Publications Office
of the European Union

Initiative for developing eProcurement Ontology

eProcurement ontology architecture and formalisation specifications

Disclaimer

This report was prepared for the Publications Office of the European Union by Infeurope.

Document metadata

Reference	WP 1.1: eProcurement ontology architecture and formalisation specifications
Corporate Author	Publications Office of the European Union
Author	Eugeniu Costetchi
Reviewers	Natalie Muric, Ioannis Rousochatzakis, George Vernardos
Contractor	Infeurope S.A.
Framework contract	10688/35368
Work package	WP 1.1
Delivery date	17 April 2020
Suggested readers	project partners, future users, legal practitioners, software developers and architects

Abstract

Public procurement is undergoing a digital transformation. The EU supports the rethinking of the public procurement process with digital technologies in mind. This goes beyond simply moving to electronic tools; it rethinks various pre-award and post-award phases. The aim is to make them simpler for businesses to participate in and for the public sector to manage. It also allows for the integration of data-based approaches at various stages of the procurement process.

With digital tools, public spending should become more transparent, evidence-oriented, optimised, streamlined and integrated with market conditions. This puts eProcurement at the heart of other changes introduced to public procurement in new EU directives.

Given the increasing importance of data standards for eProcurement, a number of initiatives driven by the public sector, the industry and academia have been kick started in the recent years. Some have grown organically, while others are the result of standardisation work.

In this context, the Publications Office of the European Union aims to develop an eProcurement ontology.

The objective of the eProcurement ontology is to act as this common standard on the conceptual level, based on consensus of the main stakeholders and designed to encompass the major requirements of the eProcurement process in conformance with the Directives and Regulations.

This document provides a working definition of what is the architectural stance and the design decisions that shall be adopted for the eProcurement formal ontology along with the specifications how to generate comprising components.

Contents

1 Introduction

This document provides a working definition of what is the architectural stance and the design decisions that shall be adopted for the eProcurement formal ontology along with the specifications how to generate comprising components.

1.1 Background considerations

Public procurement is undergoing a digital transformation. The EU supports the process of rethinking public procurement process with digital technologies in mind. This goes beyond simply moving to electronic tools; it rethinks various pre-award and post-award phases. The aim is to make them simpler for businesses to participate in and for the public sector to manage. It also allows for the integration of data-based approaches at various stages of the procurement process.

Digital procurement is deeply linked to eGovernment. It is one of the key drivers toward the implementation of the "once-only principle" in public administrations – a cornerstone of the EU's Digital Single Market strategy. In the age of big data, digital procurement is also crucial in enabling governments to make data-driven decisions about public spending.

With digital tools, public spending should become more transparent, evidence-oriented, optimised, streamlined and integrated with market conditions. This puts eProcurement at the heart of other changes introduced to public procurement in new EU directives.

PSI directive [27] across the EU is calling for open, unobstructed access to public data in order to improve transparency and to boost innovation via the reuse of public data. Procurement data has been identified as data with a high-reuse potential [3]. Therefore, making this data available in machine-readable formats, following the data as a service paradigm, is required in order to maximise its reuse.

Given the increasing importance of data standards for eProcurement, a number of initiatives driven by the public sector, the industry and academia have been kick started in the recent years. Some have grown organically, while others are the result of standardisation work. The vocabularies and the semantics that they are introducing, the phases of public procurement that they are covering, and the technologies that they are using all differ. These differences hamper data interoperability and thus its reuse by them or by the wider public. This creates the need for a common

data standard for publishing public procurement data, hence allowing data from different sources to be easily accessed and linked, and consequently reused.

In this context, the Publications Office of the European Union aims to develop an eProcurement ontology.

The objective of the eProcurement ontology is to act as this common standard on the conceptual level, based on consensus of the main stakeholders and designed to encompass the major requirements of the eProcurement process in conformance with the Directives and Regulations [28, 29, 26, 30].

1.2 Target audience

The target audience of the eProcurement ontology, defined in [48], comprises the following groups of stakeholders:

- Contracting authorities and entities, i.e. buyers, such as public administrations in the EU Member States or EU institutions;
- Economic operators, i.e. suppliers of goods and services such as businesses, entrepreneurs and financial institutions;
- Academia and researchers;
- Media and journalists;
- Auditors and regulators;
- Members of parliaments at regional, national and EU level;
- Standardisation organisations;
- NGOs; and
- Citizens [48].

1.3 Context and scope

In the past years much effort was invested into the eProcurement ontology initiative, including definition of requirements, provision of general specifications, identification of the main use cases, and laborious development of a preliminary shared conceptual model expressed using Unified Modelling Language (UML) [8, 13].

The general methodology for developing the eProcurement ontology is described in [24, 3–15]. It describes a process comprising the following steps:

1. Define use cases
2. Define the requirements for the use cases
3. Develop a conceptual data model
4. Consider reusing existing ontologies
5. Define and implement an OWL ontology

The ultimate objective of the eProcurement ontology project is to put forth a commonly agreed OWL ontology that will conceptualise, formally encode and make available in an open, structured and machine-readable format data about public procurement, covering end-to-end procurement, i.e. from notification, through tendering to awarding, ordering, invoicing and payment [48].

Work so far has concentrated on the conceptual modelling of the eNotification phase, taking into consideration the needs of other phases. The UML conceptual model has been created with the forthcoming procurement standard forms (eForms) in mind; the model has not been mapped to the current standard forms.

In the 2020 ISA^2 work programme a new project has been set up to analyse existing procurement data through the lens of the newly developed conceptual model. This means that the conceptual model needs to be transposed into a formal ontology and a subset of the existing eProcurement data must be transformed into RDF format such that they instantiate the eProcurement ontology and are conform to a set of predefined data shapes. Initially the notification phase is considered, while the subsequent datasets will be decided at a later stage.

Working under the assumption that Steps 1–4 have been completed, the current efforts channel on designing, implementing and executing the necessary tasks in order to accomplish Step 5 from the above process.

Once the formal ontology is created and the XML data is transformed into RDF representation, the data can be queried in order to validate the suitability to satisfy the business use cases defined in [24, Sec. 3].

This document comprises of architectural specification and implementation guidelines that shall be taken into consideration when developing the formal ontology.

Other related artefacts (i.e. documents, scripts and datasets) are presented in Section 3, where it is described, in detail, the process for accomplishing the generation the formal eProcurement ontology, transformation of XML data and the ontology validation.

There is a number of aspects that are excluded from the scope of this project stage:

- Change management and maintenance of the ontology content.
- Content authoring and conceptual design of the domain model.
- Practical implementation of systems that implement the ontology.

Currently in scope are the following items:

- designing an ontology architecture (this document),
- create guidelines and conventions for the UML conceptual model [14],
- develop a set of transformation scripts from the UML model into a formal ontology
- implement a set of scripts to transform the existing XML eProcurement data into RDF format,
- put forward a method to validate the generated formal ontology using the current eProcurement data.

1.4 Key words for requirement statements

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [9].

The key words “MUST (BUT WE KNOW YOU WON’T)”, “SHOULD CONSIDER”, “REALLY SHOULD NOT”, “OUGHT TO”, “WOULD PROBABLY”, “MAY WISH TO”, “COULD”, “POSSIBLE”, and “MIGHT” in this document are to be interpreted as described in RFC 6919 [59].

The above listed terms are used in lower case form for stylistic and readability reasons.

1.5 Conformance

This document describes normative and non-normative criteria for eProcurement ontology components and artefacts. The scripts, datasets, and the derived formal ontology and data shapes, must align to the normative criteria and may follow the non-normative descriptions.

The XSLT stylesheets [41] must be syntactically valid documents and executable with an XSLT engine with predictable output. They may be associated with XSPEC unit tests [12] to ensure correctness.

The source code must be syntactically valid and compilable/interpretable by the corresponding state of the art compiler/interpreter. The source code may be accompanied by the unit tests to ensure the implementation correctness.

The UML conceptual model must comply with UML standard version 2.5 [13] and be serialised as XMI document version 2.5.1 [1]. It also must comply with the conventions agreed with the Publications Office and other stakeholders described in [14].

The core ontology and the formal restrictions components developed under these specifications must be valid OWL 2 documents in conformance with the conditions listed in [45]. They should be available in at least Turtle and RDF/XML serialisation formats.

The data shapes component must be valid SHACL documents respecting the normative parts of the specification provided in [43].

The instance datasets must be valid RDF1.1 documents conform to the specifications [10].

The URIs adopted under this specification must respect the policy provided in Section 6.

2 Requirements

In Section 1 the context of the eProcurement project was presented along with explanations as to why the ontology is being built, what its intended uses are, who the end users are. This section elaborates on the general design criteria along with requirements the ontology should fulfil.

2.1 Functional requirements

This section provides the main functionalities and use cases that the ontology should support. These requirements are derived from the use cases identified in the report on policy support for eProcurement [48] and outlined in the in the eProcurement project chapter proposal [23].

1. *Transparency and monitoring*: to enable verification that public procurement is conducted according to the rules set by the Directives and Regulation [28, 29, 26, 30].
   (a) Public understandability
   (b) Data Journalism
   (c) Monitor the money flow
   (d) Detect fraud and compliance with procurement criteria
   (e) Audit procurement process
   (f) Cross-validate data from different parts of the procurement process
2. *Innovation & value added services*: to allow the emergence of new applications and services on the basis of the availability of procurement data.
   (a) Automated matchmaking of procured services and products with businesses
   (b) Automated validation of procurement criteria
   (c) Alerting services
   (d) Data analytics on public procurement data
3. *Interconnection of public procurement systems*: to support increased interoperability across procurement systems.
   (a) Increase cross-domain interoperability among Member States
   (b) Introduce automated classification systems in public procurement systems

2.2 Non-functional requirements

This section provides the characteristics, qualities and general aspects that the eProcurement ontology should satisfy.

