

Ontologies and Reasoning in Enterprise Service Ecosystems

Daniel Oberle

Introduction

The use of *ontologies* to enable, facilitate, and improve different aspects of enterprise systems is alluring, since they are a means for modeling a domain of interest in a flexible and formal way [9]. This in turn opens up automatic *reasoning* possibilities to infer implicit knowledge or to automate specific business tasks.

So far, there has been academic interest and early industrial products, but there are two inhibitors that prevent the widespread usage of ontologies in enterprise systems. First, the actual value proposition of ontologies in enterprise systems remains vague [11, 30]. That means, decision makers often do not have a clear picture of *why* and *when* to apply ontologies in enterprise systems. Second, many challenges have to be overcome when using ontologies in enterprise systems. As examples for such challenges, consider the integration of the new technology in existing IT landscapes or the lack of ontology experts. Hence, decision makers often decide against ontologies and in favor of more established technologies.

As a response, we provide a rationale for how ontologies can benefit enterprise systems. The rationale provides a qualitative argumentation line for how the technological features of ontologies can be fruitfully combined to achieve business benefits. We carefully identify the challenges of using ontologies in enterprise systems and present two approaches to tackle (i) the challenge of technical integration of ontologies in existing enterprise systems and (ii) the challenge of training existing developers to familiarize them with the new technology. The rationale and the two approaches are summarized in Sect. “Use of Ontologies in Enterprise Systems.”

The subsequent section gives a concrete example of the inhibitors and use case for the second approach. Our research brought forth both a pragmatic and an ontological scheme for *service description for enterprise ecosystems*. Both schemes allow the comprehensive description of services in order to enable communication and trade between ecosystem participants. However, the strengths of the pragmatic scheme are the weaknesses of the ontological scheme since the pragmatic scheme relies on established technologies. As a response, our second approach can be utilized to facilitate the design of both service description schemes.

The following section provides an example for the run-time usage of both service description schemes and the first approach. Our research brought forth a novel way for *engineering legally compliant services*. This contribution relies on a formalized subject matter provided by the two aforementioned service description schemes each contributing unique benefits required for a solution. The coexistence of both schemes is facilitated by our first approach to tackle the challenge of technical integration.

Use of Ontologies in Enterprise Systems

Enterprise systems are designed to integrate computer systems that run all phases of an enterprise's operations to facilitate cooperation and coordination of work across the enterprise. The intent is to

DOI 10.1007/s00287-014-0785-5
© Springer-Verlag Berlin Heidelberg 2014

Daniel Oberle
SAP Research, Vincenz-Priessnitz-Str. 1,
76131 Karlsruhe
E-Mail: d.oberle@sap.com

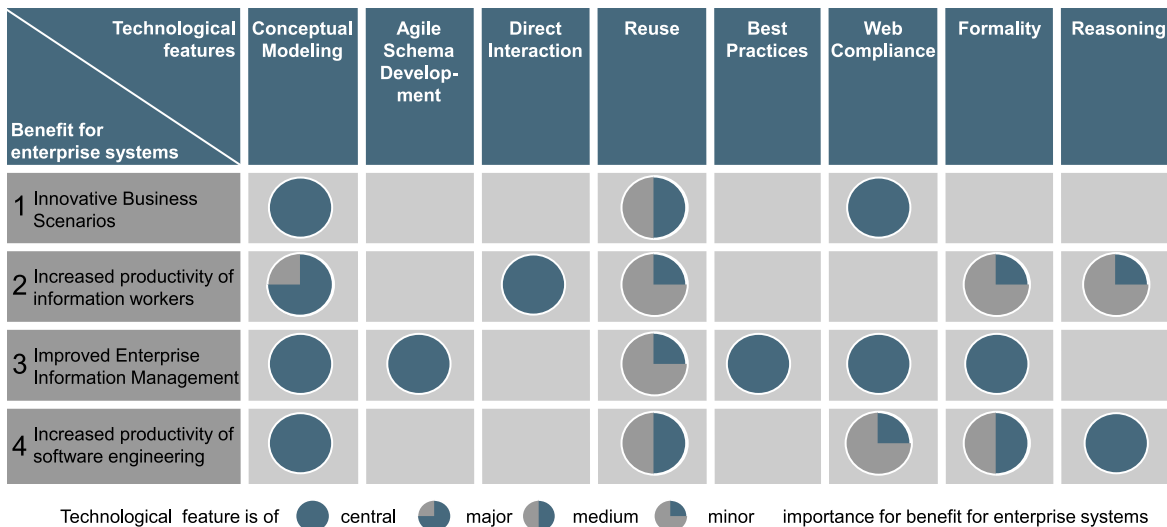


Fig. 1 Technological, value-adding features of ontologies & reasoning lead to benefits for enterprise systems (generalized version from [17, Fig. 3])

integrate core business processes (e. g., sales, accounting, finance, human resources, inventory, and manufacturing) [10].

An argumentation line for *why* and *when* to apply ontologies can foster their widespread usage in enterprise systems. Such a rationale is further discussed in Sect. “Rationale” and the main subject of our contribution in [17].

Besides, many challenges are in the way of using ontologies in enterprise systems. The often complex boundary conditions in an enterprise setting are an example for such challenges. Therefore, our research focussed on tackling some of the challenges (cf. Sect. “Tackling the Challenges”). The first approach tackles the challenge of having to train developers to enable them to work seamlessly with ontologies in their familiar environment [26]. The second approach tackles the challenge of technically integrating the new technology into existing enterprise systems [23].

