Complex Systems Informatics and Modeling Quarterly
CSIMQ, Issue 1, March, 2014, (Pages 1-23)
Published online by RTU Press, https://csimq-journals.rtu.lv
http://dx.doi.org/10.7250/csimq.2014-1.01
ISSN: 2255-9922 online

# Using Ontologies for Enterprise Architecture Integration and Analysis

Gonçalo Antunes[1], Marzieh Bakhshandeh[1], Rudolf Mayer[2],
José Borbinha[1,3], Artur Caetano[1,3]

[1] INESC-ID, Information Systems Group, Rua Alves Redol 9, 1000-029 Lisbon, Portugal
[2]SBA Research GmbH, Favoritenstraße 16, 1040 Wien, Austria
[3] Higher Technical Institute, University of Lisbon, Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal

goncalo.antunes@ist.utl.pt, marzieh.bakhshandeh@ist.utl.pt, rmayer@sba-research.at,
jose.borbinha@ist.utl.pt, artur.caetano@ist.utl.pt

**Abstract.** Enterprise architecture facilitates the alignment between different domains, such as business, applications and information technology. These domains must be described with description languages that best address the concerns of its stakeholders. However, current model-based enterprise architecture techniques are unable to integrate multiple descriptions languages either due to the lack of suitable extension mechanisms or because they lack the means to maintain the coherence, consistency and traceability between the representations of the multiple domains of the enterprise. On the other hand, enterprise architecture models are often designed and used for communication and not for automated analysis of its contents. Model analysis is a valuable tool for assessing the qualities of a model, such as conformance and completeness, and also for supporting decision making. This paper addresses these two issues found in model-based enterprise architecture: (1) the integration of domain description languages, and (2) the automated analysis of models. This proposal uses ontology engineering techniques to specify and integrate the different domains and reasoning and querying as a means to analyse the models. The utility of the proposal is shown through an evaluation scenario that involve the analysis of an enterprise architecture model that spans multiple domains.
**Keywords**: Enterprise architecture, ontology, domain integration, ArchiMate, OWL.

## 1 Introduction

Enterprise architecture is defined by Lankhorst as "a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems, and infrastructure", with a "focus on alleviating the infamous business-information technology alignment problem" [1]. Such alignment is typically achieved through the creation of models describing an enterprise, including business and IT elements, so that it can realize management requirements and be maintained over the period of its useful life [2].

Several enterprise architecture frameworks were developed through the years, providing methods and techniques, including enterprise architecture modelling languages, as can be witnessed, for instance, in [3]. However, and despite the efforts for developing comprehensive approaches to enterprise architecture, such as TOGAF [4] or ArchiMate [5], it can be noticed

that such "one language fits all" approaches are not enough for addressing the specifics of each organization, which can be attested by the emergence of situational approaches [6], [7], [8], [9]. The use of situational approaches makes possible the combination of different method and model fragments to suit the specificities of each scenario being addressed, which can be a means to more effectively address the concerns of the stakeholders.

Managing the dependencies between and within the different models created is crucial for supporting the communication between the different stakeholders, and for ensuring their alignment and consistency [10], [11]. However, the integration of multiple meta-models brings challenges at the level of coherence and model consistency [12], [13], [14]. In terms of traceability, it becomes difficult to maintain links among elements from the different models, which is a problem that gets exacerbated as the models evolve. Consequently, model consistency is also hindered, namely when the models are created using different tools.

Moreover, commonly used model representations (i.e., the way the information contained in the models is structured and organized) create challenges to the analysis and evaluation of the alignment between business goals and the IT capabilities. On the one hand, the high complexity and detail of the information contained in the models makes the analysis exclusively by human means hard. On the other hand, used model representations are typically inadequate for automatic processing of the information contents of the models, with most of the tool support available for that end providing limited analysis capabilities [15].

As a result, there is a need for an extensible approach to enterprise architecture that ensures meta-model level coherence, which concerns encoding rules on the meta-model, and supports model and cross-model verification, which concerns the evaluation of the conformance of the models to the rules specified in the respective meta-models. Moreover, enterprise architecture models should be computable so that the effort involved in their analysis is manageable. Therefore, the following research questions can be raised:

- Research Question 1 - Model representation: How to represent the models in a way that allows their integration?
- Research Question 2 - Model analysis: How to represent the models in a way that supports the production of relevant analysis for the different stakeholders?
- Research Question 3 - Analysis techniques: What information processing techniques should be used for determining the alignment of the information systems with the business goals?

In order to be able to provide answers to the aforementioned research questions, we assume that the following principles should be respected:

- Principle 1 - Flexibility: Support for covering the multiple viewpoints of stakeholders should be provided, including the capability for enriching the already existing information.
- Principle 2 - Computable models: A computable representation for the enterprise architecture models should be adopted.
- Principle 3 - Support for analysis techniques: Support for techniques allowing the detection and analysis of alignment between system capabilities and business objectives should be provided.

This paper raises the hypothesis that ontologies can contribute for solving the described problem. Ontologies describe a domain model by associating meaning to its terms and relations. A more formal and widely used definition is that of Gruber who defines an ontology as a "formal specification of a conceptualisation" [16]. Several authors recognize ontologies and associated techniques as being valuable tools in the enterprise architecture domain [17][18][19][20][21]. Enterprise architecture can benefit from the available knowledge and techniques, which can improve model creation, extension and validation, through an explicit semantic definition of the concepts present in different meta-models [22].

The main contribution of this article is thus proposing an architecture based on the use of ontologies that provides the following features:

- Representation of enterprise architecture models, providing information processing capabilities, namely computational inference and information querying, allowing for:
  - Enforcement of meta-model coherence, through the addition of axioms to the ontologies enforcing semantic rules implicitly defined in the model specifications.
  - Meta-model conformance verification of models, since models specified according to a determined meta-model can be verified against the semantic rules specified in the ontology through the use of reasoners, allowing for the verification of logical inconsistencies present in models.
  - Information processing mechanisms that can be used to support decision-making, through the usage of computational inference and querying mechanisms, allowing for better information retrieval and processing.
- Ontological integration of the viewpoints of different stakeholders, offering improved extensibility and expressiveness of the Enterprise Architecture, through the management of the inclusion of new domain-specific meta-models in a standard way, with the aim capturing specific aspects of organisations.