- The practices, technologies and standards must be aligned with the European Directive on open data and the reuse of public sector information [32], the single digital gateway regulation [31], and European Publications Office standards and practices.
- The terminology used in the ontology should be reused from established core vocabularies [57] and domain ontologies as long as their meaning fits into the description of the eProcurement domain.
- The concept and relation labels must allow for multilingual content, covering at least the official European Languages [56].
- The formal ontology, and the related artefacts, must be generated from the eProcurement UML conceptual model, serving as the single source of truth, through a set of predefined transformation rules [15].
- The content of the ontology must be consistent with the predefined set of UML conceptual model conventions [14].
- The ontology identifiers must follow a strict URI policy defined in Section 6.
- The ontology design must commit long term URI persistence.
- The ontology, and the related artefacts, must be layered in order to support different degrees of ontological commitment and levels formal specification stacked on each other (see Section 4.2).
- The ontology, and the related artefacts, must be sliced in order to support a modular organisation of the domain in terms of self contained or semi-dependent modules (see Section 4.2).

2.3 General design criteria

For the purpose of knowledge sharing and interoperation between programs based on a shared conceptualisation, Gruber [37] proposes a set of preliminary *design criteria* a formal ontology should follow:

- *Clarity.* An ontology should communicate the purpose and meaning of defined terms. Definitions should be objective and independent of social and computational context even if the underlying motivations arise from them. Formalism is the means to this end, and when possible, the logical formulation should be provided.
- *Coherence.* Ontology should permit inferences that are consistent with the definitions. At the least, the defining axioms should be logically consistent. Coherence should also apply to the concepts that are defined informally, such as those described in natural language documentation and examples.
- *Extensibility.* Ontology should be designed to anticipate the uses of the shared vocabulary. It should offer a conceptual foundation for a range of anticipated tasks and the representation should be crafted so that one can extend and specialise the ontology monotonically. This feature supports and encourages reuse and further specialisations of ontologies and creation of the application profiles.
- *Minimal encoding bias.* The conceptualisation should be specified at the knowledge level without depending on a particular symbol-level encoding.
- *Minimal ontological commitment.* Ontology should require the minimal ontological commitment sufficient to support the intended knowledge sharing activities. Ontology should make as few claims as possible about the world being modelled, allowing the parties committed to the ontology freedom specialise and instantiate the ontology as needed. An ontological commitment is an agreement to use the shared vocabulary, with which queries and assertions are exchanged between agents, in a coherent and consistent manner. We say that an agent commits to an ontology if its behaviour is consistent with the definitions in the ontology [37].

3 Process and methodology

The main effort of the current stage of the project is to develop a formal ontology. This corresponds to Step 5 of the process described in [24, 3–15] and repeated in Section 1.3.

This section expands and addresses in detail the process of defining and implementing an OWL eProcurement ontology. The underlying assumption is that the

conceptual data model developed at Step 3 serves as an input for the creation of the ontology, and that this process shall be automatic.

In addition to producing the ontology as an artefact, we also need to validates its fitness to represent existing data and test whether the functional and non-functional requirements are respected. Figure 1 depicts the sequence of steps as a BPMN process diagram [65].

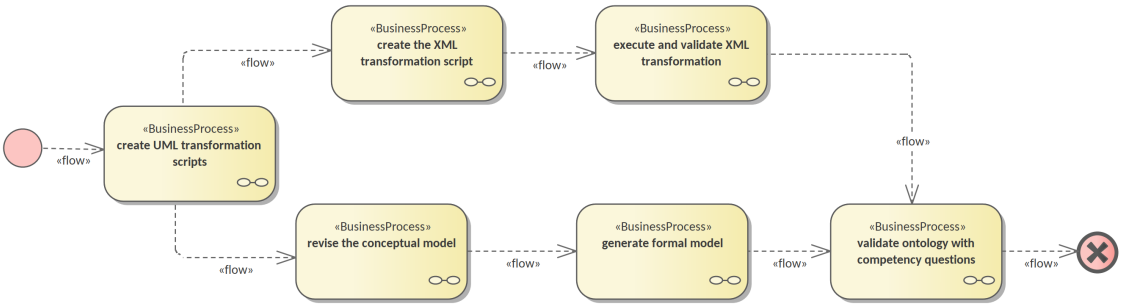


Figure 1: The main steps to implementing and validating the formal eProcurement ontology

The conceptual model serves as the single source of truth, the process starts with development of a series of transformations scripts. The conceptual model needs to be adjusted in order to fit a set of UML modelling conventions [14] making it suitable input for the transformation scripts. Provided that the conceptual model conforms, the transformation can be executed. Finally, the validation of the formal ontology can be performed using the existing eProcurement data.

The existing eProcurement data needs to be transformed from XML into RDF format. So, in parallel, after the UML transformations are created and along with them, the ontology architecture and UML conventions, then a set of XML transformation scripts can be developed. Once they are ready, they need to be executed on previously selected datasets, to convert them into RDF data instantiating the formal ontology. Only then, when the datasets are available, the ontology can be validated.

The next subsections describe each of these six steps in more detail in order to provide rationale and introduce each artefact in part.

3.1 UML transformation scripts generation

The process starts with authoring two documents laying the foundations of the entire process: the ontology architecture and the UML modelling conventions [14]. The main purpose of the ontology architecture (this document) specifications is to describe why the ontology is being built, what its intended uses are, who the end-users are, and which requirements the ontology should fulfil. Moreover, it states how the ontology should be structured in order to facilitate maintenance and usage patterns.

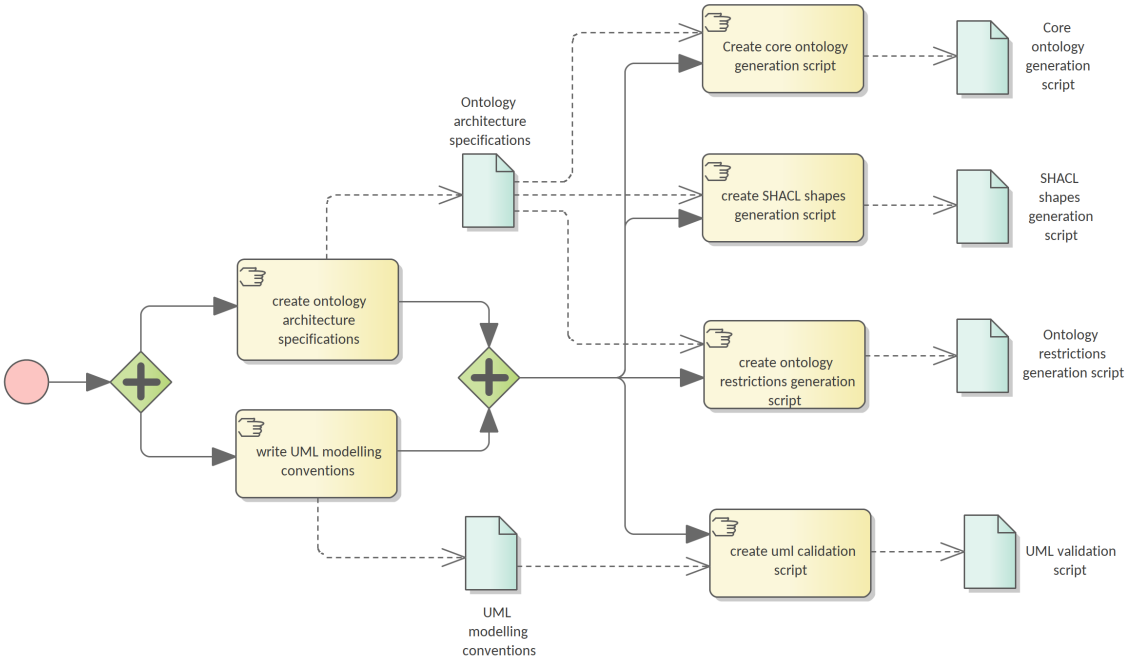


Figure 2: Creation of the specifications documents and the UML transformation scripts

The conceptual model must comply with a set of UML modelling conventions making it suitable input for the transformation scripts, which implement the same conventions. The two parallel actions starting the process are depicted in Figure 2.

The UML conventions document serves, at large, as the requirements specification for the XSLT script that checks whether the UML conceptual model conforms to the conventions.

The ontology architecture specification (this document) serves, at large, as requirement specifications for the development of three XSLT scripts to generate the formal ontology. These scripts can be developed independent of each other as they refer to different aspects of the formal ontology as described in Section 4.1.

The input for these scripts is the UML conceptual model, authored using Enterprise Architect, and serialised in XMI 2.5.1 format [1].

3.2 Conceptual model revision

The current project runs under the assumption that the conceptual model is organised and expressed in accordance to the UML conventions specified in [14]. However the conceptual model was developed well before the UML conventions were established. Therefore the model needs adjustments to conform to the conventions.

Having this assumption violated is a risk with critical impact. Therefore, an auxiliary process was developed (see Figure 3) to continuously improve and correct the model in case it is non-conformant.

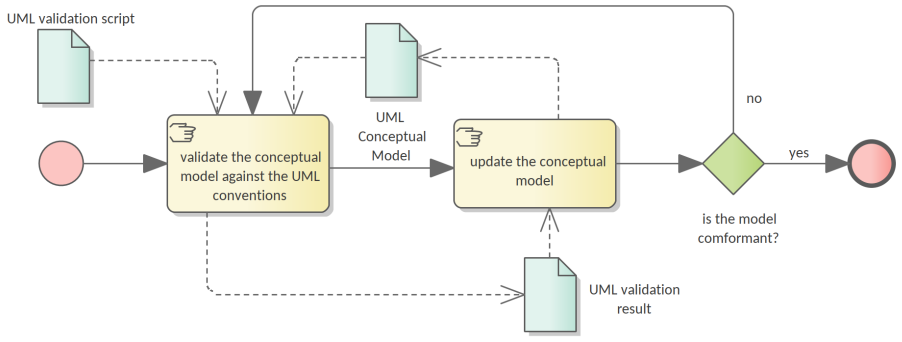


Figure 3: Adjustment of the UML conceptual model guided by the validation script

The conceptual model revision is an iterative process. A validation script is developed in a preceding step (see Section 3.1). This script it is executed on the current conceptual model outputting a report. This report is comprised of errors and warnings with detailed description of what is the deviation from the UML conventions. It also provides hints for the conceptual model designer what are the necessary actions to resolve the issues.