Rationale

Ontologies and reasoning inherit from several precursory technologies, e. g., logics, theorem provers, deductive databases, AI, or semantic nets. Therefore, many of the technological features are not unique but are also offered by related technologies. In [17, Sect. 3], we identify the following technological features of ontologies (depicted as columns in Fig. 1):

Conceptual modeling: for capturing a universe of discourse with intuitive modeling primitives (classes, relations, instances).

Agile schema development: for evolving an ontology at run time both on the instance and schema level.

Direct interaction: for creating, changing, and populating ontologies by generic tools.

Reuse: for sharing the knowledge in an ontology between users.

Best practices: for improving design via foundational ontologies, design patterns, and quality criteria.

Web compliance: for publishing (RDFS [4], OWL [15]), querying (SPARQL [24]), and annotating (SA-WSDL [5], SA-REST [8], RDFa [1]) ontological information in the Web.

Formality: for having a sound basis of the modeling language.

Reasoning: for subsumption and consistency checking as well as instance classification and retrieval by inference engines.

The technological features of ontologies and reasoning can be applied to benefit enterprise systems as shown by the rows in Fig. 1. The figure represents a generalization, i. e., several pilots, prototypes, and products were analyzed to learn which technological features are *typically* applied to achieve benefits for enterprise systems:

Innovative business scenarios: The technological feature of *Web compliance* allows information to be published and queried world-wide. A high degree of *reuse* lets the ontology act as a globally distributed schema and database. This, paired with the feature of *conceptual modeling*, opens up hitherto impossible business scenarios (a concrete example is given further below).

Increased productivity of information workers: The starting point for obtaining this benefit is the integration of all relevant information in a way that can easily be consumed by the information worker (*conceptual modeling*). The second crucial technological feature is *direct interaction*, i. e., the user-friendly way of viewing and editing. Most examples in this category require *reuse* since a group of information workers typically cooperates. Some examples also exploit *reasoning*, and consequently rely on the feature of *formality*.

Improved enterprise information management: *Conceptual modeling* is applied here for an easy-to-understand view of enterprise information. To ensure sustainable modeling, building the conceptual model with *best practices* can help to capture enterprise information with particular high quality. Also, modeling a domain once and its enterprise wide *reuse* is one of the sought-after goals. *Agile schema development* allows to access and evolve the ontology over time according to the enterprise's representation needs. *Web compliance* avoids information silos and enables integration with information on the Web. Finally, *formality* can help to counter ambiguities when modeling enterprise information.

Increased productivity of software engineering: *Conceptual modeling* of a relevant domain is the prerequisite for obtaining this benefit. *Formality* and *reasoning* help to automate specific task in the realm of software engineering. In some cases, *reuse* is required and *Web compliance* can help to annotate existing resources, e. g., Web service descriptions.

As a concrete example for achieving the benefit of innovative business scenarios, consider the British Broadcasting Corporation (BBC). The BBC publishes large amounts of content online, such as text, audio, and video about programmes or music. The BBC Programme ontology is used to provide

machine-readable representations enabling richer applications on top of the BBC's data. The system gives the BBC the flexibility and a maintainability benefit: the web site becomes the BBC's API. The ontological representations allow third-party developers to use the BBC's data to build and monetize new applications [14].

The BBC Programme ontology is an intuitive conceptual model allowing third-party developers to easily understand the BBC's data. This technological feature of *conceptual modeling* paired with the feature *Web compliance* is of central importance here. Ontological content descriptions are published and interlinked across the BBC's websites according to W3C Semantic Web recommendations such as RDFS or OWL. In addition, the technological feature of *reuse* is of importance since the BBC Programme ontology extends the already existing Music Ontology and FOAF vocabulary.

Tackling the Challenges

The use of ontologies in enterprise systems is also hindered by challenges which we identify in [17]. The challenges can be aligned on an axis that moves from technology-agnostic to ontology-specific (cf. Fig. 2). The first challenge concerns the often tedious *technical integration*. In the case of the BBC example above, ontology mappings as well as an ontology store have to be integrated with the existing landscape of BBC's enterprise system.

The second challenge concerns different use cases that might require different mappings, stores, and reasoners, however. In this case, the challenge of technical integration might have to be faced n times.

As a consequence, the third challenge is to arrive at a positive overall *cost-benefit ratio*. The hope is that modeling a domain once in an ontology will suffice for all use cases. However, different use cases typically have different representation needs, i. e., the *modeling depends on the use case* (challenge 4). Correspondingly, the BBC Programme ontology might not suffice for the representation needs of all new applications envisioned by third-party developers.

Since one hardly finds metrics or scientific methods to prove the advantages of ontologies, *how to measure the benefits?* represents the fifth challenge. Finally, the *training* of existing employees or the acquisition of ontology experts is difficult for every new technology. BBC first needed to ac-

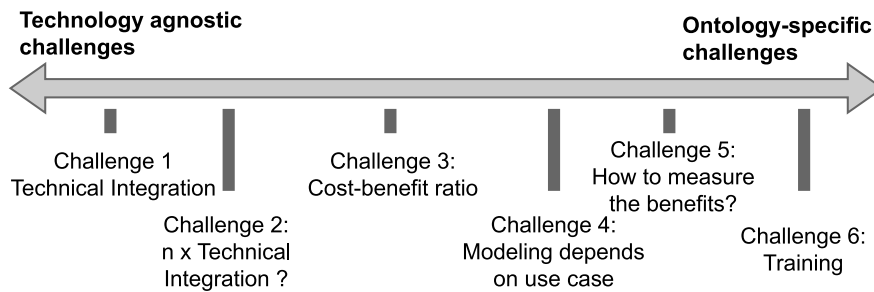


Fig. 2 Overview of challenges aligned on an axis that moves from technology agnostic to ontology-specific challenges (derived from [17, Fig. 4])

quire ontology expertise either by training existing developers or by hiring new ones.

