Discovery and Integration (UDDI), i.e., a specification for a platform-independent registry. UDDI services shall be discovered via white pages (address, contact, and known identifiers) or yellow pages (industrial categorizations based on standard taxonomies), as well as green pages. However, UDDI does hardly prescribe any schema for such information.

As the concept of SOA matured, calls for support in software and service engineering increased. Hence, the OMG standards body dedicated its focus to software engineering for SOA, and, subsequently defined the Service-oriented architecture Modeling Language (SoaML) [1]. SoaML actually originates in the publicly funded EU research projects SHAPE[4] and MODELWARE. The predecessor of SoaML is the UML Profile and Metamodel for Services (UPMS [2]).

Finally, the multitude of description efforts and different definitions of SOA led to a Reference Model for Service Oriented Architecture (SOA-RM) from OASIS [3]. Similarly, The Open Group drafts an alternative reference model in form of an ontology for Service-Oriented Architectures (SOA Ontology) [4].

Current research in the SOA strand mainly concerns RESTful services and their description. Until recently there was no counterpart to WSDL for RESTful services. The W3C submission Web Application Description Language (WADL) is about to fill this gap [5]. Note, however, that version 2.0 of WSDL can be used to describe REST Web services, thus competing with WADL. Another recent research is the work of [6], which provides an ontological account of Web services. The resulting core ontology of web services is built according to the principles of ontological analysis on top of the DOLCE foundational ontology, and, thus, can be regarded as a reference model.

2.2 Semantic Web Services Efforts

A second strand consists mainly of ontologies in the field of Semantic Web Services. As presented in the seminal paper, viz., [7], the main goal of Semantic Web Services approaches is automation of discovery, composition, and invocation of services in a SOA by ontology reasoners and planning algorithms. The most prominent efforts are OWL-S [8] and WSMO [9]. Many surrounding and similar

[3] Cf. http://en.wikipedia.org/wiki/WS-*
[4] http://www.shape-project.eu/

efforts have surfaced in literature. For example, [10] try to unify existing ontologies that capture qualities-of-services in order to automate service discovery.

Considering the need to attach semantic descriptions to existing WS-* descriptions and the multitude of ontologies available, the W3C came up with a recommendation called *Semantic Annotations for WSDL (SAWSDL)* [11]. SAWSDL introduces XML attributes to establish correspondences between tags in WSDL (or arbitrary XML Schema documents) and concepts or relations in an arbitrary ontology. A similar idea to SAWSDL is already adopted by the W3C Member Submission called *Semantic Annotations for REST (SA-REST)* [12]. SA-REST defines three basic properties that can be used in a non-intrusive way to annotate HTML/XHTML documents, typically to embed ontological meta-data. These properties are included as part of the XHTML document allowing a capable processor to gain extra information about the content of the document.

With the many approaches around came the need to specify a reference model for semantic SOAs. Consequently, the OASIS is also about to specify a *Reference Ontology for Semantic Service Oriented Architectures (RO-SOA)* [13].

Current research topics of this strand tackle the new concepts of Linked Open Data, RESTful services, and Linked Services [57]. Consequently, the ongoing publicly funded research project SOA4All brought forth *WSMO-Lite (Lightweight Semantic Descriptions for Services on the Web)* which is in W3C Member Submission state [14]. *MicroWSMO* [55][56] is another approach to tackling RESTful services and Web APIs. It is similar to SA-REST but relies on a different serialization (microformat) and acknowledges the need for a service model. The latter is captured in the *Minimal Service Model* [57]. This model essentially captures the basic notions of a Web service from a technical point of view. The minimal service model can be used to bring together MicroWSMO, SAWSDL, WSMO-Lite, OWL-S, etc., for instance.

2.3 Software-as-a-Service Efforts

The third strand is rooted in the rise of on-demand applications that led to the notion of software-as-a-service (SaaS), covering software applications (e.g., CRM on-demand) and business process outsourcing (e.g., gross-to-payroll processing, insurance claims processing) to cloud and platform services. The emphasis of service here implies that the consumer gets the designated functionality he/she requested together with hosting through a pay-per-use model. Thus, software-as-service is not synonymous with Web services and the service providers need to carefully disclose non-functional aspects like pricing and availability and to factor these into the overall service they deliver. Services, in other words, are more than core functions that are accessed by consumers. They are *delivered* by a provider to a consumer possibly over a specified period of time, in a particular geographic context, with a pricing model and payment structure, monitored with a service level agreement, and related legal obligations of the consumer and the provider [15]. The functionality together with the aspects of delivery (e.g. hosting), and constraints, rights, obligations and penalties understood between providers and consumers for delivery, moves toward an understanding of service encountered in commercial practice.

The strand of SaaS contains a standard, namely, the W3C recommendation called *SML (Service Modeling Language)* [16]. SML is a strict superset of XML Schema adding the capability to define constraints on a model (using Schematron rules) and the capability to define and reference model elements in separate files. In addition, SML-IF defines an exchange format so that SML model instances can be easily exchanged between producers and consumers. One anticipated use for SML is to define a consistent way to express how computer networks, applications, servers, and other IT resources are described or modeled so businesses can more easily manage the services that are built on these resources. Therefore, we have classified it in the SaaS strand. Note however, that the use of SML is not limited to SaaS scenarios.

Current research is represented by the *Software-as-a-Service Description Language (SaaS-DL)*. SaaS-DL builds on WS-* to capture SaaS specificities in order to support model-driven engineering [17].

2.4 Service Network Efforts

The fourth strand draws attention mainly to describing Service Networks, i.e., the ecosystem and value chain relationships between services of economic value. So far, this strand has not produced any standards and is only represented by academic approaches. An early work is [18] which is continued by the ontology of the publicly funded EU research project *OBELIX* by [19]. The latter is an application ontology that helps users with configuration of service bundling and graphical modeling of service networks. Similarly, the work presented in [20][21] introduces the *e³Service* ontology to model services from the perspective of the user needs. This offers constructs for service marketing, but in a computational way, such that automated reasoning support can be developed to match consumer needs with IT-services. The focus of this work is to generate service bundles under the consideration of customer needs.

The research presented in [22] introduces the *Service Network Notation (SNN)*, which captures similar aspects to the e³Service ontology. However, SNN is a UML model that can be analyzed for measurements of extra value for each single participant as well as for the whole network optimization of value flows.

2.5 Service System Efforts

Fifth, there are overarching efforts that concentrate on the bigger picture of service systems or service science also taking into account socio-economic aspects. Steven *Alter* was one of the first to realize that the concept of a service system is not well articulated in the service literature. Therefore, he contributes three informal frameworks as a first attempt to define the fundamentals of service systems [23].

Although the background of the *OASIS Reference Architecture Foundation for SOAs* [24] is Service-oriented architectures, the specification argues that SOA-based systems are better thought of as ecosystems rather than stand-alone software products. Therefore, the specification is put into the service system category. The reference architecture foundation is based on the OASIS SOA-RM as its starting point by building on its vocabulary of important terms and concepts.

Another effort considering the wider scope of the service system is the *Service Design Model* of [25]. Other than the aforementioned efforts, the Service Design Model is geared towards a software engineering purpose. It comes in the form of an Ecore model for the Eclipse Modeling Framework (EMF). The model takes into account the business organization, the customer, and the delivery organization during service design. The model aims at providing a foundation for the design for service quality by envisioning different modules (such as financial, resource, process, etc.), configurability, variability, and extensibility.

Current research in this strand is represented by the work of [26], which can be seen as a continuation and formalization of Alter's approach. Although differing in its main notions, they present a reference ontology for *ontological foundations of service science* which is founded on the basic principles of ontological analysis. In turn, this reference ontology forms the core part of the *TEXO Service Ontology,* which extends it by ontology modules for pricing, legal, innovation, or rating information [27].

2.6 Business Efforts

A further dichotomy in the understanding of services is the distinction between business and software or technical services. Ironically enough, software practitioners appeal to the notion of business or enterprise services to emphasize the business relatedness of software solutions, while business practitioners take it for granted that their software applications are used in commercial operations. Different parts of commercial organizations catalogue their services assets to different ends. The focus of governance portfolios tends to focus on services (and business processes) as integral to business operations, meaning their cost centers, organizational objectives, customer segments and volumes, operating margins, profits, revenue targets etc. While some alignment with

software and IT services is made, the thrust of information captured about services in such portfolios is how they are directly understood and managed in the business. At the other extreme are software registries, which describe software services, their technical dependencies and supportive platforms. The separation of concerns for cataloguing services for business and technical portfolios leads to a gap in understanding how services can be more comprehensively described and managed across their business and technical manifestations.

Therefore, the final strand is driven by schools of business administration and business informatics and focuses on capturing the purely economic aspects of services regardless of their nature (with less or no focus on IT services and software architectures). Led by the Fraunhofer IAO research institute, the German standard *DIN PAS 1018*[5] [28] essentially prescribes a form for the description of services for tendering. The structure is specified in a non-machine-readable way by introducing mandatory and optional, non-functional attributes specified in natural language, such as, classification, resources, location, etc. The standard is driven by needs of the services industry in Germany whose expectations are the professionalization and industrialization of the service industry, the increase of transparency, and eventually the overall development of the service economy.

One of the first attempts at comprehensively describing services was the work of *O'Sullivan* [29]. This work drew from practical insights into how everyday services like house building, insurance and hotel services are advertised and offered, to a scheme for describing services and a variety of delivery aspects including locative, temporal, pricing, payment, security, trust and rewards. As O'Sullivan observed, "The everyday services that surround us, and the ways in which we engage with them, are the result of social and economic interaction that has taken place over a long period of time. Any attempt to provide automated electronic services that ignores this history will deny consumers the opportunity to negotiate and refine over a large range of issues, the specific details of the actual service to be provided". The proposal aims to advance the way services are self-discovered and accessed online, shedding insights for e-commerce applications concerned with services, as opposed to solely goods.