We demonstrate the applicability of the proposal through the application of formal ontologies to model a set of different enterprise architecture domains and through the consistent integration of these domains. In particular, we develop an ontology to specify the ArchiMate 2.0 meta-model and then create traceable maps to it from a set of domain-specific languages. We also describe an example that maps the sensor technology domain to ArchiMate in the context of a real-world scenario. This demonstration shows that the application of ontologies to Enterprise Architecture modelling effectively assists consistently aligning and analysing different domains.

This work is organized as follows. In section 2, related work is described. In section 3, some of the current issues existing in Enterprise Architecture practice which this work intends to approach are discussed. Section 4 presents a proposal based on ontology techniques for improving the representation of architecture models. Section 5 presents a proposal based on ontology techniques for improving the analysis of architecture models. Section 6 describes an implementation of the proposal. Section 7 evaluates the proposal through its application to a concrete scenario. Finally, Section 8 concludes the paper and provides directions for future work.

# 2 Related Work

## 2.1 Enterprise Architecture Model Integration

When integrating the architectural data specified in different models two approaches are possible: a holistic approach or a federated approach [23]. In the holistic approach, a single model comprising all the needed artefacts is used, which means that any information specified using a different meta-model as a basis has to be converted into it, in order to be integrated. On the other hand, a federated approach based on meta-model integration uses specialized models that are linked to a main model.

Different works dealing with federated model integration in enterprise architecture have been published through the years. One of the earliest mentions to model integration in this specific domain was the Enterprise Model Integration (EMI) [24] approach. EMI is based on "object-oriented meta-modelling" concepts to describe specific languages, which can then be integrated vertically (i.e., top-down or bottom-up), horizontally, or using an hybrid approach. Different types of patterns are then specified for describing the types of mappings between two meta-models, addressing the cases where the meta-models complement each other or when they are used concurrently. In the first case, reference links or transformation rules can be used, whereas in the second case merge rules are used.

In [25], the ArchiMate language is first proposed as a way to "bridge between other existing modelling languages", which provides an high-level set of concepts that can be specialized using

specific models. In order to support this proposal, a workbench for the integration of modelling languages and tools is presented. The integration is achieved through the use of bi-directional transformational rules that can be used to detail ArchiMate model elements into more detailed models, specified in a more specific language. The proposal made in this work and its implementation is greatly influenced by this work. Follow-up works include [26], which approaches the integration of functional with non-functional models and design with analysis models, and [27], which deepens the integration role of ArchiMate by discussing its integration with symbolic, semantic and subjective models.

The work in [28] extends the EMI approach with mappings and rules for meta-model integration, providing a catalogue of rules that can be instanced in specific integration scenarios, thus making EMI situational. The work in [29] advocates the use of a federated approach to business architecture engineering that integrates generic meta-modelling methods with business architecture meta-models. For that it proposes a method based on the identification of the stakeholder concerns and specification of meta-models for addressing them, and integrating the different meta-model fragments resulting from the exercise. The integration is done through the creation of mappings reflecting similarity, generalisation or specialisation.

More recently, an approach based on the use of meta-model fragments extending a core meta-model, focusing in different qualities is described in [6]. The work in [30] maps concepts from a risk management domain model into ArchiMate using generalisation and specialisation relationships between the concepts of the two meta-models. In [31], the Enterprise Services Architecture Meta-model Integration (ESAMI) is described as being a "correlation-based integration method for architecture viewpoints, views, and models", for integrating service-oriented enterprise architectures.

### 2.2 Ontologies

The term ontology comes from the Greek *ontos* (being) + *logos* (word), literally meaning existence [32]. From the perspective of philosophy, ontology is the systematic explanation of being [32]. In computer science, the most widely used definition characterizes ontologies as "formal, explicit specification of a shared conceptualization" [16]. In this definition, "conceptualization" refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena, "explicit" means that the type of concepts used and the constraints on their use are explicitly defined, "formal" refers to the fact that the ontology should be machine readable, and "shared" reflects that the ontology should capture consensual knowledge accepted by the communities [34].

Uschold divides the uses of ontology into three categories [35]: human communication, interoperability, and systems engineering. In human communication, ontologies reduce conceptual and terminological confusion and enable shared understandings between people with different needs and viewpoints. When used for interoperability ends, it improves the exchange of data among heterogeneous sources. When engineering systems, a shared understanding of problems can also assist in the systems' specification. Informal (simple) ontologies can be the basis for manual checking of designs against specifications, to improve systems reliability. Ontologies can also foster reusability, enabling the reuse of knowledge models in new applications. In this case, ontologies are used to make the underlying assumptions of software component design explicit [32], [35]. Guarino has also provided a listing of the fields making use of ontologies, which includes: knowledge engineering, knowledge representation, qualitative modelling, language engineering, database design, information retrieval and extraction, and knowledge management and organization [36].

According to Uschold [35], there are several formalization degrees in ontology, whose range extends from the most informal to the more formal including:

- Informal (natural language)

- Semi- informal (restricted and structured form of natural language)
- Semi-formal (languages such as Ontolingua, UML, OWL)
- Rigorously formal (meticulously defined terms, formal semantics and theorems, soundness and completeness proofs )

The spectrum of ontology formalization in computer science [32] is depicted in Figure 1. According to this figure, there are two main groups of ontologies: non-structured or simple (i.e., left of the spectrum) ontologies, and structured ontologies (i.e., right of the spectrum). The degree of structure of the ontology depends on the purpose of use of ontology. The simplest form of ontology is a catalogue which is a controlled vocabulary (i.e., a finite list of terms). Another potential ontology specification is a glossary (i.e., a list of terms and meanings). The concepts and meanings in these types of ontologies are typically specified in natural language statements. This provides a kind of meaning description for human consumption since humans can read the natural language statements and interpret them.

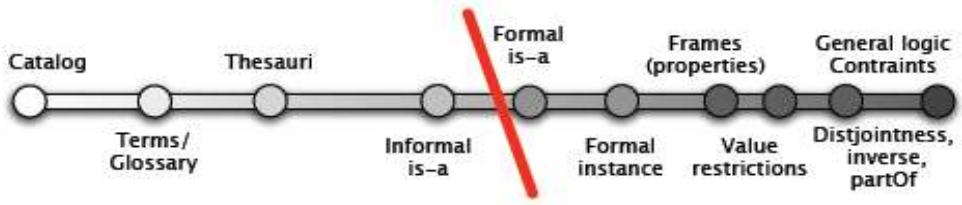


**Figure 1.** Ontology Spectrum [37]

However, interpretations of this type of specification are not unambiguous and thus they are not adequate for computer agents, not meeting the criteria of being processed by machines. Thesauri provide some additional semantics concerning the relations between terms, for instance, synonym relationships. In many cases, the relationships defined therein may be interpreted unambiguously by agents. Typically, thesauri do not provide an explicit hierarchy.