In the following we discuss our approaches to counter two of the challenges. Our first approach counters the challenge of *training* by enabling software engineers to develop enterprise systems on the basis of an ontology in their familiar environment [26]. The second addresses the challenge of *technical integration* by mapping between pragmatic class models and ontologies [23]. Recommendations of how to deal with the remaining challenges are discussed in [17, Sect. 7].

Tackling “Training:” An Adjustable Transformation from OWL to Ecore. In the BBC example, developers have to map media content residing in BBC’s established enterprise systems to ontological representations and publish the latter on Web sites. In an ideal case, the developer is supported in the authoring of and programmatic access to the BBC Programme ontology by his or her familiar development environment (e. g., Eclipse). The development environment should make working with ontologies as seamless as possible, especially if the developer is unfamiliar with ontologies.

At first sight, the task of incorporating ontologies seems straightforward, since the main modeling primitives (classes and relations) in common software engineering languages, such as UML or its Eclipse-specific variant *Ecore* [29], and ontology languages, such as W3C OWL [15], are similar. At second sight, there are less similarities between both languages, which makes the transformation intricate. For example, Ecore relies on the unique name assumption unlike OWL. In addition, this lack of similarities opens configuration options for transforming OWL ontologies.

Therefore, we contribute a unidirectional transformation from OWL to the modeling language Ecore at design time (cf., [26]). The transformation

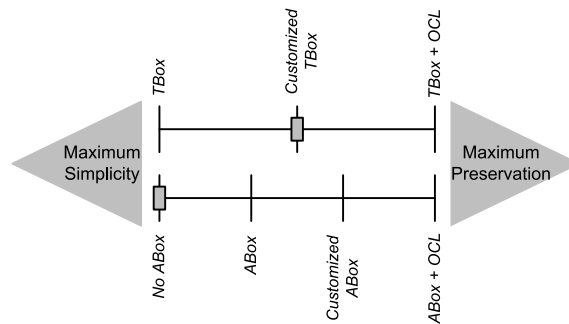


Fig. 3 Slider metaphor for the adjustable transformation from OWL to Ecore [26, Fig. 3]

allows developers to “import” an existing ontology, such as the BBC Programme Ontology, into Eclipse. Once imported, the developer can utilize all of the familiar Ecore plug-ins to manage the conceptual model.

Because of the configuration options, we allow the developer to adjust the transformation between the two extremes of a result simple to understand or a result that preserves as much as possible of the source ontology. Hence, the realization of our transformation features two “sliders” as depicted in Fig. 3.

The upper slider concerns the TBox of the source ontology which basically contains classes, relations, and axioms. The minimal transformation is to capture only classes and relations (position *TBox*). The position *customized TBox* allows the handling of equivalent classes in OWL and the creation of helper classes to represent them. At position *TBox + OCL*, the transformation tries to preserve as much of the ontology as possible by expressing OWL axioms in the Object Constraint Language of UML/Ecore.

The lower slider concerns the ABox which contains concrete instances of classes and relations. It is possible to disregard the ABox completely (position *No ABox*), to transform instances in general

(ABox), to handle also equivalent instances (*customized ABox*), and to preserve as much as possible ($ABox + OCL$).

Tackling “Technical Integration:” Mapping Pragmatic Class Models to Reference Ontologies. The challenge of *technical integration* concerns the handling of software components such as ontology stores or mappings in existing and established enterprise systems. Ontologies often play the role of a reference model, i. e., a generic, commonly agreed upon conceptual model of a domain. To this end, modeling decisions are taken in a way such that the intended meaning of domain terms is captured as precisely as possible. One goal of reference ontologies is to facilitate and enable information integration across enterprise systems. Indeed, the BBC Programme ontology acts as a reference model to facilitate information exchange between BBC and third-party enterprise systems to enable new applications.

However, existing enterprise systems on the side of the third party are often based on database schemas or class models that deviate from the reference ontology although they potentially capture the same domain. The reason is that class models are task-specific, with the focus on an efficient implementation of an application. In contrast to reference ontologies, modeling decisions are geared towards a pragmatic and efficient model. Due to those differences, one often faces the situation where class models and reference ontologies are incompatible in the sense that a 1:1 mapping between them does not exist.

Since the reference ontology is typically designed *ex post*, i. e., long after enterprise systems have been put in place, the enterprise systems’ pragmatic class models have to be mapped to such a reference ontology. Mapping between the BBC Programme ontology and a pragmatic class model in the enterprise system of a third party is typically a non trivial task which requires a flexible mechanism to cope with all kinds of disparities between the two kinds of models. Further, the mapping has to be bidirectional, since the third party’s enterprise system has to send and receive messages adhering to the BBC Programme ontology. In addition, the mapping process itself must happen at run time and on the instance level for coping efficiently with the disparities between reference ontology and class

model. Finally, the established enterprise systems cannot be touched in most cases. That means, the mapping mechanism has to be realized in a non intrusive way.

In [23], we contribute a novel approach to facilitate the technical integration by mapping between reference ontologies and class models. As required for integrating existing enterprise systems, the approach is non intrusive, i. e., it can be implemented without having to change the class model. The approach is flexible, i. e., it does not rely on 1:1 mappings between the class model and the reference ontology.