The PhD thesis of *Emmrich* [30] has a similar motivation only that this work focuses on product-related services, such as maintenance, and is specified in UML. It basically merges existing standards and models for products, companies, organization, and resources. *Toma* [31] presents a syntactic translation of O'Sullivan's work in the proprietary WSML language. The goal is to extend the aforementioned WSMO by non-functional properties for automation of discovery, composition, invocation, and, in particular, ranking of services in a SOA [32].

[5] DIN is the German abbreviation for „German Institute for Standardization." PAS stands for Publicly Available Specification.

3 Requirements

In order to understand what is required in describing services through dedicated languages, we use a constructivist approach to establish the common basis that can be assessed and negotiated for different perspectives and views, i.e. an inter-subjective view. The constructivist approach has been used in the philosophical treatment of Information Systems [33] and conceptual modeling of information systems [35], which to varying degrees embody services. We provide a constructivist synthesis by first laying out a general discourse on services. The general discourse on services (Section 3.1) allows us to identify the relevant universe of discourse for service description languages (*what* has to be covered?). Section 3.2 continues and proposes requirements, as design principles, to guide the formation of service description languages (*how* to design the language?). The requirements are grounded in literature, e.g., [34], and are underlined by a double round delphi study with 20 organizations.

3.1 Universe of Discourse

An analysis of the nature of services is a dedicated topic in its own right. For the purposes of this document, several salient aspects of services are discussed as a background for motivating the relevant universe of discourse for service description languages.

3.1.1 Core Functionality

The challenge of describing services lies in the fact that a wide range of artifacts, across economic, political, commercial, community and individual settings, are referred to as services. These range from purely human (e.g. consultancy services), to purely automated (e.g. message store/forward system), with a mixture of human and automated resources used for others in between (e.g. purchase order requisitions). The most basic characteristic of all sorts of services is that they are intangible, meaning they are not physically existent, as is the case with goods. Services fundamentally involve operations or actions, exposed to and provided for consumers as *capabilities*. Service capabilities aim at generating value to both the consuming and providing party (value co-creation), which is measured in different ways, for expectations manifested as requests and remunerations. The value is attained when the outcomes or responses meet the expectations of requests and the remunerations are transferred.

3.1.2 Service Agents

A further challenge relates to the positioning of services against other forms of organizational artifacts. Unlike business processes, as well as business objects and business resources, which are implemented through business applications, services are encapsulated and exposed functionality that draws from these core artifacts. They are cast in service-related languages (like WSDL and REST) as pure interfaces over implementations. Hence, services share similarities with these, so that they are sometimes regarded synonymously. This is especially the case with business processes. Business processes are composed as a “pipeline” of actions, with assigned resources (human or automated) for each, operating on business operations, applications and business resources. While being similar to services, their focus is on the internal details of organizations and their systems, i.e., “how” requests, actions and responses are processed to fulfill consumer goals. By comparison, services are mostly targeted for external consumers and focus on requests and responses through exposed capabilities but are internally implemented. Where business process activities are orchestrated, service capabilities are *delivered*.

Fundamentally, services are delivered to *consumers* by *providers*. With service delivery increasingly taking place in wider settings such as business networks, a distinction has emerged between owners of services and the providers to whom owners outsource core delivery responsibility. For example, a

globally acting bank owns account management services. However, its subsidiaries in different parts of the world act as providers of the services. In relation to this, the notion of *stakeholder* is important since certain partners in a business network have an interest or involvement in a service without having any full responsibility for its provisioning and delivery. Examples are government regulation authorities that define compliance requirements for commercial services, e.g. customs/quarantine procedures and carbon footprint levels. Furthermore, the provisioning of a service is often thought as being the responsibility of a service provider. However, the rise of service marketplaces, cloud computing and business processing hubs are extending the distribution of provisioning to third party service brokers. On service marketplaces, for example, providers from diverse agencies expose services with their pricing and conditions of delivery. In addition to brokers, other *intermediaries* are emerging like cloud providers who can offer third-party hosting, B2B gateways that can offer message translation as a service, third-party aggregators that extend the value and capabilities of services through innovative repurposing, bundling etc., and channel partners that allow services to be interacted with through new user-interfaces and experience.

3.1.3 Non-functional Properties

Delivery of services is subject to constraints so that expectations of requests and responses are firmly in place for consumers, providers and other agencies involved like intermediaries and stakeholders. Constraints include pricing, licensing, temporal availability, geospatial availability, dependent resources with necessary capabilities, and so on. Given the range and complexity of services across varying durations, the delivery of services entails rights, obligations, and penalties on the part of providers, consumers, and others. These are described in formal documents like service contracts and service level agreements. The risks of contention, negligence or malpractice are considered significant. Hence, non-functional properties need to be made explicit for service delivery. This is in contrast to business process orchestration where pricing and similar information tends to be regarded as tacit.

3.1.4 Dependencies

We generally understand the procurement of a service as follows. A service is assembled and *provisioned* to meet its capabilities. It is exposed where it can be *discovered*. It is requested by consumers. It is invoked and *delivered* for the required outcomes, noting that the same running service may be used for several requests. As part of delivery, there may be several interactions with consumers and a formal phase generally known as *settlement/fulfillment*, where the value of a service is ultimately transferred, subject to constraints such as payment and exchange of legal documents. Whether services are provided through human/professional activity, Web services over business applications, widgets and information look-ups, digital media, platform, infrastructure or other machine related services - we observe that this same generic pattern applies, more or less, to the way in which services are used for delivering value to their markets and audiences. Variations of it arise from different forms and granularities of services, different organizational settings for provisioning and delivery, and different ways in which services are consumed.

Services have different degrees of *complexity* and *granularity*. Generally speaking: The more complex a service, the greater the number of capabilities it exposes and the more likely it is to use other services and resources and impose constraints. A rather simple service like a price comparison may require some basic input from a consumer and display output through a single interaction without payment or any notion of settlement/fulfillment. Such a service is said to have a single capability, although it should be noted that the output could be drawn from complex application configuration because many applications are running in remote environments. At the other extreme, a service implemented by a business process like obtaining a passport executes through different human/automated resources and may involve different stakeholders (supportive providers) and services that contribute to its capabilities. Not all capabilities may be relevant to consumers and some capabilities result from negotiations between provider and consumer, which increase/decrease

the cost and reliability for example. Similarly, a service implemented as a complex, multi-stakeholder business process like property checks for a conveyance lawyer could be simple from a consumer's point of view in that it may involve a single capability and a few interactions. A cloud hosting service has a few up-front interactions with a consumer and on-going CPU, storage, memory and network services, with ongoing monitoring and billing. Hence, we can see that a service's capabilities, its granularity and dependencies, and its interactions can have different and subtle interplays that must be captured by a service description language.

To provision a service, it has to be prepared in a way that it can be delivered for the capabilities it exposes to consumers. This means that the service had all the necessary functional components and resources available and is placed in the environment where they can be executed and delivered. All the dependencies between resources and services need to be understood and risks of execution across loose-couple environments need to be mitigated. Services exposed from corporate environments, for example, have complex dependencies in their business and IT landscapes where there could be thousands of resources and internal services contributing to capabilities of exposed services. There are different forms of service dependencies for different purposes. Classic service composition generally applies for functionally composing services for greater value-add. Other forms of dependencies relate to ensuring the right components are in place for delivery, e.g. metering and payment engines. For market competitive reasons, services are *bundled* together without any particular order of execution or data dependency involved.

3.1.5 Service Access

When services are accessed, they need to be delivered across all their dimensions – technical, human, business and material. As elaborated above, services are interactional and they have effects on consumers and other entities with which they interact. They need to be understood in terms of all interactions that can occur at different time intervals, in different locations, with different entities be they consumers, stakeholders, intermediaries, or providers. Services are accessed through designated points of consumption known as channels. Business channels (e.g. a public/private local business development portal, mobile banking, and whole-of-government citizen services point) are forms of business resources, which have consuming applications that allow access to services, or specific operations of services. They may be pre-configured to access particular services or they may allow run-time discovery, selection and ordering. Consumers may directly interact with these applications or the applications may entail automated interactions with services. In the case of the latter, it is crucial for interactions and their data contexts to be described in a way that permits the interoperation of consuming applications and services. For end consumer interactions, different operations of a service may be required in different channels. For example, a passport application requires full authentication of a consumer through a counter channel. Progress in processing the passport may permit a self-serve channel while final handover of the passport can only occur through the counter. Channels may have time and geographic constraints in which services are accessed. They can support different technical channels for different types of devices that are permitted when interacting with services. While it is typical to think of service as executing with a hosted environment of a provider or intermediary, certain classes of services like digital media and lightweight software components may need to be downloaded and executed in the consuming environment.

3.2 Language Formation

The general discourse on services in the previous sections allows identifying the relevant universe of discourse for service description languages. In the following, requirements are proposed that guide the formation of service description languages. These are split into *generic language requirements* (Section 3.2.1) and *service concept formation requirements* (Section 3.2.2).