An inclusion of strict sub-class hierarchies (i.e., is-a relationships between concepts) improves deduction possibilities over ontologies. Another step further in terms of ontology formalization includes using frames. Frames are a knowledge representation scheme very similar to class diagrams. When using frames, concepts are represented through classes. Each class has a set of properties, with each property having a type, a set of values and value restrictions.

Description logics (DL) are a family of logic based knowledge representation formalisms used to describe domains in terms of concepts, roles and individuals, including their relationships. DL allows the definition of axioms that are logical statements about the "relationships between properties and/or classes in the domain" [38]. In that sense, DL are a step further in ontology formalization as they allow the expression of logic constraints that are not possible outside the specification of a frame.

### 2.2.1 Ontology Integration

The word integration has been used with different meanings in the ontology field. In simple terms, ontology integration is the process of identifying common concepts and relationships shared between two or more ontologies [39]. Three main techniques of ontology integration can be described as follows.

- Ontology alignment: is the process of building a new ontology by identifying correspondences between all the concepts of two ontologies, which are said to be equivalent.
- Ontology mapping: is the process of building a new ontology by finding common concepts between two (or more) concepts belonging to two (or more) different ontologies.

- Ontology merging: is the process of building a new ontology by merging several ontologies into a single one that will "unify" all of them, and create a more general ontology about a subject.

One of the big challenges in ontology mapping is finding the semantic mappings between two given ontologies. Figure 2 shows a high-level view of the mapping process [40], [41]. An essential element when performing the mapping is the matcher. The matcher tries to discover correspondences between different ontology entities from narrow perspectives [42]. It builds correspondences between different ontologies by first selecting an appropriate feature (e.g., entity label, structural description of concepts, and range for relations, instantiated attributes or extensional descriptions) from the ontologies [40].

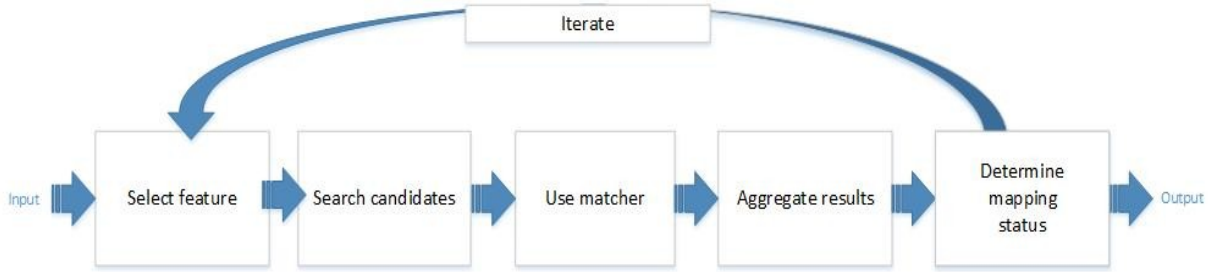


**Figure 2.** A simplified high-level view of a mapping process

Common mapping approaches link candidate ontologies to a common ontology using anchors [43], [40], [42]. Anchors are entities which are declared to be equivalent. Finally, matchers evaluate the similarity criteria of both ontologies. Frequently, functions based on heuristic similarity instead of exact logical similarity are used to avoid a costly complete search. Iterating stops when there are no more new mapping proposals [44].

Different types of mismatches may occur between different ontologies [40], [45], [46], [47]:

- Syntactic mismatches: Two ontologies are syntactically heterogeneous if they are represented by different representation languages. To resolve this type of mismatches, simply transform the representation language of one ontology to the representation language of the other ontology. However, many times translation is difficult and even impossible and may lead to source information omission.
- Lexical mismatches: Describes the heterogeneity between the names of entities, instances, properties or relations. They are four forms of heterogeneity:
    - Synonyms: The same entity is represented by two different names.
    - Homonyms: The same name represents two different entities.
    - The same name in different languages labelling the same entity.
    - The same entities are named with the same words but with different syntactic variations, such as abbreviations, optional prefixes or suffixes
- Semantic mismatches: The mismatches identified at this level are related to the content of the input ontologies. This is the most challenging mismatch. Three abstract forms of mismatches can be described:
    - Coverage: Two ontologies are different from each other in that they cover different (possibly overlapping) portions of the world (or even of a single domain).
    - Granularity: Two ontologies are different from each other in that one provides a more respectively less detailed description of the same entity.
    - Perspective: Two ontologies are different from each other in that one may provide a viewpoint on some domain which is different from the viewpoint adopted in another ontology.

### 2.2.2 Ontology Reasoning

One of the key features of ontologies is that they can be validated and analysed by reasoning methods. The term reasoning is used, in the context of ontologies, to denote any mechanism for making explicit a set of facts that are previously implicit in an ontology. The two main purposes of reasoning are validation and analysis [48].

Validating an ontology means ensuring that the ontology is a good representation of the domain of discourse that you aim at modelling. Reasoning is extremely important for validation. For example, consistency checking is a kind of reasoning, which can be performed to capture possible inconsistencies in the definition of the classes and properties of the ontology. Also, a reasoner can then be used to obtain inferred information about the model, such as inferred super-classes, inferred equivalent classes, and inferred types for individuals.

In analysis one assumes that the ontology is a faithful representation of the domain, and tries to deduce facts about the domain by reasoning over the ontology. Moreover, it tries to collect new information about the domain by exploiting the ontology. Clearly, analysis can also provide input to the validation phase.

In the recent years logical reasoning has been widely used in the field of ontology engineering. There are some forms of logical reasoning, including reasoning in classical logic and non-classical logic [48], [49], [50]. Reasoning in classical logic is related to the idea of proving theorems in classical first order logic. This can be characterized as the task of checking whether a certain fact is true in every models of a first-order theory. A well-known example of reasoning that cannot be captured by classical logic is the so-called defeasible reasoning. Although we can state in an ontology that all birds fly, it is known that penguins are birds that do not fly. This is a form of non-monotonic reasoning: new facts may invalidate old conclusions. Since classical logic is monotone, this goes beyond the expressive power of classical logics.

## 3 Problem: the Need for Flexible and Computable Enterprise Architecture

In this section, we describe and analyse the problem addressed by this work. In order to ground our proposal in good practice, the architecture principles that a solution for the problem should abide for are defined. An architecture principle can be described as "a declarative statement that normatively prescribes a property of the design of an artefact, which is necessary to ensure that an artefact meets its essential requirements" [51].