Instead of using static mappings between the class model and the ontology, our approach uses declarative mapping instructions. For each class of the pragmatic model, mapping instructions are written that explain the classes (or other constructs) of the class model in terms of the reference ontology. Formalizing the mapping instructions is performed at design time and by an expert. The mapping instructions are processed by a mapping execution engine. For each enterprise system, two separate mapping execution engines are established: one for mapping from the class model to the reference ontology, and one for mapping from the reference ontology to the class model.

Service Description for Enterprise Ecosystems

The following section gives a concrete example of the inhibitors and use case for our adjustable transformation from OWL to Ecore. The example concerns *enterprise service ecosystems* where several parties have a share in delivering a service. A good example is third-party logistics services: in order to transport a good, several carrier services might be combined with customs clearing and warehousing. Each individual carrier, the customs clearance, and warehousing might be different companies with different service offerings. These constitute a service ecosystem where it is the task of a third-party logistics provider (3PL) to combine offerings to a service bundle meeting the specific needs of a customer.

This situation requires a shared and comprehensive description of services in order to enable information exchange between ecosystem participants. The service description has to take into account business, operational, and technical aspects of a service. For example, the description needs to

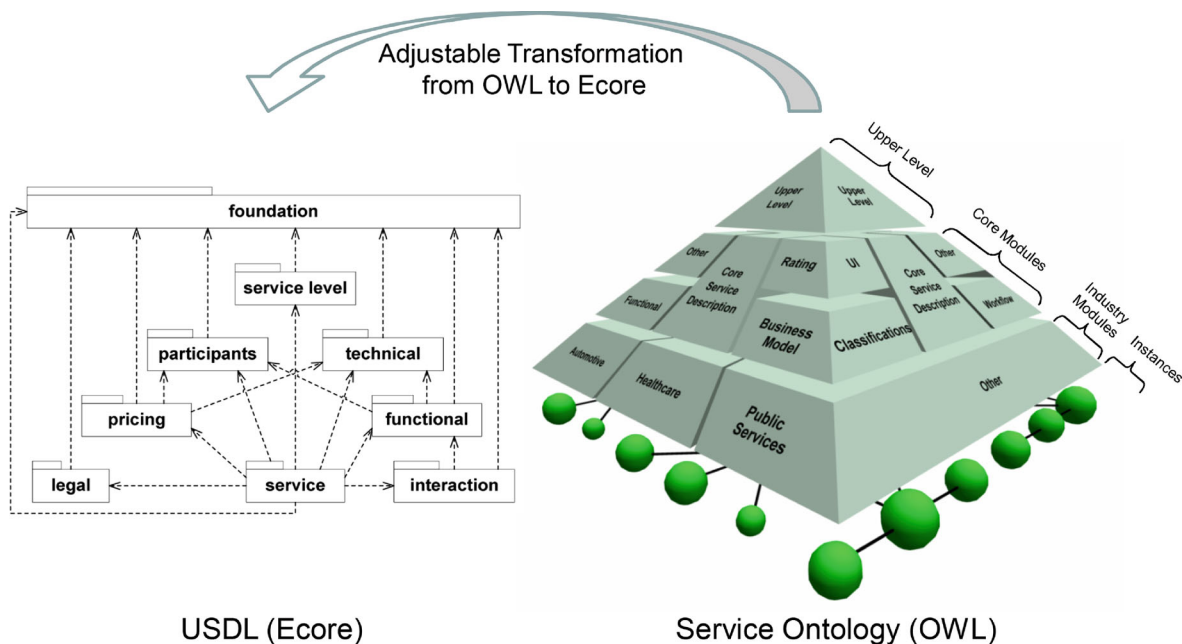


Fig. 4 The design of USDL [18, Fig. 5] and Service Ontology [19, Fig. 2] is facilitated by the adjustable transformation from OWL to Ecore

capture the price plan, licensing issues, functionality, service level agreements, or the interaction protocol. The description should be (i) normative for all participants, (ii) formal in order to be machine processable, and (iii) serializable in XML for use in the Web. This allows ecosystem participants to exchange relevant information uniformly between their corresponding enterprise systems, and, thus, lower the integration costs.

As a response, our research approached *service description for enterprise ecosystems* both as a reference ontology and as a pragmatic class model. The first scheme is the *Service Ontology* (cf. [19]) and the second scheme is the *Unified Service Description Language (USDL)* (cf. [18]).

The coexistence of both schemes is a direct consequence of the inhibitors of ontologies and reasoning. The Service Ontology leverages technological features of OWL and is afflicted with several challenges of adopting the new technology. Vice versa, the strengths of USDL are the weaknesses of the Service Ontology since USDL relies on the more established Ecore language. Our *adjustable transformation from OWL to Ecore* facilitates the design of both service description schemes: aspects modeled in OWL can be imported into Eclipse. This avoids manual remodeling of every aspect in USDL from scratch.

The Service Ontology

Our first scheme for a comprehensive service description is the *Service Ontology*. The Service Ontology allows each provider to describe different facets of their services as instances of predefined classes and relations. Thus, the third-party logistics provider could rely on concise ontological descriptions of carrier, custom clearance, and warehousing service to create a desired bundle.

The Service Ontology is depicted by a pyramid on the right hand side in Fig. 4. The pyramid is a metaphor for the number of classes and relations that increases from top to bottom. The Service Ontology is specified in OWL and consists of several modules. The modules are depicted as parts of the pyramid and can be divided into four layers.