3.2.1 Generic Language Requirements

3.2.1.1 Conceptualization

First and foremost, a comprehensive description of services, across all the facets and perspectives of the wide variety of stakeholders, is best supported on the conceptual level. In general, the conceptual level, as described by the well-known Conceptualization Principle [36], focuses specifications on essence, allowing users to understand these free of implementation details. For services, the conceptual level best supports the variety of needs for service descriptions such as discovery and match making, composition and extension, and their exploitation through of a variety of tools and execution platforms. Indeed, the requirement for the conceptualization of a service description language allows specialized languages of different aspects of services, such as business process modeling techniques and WSDL, to be leveraged, rather than being subsumed into a single, “silver bullet” language. The core part would rather serve to capture the most common and essential features of a service while other parts captured through dedicated languages would be integrated through the core. Conceptualization allows different forms of integration such as referencing dedicated parts or model-based mapping to implementation specification languages.

3.2.1.2 Expressive Power

Correspondingly, a service description language should have a sufficient expressive power so that full conceptualization is possible. The expressive power of a language is the computational measure of language’s capability to capture its target. When considering the different aspects of services like service pricing and contract documents encountered in commercial trade, it is clear that the concepts, relationships, constraints and behavior needed for full service description are large and complex. A trade-off is often sought for the completeness of descriptions versus their application needs. For example, service level agreements involve rights, obligations and penalties. However, these are captured as clauses without any semantics in current SLA and service description languages. Thus, a highly expressive service description language is imperative to capture services adequately and to allow for other parts of services to be integrated so that a service can be effectively discovered, extended, and interacted with in a collectively coherent way. In addition, a service description language should be sufficiently expressive to support the requirements of modularity and extensibility (discussed below).

3.2.1.3 Modularity

Given the size and complexity of service related information, a service description language should explicitly support modularity in order to improve maintainability and extensibility of service descriptions. When describing services ranging from purely professional/human services such as project management to cloud-based infrastructure services, not all concepts/attributes are applicable. It should be possible to use only those sets of service descriptions or modules that are relevant, including using new modules (or "extensions") where existing modules do not adequately allow for a complete description of the service.

3.2.1.4 Extensibility

Given the diverse industries and domains involving services, a “one-size-fits-all” service description scheme is infeasible. Even on a generic level for aspects such as service ownership, pricing and availability, new and unforeseen requirements for descriptions should be expected. In other words, service description languages should be extensible so that they can be used in specific business contexts involving new requirements that have not been factored into the supported set of service descriptions. To support extensibility, a service description language should adopt a layered approach from core generic concepts (i.e. core concepts that apply to all cases), to specialized generic concepts, and all the way to specialized, domain-specific concepts (i.e. those concepts that are relevant to a specific application or domain) such as those available through B2B industry

standards (e.g. VICS and eTOM). As part of this, a service description language should support concept specialization and configuration to allow it to be readily tailored to satisfy the requirements of a variety of domains and changing needs.

3.2.1.5 Comprehensibility

In light of the size and complexity of service related information, and the fact that service domain experts are non-technicians, service descriptions languages should be comprehensible. The conventional ways for making large and cumbersome models comprehensible include the use of graphical representation, model views, and stepwise decomposition from abstract to detailed levels. Since services are conveyed through textual form, different and multi-modal forms of service description should be supported. For example, it should be possible to view service contracts in structured, semi-structured or unstructured forms (textual narratives).

3.2.1.6 Formal Foundation

As with languages used for other applications, it is crucial that a service description language has a formal foundation. In particular, it should be assigned a formal semantics, in order to support for the full repertoire of concepts and features (alluded to in the aforementioned requirements) in a clear and unambiguous way and to enable formal reasoning or constraint-based model querying.

3.2.2 Service Concept Formation Requirements

In information systems' conceptual modeling literature, the requirement of the *suitability* of a language or technique for its intended domain is prominent. Different domains require different language concepts and features suitable for them. To illustrate the point, consider that a program could be written in a Turing Machine given its expressiveness and precise behavior; however, a Turing Machine is clearly unsuited for the task. The issue of suitability gets to the heart of a constructivist synthesis of concept formation for a service description language. In [34], principles were proposed to guide the identification of suitable modeling concepts for business processes. Since business processes are closely related to services (cf. Section 3.1.2), we adapt these requirements for the suitability of service description languages, described below.

3.2.2.1 Organizational Embedding

In reality, services are organizational systems or subsystems and so all concepts of services relate directly or indirectly to organizational concepts. In that sense, the concepts, as with those of other types of organizational artifacts like business processes, objects and resources, are said to be organizationally embedded [33]. More specifically, services are grounded in organizational functional structures. As "wrappers" of functionality procured by organizations, their underlying components like business processes, application operations, UIs and resources are ultimately grounded in the organization's functions at strategic, tactical and operational levels. Organizations represent their functional structures in different ways, e.g. informally expressed "org charts" to more formal enterprise models. The requirement that a service description language be organizationally embedded means that its concepts and features should be aligned with models capturing organizational functional structure (like enterprise models). This requirement does not mean that service description language subsume concepts for organizational functional structures. These serve a wider purpose than the detailed descriptions of services like planning applications (governance, reorganizations, mergers and acquisitions, outsourcing/in-sourcing) and they refer to all sorts of organizational artifacts, not just services. The requirement, however, does entail alignment of service descriptions with organizational concepts, whether they reference organizational concepts elsewhere maintained (e.g. LDAP directories) or whether they provide corresponding concepts.

A significant aspect of organizational alignment for information systems and thus services is resourcing. In business processes, resources are allocated from within organizations to undertake

ordered activities whereas at the level of services, the emphasis is on the delivery partners involved: owners, providers, stakeholders, intermediaries, and consumers. Each of these may be an individual or a role within an organization. The extent to which partners are described in a service language aligns with their organizational roles (and other organizational artifacts for that matter). This is challenging in a distributed context. A way to manage this is to allow publicly exposed artifact descriptions through a central repository so that these could be reliably used in service descriptions. In this set-up, synchronization with the local organization repositories is required.

Further aspects of service concepts that should be organizationally embedded include delivery aspects such as pricing, availability as well as functional composition and bundling, whose constraints are ultimately determined by organizational policies.

3.2.2.2 Cognitive Sufficiency

In [34], the need for business process models to provide a sufficient cognizance for human interpretation was identified. This need led to the requirement for integrating information about control and data flow, resources, events and message exchange, task pre- and post-conditions including temporal constraints, task UIs and others, in order to combine them effectively in a rich model. Accordingly, we require that practitioners reliably understand the different aspects of services including models of their different artifacts, documents, code, and non-functional descriptions. For services, the challenge of cognizance is more pronounced than for business processes since services are the exposed units of functionality yielded from prior artifacts like applications and business processes. It is unclear how a logical service entity can be coherently presented across its different sub-models and documents captured through different techniques and languages.

A service, as we described above, logically consists of capabilities implemented by operations. Those operations are the ones related to different artifacts (like business process activities or update transitions of business objects). Moreover, capabilities are subject to delivery constraints and constraints of ownership, provisioning, pricing, availability etc. If such constraints are separately captured as we have seen in current SLA languages (e.g., WS-SLA as mentioned in Section 2.1), then the association is lost as to which parts of a service specific constraints and dependencies relate.

Therefore, the requirement of cognitive sufficiency can be supported by a language that abstracts the core functionality of a service (capabilities and operations) through its descriptions. This allows for delivery constraints like pricing as well as references to models of service artifacts to be associated on this fine-grained level. The intuitive meaning of a service is then conveyed coherently without requiring practitioners to make mental correlations across different service documents.

3.2.2.3 Service Information Hiding

The well-known principle of information hiding is intrinsic to services since they are encapsulated, reusable and deployed units of functionality. Current service languages allow services to be captured through well-defined service interfaces while service composition languages allow services composed on business process (e.g. BPMN), service operation (e.g. SCA) and UI levels (e.g. widget composition). However, services may be realized as complex structures and can recursively consist of other services of different sorts and with different types of artifacts. Consider for example, a service for business formation, which is implemented through a long-running business process orchestrating many other processes for fulfilling provisioning of licenses, with value-added user interface components available at different stages for data sensing (e.g. market monitoring of different business lines, visualization of local supply chains). Furthermore, consider that for certain types of business licenses, other services are bundled as part of the fulfillment stage, like monthly tax reporting and office/HR support, through software-as-a-service applications with different business processes operating through them. The question is how to describe such a complex service package so that its different parts and dependencies can be understood and related to delivery constraints. The issue is similar to cognitive sufficiency (described above) in that a consistent logical structuring of

services is sought, in this case for complex services. However, it specifically relates to describing complex services independent of specific techniques like BPMN.

If a provider deploys a service that is implemented through a business process, then such a service could be described as a single service with different capabilities and operations, i.e., individual activities are not represented as services. If on the other hand, the business process makes use of other services that have been deployed (and catalogued as different services), the composition should be reflected on the level on service descriptions through service dependencies and should not be hidden in the business process model. Thus, complex and recursive structures of services can be represented through logical service descriptions and their particular artifacts (e.g. BPMN models) can be pointed to. In fact, such support of service information hiding allows different types of service artifacts to be composed in a way that is independent of particular composition types and techniques. For this purpose, different forms of composition should be described through a service description language without replicating specific techniques. The classification of compositions should include ordered triggering (business processes), data correlation (widgets and service operations), implementation inheritance (package imports) and unordered bundling.

3.2.2.4 Deployment Symmetry

As discussed above, the provisioning and delivery of services can be extended to third-parties such as service brokers and cloud hosts. For example, service brokers allow services to be advertised and delivered through their (marketplace) platforms. They "on-board" services so that their discovery, ordering/CRM, payment, orchestration engines etc., can be exploited. In doing so, new service offers can be created with new monetization models, e.g. advertising and rewards can be used to offset prices set by providers. Different service offers could be created for the same service with different pricing, reliability and risk mitigation constraints. Since delivery and execution of new service offers shifts to intermediaries, we require that languages allow service descriptions to be consistent and coherent, i.e. symmetric, regardless of how they have been extended and where they have been deployed.