According to the ISO/IEC/IEEE 42010 [52], a system architecture description includes views that address the concerns of the system stakeholders. Those views are created according to the viewpoints of the stakeholders. Correspondence rules are created between the views to enforce composition, refinement, consistency, traceability, dependency, constraint and obligation relationships between the views [52]. In this way, architecture can function as a communication tool between different stakeholders, as each is presented with its own consistent view on the system of interest, displayed in a manner that is understood by him.

- Architecture Principle 1 - Concern orientation: The architecture of the solution shall represent the concepts necessary and sufficient to address an explicit set of modelling concerns. This means that the model shall be derived from the questions that need to be addressed and to provide answers to those questions. This also means that the model shall not support any concepts that are not explicitly derived from stakeholder concerns.
- Architecture Principle 2 - Expressiveness: The architecture of the solution shall be able to represent the domain concepts without ambiguity in order to improve communication. This entails defining the minimum set of types and relationships to describe a domain

Although the need for multiple views on the system is recognized by the standard, the truth is that it is a challenge to maintain these relationships when multiple independent meta-models and models are involved [25]. As such, some enterprise architecture modelling approaches try to be

as comprehensive as possible up to a certain level of abstraction, providing a meta-model that approaches the different layers of an organization [23]. But the fact is that, many times, the integration of many meta-models is imperative in order to provide project or domain-specific solutions to many problems [23][28].

- Architecture Principle 3 - Extensibility: The architecture of the solution must cope with extensions because context modelling entails using multiple concurrent perspectives on the same problem. This derives from being able to answer to multiple concerns. Therefore, domain-specific and domain-independent models must coexist and the overall architecture must cope with multiple model transformation and integration. A specific concern is that the architecture is extensible to new application domains.

Using multiple meta-models often requires using multiple specialized tools, making it especially problematic to ensure the consistency across models, especially when the meta-models evolve [53]. The automatic or semi-automatic validation of the conformance of the models to the meta-models might be available or not, depending on whether the models are fully computable (abstract syntax and semantics) or not, or on the existing tool support.

- Architecture Principle 4 - Modularity: The architecture of the solution must follow the principles of high-cohesion and low-coupling. Observing these principles contributes to expressiveness and extensibility of the architecture. It is especially important that adding new domain-specific aspects to the model does not interfere with the concepts already present in the model.

Enterprise Architecture should also provide support to governance and decision making [1], [54], functioning as an important information source for informed decisions. Given that the purpose of models is to answer questions about the modelled entities [54], the ability to analyse the models for retrieving answers to the stakeholders is also desirable [55]. Stakeholders should be able to obtain as much useful information as possible from the knowledge contained in the models, which might reach a great level of complexity when elaborated with detail [56]. Such analysis can of course be made without any automation, however, it can be difficult obtain useful information when dealing with complex scenarios [57].

Given this, creating computable representations for enterprise architecture models comes out as a relevant need [58]. The combination of the computable models along with the enforcement of the dependencies brings benefits for enterprise architecture, such as retrieval, management, and processing of information. One example of such benefits is dependency analysis, which can be used for evaluating the alignment between the business and IT concepts.

- Architecture Principle 5 - Viewpoint-orientation: The architecture of the solution must support defining views over subsets of its concepts. This serves to facilitate the communication and the management of the models as viewpoints act as a separation of concerns mechanism. Viewpoints will facilitate addressing multiple concerns and can improve decision-making by isolating certain aspects of the architecture in views according to the needs of decision makers. In that sense, viewpoint specifications can be as simple as a filter that is applied on the overall constellation of enterprise architecture models, or as complex as an algorithm that uses the information contained on the models for performing a determined computation.

Next section described a proposal for the representation of enterprise architecture models that addresses the listed design principles.

# 4 Proposal: Representation of Enterprise Architecture Models Using Ontologies

Addressing the need for a representation that allows integration of different meta-models and their analysis by computational needs, our proposal is based on the hypothesis that ontologies are

able to represent, integrate and extend enterprise architecture models. Our approach is grounded in the following concepts:

- Upper Ontology (UO): a core meta-model, which represents a minimum set of enterprise architecture concepts and which can be considered domain-independent (i.e., that does not address any specific domain-dependent concerns and that can be applicable to the majority of the scenarios), thus addressing the architecture principles 1 and 2 (cf. section 3).
- Domain-Specific Ontologies (DSOs): In order to address the architecture principles 2 and 3 (cf. section 3), the upper ontology can be integrated with other domain-specific meta-models, in a varying number depending on the situation at hand, which are termed domain-specific ontologies. Each DSO represents a domain-specific language that addresses a particular set of concerns, and should also have the minimum set of concepts required for describing a determined domain, therefore addressing the architecture principle 2 (cf. section 3).

Following the architecture principle 5, the number of dependencies between the DSOs and the UO should be minimal, and each DSO should deal only with a set of domain-specific concerns. In this way, the addition of DSOs should have a minimum impact on the UO and existing DSOs.

For achieving traceability between the UO and the DSO and to fulfil the architecture principle 3, it is necessary to integrate the ontologies. Thus, ontology integration deals with the combination of the different ontologies for ensuring consistency and maximum coverage of the domain being addressed. In this particular case, the adopted technique is that of ontology mapping (cf. section 2.1), due to the fact that a DSO specializes some of the concepts present in the UO, but not all of them.

The simplest case is that of integrating the DSO with the core concepts represented in the UO. Cross DSO-DSO integration can also occur in cases where there is the need for adding more expressive power to specific domains. The ontology integration makes use of model transformation, which involves defining a mapping strategy from a source model to a target model [59], [60]. Figure 3 depicts the different types of transformation mapping strategies between the UO and the DSOs.