The *Upper Level* consists of a concise foundational ontology which (i) is used as a modeling starting point because it provides a basic set of generic classes and relations valid in any domain. Using a foundational ontology as a modeling basis means relating core classes and relations to some proposed invariant categories of human cognition. What is typically gained is an increased understanding of one's own ontology as well as a cleaner design. (ii) The foundational ontology helps defining general ontology design patterns as best practices for recurring modeling needs. All this represents one

reason for choosing an ontology to capture a service description (i. e., the technological feature of *best practices*). This level introduces fundamental classes, such as Object or Event, a whole taxonomy of specializations, as well as basic relations, such as part-of.

Several *Core Modules* are built around the Core Service Description module, which captures information common to every service (e. g., information about the service provider, quality of service, etc.). In addition, different aspects of a service description (legal, business model, technical, rating, UI, etc.) are placed in separate modules and linked to the classes belonging to the Core Service Description module. So far, several core modules have been designed to a mature state and published in diverse literature:

Core Service Description Module: Introduces necessary information common to every service and independent of a specific industry or aspect, e. g., classes such as Service, Service Provider, Service Action, etc., further discussed in [7].

Pricing Module: Allows to capture the complete Price Plan of a service with its Price Components and Price Levels (cf., [12, 13]).

Legal Module: Formalizes service licenses according to German or US copyright law (discussed in [3]).

Classification Module: Ontological representations of existing classification schemes, such as UN-SPSC or eClass [16, Sect. 10].

Documentation Module: Covers organizational knowledge including, e. g., knowledge about which service is described in which document [16, Sect. 11].

Idea Module: Its task is to represent information about the service engineering phase as discussed in [27].

Rating Module: Captures ratings of customers after consumption of a service [16, Sect. 9].

Industry Modules (e. g., logistics, automotive, health-care, or public services modules) can be modeled by exploiting the aforementioned ontology modules. The core knowledge specified in the Core Service Description module and adjacent aspect-related core modules can be specialized for specific industries. In our example, a logistics module would introduce Logistic-Service as a subclass of Service of the Core Service Description module. Logistic-Service

could be further specialized in Road Transport, Airfreight, etc.

Finally, concrete logistics providers are able to describe their services as *instances* of the classes and relations of the modules at run time. For example, UPS would instantiate the class Road Transport of the logistics industry module together with a concrete price plan, license description, and so on. The instances are depicted as a mesh below the pyramid and can potentially be distributed across the Web according to the principles of Linked Data. The latter represents a second reason why an ontology was chosen for capturing a service description (i. e., the technological feature *Web compliance*).

The Unified Service Description Language (USDL)

Although the Service Ontology provides benefits by leveraging the technological features *best practices* and *Web compliance*, the challenges of *training* and *technical integration* prevent a successful and widespread usage.

Therefore, our second scheme for a comprehensive service description, viz., *USDL*, was created in parallel. The schema of USDL is modeled with the more common Eclipse Modeling Framework and its UML dialect called *Ecore* [29]. Thus, it relies on well established tooling and is accessible to the vast majority of software engineers. USDL's structure is depicted on the left side of Fig. 4 and basically covers the Core Modules of the Service Ontology, where each aspect (legal, pricing etc.) has a dedicated UML package. In essence, each UML package consists of one comprehensive class model with associations to class models in other packages.

As a consequence, several aspects were modeled twice at design time: once with ontological analysis applying best practices and once as a pragmatic class model. In order to avoid this double effort and facilitate the design of both schemes, our *adjustable transformation from OWL to Ecore* can be used. Thus, the knowledge can be captured and maintained in the more expressive Service Ontology and transformed to USDL in widespread software engineering environments when required.

Engineering Legally Compliant Services

The following section gives an example for the usage of the Service Ontology and USDL at run time. The

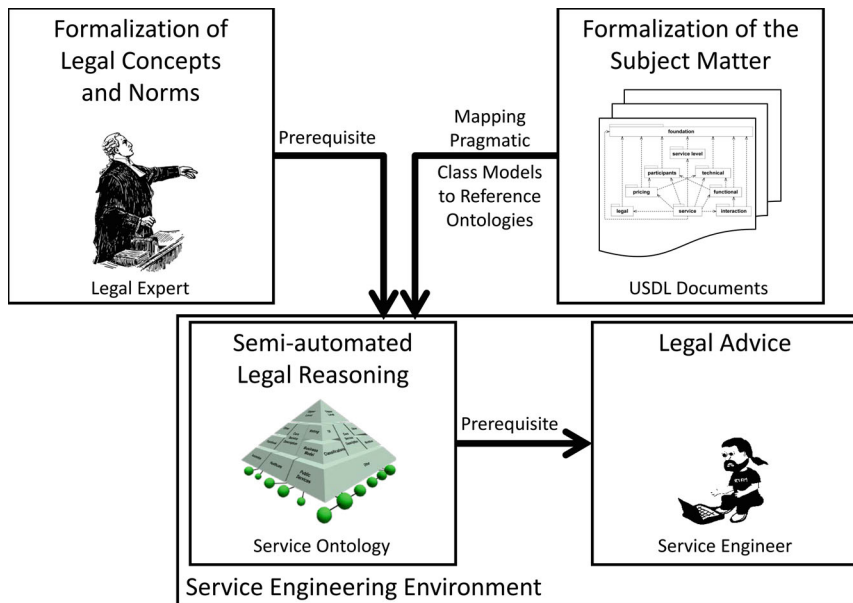


Fig. 5 Overview of the framework for engineering compliant services adapted from [21, Fig. 2]. The coexistence of USDL and the Service Ontology is facilitated by our approach for mapping pragmatic class models to reference ontologies

example concerns service ecosystems in the Web, where service value chains can be created ad hoc, especially with fully automated, software-based services. As an example, consider the Google Maps and Facebook Web services. A third party can mash up both services to provide a value-adding ‘person locator app’ to track and display the location of Facebook friends.