In other words, as services are extended, there should be no loss of information in terms of their functional and non-functional aspects. As extensions of existing services, core descriptions and constraints set in place by providers concerning future extensions should not be overridden, e.g. a cloud or channeled version of a service does not change the core functionality of a service, but its price, response time and point of consumption does change. As extensions are created, services should be traced back to their previous versions and extensions should occur with reference to prior constraints and core service details.

Consider two implications of this requirement for service description languages. The first concerns the organizational context of services. Take for example, a business formation service, traditionally operated through the government, opened up to allow the private sector to repurpose and target the service to different market segments, e.g. supporting start-ups, retail outlets or building trades. A variety of service specializations may be created where private sector agencies take up the task of overall orchestration of the service, however the government retains the core responsibility of evaluating and granting individual business licenses. Moreover, the government requires that certain sensitive tasks like settlement may be delegated to outside agencies, however they need to be accredited/certified. Such constraints of delegation and retention of control can be made transparent through well-defined organizational contexts supported in service description languages. In general, the context and structure of services can be defined abstractly and configured through concrete bindings in specializations of services, thus ensuring operational compliance in new deployments of services.

A second implication of the deployment symmetry requirement is pricing. As services are extended, e.g. to run in a cloud environment, new pricing should be incremental and not displace the constraints put in place by the originating providers. For example, if providers require a certain fee for a service, new offers of the service created by brokers or other intermediaries (e.g. broker fees)

result in additional pricing constraints. Some parties provide credits, e.g. advertising revenue. In general, a symmetric description of service pricing for multiply extended services sees debit or credit pricing constraints, or price apportions, in the overall pricing schedule.

3.2.2.5 Execution Resilience

An important aspect of service delivery relates to exception handling. The natural place for exception handling is in detailed specifications of service tasks, for example in pre- and post-conditions and action specifications. Therefore, a service description language should support the handling of exceptions, so that the execution resulting from a specification can be validated as being resilient. This includes rollbacks, cancellation policies, contingencies etc.

4 Overall Design

The language formation requirements of Section 3.2 guide the formation of USDL and act as design principles. The requirement for Conceptualization (3.2.1.1) prompts the use of a conceptual modeling language, in our case UML. More specifically, the Ecore meta-modeling specification of the Eclipse Modeling Framework (EMF) was chosen. Ecore is an implementation of the OMG EMOF specification [49] developed by the Eclipse community, thus can be considered the "UML meta-model in Eclipse." In turn, EMOF features abstract semantics in terms of the OMG Object Constraint Language (OCL) responding to the requirement of Formal Foundation (cf. 3.2.1.6).[6]

In combination with the requirement for Expressive Power (cf. 3.2.1.2), USDL has to capture the universe of discourse outlined in 3.1. This leads to USDL being a domain-specific language. This fact already sets USDL apart from many related approaches discussed in Section 2. For example, UDDI, WSMO, or OWL-S only prescribe tiny schemata and leave the modeling of universe of discourse concepts (such as a generic schema for defining a price model or licenses) to the user. W3C SML, SAWSDL, and W3C SA-REST are also designed to be agnostic of any service description schema.

Table 2 : Generic language requirements and their corresponding design elements and decisions

Requirement	Design Element / Decision
3.2 Language Formation 3.2.1 Generic Language Requirements	
3.2.1.1 Conceptualization	UML
3.2.1.2 Expressive Power	USDL as domain-specific language
3.2.1.3 Modularity	USDL is split in different packages (modules)
3.2.1.4 Extensibility	Future work (cf. 6.2)
3.2.1.5 Comprehensibility	Future work (cf. 6.3)
3.2.1.6 Formal Foundation	EMOF Abstract Semantics

The distinction in business, operational, and technical information put forward in Figure 1 carries the main idea of USDL, namely the unification and interconnection of service information from all areas of the spectrum. Yet the distinction in business, operational, and technical aspects proved to be too coarse-grained for an adequate structuring. Instead, USDL is split into several packages (according to UML terminology) as a response to the requirement of modularity (cf. 3.2.1.3). Each package represents one module and contains one class model. The resulting modules are depicted in Figure 5 and are briefly explained in Table 3.

The Foundation module factorizes common parts of the remaining modules as a consistent continuation of modularization. Because of its basic character, all other modules depend on the Foundation module meaning they reference one or more of its elements.

[6] Ecore is an implementation of EMOF, the OMG standard meta-modeling language, which was used to define the UML language. Ecore is a widely adopted and wide-spread meta-modeling language that has appropriate tooling available based on the Eclipse development environment. EMF provides generation capabilities and several automatic transformations that facilitate the development of USDL tools.

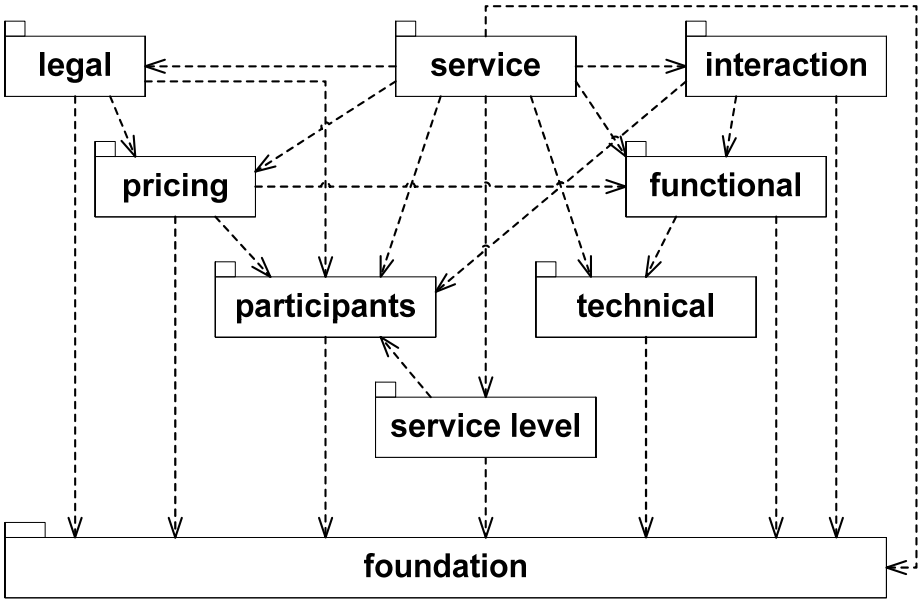

Figure 5 UML package diagram of USDL (dependencies between packages are represented as arrows)

Table 3 : Overview of USDL modules and their content.

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 the works. 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

The requirements of Extensibility (3.2.1.4) and Comprehensibility (3.2.1.5) have not been tackled yet. They remain future work and are discussed in Sections 6.2 and 6.3. Extensibility is a major topic that concerns the handling of deviations of USDL for specific industries, countries, or proprietary use cases. Special care has to be taken to prevent a proliferation of USDL derivations that require costly manual mappings as has happened with the multitude of product standards. Comprehensibility is

required because of the complexity of USDL on two levels. First, modeling services in USDL requires knowledge in several disciplines (legal, business economics, computer science). Second, the sheer number of modules, classes, and relations of USDL introduces a steep learning curve. One idea to remedy the problem is to introduce an intuitive graphical representation for USDL.

5 Modules' Design

Further requirements concerning the service concept formation are listed in Table 4 below. These are mostly addressed within the individual modules and are discussed in this section. We briefly explain the purpose, the content (in terms of central classes), and give corresponding examples. Besides, we position the modules to the related approaches of Section 2 and to further related approaches outside the realm of services (e.g., approaches to capture price plans for products). In addition, each of the modules features its own comprehensive specification available at www.internet-of-services.com.