3.3 Formal ontology generation

The UML conceptual model constitutes the sole main input for three transformations scripts. Therefore, it is very important to ensure that the conceptual model is conform to the set of UML conventions [14]. They ensure that the conceptual model represents an adequate input for the transformations script as they are developed based on the same set of conventions and assumptions.

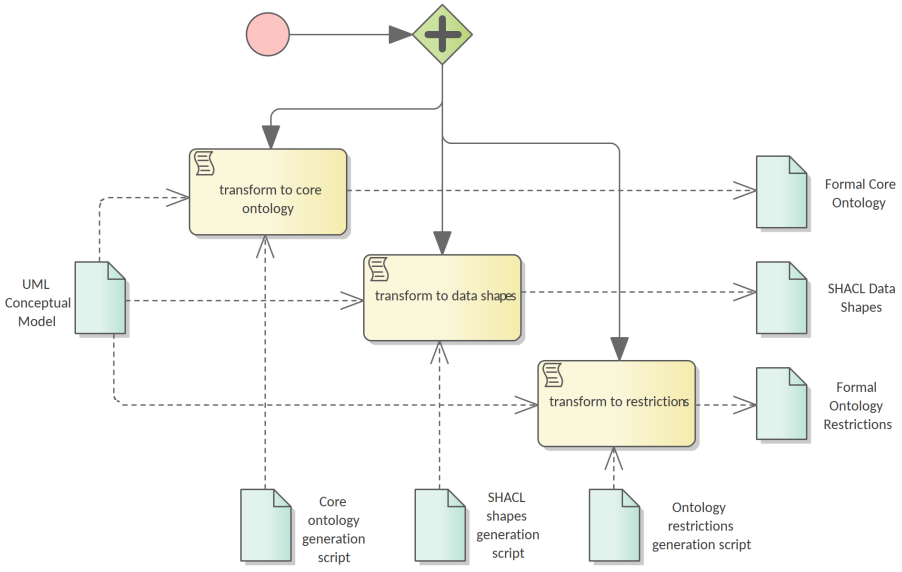


Figure 4: Generation of the formal ontology from the UML conceptual model

The transformation scripts, developed in a preceding step described in Section 3.1, are: the core ontology transformations script, the SHACL shapes transformation script, and the ontology restrictions generation script. Each are executed on the current conceptual model resulting in three output artefacts, or three sets of outputs artefacts, that depends on the implementation decisions. Each output corresponds to one of the components of the eProcurement ontology addressed in Section 4.2: the formal core ontology, the SHACL data shapes, and the formal ontology restrictions. And this concludes the ontology generation process depicted in Figure 4.

3.4 XML to RDF data transformation

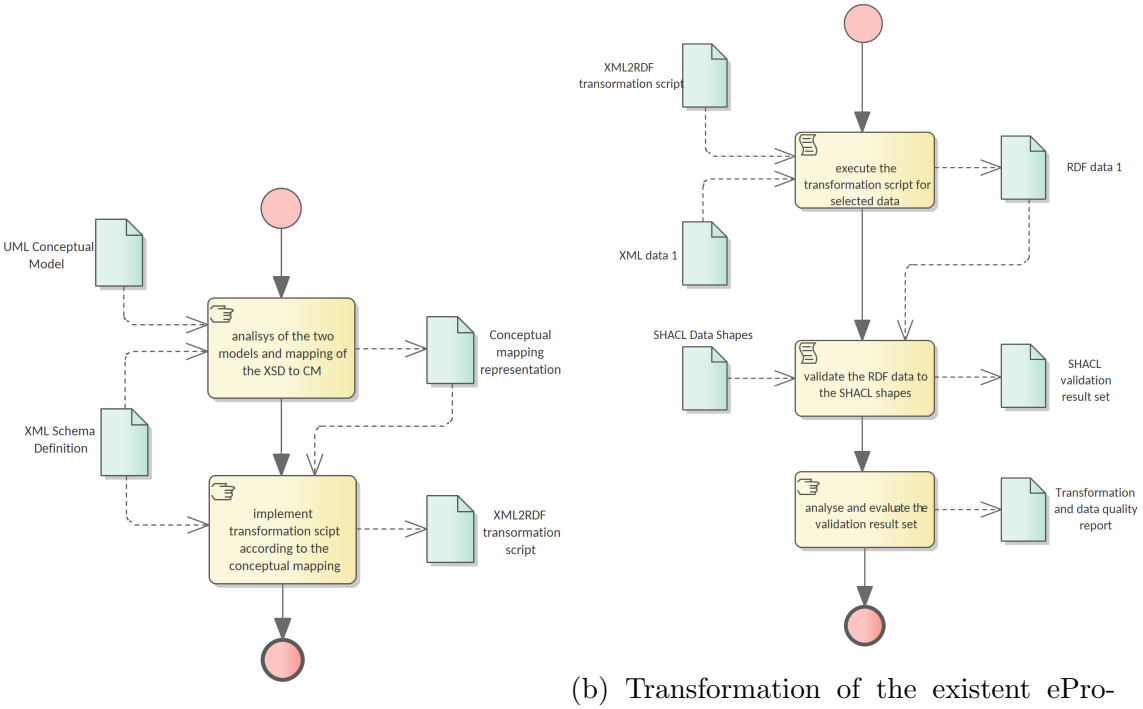
In the introduction of this document, Section 1.3 explains that it is necessary to transform the current eProcurement data from XML formal (structured according to the standard forms) into RDF format. This process is depicted in Figure 5.

The data must represent instances of the ontology generated in previous step explained in Section 3.3, which is structured according to the forthcoming eForms. This means that the transformations script must consider a carefully established conceptual mapping between the two standards in addition to implementing the format transformation itself. Figure 5a reflects that, and in addition, the XSD schemas, to which existent XML data conform, must be consulted and considered in the design and implementation process.

Current specification is agnostic to the technology used for implementing data transformation process. A noteworthy candidate is XSLT – a language for expressing XML transformations. Yet, for the standard programming languages, such as Java, Python, JavaScript and many others, mature libraries to process XML and RDF are available.

The execution of data transformation process (see Figure 5b) unfolds in three steps as follows. First, the selected data is fed as input to the transformation script. The output RDF data are then validated for conformance to the formal ontology using the SHACL data shapes. The validation output is a report in RDF format listing possible violations of the data. This report is transformed into a human readable form using a SPARQL query and used to interpret the conformance of the data, and eventually spot mistakes in the transformation script of in the data shapes.

These reports must be used during the transformation script development, as additional stress tests for whether the script performs correctly or it needs further adjustments. Of course, these reports may indicate problems stemming from either the transformation script, the input data or the SHACL shapes. Therefore, a developer's assessment is necessary to decide on the source of the issue and provide the necessary feedback. The script development process may be considered complete, when for a random set of input data, the analysts and developers can assert that only the input data caused the exceptions. The data driven conformance exceptions constitute an important input in the ontology validation step of the process presented in Section 3.5.



(a) Implementation of the XML transformations script based on the mappings from XSD schemas onto the conceptual model

(b) Transformation of the existent eProcurement XML data into RDF representation and validating the end result conformance

Figure 5: XML to RDF data transformation

3.5 Ontology evaluation

The ontology evaluation process aims at assessing how well the use cases listed in the functional requirements (see Section 2.1) are enabled and supported by the formal ontology and consequently the conceptual model. The evaluation process needs to be designed elsewhere, but at this stage it is possible to foresee two processes that needs to take place: loading the data into a triple-store and querying the data and analysing the result-set for fitness.

Figure 6 depicts the process of loading data into a triple store. Three components must be ingested before it is ready for query: (a) the RDF datasets generated from the existent XML datasets, (b) the core ontology and (c) the formal ontology restrictions. These three components correspond to a complete ontology where the

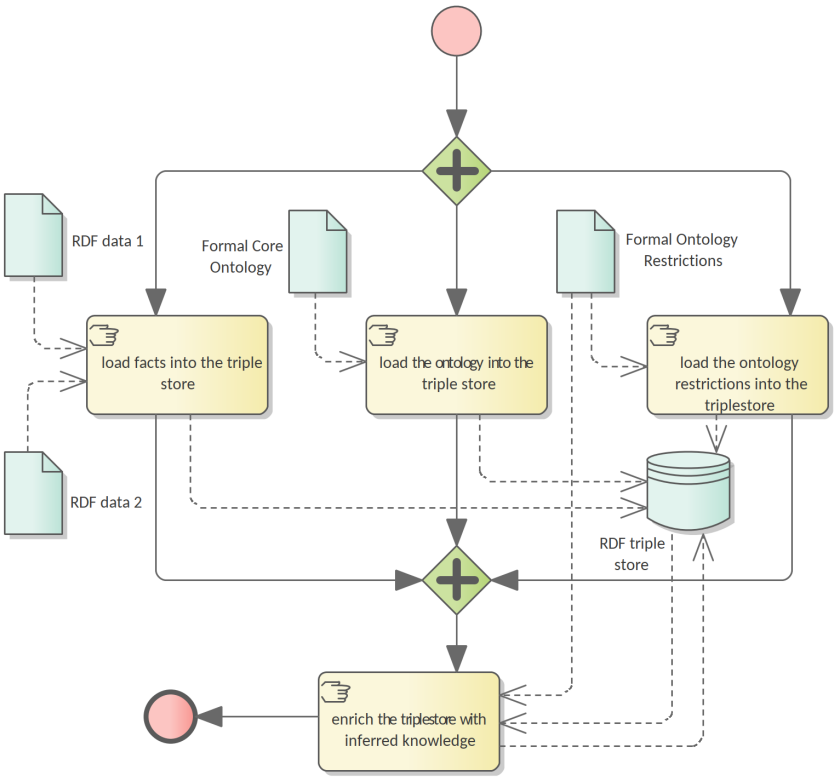


Figure 6: Loading the data into the triple store

model and the data, i.e. ABox and TBox (see Section 5.1), are brought together into a whole.