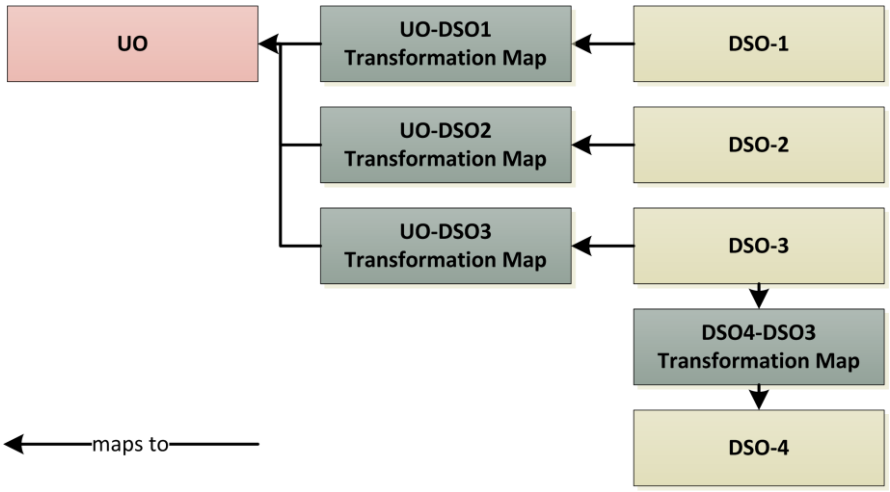


**Figure 3.** UO-DSO Integration Architecture

Considering the mapping between the UO and a DSO, it might be as simple as a 1:1 correspondence between the concepts of the two ontologies. However, it is expected that semantic mismatches might occur, due to evident differences in coverage, granularity and perspective (cf. section 2.3.1), as domain-specific languages might contain very specific concepts that cannot be mapped at all into the UO. Using the Bunge-Wand-Weber representation

model as an inspiration [61], one concept on the DSO might map to several concepts in the UO (overload), several concepts in the DSO might map to one concept in the UO (redundancy), a concept in the DSO might not map at all to the UO due to inexistence of adequate concepts in the upper ontology (excess), or some concepts in the UO might not be mapped by concepts in the DSO due to the inexistence of adequate concepts in the DSO (deficit). Figure 4 depicts the different types of semantic mismatches.

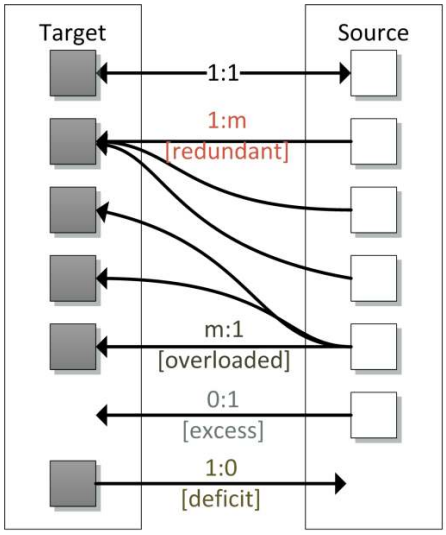


**Figure 4.** Types of potential semantic mismatches [62]

It is assumed that the mapping between some of the concepts of the DSO and some of the concepts of the upper ontology will always be possible, as the kinds of problems we are dealing are situated in the information systems domain. Redundancy, overload and deficit cases are thus expected to occur frequently. Cases where doubts might occur concerning the mapping between two concepts are carefully judged, with the mapping not being considered in cases of incompatibility.

One important aspect to refer is that this proposal would still acknowledge the use of existing tools and representation techniques for creating and managing the independent models. The models should then be converted to the UO or DSOs (i.e., creation of individuals), depending on the case. As the models are being converted, they should be verified for any inconsistencies according to an already existing meta-model expressed in an ontological representation. After that, the models can be used for performing analysis and can even be converted back into the original representation format if desired.

## 5 Proposal: Analysis of Enterprise Architecture Models Using Ontologies

In order to be able to address the need for information processing techniques in enterprise architecture, it is proposed that advantage is taken from the computational inference features of ontologies for performing enterprise architecture model analysis. The proposed architecture makes possible the usage of reasoning for performing analysis of the models, providing the required views for stakeholders, thus addressing the architecture principle 5 (cf. section 3). Four possible analysis configurations are shown in Figure 5 and are described below:

- Intra-UO reasoning, when inference is limited to the concepts of the UO. If the elements of the UO are organized in different layers, then inter-layer UO reasoning or intra-layer UO reasoning is possible.
- Intra-DSO reasoning, when inference is limited to the concepts of the DSO;

- Cross-DSO reasoning, whenever a mapping transformation between different DSOs is available, inference can use concepts from different DSOs;
- Cross UO-DSO reasoning, when inference uses concepts from both the UO and one or more DSOs, requiring a mapping transformation between each UO-DSO pair.

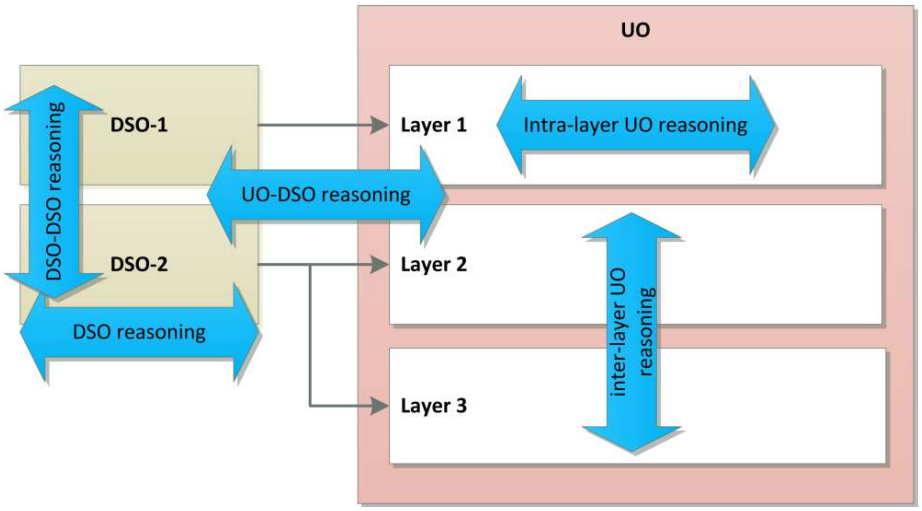


**Figure 5.** Reasoning configurations

By using DL as the formalism for the representation of the UO and DSOs, features such as instance checking, relation checking, consistency checking, dependency inferring and completeness checking are available through the use of reasoners. These features can thus be used for checking of the correctness of the enterprise architecture models. Some of the popular DL reasoners available are: FaCT++ [63], HermiT [64], Pellet [65], and TrOWL [66].