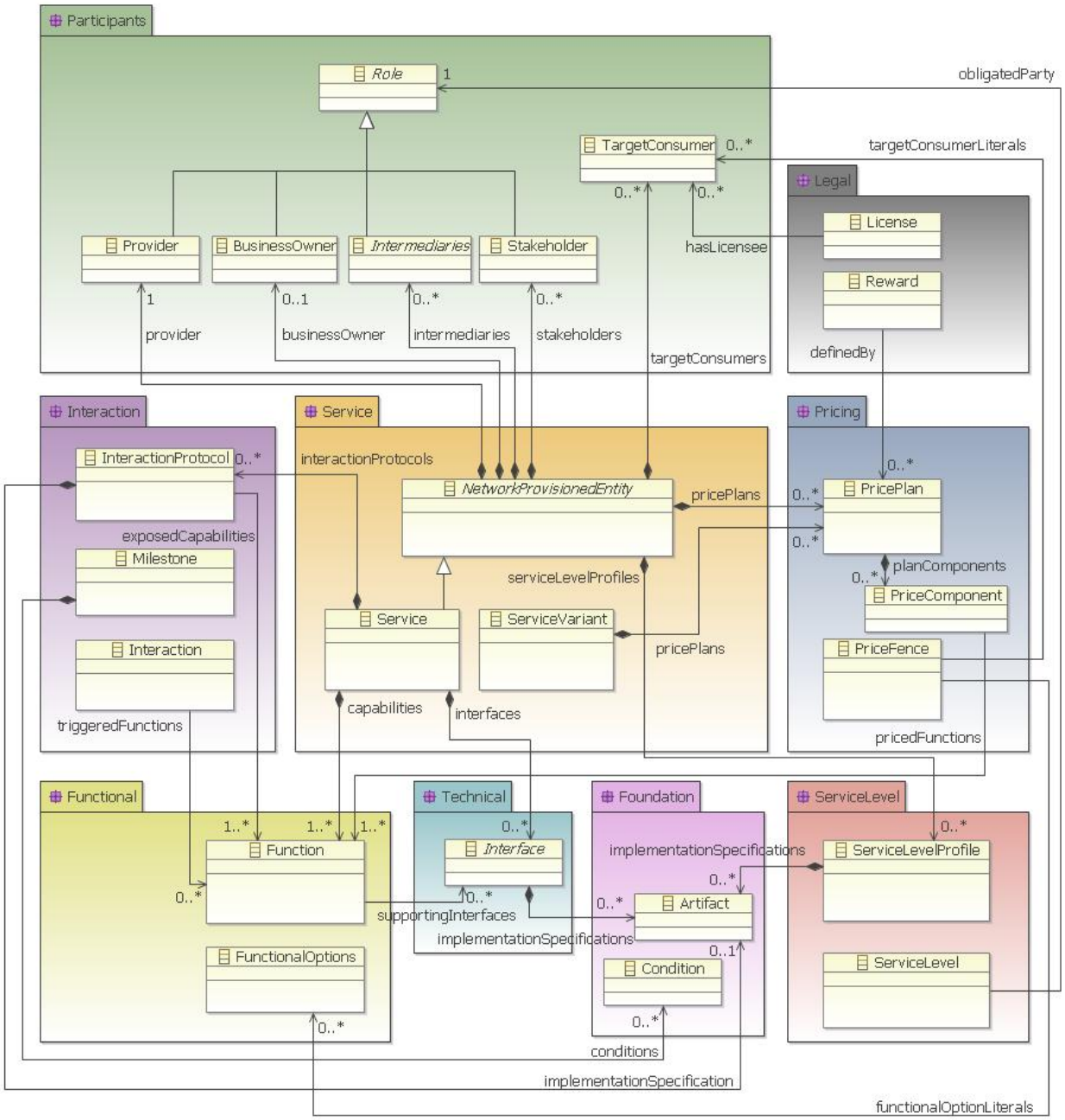

Figure 6 : Meeting the requirement of cognitive sufficiency leads to interconnected modules. Note that only interconnected elements of the individual modules are shown to give an impression. For comprehensive class diagrams of individual modules please refer to the module's specification at www.internet-of-services.com.

Regarding the requirement for Cognitive Sufficiency (cf. 3.2.2.2), a strong interconnection between USDL modules allows users to reliably understand the different aspects of services including models of their different artifacts, documents, code, and non-functional descriptions. Figure 6 highlights the interconnection (note that only interconnected elements of the individual modules are shown to give an impression).

Table 4 : Service concept formation requirements and their corresponding design elements and decisions

Requirement	Design Element / Decision
3.2 Language Formation 3.2.2 Service Concept Formation Requirements	
3.2.2.1 Organizational Embedding	Several concepts in Participants, Technical, and Interaction modules
3.2.2.2 Cognitive Sufficiency	Interconnection of USDL modules
3.2.2.3 Service Information Hiding	Recursive bundles, composition and dependencies in the Service Module; Recursive functions in Functional Module
3.2.2.4 Deployment Symmetry	Concept of AbstractService in Service Module
3.2.2.5 Execution Resilience	Concept of Fault, Preconditions, and Postconditions in the Functional Module

5.1 Service Module

Purpose: The Service Module can be regarded the center of USDL. The module ties together all aspects of service description distributed across other USDL modules (cf. Figure 6). Therefore, the requirement of Cognitive Sufficiency (cf. 3.2.2.2) is mainly met by the Service Module. The module also responds to 3.2.2.3 Service Information Hiding, since it allows the recursive specification of service compositions and bundles. That means existing and separate USDL descriptions can be combined or bundled by establishing links. Similar to establishing links to existing USDL descriptions for compositions and bundles, arbitrary dependencies can be modeled (cf. Dependencies in Section 3.1.4). On the one hand, dependencies comprise the actors providing and performing services, which normally happens under well-defined business and technical constraints. On the other hand, dependencies can exist on resources utilized by actors to consume or provide the service. Finally, the concept of AbstractService responds to the requirement of Deployment Symmetry (3.2.2.4).

Content: The concepts of Service, ServiceBundle, and CompositeService are the main concepts of the module as outlined above. 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 between the main concepts and resources. Besides, 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.

Example: Consider a carrier service such as FedEx International Economy. Such courier and freight services typically come with accompanying IT services, in this case a Web front and Web services for looking up rates, initiating or tracking shipments (e.g., FedEx Shipment Manager), and incident reporting. The Service Module allows capturing this setting via a ServiceBundle. Attributes such as Name, PublicationTime, or additional free-text descriptions are available to capture the most basic

information of the bundle. Links can be established to a PricePlan, Licenses, Functions, ServiceProvider, and ServiceLevels in the corresponding modules. Besides, dependencies can be specified to separate USDL files capturing the actual carrier and management services. Both would be instances of the class Service, which in turn also feature the above mentioned attributes and links, complemented by an attribute indicating their nature (here professional and automated, respectively).

5.2 Pricing Module

Purpose: The fundamental challenge of the Pricing Module is modeling the segmentation rules within the price structure, i.e., rules determining when and how different consumers are charged different prices. Our modeling is based on state-of-the-art [44] and research in business economics such as our own contribution in [58]. Besides, the modeling takes into account common segmenting practices, e.g., [43], in order to assess how they affect the price structure design. As such, the Pricing Module is relevant for capturing Non-functional Properties (cf. 3.1.3). It is also relevant for Cognitive Sufficiency (3.2.2.2) because of several relations to other modules as shown in Figure 6.

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 readily modeling scenarios with alternative price plans that may be assigned to an offered service or bundle. Each plan is possibly 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,[7] i.e., the criteria a customer must meet or the service limitations he/she needs to accept to qualify for a certain price.

Example: Consider a car rental service that would feature the following price components: car type, number of rental days, number of driven kilometers, use of special equipment, e.g., a navigation system. The USDL price model allows capturing the complexity of such a price plan including the information below

1. Each car type, classified according to the ACRISS[8] classification, is priced differently.
2. Every car type is assigned a price per day. This price is multiplied by the number of rental days.
3. A navigation system costs 5€ per day, if not already included in a car. Otherwise, the base price of the car will be higher.
4. A discount of 25% is given on the price of the navigation system if the customer is a member of the German Automobile Association (ADAC).

Related work: The Pricing Module progresses the state-of-the-art compared to (*i*) the state-of-the-art of Section 2 and (*ii*) additional related work of price models independent of the service domain such as [50] for electronic product catalogs. Therefore, the USDL price model provides a scientific contribution in itself as presented in [51]. Regarding (*i*), the USDL Pricing Module allows tiered pricing and a structuring in price components according to established business literature. Both is not possible in [29] and [31], for instance. In [20][21], the price model is represented as a mathematical formula included in the ontology in order to enable the price determination. Therefore, the work is not directly comparable to ours since the structure of a price plan is not declaratively modeled as discussed in [52]. Regarding (*ii*), we refer the reader to [51], where related approaches are grouped and positioned in four categories: (a) economic contributions, (b) established enterprise software, (c) standalone billing engines, and (d) explicitly specified pricing models.

[7] This terminology originates in "*The Strategy and tactics of Pricing*" [44], 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.

[8] The Association of Car Rental Industry System Standards, cf. http://www.acriss.org/

5.3 Legal Module

Purpose: The Legal Module of USDL represents a scientific contribution by itself, providing a novel approach to represent copyrights in a formal way as presented in [47]. In a first step, the general licensing aspects of the German legal code (Urheberrechtsgesetz) is captured by a generic copyright model. In a second step, the generic copyright model is further specialized to the subject matter, i.e., services. Only then it is guaranteed that USDL service license descriptions are compliant to the law by default according to [47]. However, this also means that each geopolitical context requires its own representation to capture the law correctly. The Legal Module captures Non-functional Properties of a service as outlined in Section 3.1.3. It is also relevant for Cognitive Sufficiency (4.2.2.2) because of several relations to other modules as shown in Figure 6.

Example: Let us briefly demonstrate the module by looking at the licenses of the GeoNames IT service, i.e., a database containing geographical information about more than 8 million locations. GeoNames is available as a free and commercial service with different licenses. Let us consider the free version, which is licensed to be used without payment. However, when using the service, it is required to attribute to GeoNames. Moreover, the usage is restricted to 50.000 credits per day and IP address. The service output needs to be attributed by the Creative Commons license CC-BY. CC-BY requires the licensee to attribute the licensor in the way he wishes.

Content: In order to capture such information, 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.

Related Work: The Legal Module is progressive compared to (i) the state-of-the-art of Section 2 and (ii) additional related work of license description approaches independent of the service domain. With respect to (i), we surpass [29] and consequently [31] since both are not based on a generic copyright model in a specific legal code. Therefore, they also do not consider geopolitical differences in copyright law. One of the most detailed approaches in dealing with service licensing is from [54]. They developed a service license model considering traditional software licenses. Based on this model, they have created an ODRL (Open Digital Rights Language) Service Profile (ODRL-S), which can be used to describe service licenses. The service model incorporates the WIPO framework and therefore lacks a profound legal foundation. [53] talks about usage policies instead of licenses and identifies different kinds of usage policies. It focuses on the technical formalization of such usage policies without any legal foundation. With respect to the body of related work of (ii), we refer the reader to the positioning given in [47]. In general, the domain of services is more diverse than the product domain.