The triple-store organisation, whether it acts as an eProcurement data dissemination service, URI dereferencing or other publications related issues is not in scope of the current architecture specification. Nevertheless, the data organisation and dissemination should be treated elsewhere in detail considering the best practices for publishing linked data [7].

For the purposes of the evaluation efforts, the TBox axioms, i.e. the core ontology and the formal restrictions, must be saved in a dedicated graph. It is recommended that the instance data is also partitioned into a set of logical and manageable graphs.

Once the data and the ontology are loaded, the benefits of the underlying logical formalism can be drawn: checking the consistency of the data dn model and inferring

new knowledge from the existent one. In case the triple store has an incorporated reasoner it should be enabled following an OWL 2 direct semantics described in Section 5.2. Otherwise an external reasoner such as Fact++ [64], Pellet [62], HermiT [60] or CB [42], must be used to enrich the triple-store. A comparison of OWL 2 EL reasoners is provided in [25]. It is, however, dated and a thorough state of the art research needs to be considered before commencing the work on reasoning.

It is not yet possible to decide at this stage about the range of reasoning capabilities that can be engaged for the eProcurement ontology. Experiments must be conducted on available data to evaluate the speed of reasoning against the expressivity of the considered formalism expressivity and the coverage of inference rules as described in Section 5.4.

In principle, it is possible to engage with ontology evaluation process directly after the data loading steps and skipping the inferencing step, but the range of potential answers will be limited to an extent, which currently is not possible to evaluate.

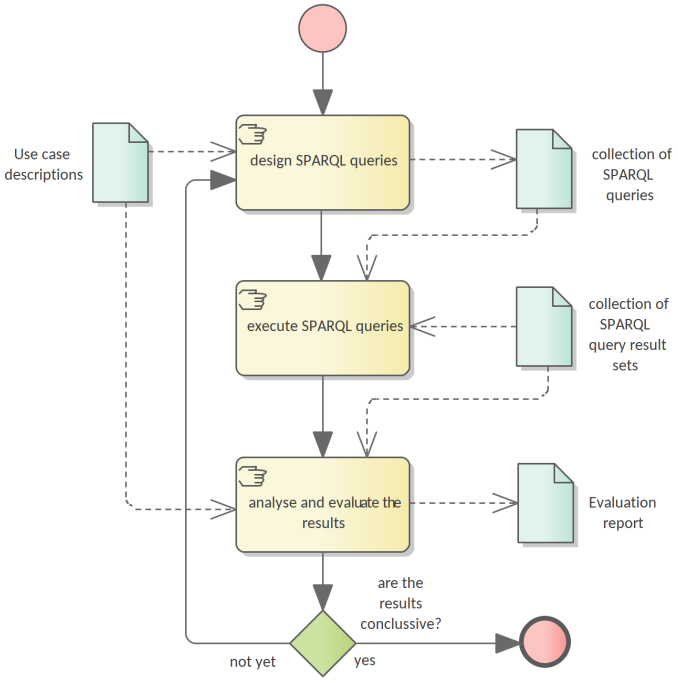


Figure 7: Conceptual validation of the eProcurement ontology against selected use cases

Once the triple store is enriched with inferred information, it can be used to address the use case related information needs. The process is depicted in Figure 7.

Each use case carries a set of information needs which must be derived and expressed in the form of SPARQL queries. The ontology evaluation process commences with developing a collection of such queries. Then they are executed on the triple-store and the result sets are collected. Empty sets are also important results as they indicate possibly an ill-formed query, a lacuna in the existent data or an incorrectly modelled ontological segment.

The result sets are evaluated and aggregated into a conclusive report explaining the strong and weak points of developed ontology in addressing the selected use cases given the existent data.

4 Architectural considerations

The previous section presented the processes used to generate eProcurement ontology components, instance data and how to validate them. This section provides in depth specifications of how these artefacts are structured and how they relate to each other.

4.1 Separation of concerns

The successful application of an ontology or the development of an ontology-based system depends not just on building a good ontology but also on fitting this into an appropriate development process. All computing information models suffer from a semantic schizophrenia. On the one hand, the model represents the domain; on the other hand, it represents the implemented system, which then represents the domain. These different representation requirements place different demands upon its structure [53].

One of the common ways to manage this problem is a separation of concerns. OMG's Model Driven Architecture (MDA) [63] is a well-documented structure where a model is built for each concern and this is transformed into a different model for a different concern.

Transformation deals with producing different models, viewpoints, or artefacts from a model based on a transformation pattern. In general, transformation can be used

to produce one representation from another, or to cross levels of abstraction or architectural layers [61].

The process described in Section 3 incorporates some of these principles and employs model transformation as means to achieve the project objectives.

4.2 Layers and components

This architecture is organised in *horizontal layers* and vertically slicing *components*. The *components* reflect the organisation of the formal ontology based on a logical content division of the UML conceptual model into packages and modules. This division increases the maintainability of entire content. The models and their components are not disconnected from one another. The relations between them are represented in Figure 8 where each component is represented as an UML package. The conceptual model serves as the single source of truth, from which the three components of the formal model are derived. Each of these components can be further divided into modules.

The main ontology packages are: the *core ontology*, *data shapes*, and *formal ontology restrictions*. These formal modules are derived from the conceptual model through model transformations as described in Section 3 following the rules laid out in [15].

The *core ontology* is the foundational, and serves as a backbone for the other components. It establishes the identifiers and the basic definitions of classes and properties.

The *data shapes* represent constraints on how the core ontology can be instantiated and the set of controlled value lists associated to it.

The *formal ontology restrictions* cover intensional class and property definitions used for deriving additional knowledge from factual information. Both, the data shapes and the ontology restrictions are defined as extensions that are flashing out the core ontology which plays, in this case the role of a backbone.

This architecture distinguishes the following layers: the *conceptual layer*, *core definition layer*, *validation layer* and *reasoning layer*. These *layers* can be thought of as formal languages with well-defined boundaries and extents. The diagram in Figure 9 presents the organisation the ontology layers along two axes: *expressivity* and *detail*.

The *expressivity* of a language is the breadth of ideas that can be represented and communicated in that language. The more expressive the language is, the greater the

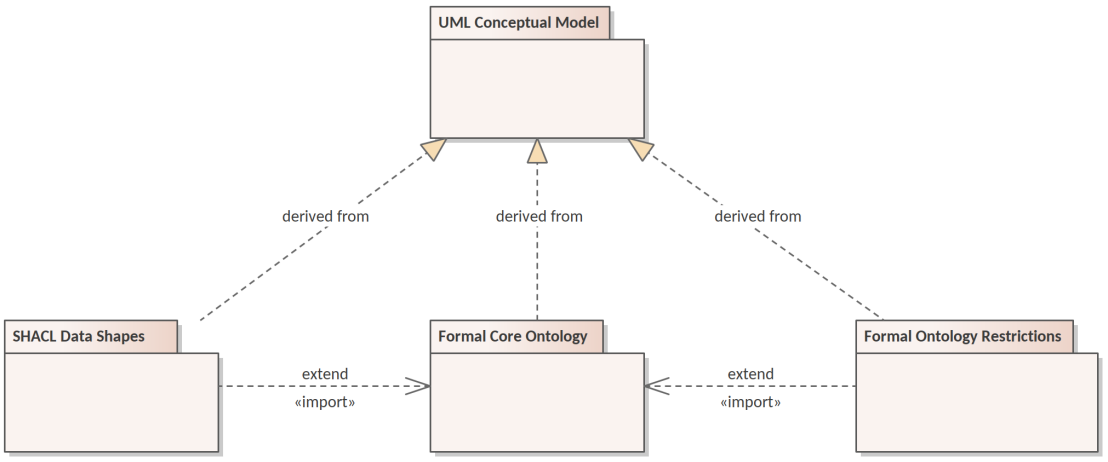


Figure 8: The main components of the eProcurement ontology and their relation to each other and the UML conceptual model

variety and quantity of ideas it can be used to represent. The design of a language and its formalism involves an inevitable trade-off between the expressive power and its "analysability", which translates directly into computation difficulty. The more a formalism can express, the harder it becomes to understand, i.e. compute, what do instances of it mean.

The *detail* refers to how much description and aspects of the domain concepts are considered. This dimension plays a pragmatic rather than formal role. The rationale is that lower level of detail is useful and re-usable to a wider public, but for a set of relatively simple tasks. As the level of detail increases, the difficulty to operate on it also increases thus the user base shrinks and the task complexity rises.

The detail axis starts, on one side, from establishing the concepts identity, labels, natural language definitions; continues through establishing relations and constraints; and ends, on the other side, with formulating special logical conditions, implications and inference rules.

The *conceptual layer* accommodates informal representation of the business objects, places, things, actors in the "real world" not representations of these things in the information system. It is situated at the base of the diagram with lowest expressivity because it has not formal semantics but with a wide coverage of detail. The dotted line delimits the border between formal and informal layers.