The flexibility and volatility of such dynamically created service value chains is powerful, yet it also creates new threats. In particular, the prevailing paradigm of checking the legal compliance of the individual services *before* they are ever used is no longer possible. The unanticipated combination of Google Maps and Facebook services by a third party might indeed violate data privacy laws. For example, the German Federal Data Privacy Act (FDPA) prohibits the transfer of personal data (potentially the Facebook ID to Google) by a third party without consent by the user in most cases.

As a reaction to this paradigm shift, we contribute a solution for enabling service engineers to ensure legal compliance by design in [21]. We claim that a technology-supported framework must be established to convey and manifest legal requirements in software-based services as early as possible. The framework semi-automatically advises service engineers – who are usually legal laymen – about legal reasoning. This support is integrated in the

development environment of the service engineer (e. g., [28]). An overview of the framework is given in Fig. 5.

The framework relies on a formalized subject matter provided by USDL descriptions. Yet, an ontological representation of the subject matter is required to enable legal reasoning. Consequently, our approach for *mapping pragmatic class models to reference ontologies* is required to facilitate the coexistence of both service description schemes.

Formalization of Legal Concepts and Norms

The prerequisite for our framework is the formalization of legal concepts and legal norms of a given statutory provision, e. g., the FDPA, in an ontology (cf. also Fig. 5). The task of formalizing legal concepts benefits from the technological features *conceptual modeling* and *best practices* which help to capture the intended meaning of legal concepts. As an example, consider the legal concept of Personal Data in the FDPA below. A legal expert has to capture its main properties, viz., PersonalData subclass of Information, concerns Circumstances and about Natural Person.

Section 3 FDPA: Further definitions

(1) “Personal data” shall mean any information concerning ... circumstances about a ... natural person.

The task of formalizing legal norms leverages the technological features of *formality* and *reasoning*.¹ The structure of a legal norm often takes the form of an “if ... then” rule as demonstrated by Sect. 4 FDPA below. Ontology languages usually allow to represent and reason with such rules.

Section 4 FDPA: Lawfulness of data collection, processing and use

(1) The collection, processing and use of personal data shall be lawful only if permitted or ordered by this act ... or if the data subject has provided consent.

Consequently, the formalized legal concepts (written in sans serif font) can be assembled by logical operators (AND, OR, \rightarrow) resulting in the formalized legal norm below. This allows to capture the rule-like nature of a legal norm in the same language as the formalized legal concepts.

```
((Collection(X) OR Processing(X) OR Use(X))
AND performedUpon(X, Y) AND Personal Data(Y))
AND
(Permission(P) OR Order(P)) AND givenFor(P, X))
OR
(Consent(C) AND Data Subject(D) AND about(Y, D)
AND gives(D, C) AND permits(C, X))
 $\rightarrow$ 
Lawfulness(L) AND givenFor(L, X)
```

Formalization of the Subject Matter

Another prerequisite for the framework is the formalization of the subject matter (cf. also Fig. 5). In legal terms, the subject matter is the real-world situation which is under consideration. In our case, the subject matter can be formalized by our service description schemes.

It is expected that USDL becomes the lingua franca for service descriptions in the Web, since it is a pragmatic approach relying on a well-established and widespread modeling language and infrastructure. Therefore, the main source for a formalized subject matter will be USDL documents. Yet, the subject matter must also be represented in an ontology language to facilitate the legal reasoning process described in the subsequent section. Therefore, our framework internally relies on ser-

vice descriptions given by means of the Service Ontology.

The coexistence of USDL and the Service Ontology is facilitated by our approach for *mapping pragmatic class models to reference ontologies* as depicted in Fig. 5. It allows to flexibly represent USDL documents as instances of the Service Ontology at run time.

Semi-automated Legal Reasoning

Legal reasoning is a complex intellectual process, so far only executable manually by a legal expert and involves basically two steps. In the first step, the legal expert identifies the legal consequence he or she wishes to attain. In our example, this would be the legal consequence of “lawfulness” of the abovementioned transfer of Facebook IDs. This determines a limited set of norms from which to start, namely Sect. 4 FDPA in our case. In turn, the legal expert has to look for further norms that influence the desired legal consequence. Thus, the legal expert mentally constructs a *norm graph* that enables him or her to decide whether the originally intended legal consequence can be reached or not.

This step can be automated by an inference engine based on the formalized legal norms. In our example, the legal consequence of “lawfulness” would represent the primary norm in our graph. Note that further formalized legal norms are required to infer whether lawfulness is given. For example, Consent requires another set of formalized legal norms that specify whether the consent is effective according to Sect. 4a (1) FDPA and so on.

The second step requires the legal expert to legally subsume an actual subject matter under the legal concepts of the norm graph. The legal expert tries to translate and align elements of the subject matter to legal concepts. As an example, the legal expert has to answer whether Facebook IDs represent personal data as defined in Sect. 3 (1) FDPA. This legal subsumption process cannot be fully automated since it requires human interpretation of the individual subject matter. A Facebook ID can be personal data if it allows to identify the corresponding natural person with some effort. This may vary depending on the actual Facebook ID and requires the legal expert to check accompanying legal documents or earlier decisions of legislation.

¹ A norm is typically represented by a paragraph, section, or sentence in a provision.