5.4 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 or are to be delivered. However, SLAs in practice are specified at the top-level interface between a service provider and a service customer only. Customers and providers can use top-level SLAs to monitor whether the actual service delivery complies with the agreed SLA terms. In case of SLA violations, penalties or compensations can be directly derived. The Service Level Module captures Non-functional Properties of a service as outlined in Section 3.1.3. It is also relevant for Cognitive Sufficiency (4.2.2.2) because of several relations to other modules as shown in Figure 6.

Example: Consider the following service level specification of a logistics service, where

- Customers can specify their expected delivery duration,
- A provider-obligated term guarantees the delivery within the specified duration,
- A provider-obligated term guarantees that goods are maintained at -5 degrees Celsius,
- A provider-obligated action regulates penalty payments for delayed deliveries.

As an example for an IT service, the service level description could comprise

- Customers can specify the preferred availability and their expected usage rate,
- A customer-obligated term then regulates the usage rate,
- Provider-obligated terms specify availability and response time, and a provider-obligated action increased the storage capacity whenever actual utilization goes beyond 90%.

Content: The USDL Service Level Module allows modeling such information. The module is derived from our research presented in [60] as part of a multilevel SLA management platform. The 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 (e.g., as gold, silver, bronze profile). Further classes are ServiceLevelAttribute, GuaranteedState, GuaranteedAction, etc., which are interrelated and linked to Service in the Service Module and Role in the Participants Module.

Related work: SLA modeling is about the formal description of SLAs and other related artifacts (e.g., software and infrastructure) that matter within the overall SLA management process. One of the approaches implicitly mentioned in Section 2 is WS-Agreement [59], which can be seen as the prevalent standard for expressing SLAs. It defines a representation of SLA templates containing terms free to modify, SLA offers based on these templates, and agreements themselves. A large body of work exists in the areas of software modeling. The USDL Service Level Module goes beyond these approaches as it realizes an SLA foundation model with higher expressiveness, which is at the same time also integrated with other modeling artifacts from software or system level. For a detailed comparison to related work, please cf. [60].

5.5 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. As shown in Figure 4 on page 10, such parties are service consumer, service providers, and potentially various intermediaries in between. The Participants Module captures information about and dependencies between these service agents (cf. Section 3.1.2). It also enables Organizational Embedding (3.2.2.1) by relating such roles with contact persons, for instance.

Example: While the provider might act as the trading partner to consumers and define business aspects of delivery, there are scenarios in which this function is performed by another legal entity. This could be, for example, a national subsidiary of a multi-national organization providing the actual service. There are also third-party providers of delivery functions (e.g., billing or authentication) that can be orchestrated with a service enabling the outsourcing of these functions.

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

Related work: Most of the related approaches covered in Section 2 limit their focus on capturing service information relevant for service consumer and provider.

5.6 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 equally enable 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. Interface in this context means a set of concrete technologies through which the service can be accessed. A simple example is an automated service that has a WSDL-based Web service interface *and* a REST interface. Both modules capture Core Functionality (cf. Section 3.1.1), and respond to the requirement of Cognitive Sufficiency (4.2.2.2) by interrelating the functional and technical aspects of a service. The Technical Module also tackles Organizational Embedding (3.2.2.1) since it allows linking to an IT resource (the interface description of an IT asset) within the organization. Further, it addresses Execution Resilience (4.2.2.5) through the concept of Faults, Preconditions, and Postconditions.

Content: In case of IT services, the module is able to refer to the interface description. Similar to WSDL, the Functional Module introduces classes such as Interface, which represents a technical interface containing interface elements, e.g., operations or parameters that can be referenced explicitly. In line with the idea of Semantic Web Services (cf. 2.2), such 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 thus responding to Service Information Hiding (3.2.2.3). 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.

Example: Consider a geo information Web service. There might be a WSDL file describing an Interface with several Operations including *getAdr*. The *getAdr* operation might feature two input parameters of type *xsd:int* as well as a complex output parameter *Adr*. The corresponding Function could be called *Retrieve Postal Address*, thus capturing the conceptual meaning of the WSDL operation. The corresponding parameters could be annotated with *Geo Coordinates* and *Postal Address*, respectively. With this kind of annotation, we capture the business semantics of the interface whose functions can be put into a business protocol in the Interaction Module.

Related work: Currently there are no standardized means to capture the abstract functionality provided by a service. A common conceptualization, used for example in SoaML (OMG) or the SOA Reference Model (OASIS), is capability modeling. In [39] the concept of capabilities is discussed as expressing the ability to perform a course of action that achieves a result. When viewed on a whole, this constitutes the functionality offered as *the* service. Other scholars (e.g. in [26]) have extended this definition by adding the notion of commitment. This means that the entity producing/performing the service is not only capable of doing so, but also committed to do so upon request.

Capability modeling usually takes a *black-box* view because it only captures the capability as something that is externally visible and does not reveal how it is realized internally (cf. [40]). Other approaches aimed at modeling functionality provide a more detailed formalization, e.g., software function/component models and business process models. These approaches trace their roots to structured systems analysis and design techniques, which introduced the principle of functional decomposition, and offer a *white-box* or *gray-box* view depending on the level of detail provided about the components/processes. For example, drilling down to the lowest level of atomic functions performed in an organization including information about the roles and systems involved in the process can be regarded as a true *white-box* view (sometimes also called *glass-box*).

It should be pointed out that capability modeling does not have to be limited to a flat, single-layer model. In fact, there are approaches that propose quite diverse hierarchical models (cf. [40]), which also capture interconnections between individual capabilities in terms of inputs, outputs and exceptions. In using the principles of decomposition, they slightly extend their scope from *black-box* to *gray-box* and thus share similarities with function or component modeling. The difference is that they describe functionality from a business point of view, which does not go beyond a certain level of

detail, as opposed to an IT systems point of view that usually covers all aspects of realization/implementation.

5.7 Interaction Module

Purpose: As explained in the previous section, the Functional Module captures knowledge about the capabilities provided by a service and the means of communication employed to access them. In addition to that, it might be necessary to know as well in *what order* individual functions have to be performed or *when* it is necessary to interact with the service, respectively the actors performing it. This is important in order to access the service properly. These aspects are represented by the Interaction Module. They mostly apply to services that comprise a complex course of action. At the end of such a course of action, e.g., a multi-step visa application process, one or more capabilities are successfully rendered. In simple terms, the interaction protocol captures the externally observable behavior of services. Therefore, the Interaction Module captures information relevant for Service Access (cf. Section 3.1.5). It is also relevant for Cognitive Sufficiency (4.2.2.2) because it relates to the Participants Module to assign roles, and to the Functional Module. Further, Organizational Embedding (4.2.2.1) is enabled by linking interaction steps to existing workflow and business process descriptions such as WS-BPEL or BPMN resources.

Related Work: Interaction structures are well known in the domain of workflow and business process modeling (e.g., BPMN [61]) and have been applied in choreography languages to define collaborations (e.g. WS-CDL [62]). Within the spectrum of different modeling approaches, two general styles of representing control flow can be identified. On the one hand, there are flat, graph-based modeling notations that capture ordering as a set of relationships between vertices and edges (examples include BPMN and Petri Nets). On the other hand, there are hierarchical, block-structured languages that capture ordering through the semantics of the blocks (e.g. BPEL).

Content: For reasons of simplicity and traceability, it was decided to employ a basic version of the block-structured approach (only capturing sequences) in USDL. This was deemed sufficient for services that do not actually have an existing formal description of their externally observable behavior (manual services in most cases) and would need such a description. In order not to exclude complex interaction protocols, providers get the possibility to attach the definition of their protocol(s) in a formalism of their choice and potentially have an abstraction of the protocol(s) mapped to the USDL model. Phases may be introduced as necessary, as they usually only make sense for long running process-based services. The main class InteractionProtocol[9] defines an order among the Functions defined in the Functional Module including grouping them into Phases with Milestones. 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).

Example: Consider the InteractionProtocol to follow when opening a new term deposit account. There might be a Phase for capturing the application details followed by a Phase for submitting the application as well as a Phase for representing the state when the application is submitted. A Milestone might be reached when the required data is captured through a questionnaire, i.e., all relevant information for opening a new bank account is captured (including customer ID and contact details as well as options for the selected account. Phases are decomposed into Interaction, e.g., an Interaction for providing personal details. Each Interaction implicitly includes the Consumer. In addition is assigned a direction and a specific Role (cf. Participants Module) covering the opposite end of interaction, in this case the Provider. It may also link a specific Function (cf. Functional and Technical Modules), which is performed during or after the interaction takes place.

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

5.8 Foundation Module

Purpose: The Foundation Module factorizes common parts of the remaining modules as a consistent continuation of the requirement for Modularity (3.2.1.3). Because of its basic character, all other modules depend on the Foundation Module. That means that they reference one or more of its elements. In order for concepts to be moved to the Foundation, there need to be at least two modules that use a concept in the exact same way. If it is not the exact, but a similar way, a conscious evaluation is conducted, after which several individual concepts from two or more modules may be unified into a single (set of) concept and placed into the Foundation module. The Foundation Module also serves as the container for concepts that logically cannot be assigned to any USDL module, as well as concepts that by themselves are not dependent on the 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.

Related Work: Concepts of location (space) and time are not an invention of USDL. Their conceptualization in the model has been designed to specifically cater for the needs (and challenges) of service description and is aligned with concepts in the domain of geographic information systems and international time standards. Specifications that were consulted include the Geographic Markup Language (GML, ISO 19136) of the Open Geospatial Consortium,[10] as well as ISO 8601 and ISO 19108 (standard associated with GML) for the exchange of time-related information. Whereas GML and ISO 19108 were deemed too complex to be referenced in USDL (the tooling support was considered to be too much overhead), ISO 8601 covers only a part of what was identified as relevant. Consequently, both sets of concepts (time and location) were re-modeled in USDL and a clear indication of their relation to the above-mentioned standards is provided as part of the specification text.