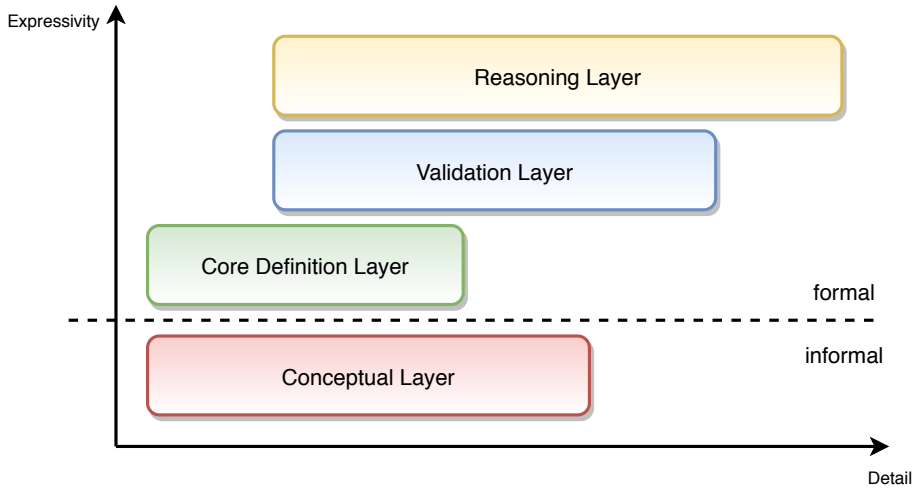


Figure 9: Expressivity and extent of the eProcurement ontology layers

In the conceptual layer is situated the *UML conceptual model*, which is described in Section 4.3. This model is also called *computation independent model* because it is informal and its main purpose is to interface the domain experts with an explicit conceptual representation of the domain.

In the formal section of the diagram, the expert knowledge is expressed using descriptive languages with a formal semantics (see Section 5.2), which in this architecture are primarily OWL 2[52] and SHACL[43].

The *core layer*, is situated in the core ontology, which has as a primary goal to define the main concepts and relations of the ontology. It is limited to the declarative axioms of the ontology and therefore serves primarily for vocabulary and identity establishment. For this reason, it is represented as having the lowest formal expressivity and the lightest level of detail in comparison to other layers. It plays the role of a backbone which is flashed out by other layers.

The *validation layer* accommodates declarations of data shapes which represent constraints on the instance data, i.e. the ABox (see Section 5.1). The assertions in this layer are interpreted done under the closed world assumption, making it possible to support validation functionality. The SHACL shapes are situated in this layer and are addressed in detail in Section 4.5.

The validation layer is situated immediately above the core definitions, being more

expressive but also extending it with additional detail. Therefore, it is shifted, on the detail axis, to the right towards more detail and leaving out the simple axioms as they are already covered in the core later.

The *reasoning layer* accommodates formal intensional definitions of the classes and properties. In Logics, *intensional definition* gives the meaning of a term by specifying necessary and sufficient conditions for when the term should be used. This layer is mostly formed of subclass restrictions and complex class definitions, and as well, domain and range specifications for properties. This layer provides assertions necessary to support consistency checking and classification reasoning functionalities for eProcurement ontology. The formal ontology restrictions component is situated in this layer and is described in Section 4.6.

Just like in the case of the restrictions layer, the reasoning layer extends the core layer with higher level of detail and comprising assertions with higher expressive power. Therefore, the simple details rest in the core layer permitting this one to focus on other ones. For this reason, it is depicted as shifted to the towards the right side of detail axis.

4.3 UML conceptual model

The conceptual model is represented in UML [8] serves as the single source of truth. Thus, the scope of this architecture is limited by what can be expressed in UML and how that information is utilised to generate formal statements. Each of the above functions will lead to different interpretations of the same UML model.

The primary application of UML [34] for ontology design is in the specification of class diagrams for object-oriented software. However, UML does not have a clearly specified declarative semantics, so that it is not possible to determine whether an ontology is consistent, or to determine the correctness of an implementation of the ontology. Semantic integration in such cases becomes a subjective exercise, validated only by the opinions of the human designers involved in the integration effort [38].

On the other hand, UML is closer than more logic-oriented approaches to the programming languages in which enterprise applications are implemented. For this reason, in the current project we have decided to develop agreements on the informal semantics of the UML-based conceptual model. It consists of a set of explicit conventions for naming and structuring UML elements [14] and for transforming UML to OWL [15].

4.4 Core ontology component

This component constitutes the backbone of the eProcurement ontology. It is the simplest from the formal point of view and lightest in terms of detail.

The main purpose of this component is to declare the classes, properties datatypes and controlled lists. It established, in a machine readable format, the concepts by assigning each an URI and decorating it with a human readable labels and descriptions. This represents a mechanism that established a common understanding between humans and machine.

It is void of any constraints or restrictions and may be used as a formal ontology or as a data exchange vocabulary.

4.5 SHACL data shapes component

In OWL the constraints are formed at the semantic level in the way the logic of the entire knowledge base holds together consistently. It is not always easy or obvious how the constraints should be formed so that they fulfil a business or application requirement. Moreover, in practice often the kind of constraints necessary are those aiming at the surface representation of the data. Take, XSD for example, it provides a description of how an XML document ought to be structured.

In the eProcurement domain, before the RDF data is utilised as a semantic resource, it must first respect a more formal conventions on how it is instantiated and organised. There is a need for the XSD counterpart for the RDF graphs.

Shapes Constraint Language (SHACL) [43] is a specification for validating graph-based data against a set of conditions. It provides a concise, uniform syntax for both describing and constraining the contents of an RDF graph. Among others, SHACL includes features to express conditions that constrain the number of values that a property may have, the type of such values, numeric ranges, string matching patterns, and logical combinations of such constraints.

Application profiles (AP) represent a set of constraints on the logical model tying it to a particular system implementation. The application profiles, in this project, must be expressed using SHACL language. The approach taken by the Publications Office to develop APs is described in [16]. The same style should be maintained for the eProcurement APs.

The AP must be conceived as extending the core ontology and fleshing the classes and properties with node and property shapes (see [43, Sec 2.2–2.3]). The constraints available in the conceptual model are generic and should be automatically generated. However, when it comes to integration with specific systems, these constraints may not be sufficient, possibly even too rigid at times. Therefore, it is recommended to conceptualise the data shapes not a single fit it all AP but rather to create specific APs as the needs arise.

4.6 Formal ontology restrictions component

This component accommodates formal intensional definitions of the classes and properties. It is mostly formed of subclass restrictions, complex class definitions, domain and range specifications for properties; which can be derived from the conceptual model.

This component provides the rules and logical conditions for reasoning with eProcurement ontology. And so, the statements from this component play the role of necessary and sufficient conditions to support consistency checking and classification reasoning functionalities for eProcurement ontology.

Mostly expressions in OWL 2 should be acceptable for the reasoning purposes of eProcurement ontology. It is possible, however, that OWL 2 is too expressive, leading to slow reasoning, and thus downgrading to OWL dialect (EL, QL, RL) might be necessary. This implies that multiple variants of this components should be generated, one for each OWL dialect as described in Section 5.4. Each of these variants shall be tested and the most appropriate choice selected for reasoning in the validation phase (see Section 3.5).

5 Formal considerations

This section addresses the aspects of logics in the eProcurement ontology. It specifies the semantics, the relationship between the abstract layer of the ontology and the instance data, under what assumptions is the inference done and what level of logical formalism should the ontology components adapt in order to maintain flexibility and reasoning capacity.

5.1 Model and data relationship

DL ontologies are structured into two sets: *ABox* and *TBox*. The *ABox* consists of all (class or property) instance assertions. The *TBox* consists of all terminological axioms, i.e., of all subclass inclusion axioms. The ABox provides information about concrete individuals while the TBox describes general rules that hold for all individuals. In consequence, ABoxes tend to be much larger than TBoxes [44]. In Figure 10 is depicted the delimitation and relations between the TBox and box ABox components of the eProcurement ontology.

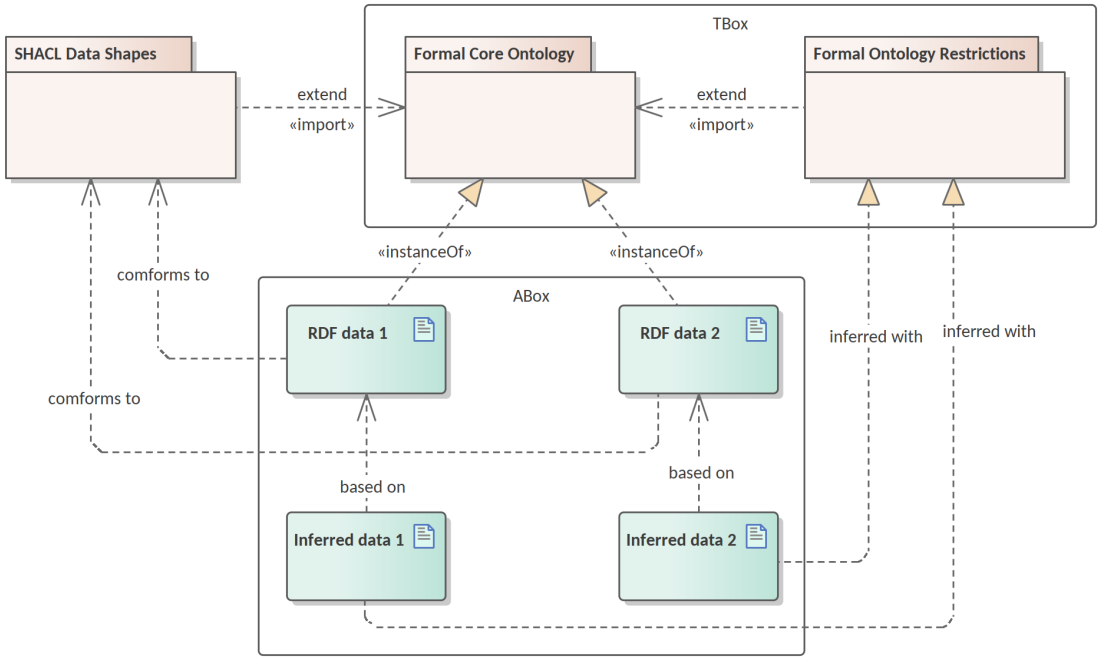


Figure 10: The relationship between the data and each of the ontology components

To explain the relevance of each component, in Figure 10, two arbitrary datasets are brought as examples. They instantiate the core ontology, which means that they comprise of factual statements of concrete entities of eProcurement classes. In order to ensure that the instance data follow minimally the intended ontology design they need to comply with a set of data shapes. Once this condition is satisfied, then, new knowledge can be inferred from the provided facts, given the domain inference rules. The new knowledge is mainly oriented to solve the classification task and does not

cover other types of inference.