Legal Advice

In the prevailing paradigm, the third party, which is about to mash up the Google Maps and Facebook services, would have to consult a legal expert to perform the legal reasoning steps. This is both time and cost intensive. Hence, the challenge is to enable the service engineer – who usually is a legal layman – to perform the legal reasoning steps.

Our approach is to automate the legal subsumption process as much as possible. For example, the legal concept of Age can be subsumed automatically since it does not require further interpretation. For all other legal concepts, a wizard guides the service engineer by giving explanations and displaying accompanying legal documents, such that a layman is enabled to make the bulk of the decisions. If the service engineer is not capable of making such a decision, there still is the possibility to consult a legal expert.

If the legal subsumption process can be completed and the desired legal consequence is not attained, the service engineer is advised about how to react. For example, the user of the ‘person locator app’ must be informed about the pending transfer of his Facebook ID and must provide explicit consent.

Conclusion and Outlook

In enterprise settings, decision makers often have a hard time deciding in favor of ontologies as opposed to more established technologies. Therefore, our rationale for *why* and *when* to apply ontologies can foster the widespread usage of ontologies in enterprise systems. In addition, our research contributes an *adjustable transformation from OWL to Ecore* as well as an approach for *mapping pragmatic class models to reference ontologies*. Both help to tackle challenges when adopting ontologies in enterprise systems. We have demonstrated the usefulness of both approaches for the Service Ontology and USDL as well as for the task of engineering compliant services. All of them represent scientific contributions in their own right.

Appendix

Personal Contributions

This article represents a summary of my cumulative habilitation treatise consisting of [17–19, 21, 23, 26]. As can be seen in the bibliography, several coauthors have contributed to the individual publications as

follows: Although I have been the initiator and driver behind [26], the work was executed in-depth by Tirdad Rahmani, who has been a researcher at that time and in turn supervised the student Marco Dahms. Our publication in [23] represents a major part of Heiko Paulheim’s dissertation (cf., [22]). However, I have been responsible for further developing, generalizing, and positioning the approach in the larger context of software engineering.

USDL required more than a dozen man years by an interdisciplinary team of colleagues at different SAP Research locations. I have been responsible for USDL from May 2010 to December 2011. Besides acting as product manager, my main contribution during this time span was to scientifically position USDL resulting in [18] and [2]. The efforts and overall design of the Service Ontology have been conceived and coordinated by me as documented in [19] (an English translation of the German *Wirtschaftsinformatik* journal [20]). However, similar to USDL, several colleagues contributed several modules to the ontology, e. g., [3, 27]. I have initiated and driven further development of the central ontology module (cf. [7] – authors are listed in alphabetical order) based on Ferrario’s and Guarino’s seminal work in [6]. In addition, I have been contributing to the Pricing Module [12] by supervising master student Tom Kiemes.

The original ideas of *engineering legally compliant services* stem from Dr. Oliver Raabe at the KIT. Contributions were worked out in close cooperation with Richard Wacker, Christian Baumann, and Christian Funk, who were PhD-students at that time. Oliver Raabe acted as lead and supervisor for all aspects of legal science. I acted as lead and supervisor for all aspects concerning computer science. Our main contribution in [21] is a peer-reviewed fragment of [25].

Acknowledgements

I would like to express my gratitude to all my coauthors, supervisors, as well as the reviewers of this document: Alistar Barros, Christian Baumann, Nadeem Bhatti, Saartje Brockmans, Marco Dahms, Felix Drefs, Jürgen Ebert, Roberta Ferrario, Andreas Friesen, Christian Funk, Nicola Guarino, Steffen Heinzl, Pascal Hitzler, Christian Janiesch, Tom Kiemes, Uwe Kylau, Jens Lemcke, Michael Niemann, Heiko Paulheim, Roland Plendl, Florian Probst, Oliver Raabe, Tirdad Rahmani, Sebastian Rudolph,

Ansgar Scherp, Steffen Staab, Rudi Studer, York Sure, Susan Marie Thomas, and Richard Wacker.

Funding

The work presented here was funded by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS/Texto project as well as the Smarter Privacy project.