[10] http://www.opengeospatial.org/

6 Future Work

6.1 Standardization

A comprehensive, yet generic, description language for services constitutes a major building block in the wider vision of the Internet of Services. The Internet of Services connects many participants for the purpose of provisioning, trading and delivering services. In order to facilitate interoperability especially on the business level, the Internet of Services has to be built on a set of well-defined standards. As explained in the context of USDL, current widely-used service description efforts are not sufficient to support all requirements and thus need to be supplemented with a description language unifying business, operational (functional) and technical characteristics of services. USDL is a proposal for such a language.

Therefore, the strategy is to put modules of USDL before the community when they reach a certain maturity level. Experts from academia and industry are invited to review the modules and interested parties are free to take USDL and apply it to their use cases and domains. This will further validate the concepts of the language and eventually provide feedback to the team working on it, which will help to consolidate the model.

Standardization is a logical and necessary step to make USDL a true building block of service networks in general and the Internet of Services in particular. In order to prepare USDL for this, a W3C incubator group has been set up (cf. http://www.w3.org/2005/Incubator/usdl/), which is open to W3C members and other interested parties from industry, academia, governments, and other institutions. This body assumes ownership of USDL after version 3.0 has been released and its main goal is to drive USDL towards standardization. SAP Research will take a lead role in the incubator group but will cease to be the sole owner of USDL.

6.2 Extensibility

USDL aims at conceptualizing the services domain to an extent that is sufficient to support a wide range of services. Nevertheless, covering all possible viewpoints on services and all conceivable usage scenarios in every domain is difficult, at least, and requires a wider audience. A generic service description language acting as a “one size fits all” for domains as diverse and complex as banking/financials, healthcare, manufacturing and supply chains is not sufficient. Not all aspects of USDL apply to all domains. USDL rather needs to be configured for the particular needs of applications where some concepts are removed or adapted while new and unforeseen ones are introduced. A particular consideration of this is allowing specialized, domain-specific classifications such as those available through vertical industry standards to be leveraged through USDL. In addition to this, the way in which USDL is applied for deployment considerations, e.g. the way lifecycle versioning of services and their descriptions applies, needs to be managed in a way that does not compromise the fundamental concepts of USDL.

This requirement of Extensibility (3.2.1.4) has not been tackled so far. Extensibility is a major topic that concerns the handling of deviations of USDL for specific industries, countries, or proprietary use cases. Special care has to be taken in order to prevent a proliferation of USDL derivations that require costly manual mappings much like has happened with the multitude of product and EDI standards. A possible solution is presented in [63], which consists of three pillars. First, a common and controlled vocabulary specified by means of the UN/CEFACT Core Component Specification (CCTS). Second, a context-logic that allows to define which elements of the language are valid in which contexts. Examples for contexts are industry, country, business process, or business role. The UN/CEFACT is currently working on a corresponding specification called Unified Context Methodology Technical Specification. Third, a USDL Schema Repository is required that implements the first two points and allows users to retrieve their corresponding rendition of USDL.

6.3 Comprehensibility

USDL is rather generic, i.e., it allows capturing different aspects of different kinds of services. The downside of the generality is that it leads to a complex model with a steep learning curve for potential users. Not only do the individual modules exhibit a complex structure but also understanding their rationale requires expertise in different disciplines. While a computer scientist might be able to "read" the UML class diagrams, a full understanding of the Pricing Module requires background in business economics. Likewise, a full understanding of the Legal Module requires basic knowledge of copyright law and licenses, etc.

This complexity is reflected in the current version of the USDL editor. The editor has been developed by model-driven engineering. It exhibits the full complexity of the individual modules. Therefore, users are typically overwhelmed by this complexity and Comprehensibility (cf. 3.2.1.5) becomes a major problem, and thus requirement.

There are first ideas that elaborate on deriving a natural language description from formal USDL descriptions [38]. However, this solution does not address the problem of creating a USDL description in the first place. The challenge is rather to counter the complexity of USDL not by sacrificing its generality but by finding an intuitive representation. There might be different kinds of intuitive representations for different modules of USDL. According to Strahringer [64], a (conceptual) language features a conceptual and a representational aspect. The conceptual aspect of USDL describes the constructs that the service description is built upon, i.e., their meaning and which relations exists among them. The representational aspect defines a diagrammatic notation that links graphical symbols to these constructs [65]. Apart from the pool of symbols and valid links among them, the representational aspect defines rules for the layout of the conceptual models. For example, in the language EPC (Event-driven Process Chains), an event is represented by a hexagon and a function is symbolized by a soft rectangle.

Such diagrammatic notations might be embedded in a modeling method that makes use of role-specific views. Role specific views to the services description might represent different competences in and requirements to the service description that demand alternative representations. An example for the application of views to information modeling is the ARIS reference architecture [66] that distinguishes a data, a function, an organization, and a processes view to modeling an information system.

7 References

[1] Berre, A. (2008): Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS). Revised Submission. OMG Document.

[2] Berre, A. (2008): UPMS - UML Profile and Metamodel for Services - an Emerging Standard. Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference.

[3] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F. & Metz, R. (2006) Reference Model for Service Oriented Architecture 1.0, Official OASIS Standard, October 2006, http://www.oasis-open.org/committees/soa-rm/

[4] Harding, C. (2008): Service-Oriented Architecture Ontology. The Open Group Draft 2.0, http://www.opengroup.org/projects/soa-ontology/

[5] Hadley, M. (2009) Web Application Description Language (WADL), W3C Member Submission, 31 August 2009, http://www.w3.org/Submission/wadl/

[6] Oberle, D., Lamparter, S., Grimm, S., Vrandecic, D., Staab, S., Gangemi, A. (2006): Towards ontologies for formalizing modularization and communication in large software systems. Applied Ontology 1(2): 163-202

[7] McIlraith, Sheila A., Son, T., Zeng, H (2001): Semantic Web Services. In: IEEE Intelligent Systems 16(2), p. 46-53.

[8] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H (2001): DAML-S: Semantic Markup for Web Services. In: SWWS 2001, p. 411-430.

[9] Roman, D., de Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C., Fensel, D., (2006): WWW: WSMO, WSML, and WSMX in a Nutshell. In: ASWC 2006, p. 516-522.

[10] Dobson, G. & Sánchez-Macián, A. (2006) Towards Unified QoS/SLA Ontologies. In: Proceedings of the IEEE Services Computing Workshops (SCW 2006), Third International Semantic and Dynamic Web Processes Workshop (SDWP 2006), pp. 169-174, IEEE Press.

[11] Farrell, J. & Lausen, H. (2007) Semantic Annotations for WSDL and XML Schema (SAWSDL), W3C Recommendation, August 2007, http://www.w3.org/TR/sawsdl/

[12] Gomadam, K., Ranabahu, A. & Sheth, A. (2010) SA-REST: Semantic Annotation of Web Resources, W3C Member Submission, April 2010, http://www.w3.org/Submission/SA-REST/

[13] Norton, B., Kerrigan, M., Mocan, A., Carenini, A., Cimpian, E., Haines, M., Scicluna, J. & Zaremba, M. (2008) Reference Ontology for Semantic Service Oriented Architectures. OASIS Public Review Draft 0.1, November 2008, http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/pr01/see-rosoa-v1.0-pr01.html

[14] Fensel, D., Fischer, F., Kopecký, J., Krummenacher, R., Lambert, D., Vitvar, T. (2010) WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web, W3C Member Submission 23 August 2010 http://www.w3.org/Submission/WSMO-Lite/

[15] Dietrich, B. (2006): "Resource planning for business services, Communications of the ACM, 49(7), ACM, 2006, pp. 62-64.

[16] Pandit, B., Popescu, V., Smith, V., (2009): SML Service Modeling Language. W3C Proposed Recommendation 12 February 2009. http://www.w3.org/TR/sml/

[17] Sun, W., Zhang, K., Chen, S., Zhang, X. & Liang, H. (2007) Software as a Service: An Integration Perspective. In: Proceedings of the Fifth International Conference on Service-Oriented Computing. LNCS 4749, Springer Berlin.

[18] Baida, Z., Gordijn, J., Akkermans, H., (2001): Service Ontology. 2001, Free University Amsterdam.