It is important to note that the data shapes fall out of the TBox as they serve the validation function and are based on a different set of assumptions. First, they are interpreted under the closed world assumption, like in XML or RDBM contexts (see Section 5.3). Secondly, they follow a RDF graph based semantics 5.2.

5.2 Semantics

Users of OWL [52] can actually select between two slightly different semantics: *direct semantics* that corresponds to the Description Logics (DL) [4], and *RDF-based semantics* that is based on translation of the OWL axioms into directed graphs. In this document we assume by default the direct semantics. In particular cases (i.e. SPARQL entailments and SHACL data shapes) RDF-based semantics is adopted and is explicitly mentioned in the document.

Description logics provide a concise language for OWL axioms and expressions. DLs are characterised by their expressive features. The description logic that supports all class expressions with $>, \bot, \sqcap, \sqcup, \neg, \exists$ and $\forall$ is known as $\mathcal{ALC}$ (which originally used to be an abbreviation for Attribute Language with Complement). For a formal introduction into DL please consult [4].

Inverse properties are not supported by $\mathcal{ALC}$, and the DL we have introduced above is actually called $\mathcal{ALCI}$ (for $\mathcal{ALC}$ with inverses) [44]. Many description logics can be defined by simply listing their supported features. The letter $\mathcal{S}$ is often used as an abbreviation for the "basic" DL consisting of $\mathcal{ALC}$ extended with transitive roles (which in the $\mathcal{AL}$ naming scheme would be called $\mathcal{ALCR}+$).

The letter $\mathcal{H}$ represents sub-roles (role Hierarchies), $\mathcal{O}$ represents nominals (nOminals), $\mathcal{I}$ represents inverse roles (Inverse), $\mathcal{N}$ represent number restrictions (Number), and $\mathcal{Q}$ represent qualified number restrictions (Qualified).

The integration of a concrete domain/datatype is indicated by appending its name in parenthesis, but sometimes a "generic" $\mathcal{D}$ is used to express that some concrete domain/datatype has been integrated. The DL corresponding to the OWL DL ontology language includes all of these constructors and is therefore called $\mathcal{SHOIN}(\mathcal{D})$. We will use this notation when discussing degrees of expressivity for the ontology layers in Section 5.4.

Computing all interesting logical conclusions of an OWL ontology can be a chal-

lenging problem, and reasoning is typically multi-exponential or even undecidable. To address this problem, the recent update OWL 2 of the W3C standard [55, 52] introduced three profiles: *OWL EL*, *OWL RL*, and *OWL QL*. These lightweight sublanguages of OWL restrict the available modelling features in order to simplify reasoning. This has led to large improvements in performance and scalability, which has made the OWL 2 profiles very attractive for practitioners [44].

On the other hand, the validation data shapes are expressed using Shapes Constraint Language (SHACL) [43]. Its semantics is based on RDF graphs but full RDFS inferencing is not required. SHACL processors may operate on RDF graphs that include RDF entailments [54] and SPARQL specific entailments [51]. The entailment regime specifies conditions that address the fourth condition on extensions of basic graph pattern matching [39, 54].

This architecture delimits different concerns in Section 4.2 in a stack of layers and assigns levels of expressivity to each of the layers in Section 5.4.

5.3 Open and closed world assumptions

In the formal systems of logic used for knowledge representation, reasoning is the process through which logical conclusions are derived from a set of premises known to be true or assumed to be true by the laws of valid inference (inference rules).

In the eProcurement ontology checking consistency and deriving new knowledge is foreseen as a valuable functionality beyond the knowledge representation and interoperability establishment.

Inferencing is impacted by what is assumed about the knowledge base. The important assumptions to consider are: (a) whether the knowledge base is considered complete – the closed-world assumption (CWA); or (b) whether the knowledge base (proper knowledge base) is in a state of continuous progression – the open-world assumption (OWA) [19].

Under the *closed-world assumption* it is the presumed that a statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false [58]. The opposite of the closed-world assumption is the *open-world assumption* stating that lack of knowledge does not imply falsity. The truth value of a statement may be true irrespective of whether or not it is known to be true.

The eProcurement data is fragmented across information systems. It represents

concerns specific to different steps in the procurement process. Performing local reasoning with such incomplete knowledge is therefore necessary functionality for the eProcurement project.

Semantic Web languages, including OWL, make the open-world assumption. The absence of a particular statement within the web means, in principle, that the statement has not been made explicitly yet, irrespective of whether it would be true or not, and irrespective of whether we believe that it would be true or not. This stance is also very convenient for decentralised knowledge bases over the internet, where information may be accessible, outdated, contradictory, inaccessible or missing [19].

For validation purposes, in particular, a closed world assumption needs to be made. This concerns the data shapes expressed in SHACL language. In this case, the knowledge base must be considered complete in order to assess whether it fulfils the imposed constraints or recommended shapes. Therefore, everything that is not known to be true must be considered as false.

The eProcurement data must be validated within its local context. The data must be conformant to the information needs and aspects specific to the procurement phase and, possibly, to the information system that handles it. It is, therefore, foreseeable that multiple validation schemes and application profiles have to be developed specific to different phases and aspects of the procurement process. These schemes must extend and flesh out the core ontology, which is the ontology backbone, with levels of specificity and detail as necessary.

5.4 Expressivity levels

In the layered approach described in Section 4.2, different expressivity levels are necessary for each layer. This section briefly describe these levels of expressivity and relate them to OWL sublanguages [20] and profiles [49].

Table 1 summarises the recommended sublanguage for each component, gradually advancing from lightest towards the more expressive ones. The last item, special inference rules, is mentioned but fall out of the scope of current specification. It may be considered in the ontology validation process (see Section 3.5) involving *Semantic Web Rule Language (SWRL)* expressions. SWRL [40] is more expressive than OWL 2. It allows inequality and equality expressions (IE). Punning (PN) is included, which means that class terms can be used as properties. Language overview and a few flavours of SWRL are discussed in [47].

The conceptual model must be expressed UML. It is mostly scoped to class diagrams and the corresponding elements, but additional ones may be employed, provided that there is a clear convention on their interpretation.

The core ontology must be expressed in the simplest OWL language - OWL Lite [20]. [20, Sec.8.3] describes the extent of the language in detail specifying what type of axioms are permitted and which are not. Anyway, this language is beyond the expressivity needs of this component, which are enumerated in Section 4.4.

Model/Component	OWL profile	DL language
Conceptual model	UML (informal)	-
Core ontology	OWL Lite	$\mathcal{SHIF}(\mathcal{D})$
Simplified restrictions	OWL EL	$\mathcal{EL}tiny$
Simplified restrictions	OWL RL	$\mathcal{RL}tiny$
Simplified restrictions	OWL QL	$\mathcal{QL}tiny$
Data shapes	SHACL	-
Complete restrictions	OWL 2	$\mathcal{SROIQ}(\mathcal{D})$
Additional inference rules (out of scope)	*SWRL*	$\mathcal{SROIQ}(\mathcal{D}) + IE, PN$

Table 1: The components and the corresponding language dialect

The formal ontology restrictions are divided into four variants. Three variants must be generated for each of the OWL 2 sublanguages. The transformation rules that require more complex statements must be omitted when generating these (simpler) variants. OWL 2 EL is a lightweight language with polynomial time reasoning, which guarantees very fast termination time. OWL 2 QL is specifically designed for efficient database integration where the amount of instance data is very large. OWL 2 RL is designed for compatibility with rule-based inference tools. An in depth analysis of each dialect is available in [44].

The complete set of restrictions should be expressed using OWL 2 language, which offers "maximum" expressivity while keeping reasoning problems decidable, but still very expensive. The reason these variants are mentioned is because reasoning configuration for eProcurement is still difficult to decide, as was explained in Section 4.6. Each variant must be generated from the UML model, of course, limited to the level of detail available in the UML model, and to the expressivity of each sublanguage indicated in Table 1.

Data shapes must be expressed in SHACL language. No particular restrictions apply to this component as the SHACL engines are known to perform in polynomial time.

6 URI policy

The report on high value datasets from EU institutions [3] mentions eProcurement data as being of special importance and high value. It also provides guidelines and recommendations for publishing government data approached both from the publisher's point of view, and the reuser's point of view.

eProcurement ontology must be published in multiple formats, including RDF. This entails the assignment of identifiers to each term and, to be useful in the kind of linked data applications envisaged, those identifiers should be HTTP URIs and commit long term URI persistence.

6.1 General considerations and recommendations

There is a five star rating [6] to measure published data reusability. This rating system is based on four design principles proposed in [5], to which eProcurement ontology subscribes.

- Use Uniform Resource Identifiers (URIs) to uniquely identify things (data entities);
- Use HTTP URLs, corresponding to these URIs, so that information can be retrieved;
- Provide metadata using open standards such as RDF;
- Include links to related URIs, so that people can discover more things.

Moreover the eProcurement URI scheme must subscribe to the "Cool URIs" recommendations [17] and ensure that they don't change [11].

- *Simplicity.* Short, mnemonic URIs will not break as easily when sent in emails and are in general easier to remember.
- *Stability.* Once you set up a URI to identify a certain resource, it should remain this way as long as possible.
- *Manageability.* Issue your URIs in a way that you can manage. [17]

The ISA^2 study on persistent URIs [2] provides a set of design and management principles. They are completed by a more recent study on URI design patterns [21], in the context of promoting semantic interoperability, identified good design practices for the local part of URIs under the `http://data.europa.eu` domain (see Section 6.2).