However, and in order to produce an analysis that is meaningful to the stakeholders, the models need to contain the information needed for performing the analysis, which always depends on the scenario at hand [54]. Thus, the addition of DSOs should take into account the following steps:

- Step 1 - Identify stakeholders and analysis needs: in this step, the identification of the stakeholders, their concerns, and analysis needs is performed. For this purpose, after the stakeholders are identified, their information needs are gathered under the form of questions and expected answers using a predefined template. After this, the questions need to be analysed in order to identify the concepts and instances if the concepts referred therein. Table 1 depicts example stakeholder questions after being analysed, with concepts highlighted in bold and instances highlighted in italics.

**Table 1.** Example stakeholder questions

| Question | Expected Output |
|---|---|
| Which **business processes** *BPA* depend on **business process** *BPB*? | List of **business processes** |
| Which **business actors** *BA* are required to execute **business process** *BP*? | List of **business actors** |
| What are the **technological entities** *T* supporting **business process** *BP*? | List of **technological entities** |

- Step 2 - Review enterprise architecture models: this step determines what is required from the models, and checks if the existing UO and DSOs can account for those requirements. If the existing models are not expressive enough for to cover the stakeholder needs, new concepts are to be added through a new DSO, which can be created from scratch, or by adopting an existing meta-model.
- Step 3 - Instantiate model: this step deals with the creation of a model instance for a specific scenario.
- Step 4 - Perform analysis: this step deals with the posing of the reasoning queries to the model instance. Questions such as the ones displayed in Table 1 need to be converted into DL to be posed to the models. The DL queries might involve any of the reasoning configurations discussed earlier in this section.

The next section describes an implementation of the proposal and a short example demonstrating the approach.

## 6 Implementation of the Proposal

For the implementation of the proposal, it was decided that an existing meta-model was to be adopted as the UO, describing the domain-independent aspects of the architecture, since several enterprise architecture meta-models are already available for use. Moreover, the focus of this research is not on creating enterprise architecture meta-models, but on providing an architecture where existing meta-models can be used and extended, using ontology technology.

This scenario was modelled using the standard ArchiMate 2.0 architecture description language. ArchiMate is a standard modelling language and framework from The Open Group[1] that covers the domain of Enterprise Architecture [5]. The language includes a minimum set of concepts and relationships and the framework includes a minimum set of layers and aspects required to enable modelling of the majority of cases. ArchiMate fits within the defined architecture principles (cf. section 3), namely:

- It is concern-oriented, thus addressing the architecture principle 1.
- It provides a high-level of abstraction, including a minimum set of concepts and relationships, with the framework including also a minimum set of layers and aspects to enable modelling of the majority of cases, thus addressing the architecture principle 2 .
- And it was designed with extensibility in mind, thus addressing architecture principle 3.
- It is viewpoint-oriented, thus addressing architecture principle 5.

The extensions already described in ArchiMate's specification are also considered part of the UO as they address aspects that are traversal to all the organizations.

To create the UO, the ArchiMate meta-model, along with the Motivation and Implementation and Migration extensions were specified in OWL-DL. OWL is a family of semantic web languages designed to represent rich and complex knowledge about things, groups of things, and relations between things [67]. When expressed in OWL, an ontology consists of axioms that place constrains on classes and on the relationships that are allowed between the classes. These axioms allow the systems to infer additional information based on the data explicitly provided. The data described by an ontology specified using one of the languages of the OWL family is interpreted as a set of "individuals" and a set of "properties" which relate these individuals to each other. Three different languages are available: OWL Lite, OWL-DL and OWL Full.

The choice of OWL-DL enables taking advantage of the different computational inference and querying mechanisms already existing, and as a result, being able to perform analysis of the model for assessing the consistency of models against rules, verify the completeness of models, and for decision making through the production of reports based on contents of the model. The method employed for creating an ArchiMate representation in OWL involved three steps: (i)

[1] http://www.opengroup.org/

transform the ArchiMate meta-model; (ii) adding axioms and cardinalities; and (iii) transforming the ArchiMate models.

In step (i), an analysis of ArchiMate's meta-model was performed concept-by-concept, including the relations with other concepts and the constraints existing in those relations. The result was the mapping of concepts into OWL classes and the mapping of relations into OWL `ObjectProperties`. In step (ii), restrictions were added to the properties, such as `InverseObjectProperties` and `SuperObjectProperties` axioms were added to the OWL ontology, so that derived relationships can be inferred. For instance, the expression: `triggeredBy InverseOf triggers` declares that the `triggers ObjectProperty` has an `InverseObjectProperty` named `triggeredBy`. Axioms were added to ensure the compliance against the ArchiMate specification, including cardinality. For instance, the expressions

```
BusinessFunction and hasAspect exactly 1 Thing
BusinessFunction and hasAspect some BehavioralAspect
BusinessFunction and hasLayer exactly 1 Thing
BusinessFunction and hasLayer some BusinessLayer
```

specify that the OWL class `BusinessFunction` covers exactly one aspect, i.e, `BehavioralAspect`, and i.e., `BusinessLayer`. Step (iii) is scenario dependent and involves creating individuals of belongs exactly to one layer, the classes existing in the ArchiMate ontology created in the two previous steps, which correspond to the elements modelled in an ArchiMate model.

Figure 6 depicts an ArchiMate model using the ArchiSurance example [5]. After the conversion of the model into OWL (step (iii) above), and by using the reasoning capabilities, a question such as the following can be answered for validating the correctness of the ontology:

- What `BusinessServices` are used by the `Customer BusinessRole`?
- What `BusinessProcesses` are used by the `Customer BusinessRole`?
- What ArchiMate concepts belong to the `ApplicationLayer`?
- What ArchiMate concepts are `BehaviouralAspect`?