[19] Akkermans, H., Baida, Z., Gordijn, J., Peña, N., Altuna, A., Laresgoiti, I. (2004) Value Webs: Using Ontologies to Bundle Real-World Services. IEEE Intelligent Systems (EXPERT) 19(4):57-66
[20] De Kinderen, S., Gordijn, J. (2008) (a): e3Service - A model-based approach for generating needs-driven e-service bundles in a networked enterprise. In Proceedings of 16th European Conference on Information Systems, 2008.
[21] De Kinderen, S., Gordijn, J. (2008) (b): e3Service - An ontological approach for deriving multi-supplier IT-service bundles from consumer needs. In Ralph H. Sprague editor, Proceedings of the 41st Annual Hawaii International Conference on System Sciences, Pages 318, IEEE CS, 2008.
[22] Bitsaki, M.,; Danylevych, O., van den Heuvel, W., Koutras, G., Leymann, F.,; Mancioppi, M., Nikolaou, C., Papazoglou, M. (2008): An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network. In LNCS An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network. Springer Berlin / Heidelberg, Germany 2008
[23] Alter, S. (2008): Service system fundamentals: Work system, value chain, and life cycle. In: IBM Systems Journal 2008 47(1), p. 71-85.
[24] Estefan, J. A., Laskey, K., McCabe, F. G. & Thornton, D. (2009) Reference Architecture Foundation for Service Oriented Architecture. Version 1.0, Committee Draft 2, October 2009, OASIS Service Oriented Architecture Reference Model TC
[25] Dhanesha, K. A., Hartman, A. & Jain, A. N. (2009) A Model for Designing Generic Services. In: Proceedings of the Seventh IEEE International Conference on Services Computing, pp. 435-442, IEEE Press
[26] Ferrario, R., Guarino, N. (2008) Towards an Ontological Foundation for Services Science. In John Domingue, Dieter Fensel, Paolo Traverso (Eds.): FIS 2008, Lecture Notes in Computer Science 5468 Springer, Vienna, Austria, 2009, pp. 152-169.
[27] Oberle, D., Bhatti, N., Brockmans, S., Niemann, M., Janiesch, C. (2009). Countering Service Information Challenges in the Internet of Services, Journal of Business & Information System Engineering, 1(5), Gabler Verlag, 2009, pp. 370-390.
[28] DIN PAS 1018 (2002): Grundstruktur für die Beschreibung von Dienstleistungen in der Ausschreibungsphase. Deutsches Institut für Normung (DIN), "PAS 1018:2002-12", Beuth Verlag, Berlin, 2002.
[29] O'Sullivan, J. (2006): Towards a Precise Understanding of Service Properties. In: Faculty of Information Technology. 2006, Queensland University of Technology. p. 232.
[30] Emmrich, A. (2005): Ein Beitrag zur systematischen Entwicklung produktorientierter Dienstleistungen. University of Paderborn. Paderborn 2005
[31] Toma, I. (2010) Modeling and Ranking Semantic Web Services based on non-functional properties. PhD Thesis. Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck, 2010.
[32] Toma, I., Foxvog, D., De Paoli, F., Comerio, M., Palmonari, M. & Maurino, A. (2008) Non-functional properties in Web services. WSMO Working Draft D28.4v0.2. April 2008, http://www.wsmo.org/TR/d28/d28.4/v0.2/
[33] Falkenberg, E., Hesse, W., Lindgreen, P., Nilsson, B.E., J.L.H. Oei, Rolland, C., Stamper, R. K., Van Assche, F.J.M. Verrijn-Stuart, A.A. Voss, K. (1998). FRISCO - A Framework of Information System Concepts – The FRISCO Report. IFIP WG 8.1 Task Group FRISCO.
[34] Alistair P. Barros, Arthur H. M. ter Hofstede (1998): Towards the construction of workflow-suitable conceptual modelling techniques. Information System Journal 8(4): 313-
[35] Bunge, M. (1977): Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World, Reidel, Bosten, Ma.
[36] Van Griethuysen, JJ. (ed.) (1982): ISO/TC97/SC5/WG3 Concepts and Terminology for the Conceptual Schema and the Information Base
[37] Heller, M., Allgaier, M. (2010): Model-based Service Integration for Extensible Enterprise Systems with Adaptation Patterns. In David A. Marca, Boris Shishkov, Marten van Sinderen

(Eds.): ICE-B 2010 - Proceedings of the International Conference on e-Business, Athens, Greece, July 26 - 28, 2010, ICE-B is part of ICETE - The International Joint Conference on e-Business and Telecommunications. SciTePress 2010, ISBN 978-989-8425-17-1, pages 163-168

[38] Heinrich, M., Boehm-Peters, A., Knechtel, M. (2009): A platform to automatically generate and incorporate documents into an ontology-based content repository. In Uwe M. Borghoff, Boris Chidlovskii (Eds.): Proceedings of the 2009 ACM Symposium on Document Engineering, Munich, Germany, September 16-18, 2009. ACM 2009, ISBN 978-1-60558-575-8, pages 43-46

[39] Oaks, P., Hofstede, A.H.M., Edmond, D. (2003): Capabilities: describing what services can do. In Proceedings of Service-Oriented Computing – ICSOC 2003, LNCS 2910, Springer, 2003, pp. 1-16.

[40] Homann, U. (2006). A Business-Oriented Foundation for Service Orientation. MSDN, Microsoft Corporation, 2006, http://msdn.microsoft.com/en-us/library/aa479368.aspx.

[41] Stern, A.A. (1986). The Strategic Value of Price Structure. Journal of Business Strategy, 7(2), MCB UP Ltd, 1986, pp. 22-31.

[42] Faruqui, A., Wood L. (2008). Quantifying the Benefits of Dynamic Pricing in the Mass Market, Edison Electric Institute (EEI), 2008.

[43] Hogan, J. & Nagle, T. (2006). Segmented pricing using price fences to segment markets and capture value. SPG Insights, Strategic Pricing Group, 2006.

[44] Hogan, J. & Nagle, T. (2001). The Strategy and Tactics of Pricing: A Guide to Profitable Decision Making, Prentice Hall, 2001.

[45] Shapiro, C. & Varian, H.R. (1998). Versioning: the smart way to sell information, Harvard Business Review, 1998.

[46] Lovelock, C. & Gummesson, E. (2004). Whither Services Marketing? In Search of a New Paradigm and Fresh Perspectives. Journal of Service Research, 7(1), SAGE, 2004, pp. 20-41.

[47] Baumann, C. & Loës, C. (2010): Formalizing Copyright for the Internet of Services. In: Kotsis, Gabriele; Taniar, David; Pardede, Eric; Saleh, Imad; Khalil, Ismail (eds.): The 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS2010). November 8 - 10, 2010, Paris, France. Wien Österreich: Österreichische Computer Gesellschaft (OCG) (books@ocg.at, 272), pages 712–719.

[48] Baumann, C., Peitz, P., Raabe, O., Wacker, R. (2010): Compliance for Service Based Systems Through Formalization of Law. In: Filipe, Joaquim; Cordeiro, José (eds.): Proceedings of the 6th International Conference on Web Information Systems and Technology. Valencia, Spain: INSTICC Press (2), pages 367–371.

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

[50] Kelkar, O., Leukel, J. & Schmitz, V. (2002) Price modeling in standards for electronic product catalogs based on XML. In: Proceedings of the 11th World Wide Web Conference (WWW), pp. 366-375.

[51] Kiemes, T.; Oberle, D. & Novelli (2010). Towards a reusable and executable Pricing Model in the Internet of Services. In F. Kotsis, G.; Taniar, D.; Pardede, E.; Saleh, I. & Khalil, I. (Eds.) Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS2010), ACM Press, 2010

[52] de Miranda, J. G. B. & Baida, Z. (2006) Modelling pricing for configuring e-service bundles. In: BLED 2006 Proceedings.

[53] Speiser, S. (2009) Semantic Usage Policies for Web Services. In: Proceedings of the 8th International Semantic Web Conference, pp. 982-989.

[54] Gangadharan, G. R., D'Andrea, V., Iannella, R. & Weiss, M. (2007) ODRL Service Licensing Profile (ODRL-S). In: Proceedings of the 5th Intl. Workshop for Technical, Economic, and Legal Aspects of Business Models for Virtual Goods, pp. 1–17.

[55] Kopecky, J., Gomadam, K. & Vitvar, T. (2008): hRESTS: an HTML Microformat for Describing RESTful Web Services. In: The 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI2008), November, IEEE CS Press, Sydney, Australia
[56] Kopecky, J., Vitvar, T., Pedrinaci, C., Maleshkova, M. (2011). RESTful Services with Lightweight Machine-readable Descriptions and Semantic Annotations. In Wilde, Erik and Pautasso, Cesare (eds.) REST: From Research to Practice, Springer.
[57] Pedrinaci, C.; Domingue, J.: Toward the Next Wave of Services: Linked Services for the Web of Data. J. UCS 16(13): 1694-1719 (2010)
[58] Lehmann, S. and Buxmann, P. (2009) Pricing Strategies of Software Vendors. Business & Information Systems Engineering, 1(6), 452-462.
[59] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S. & Xu, M. (2007) Web Services Agreement Specification (WS-Agreement). Open Grid Forum, http://www.ogf.org/documents/GFD.107.pdf
[60] Theilmann, W., Happe, J., Kotsokalis, C., Edmonds, A., Kearney, K. & Lambea, J. (2010) A Reference Architecture for Multi-Level SLA Management. Journal of Internet Engineering, to appear
[61] Business Process Model and Notation, V1.1, OMG Available Specification, January 2008, http://www.omg.org/spec/BPMN/1.1/PDF
[62] Nickolas K., David B., Gregory R., Tony F., Yves L., Charlton B. (2005). Web Services Choreography Description Language Version 1.0, W3C Candidate Recommendation, 9 November 2005. http://www.w3.org/TR/ws-cdl-10/
[63] Stuhec, G. (2007). Using CCTS Modeler Warp 10 to Customize Business Information Interfaces. SAP Developer Network article, 22. November 2007, http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/70d6c441-507e-2a10-7994-88f6f769d6e8
[64] Strahringer, S. (1996). Metamodellierung als Instrument des Methodenvergleichs: Eine Evaluierung am Beispiel objektorientierter Analysemethoden, Shaker, Aachen.
[65] Guizzardi, G., Pires, L.F., and van Sinderen, M.J. (2002). On the role of domain ontologies in the design of domain-specific visual modeling languages. In Proceedings of the 2nd Workshop on Domain-Specific Visual Languages at the 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002), J.-P. Tolvanen, J. Gray, and M. Rossi (eds.), Seattle, WA, 2002, pp. 1-14.
[66] Scheer, A.-W. (2000). ARIS: Business Process Modeling, (3rd Ed.), Springer, Heidelberg.
[67] Buxmann, P., Hess, T. & Ruggaber, R. (2009) Internet der Dienste. In: Wirtschaftsinfomatik 5 (2009), pp. 341,342