- *Use a template.* Pre-defined approach to URI design, using for example URI templates, can help organisations follow a logical structure [2, 21].
- *Avoid stating ownership.* The URI template above does not include the name of the organisation or project that minted the URI [2, 21].
- *Avoid version numbers.* URIs should remain stable between versions and new ones minted for new terms [2, 21].
- *Re-use existing identifiers.* Where resources are already uniquely identified, those identifiers should be incorporated into the URI [2].
- *Avoid using auto-increment.* Minting new URIs for large datasets will need to be automated and the process must be guaranteed to produce unique identifiers, but not sequentially allocated [2].
- *Avoid query strings.* Query strings (e.g. `?param=value`) are usually used in URLs as keys to look up terms in a database. But these constructs should not be used in the URIs but left for particular implement [2].
- *Avoid using file extensions.* For similar reasons as above [2].
- *Mix meaningful and opaque strings.* Meaningful URIs should be avoided in the URI segments which carry a risk of renaming (see Section 6.2) for any foreseeable reason [21].
- *Employ URI sub-divisions.* When necessary, create subdivisions in the URI pattern following a logical pattern and keeping the namespace maintainable. However, this practice must be kept to minimum if at all employed [21].

6.2 Persistence

In 2014, the ISA Programme supported an informal Task Force working on a common policy for the management of persistent, HTTP-based URIs of EU institutions comparable to the virtues of DOI identification scheme [22]. A Persistent URI Service

on the `http://data.europa.eu` sub-domain was established that is responsible for the registration and management of persistent URI namespaces and the forwarding of HTTP requests (URI redirection) towards the Publications Office local register for the eProcurement ontology (see Section 6.3).

`http://data.europa.eu/collection-id/local-register-space`

The Publications Office was allocated `http://data.europa.eu/a4g` URI namespaces for usage in the context of eProcurement ontology. Section 6.3 provides the specifications on the URI structure and the local part organisation.

Regarding URI design, the main consideration of creators should be that when a URI is created, all its parts should be resistant to change. For instance, locations and organisation names can change, and therefore should not be used in URIs. First and foremost, when introducing semantics in URIs, the strings used need to reflect what the resources are (i.e. intrinsic characteristics such as the type or nature), not who owns them or where they are [21].

When creating a URI, its owner can never be certain of who will be using it and can therefore not notify every concerned individual of future changes. It is therefore paramount that URIs are designed carefully with the specific goal of making them persistent, in theory forever [11]. Persistence is a vital component of URI design. Since the local part of a URI is under the control of the institution that owns it (in this case Publications Office), it is up to the owners to ensure that the way they design local IDs enables the persistence of the URI as a whole.

It is recommended to identify all eProcurement resources with URIs which have opaque local identifiers. However, in the case of TBox resources, such as the ontology and the data shapes, mnemonic local segments may be used.

6.3 URI scheme

The URIs are best maintained using a predefined set of patterns and templates. This section defines the URI templates all together forming a URI scheme.

A simple syntax is adopted to express the scheme templates in terms of URI path patterns. A path pattern comprises a sequence of delimited segments, such as `segment1/segment2/segment3`. Each advancement by a segment leads to creation of a new hierarchically positioned namespace.

The segment denomination can be literal or variable. The literal segment means that the value provided is constant and must appear as such in the indicated position. The variable segments are marked by curly brackets ({ }) and represent the name of the slot which must be filled with a concrete value when a new URI is minted in the implementation process. For example, the pattern `segment/{varSegment}` can be instantiated as `segment/123`.

It is possible to indicate that a path sub-sequence is optional by wrapping it in square brackets ([ ]). For example `segment/[{varSubSegment}/]varSubSubSegment` pattern can be instantiated either as `segment/abc/123` or `segment/123`.

This section defines templates for two dedicated PURI namespaces corresponding roughly to TBox–ABox distinction in the ontology structure or the model–data delimitation, which is explained in Section 5.1. The first namespace, *baseVoc*, is dedicated to the ontology, vocabulary and modelling artefacts; while the second, *baseData* is foreseen to accommodate the large volume of instance data.

The instance datasets are generated by a range of institutions. Therefore, each agent should be delegated minting URIs in a controlled and conflict free manner. One way to do that is allocate dedicated base PURI spaces.

Scope	Reference	URI pattern
Base URI for vocabularies	baseVoc	`http://data.europa.eu/a4g`
Base URI for data	baseData	`http://data.europa.eu/{agentSpecificId}`

Table 2: Base PURIs employed by the eProcurement ontology

For the purpose of eProcurement ontology eight scopes have been identified and each is ascribed a path segment in order to form a separate namespace. They are as follows: ontology (`/ontology`), controlled list (`/reference`), data shapes (`/shape`), reasoning restrictions (`/rule`) and XML schemas (`/schema`), instance data (`/resource`), metadata descriptions (`/metadata`), and data services (`/service`).

Ontology core The ontology documents and content should be situated under `/ontology` path segment. Three patterns are of relevance here: to refer to the root of the ontology content space, provided as `xml:base` / `xmlns`; (b) to refer to the document of the ontology or fragment/module, the subject of the ontology header; and (c) one

to refer to the each resource defined within the ontology, such as classes, properties and special individuals. Table 3 defines the patterns.

Purpose	Pattern
Root reference	`{baseVoc}/ontology/{ontologyName}`
Document reference	`{baseVoc}/ontology/{ontologyName}[#{documentRef}]`
Resources ref.	`{baseVoc}/ontology/{ontologyName}#{resourceName}`

Table 3: URI patterns for the ontology namespace

Controlled list The controlled list should replicate the current approach taken by Publications Office for reference data as described in Table 4. Optionally the controlled lists may be managed entirely by the Standardisation and Metadata Unit, including publishing them in the established namespace for reference data `http://publications.europa.eu/resource/authority`.

Purpose	Pattern
Root reference	`{baseVoc}/reference/{listName}`
Document reference	`{baseVoc}/reference/{listName}[#{documentRef}]`
Concept scheme	`{baseVoc}/reference/{listName}`
Concepts	`{baseVoc}/reference/{listName}#{concept}`

Table 4: URI patterns for the reference data namespace

In order to keep maintainability of PURIs high and fence off from the risk of agencies clashing to maintain a common PURI, new base namespaces can be requested from the PURI committee. This risk is particularly high for controlled lists and should carefully considered at the conception.

Data shapes The data shape files should be situated in the `/shape` space. The data shapes are extending the core ontology and are intrinsically bound to it. Therefore, the ontology name must be used in the content root. Table 5 defines the patterns.

Ontology restrictions The restrictions are in fact part of the ontology definition, simply corresponding to the more complex part of it. Therefore, the restrictions

Purpose	Pattern
Root reference	{baseVoc}/shape/{ontologyName}
Document reference	{baseVoc}/shape/{ontologyName}#{documentRef}
Shape resources	{baseVoc}/shape/{ontologyName}#{resourceName}

Table 5: URI patterns for the data shape namespace

Purpose	Pattern
Root reference	{baseVoc}/ontology/{ontologyName}
Document reference	{baseVoc}/rule/{ontologyName}#{documentRef}
Resources	{baseVoc}/ontology/{ontologyName}#{resourceName}

Table 6: URI patterns for the restrictions namespace

belong in the same namespace as the ontology. The document reference, however, can be distinguished and placed in the /rule namespace, where, eventually SWRL [40] and other types of reasoning related artefacts may be placed. Table 6 reflects these patterns.

XML schema In eProcurement domain, usage of XML data is a de facto approach at the moment. In order to support current practices and help establishing a technological change, a space for managing XML schemas id designed within the same PURI space. Schemas namespaces can be minted using baseVoc/schema/schemaName pattern.

Purpose	Pattern
Root reference	{baseData}/resource{documentRef}
Document reference	{baseData}/resource/{documentRef}/{documentRef}
Fragment reference	{baseData}/resource/{documentRef}/{fragmentRef}[/{subFragmentRef}]
Resources	{baseData}/resource/{documentRef}#{resourceId}

Table 7: URI patterns for the instance data namespace

Instance data The concrete eProcurement data instantiating the ePRocurement and related ontologies must be minted in the /resource namespace. The data files

are best conceptualised as datasets, bulks or fragments. Therefore, it is foreseen the possibility to organise a dataset as a set of fragments as described in the VOID specification [18]. Table 7 reflects these patterns.

Metadata Data catalogues and work descriptions should be organised and well described using the established standards such as DCAT [66], VOID [18], Dublin Core [46] or other representations such as FRBR [50] and the CDM [36, 35]. The metadata resources must be minted in the must be minted in the `/metadata` namespace following the patterns provided in Table 8.

Purpose	Pattern
Dataset	`{baseData}/metadata/{datasetId}`
Resources	`{baseData}/metadata/{datasetId}#{resourceId}`

Table 8: URI patterns for the metadata resources namespace

Services It is very important to provide endpoints where the data are accessible. These endpoints can be identified through URIs as well combined with a 303 HTTP redirection [33] to resolve the URI to the current URL where the service is accessible. Service URIs should be minted in the `/service` namespace as described in Table 9.

Purpose	Pattern
Sparql	`{baseData}/service/sparql/{dataLakeId}`
Catalogue	`{baseData}/service/catalogue/{catalogueId}`

Table 9: URI patterns for the namespace of the services

Bibliography

[1] Xml metadata interchange (xmi) specification: Version 2.5.1. Standard formal/2015-06-07, Object Management Group (OMG), 2015. URL `http://www.omg.org/spec/XMI/2.5.1`.

[2] P. Archer, S. Goedertier, and N. Loutas. D7.1.3 - study on persistent uris, with identification of best practices and recommendations on the topic for the mss and the ec. Deliverable, ISA programme of the European Commission, 2012.

[3] P. Archer, L. Bargiotti, M. D. Keyzer, S. Goedertier, N. Loutas, and F. V. Geel. Report on high-value datasets from eu institutions. Deliverable SC17DI06692, European Commission, 2014.