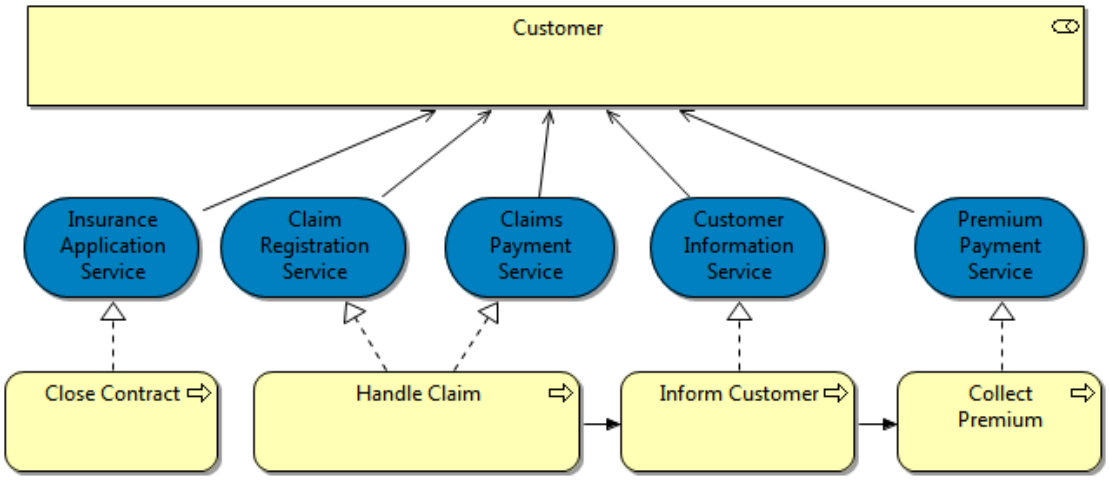


**Figure 6.** ArchiMate example: service realization view

These questions can be formalised in DL as demonstrated below.

```
BusinessService and usedBy value Customer
BusinessProcess and
  realizes some (BusinessService and usedBy value Customer)
  hasLayer some ApplicationLayer
  hasAspect some BehavioralAspect
```

Figure 7 depicts the results of the execution of the statements by the HermiT reasoner, which are according to what can be observed in Figure 6.

Concerning the ontology mapping, the classes belonging to different ontologies can be mapped in various ways, depending on the existing semantic relationship. In case of equivalence between the classes, OWL already offers the `EquivalentTo` property. Or if a concept in the DSO can be considered a specialization of a class in the UO, probably one should be state as a `SubClass` of the other. Other more specialized maps can be created between the elements of the UO and DSO by using the `ObjectProperties` already provided by the ArchiMate ontology.

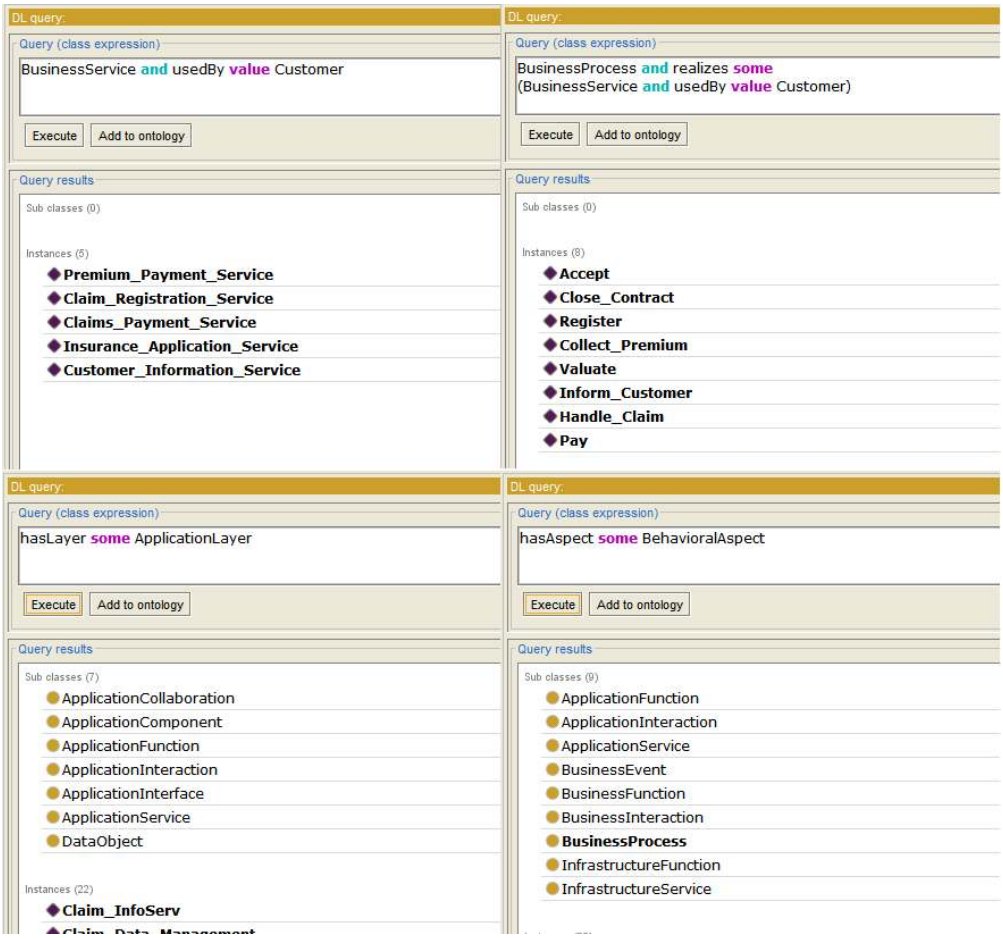


**Figure 7.** Example query results

## 7 Case Study

In order to demonstrate the utility of the proposals made in this article, this section describes the application of the proposal to a concrete scenario. This application includes the instantiation of the UO, instantiation of a DSO, and some examples of the analysis that can be performed using the two.

The case concerns a regulatory organization that assesses and monitors the structural safety of large engineering infrastructures, such as hydroelectric power plants, dams and bridges. This is accomplished through tasks performed at different stages of the structure's life-cycle, ranging from project planning, construction, and operation, to the structure's retirement. A significant part of the operational monitoring is performed by an array of specialized sensors that measure the physical behaviour of the structure. This organization is also required to document and preserve the business processes, information systems and technology that support the acquisition, processing and storage of the data pertaining to each structure. Thus EA plays an important role as a means to manage these artefacts.