References

- Adida B, Birbeck M, McCarron S, Pemberton S (2008) RDFa in XHTML: Syntax and processing. Recommendation, W3C, <http://www.w3.org/TR/rdfa-syntax/>, 22.8.2013
- Barros A, Oberle D (eds) (2012) Handbook of Service Description: USDL and its Methods. Springer, New York
- Baumann C, Loës C (2010) Formalizing copyright for the internet of services. In: Kotsis G, Taniar D, Pardede E, Saleh I, Khalil Ibrahim I (eds) iiWAS'2010 – The 12th International Conference on Information Integration and Web-based Applications and Services, 8–10 November 2010, Paris, France, ACM, pp 714–721
- Brickley D, Guha RV (2004) RDF vocabulary description language 1.0: RDF Schema. Recommendation, W3C, <http://www.w3.org/TR/rdf-schema/>, 25.2.2014
- Farrell J, Lausen H (2007) Semantic annotations for WSDL and XML Schema. Recommendation, W3C, <http://www.w3.org/TR/sawsdsl/>, 28.8.2007
- Ferrario R, Guarino N (2008) Towards an ontological foundation for services science. In: Domingue J, Fensel D, Traverso P (eds) Future Internet – FIS 2008, First Future Internet Symposium, FIS 2008, Vienna, Austria, September 29–30, 2008, Revised Selected Papers, vol 5468 of Lecture Notes in Computer Science, Springer, pp 152–169
- Ferrario R, Guarino N, Janiesch C, Kiemes T, Oberle D, Probst F (2011) Towards an ontological foundation of services science: the general service model. In: Bernstein A, Schwabe G (eds) 10th International Conference on Wirtschaftsinformatik, 16–18 February 2011, Zurich, Switzerland, vol 2, pp 675–684, [Lulu.com](http://www.lulu.com)
- Gomadam K, Ranabahu A, Sheth A (2010) SA-REST: Semantic annotation of web resources. Member submission, W3C, <http://www.w3.org/Submission/SA-REST/>, 5.4.2010
- Guarino N, Oberle D, Staab S (2009) What is an ontology? In: Staab S, Studer R (eds) Handbook on Ontologies, Handbooks on Information Systems, 2nd edn. Springer, pp 1–17
- Gartner Inc. (2004) The Gartner glossary of information technology acronyms and terms
- Inmon B (2007) The semantics mystery. BeyeNETWORK, Article no. 4605, <http://www.b-eye-network.com/print/4605>, 7.6.2007
- Kiemes T, Oberle D (2010) Generic modeling and management of price plans in the internet of services. In: Fähnrich K-P, Franczyk B (eds) Informatik 2010: Service Science – Neue Perspektiven für die Informatik, Beiträge der 40. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Band 1, 27.9.–1.10.2010, Leipzig, LNI, vol 175, GI, pp 533–538
- Kiemes T, Oberle D, Novelli F (2010) Towards a reusable and executable pricing model in the internet of services. In: Kotsis G, Taniar D, Pardede E, Saleh I, Khalil Ibrahim I (eds) iiWAS'2010 – The 12th International Conference on Information Integration and Web-based Applications and Services, 8–10 November 2010, Paris, France, ACM, pp 722–729
- Kobilarov G, Scott T, Raimond Y, Oliver S, Sizemore C, Smethurst M, Bizer C, Lee R (2009) Media meets semantic web – how the BBC uses DBpedia and linked data to make connections. In: Aroyo L, Traverso P, Ciravegna F, Cimiano P, Heath T, Hyvönen E, Mizoguchi R, Oren E, Sabou M, Paslaru Bontas Simperl E (eds) The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, 31 May–4 June 2009, Proceedings, Lecture Notes in Computer Science, vol 5554. Springer, pp 723–737
- McGuinness DL, van Harmelen F (2004) OWL Web Ontology Language Overview. Recommendation, W3C, <http://www.w3.org/TR/owl-features/>, 10.2.2014
- Oberle D (2010) Service ontology final report. Deliverable D.TEXO.9.3.2b, BMWi, Theseus Programme, Use Case Texo
- Oberle D (2013) How ontologies benefit enterprise applications. Semantic Web Journal, DOI: 10.3233/SW-130114, <http://iospress.metapress.com/content/k16n012507037044/>
- Oberle D, Barros A, Kylau U, Heinzl S (2013) A unified description language for human to automated services. Information Systems 38(1):155–181
- Oberle D, Bhatti N, Brockmans S, Niemann M, Janiesch C (2009) Countering service information challenges in the internet of services. Business & Information Systems Engineering 1(5):370–390
- Oberle D, Bhatti N, Brockmans S, Niemann M, and Janiesch C (2009) Effiziente Handhabung von Service Informationen im Internet der Dienste. Wirtschaftsinformatik 5:429–452
- Oberle D, Drefs F, Wacker R, Baumann C, Raabe O (2012) Engineering compliant software: advising developers by automating legal reasoning. SCRIPTed 9(2):280–313
- Paulheim H (2011) Ontology-based Application Integration. Springer
- Paulheim H, Oberle D, Plendl R, Probst F (2019) An architecture for information exchange based on reference models. In: Sloane AM, Aßmann U (eds) Software Language Engineering – 4th International Conference, SLE 2011, Braga, Portugal, 3–4 July 2011, Revised Selected Papers, Lecture Notes in Computer Science, vol 6940. Springer, pp 160–179
- Prud'hommeaux E, Seaborne A (2008) SPARQL query language for RDF. Recommendation, W3C, <http://www.w3.org/TR/rdf-sparql-query/>, 15.1.2008
- Raabe O, Wacker R, Oberle D, Baumann C, Funk C (2012) Recht ex machina: Formalisierung des Rechts im Internet der Dienste. Springer, Heidelberg
- Rahmani T, Oberle D, Dahms M (2010) An adjustable transformation from OWL to Ecore. In: Petriu DC, Rouquette N, Haugen Ø (eds) Model Driven Engineering Languages and Systems – 13th International Conference, MODELS 2010, Oslo, Norway, 3–8 October 2010, Proceedings, Part II, Lecture Notes in Computer Science, vol 6395. Springer, pp 243–257
- Riedl C, May N, Finzen J, Stathel S, Kaufman V, Krcmar H (2009) An idea ontology for innovation management. Int J Semantic Web Inf Syst 5(4):1–18
- Scheithauer G, Voigt K, Bicer V, Heinrich M, Strunk A, Winkler M (2009) Integrated service engineering workbench: service engineering for digital ecosystems. In: Chbeir R, Badr Y, Kapetanios E, Traina AJM (eds) MEDES '09: International ACM Conference on Management of Emergent Digital EcoSystems, Lyon, France, 27–30 October 2009. ACM, pp 446–449
- Steinberg D, Budinsky F, Paternostro M, Merks E (2009) EMF: Eclipse Modeling Framework, 2nd edn. Addison-Wesley
- White A (2009) Semantic web moving ever close to the 'semantic enterprise'? Gartner Blog Network, http://blogs.gartner.com/andrew_white/2009/04/30, 30.4.2009