[4] F. Baader, I. Horrocks, and U. Sattler. Description logics. In *Handbook on ontologies*, pages 3–28. Springer, 2004.

[5] T. Berners-Lee. Linked data, 2006, 2006.

[6] T. Berners-Lee. Star open data. *5 Star Data*, 5.

[7] C. Bizer. The emerging web of linked data. *IEEE intelligent systems*, 24(5): 87–92, 2009.

[8] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005. ISBN 0321267974.

[9] S. O. Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, Mar. 1997. URL `https://rfc-editor.org/rfc/rfc2119.txt`.

[10] D. Brickley and R. Guha. RDF schema 1.1. W3C recommendation, W3C, Feb. 2014. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[11] T. Burners-Lee. Cool uris don't change. *http://www. w3. org/Provider/Style/URI*, 1998.

[12] S. Cirulli. Xspec v0. 5.0. *XML LONDON 2017*, 2017.

[13] S. Cook, C. Bock, P. Rivett, T. Rutt, E. Seidewitz, B. Selic, and D. Tolbert. Unified modeling language (UML) version 2.5.1. Standard formal/2017-12-05, Object Management Group (OMG), Dec. 2017. URL `https://www.omg.org/spec/UML/2.5.1`.

[14] E. Costetchi. eProcurement uml conceptual model conventions. Recommendation, Publications Office of the European Union, April 2020.

[15] E. Costetchi. eProcurement uml conceptual model to owl ontology transformation. Recommendation, Publications Office of the European Union, April 2020.

[16] E. Costetchi and W. V. Gemert. Towards executable application profiles for european vocabularies. In *Smart Descriptions & Smarter Vocabularies (SDSVoc)*. W3C, oct 2016.

[17] R. Cyganiak and L. Sauermann. Cool URIs for the semantic web. W3C note, W3C, Dec. 2008. http://www.w3.org/TR/2008/NOTE-cooluris-20081203/.

[18] R. Cyganiak, M. Hausenblas, K. Alexander, and J. Zhao. Describing linked datasets with the VoID vocabulary. W3C note, W3C, Mar. 2011. http://www.w3.org/TR/2011/NOTE-void-20110303/.

[19] C. V. Damásio, A. Analyti, G. Antoniou, and G. Wagner. Supporting open and closed world reasoning on the web. In *International Workshop on Principles and Practice of Semantic Web Reasoning*, pages 149–163. Springer, 2006.

[20] M. Dean and G. Schreiber. OWL web ontology language reference. W3C recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[21] M. Dekkers and I. Novacean. D04.02.02 – local uri design patterns. Deliverable SC353DI07171, ISA programme of the European Commission, 2018.

[22] M. Dekkers and I. Novacean. D04.02.3 - comparison of using puri & doi. Deliverable SC508DI07171, ISA programme of the European Commission, 2018.

[23] M. Dekkers, E. Stani, and F. Barthelemy. D02.02 - project charter proposal. Deliverable SC378DI07171, Publications Office of the European Union, 2017.

[24] M. Dekkers, E. Stani, B. Wyns, and F. Barthelemy. D02.01 - specification of the process and methodology to develop the eprocurement ontology with initial draft of the eprocurement ontology for 3 use cases. Deliverable SC378DI07171, Publications Office of the European Union, 2017.

[25] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer. Comparison of reasoners for large ontologies in the owl 2 el profile. *Semantic Web*, 2(2):71–87, 2011.

[26] European Parliament and the Council.

[27] European Parliament and the Council. Directive 2013/37/EU of the European Parliament and of the Council of 26 June 2013 amending Directive 2003/98/EC on the re-use of public sector information Text with EEA relevance. *OJ*, L 175: 1–8, 2013. CELEX:32013L0037.

[28] European Parliament and the Council. Directive 2014/23/EU of the European Parliament and of the Council of 26 February 2014 on the award of concession contracts Text with EEA relevance. *OJ*, L 94(2014/23/EU):1–64, 2014. CELEX:32014L0024.

[29] European Parliament and the Council. Directive 2014/24/EU of the European Parliament and of the Council of 26 February 2014 on public procurement and repealing Directive 2004/18/EC Text with EEA relevance. *OJ*, L 94:65–242, 2014. CELEX:32014L0024.

[30] European Parliament and the Council. Directive 2014/55/EU of the European Parliament and of the Council of 16 April 2014 on electronic invoicing in public procurement Text with EEA relevance. *OJ*, L 133:1–11, 2014. CELEX:32014L0055.

[31] European Parliament and the Council. Regulation (EU) 2018/1724 of the European Parliament and of the Council of 2 October 2018 establishing a single digital gateway to provide access to information, to procedures and to assistance and problem-solving services and amending Regulation (EU) No 1024/2012 (Text with EEA relevance.) . *OJ*, L 295:1–38, 2018. CELEX:32018R1724.

[32] European Parliament and the Council. Directive (EU) 2019/1024 of the European Parliament and of the Council of 20 June 2019 on open data and the re-use of public sector information. *OJ*, L 172:56–83, 2019. CELEX:32019L1024.

[33] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol–http/1.1. 1999.

[34] M. Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[35] E. Francesconi, M. W. Küster, P. Gratz, and S. Thelen. The ontology-based approach of the publications office of the eu for document accessibility and open data services. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 29–39. Springer, 2015.

[36] E. Francesconi, M. W. Küster, P. Gratz, and S. Thelen. Semantic modeling of the eu multilingual resources. In *ICAIL Multilingual Workshop on AI & Law Research*, page 13, 2015.

[37] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.

[38] M. Grunninger. Enterprise modelling. In *Handbook on enterprise architecture*, pages 515–541. Springer, 2003.

[39] P. Hayes. RDF semantics. W3C recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[40] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31, 2004.

[41] M. Kay. XSL transformations (XSLT) version 3.0. W3C recommendation, W3C, June 2017. https://www.w3.org/TR/2017/REC-xslt-30-20170608/.

[42] Y. Kazakov. Consequence-driven reasoning for horn shiq ontologies. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[43] H. Knublauch and D. Kontokostas. Shapes constraint language (SHACL). W3C recommendation, W3C, July 2017. https://www.w3.org/TR/2017/REC-shacl-20170720/.

[44] M. Krötzsch. Owl 2 profiles: An introduction to lightweight ontology languages. In *Reasoning Web International Summer School*, pages 112–183. Springer, 2012.

[45] M. Krötzsch, I. Horrocks, M. Smith, and B. Glimm. OWL 2 web ontology language conformance (second edition). W3C recommendation, W3C, Dec. 2012. http://www.w3.org/TR/2012/REC-owl2-conformance-20121211/.

[46] J. Kunze and T. Baker. The dublin core metadata element set. Technical report, RFC 5013, August, 2007.

[47] A. Lawan and A. Rakib. The semantic web rule language expressiveness extensions-a survey. *CoRR*, abs/1903.11723, 2019. URL `http://arxiv.org/abs/1903.11723`.

[48] N. Loutas, N. Loutas, S. Kotoglou, and D. Hytiroglou. D04.07 - report on policy support for eprocurement. Deliverable SC245DI07171, ISA programme of the European Commission, 2016.

[49] B. Motik, I. Horrocks, B. C. Grau, A. Fokoue, and Z. Wu. OWL 2 web ontology language profiles (second edition). W3C recommendation, W3C, Dec. 2012. http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.

[50] I. F. of Library Associations and I. S. on Cataloguing. Standing Committee. *Functional requirements for bibliographic records*, volume 19. KG Saur Verlag Gmbh & Company, 1998.

[51] C. Ogbuji and B. Glimm. SPARQL 1.1 entailment regimes. W3C recommendation, W3C, Mar. 2013. http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/.

[52] B. Parsia, P. Patel-Schneider, and B. Motik. OWL 2 web ontology language structural specification and functional-style syntax (second edition). W3C recommendation, W3C, Dec. 2012. http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[53] C. Partridge, A. Mitchell, and S. de Cesare. Guidelines for developing ontological architectures in modelling and simulation. In *Ontology, Epistemology, and Teleology for Modeling and Simulation*, pages 27–57. Springer, 2013.

[54] P. Patel-Schneider and P. Hayes. RDF 1.1 semantics. W3C recommendation, W3C, Feb. 2014. http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

[55] P. Patel-Schneider, B. Parsia, and B. Motik. OWL 2 web ontology language structural specification and functional-style syntax. W3C recommendation, W3C, Oct. 2009. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.

[56] Publications Office of the European Union. Interinstitutional style guide 2011, 2011. doi: doi:10.2830/36616.

[57] PwC EU Services. e-government core vocabularies handbook. Report, ISA programme of the European Commission, 2015. URL `https://ec.europa.eu/isa2/library/e-government-core-vocabularies-handbook_en`.

[58] R. Reiter. On closed world data bases. In *Readings in artificial intelligence*, pages 119–140. Elsevier, 1981.

[59] E. Rescorla, R. Barnes, and S. Kent. Further Key Words for Use in RFCs to Indicate Requirement Levels. RFC 6919, Apr. 2013. URL `https://rfc-editor.org/rfc/rfc6919.txt`.

[60] R. Shearer, B. Motik, and I. Horrocks. Hermit: A highly-efficient owl reasoner. In *Owled*, volume 432, page 91, 2008.

[61] J. Siegel. Object management group model driven architecture (mda) mda guide rev. 2.0. 2014. URL `http://www.omg.org/cgi-bin/doc?ormsc/14-06-01`.

[62] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[63] R. Soley et al. Model driven architecture. *OMG white paper*, 308(308):5, 2000.

[64] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *International joint conference on automated reasoning*, pages 292–297. Springer, 2006.

[65] S. A. White. Introduction to bpmn. *Ibm Cooperation*, 2(0):0, 2004.

[66] P. Winstanley, A. Perego, S. Cox, D. Browning, R. Albertoni, and A. G. Beltran. Data catalog vocabulary (DCAT) - version 2. W3C recommendation, W3C, Feb. 2020. https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/.