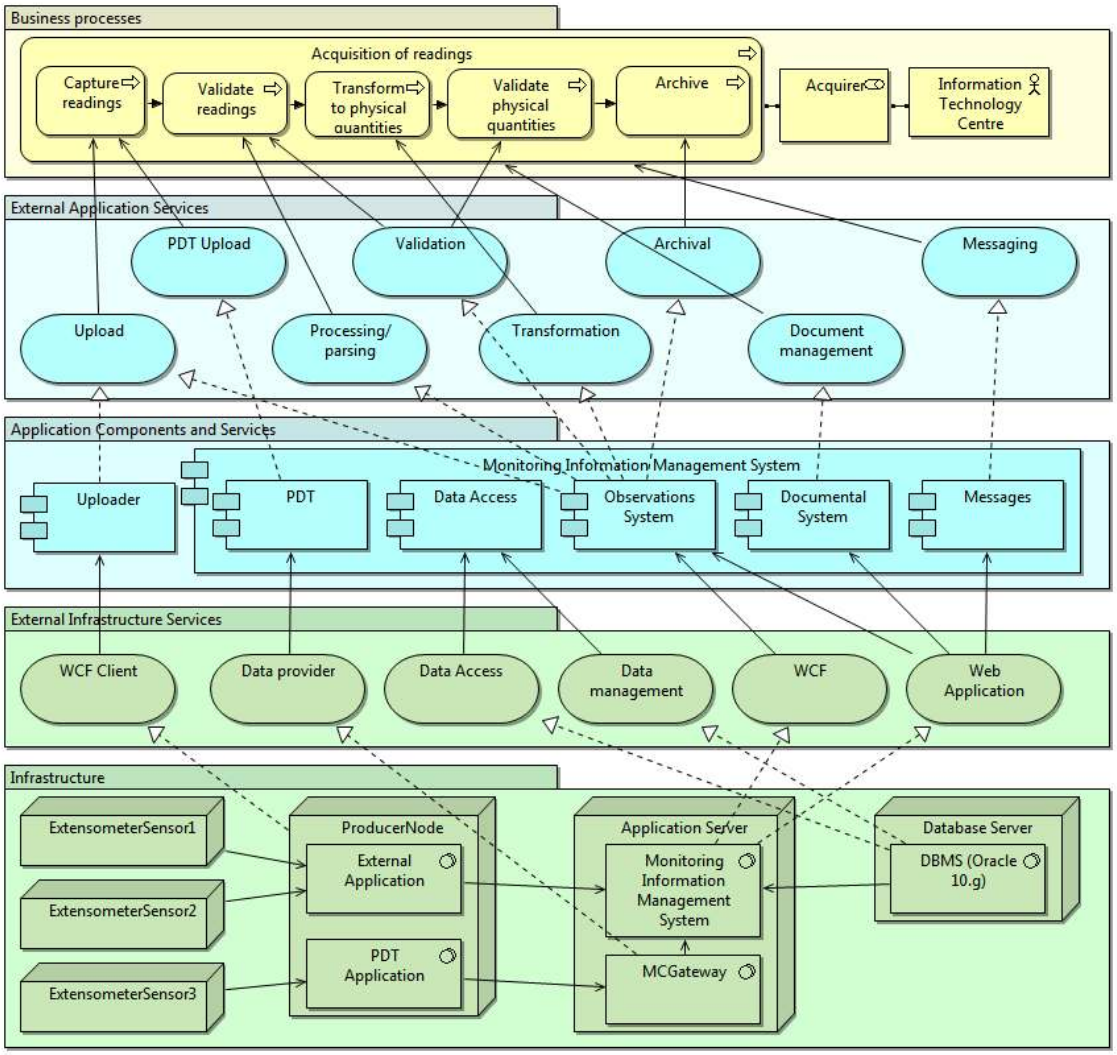


**Figure 8.** Case study ArchiMate layered view

## 7.1 Model Representation

An ArchiMate model of the case study scenario was created and converted into OWL, with Figure 8 depicting a resulting ArchiMate view. However, the organizational stakeholders required modelling and analysing specific information about sensors. Following the steps described in section 5 concerning the analysis needs of stakeholders, it was identified that the expressiveness of the concepts contained in the UO was not enough for capturing the needed information. As a result, a specific description language needed to be defined, which resulted in a DSO for capturing sensor related information as depicted in Figure 9.

After the creation of the DSO, it was time for the mapping of the ontologies. When analysing the semantic mismatches existing between the two ontologies, different alternatives were considered for creating the mappings. Statements such as `EquivalentTo`, intuitively were a clear candidate. However, in this particular case the concepts in the DSO were rather specializations of the concepts in the UO. Therefore, `SubClass` statements were employed on the mappings. Table 2 depicts the determined mappings between elements the sensor DSO and the corresponding elements in the UO. Since this DSO is supposed to capture specialized information on sensors, as expected, some of the concepts in it will have no match in the UO, and vice-versa.

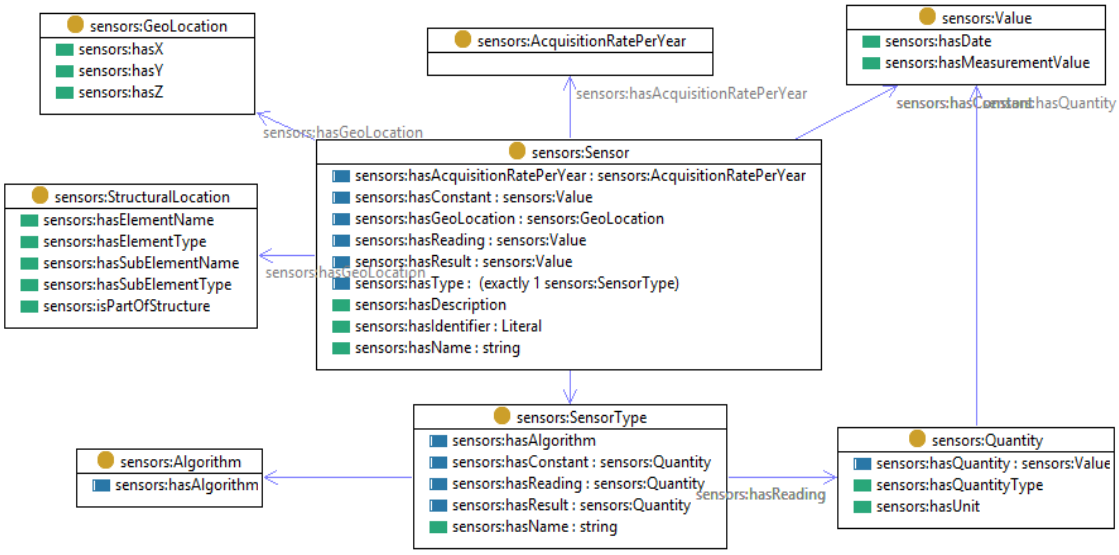


**Figure 9.** Overview of the sensors DSO

**Table 2.** Mapping between the Sensor DSO and UO

| Sensor DSO | UO |
|---|---|
| Sensor | Node |
| GeoLocation | Location |
| GeoLocation | Location |
| Algorithm | ApplicationComponent |
| Value | DataObject |
| AcquisitionRatePerYear | DataObject |

The process could be adopted to any other DSO to be integrated according to the needs of the stakeholders, thus demonstrating that the concern orientation, expressiveness, and extensibility principles are respected. Additionally, the mapping statements are done on a different ontology that references the concepts declared in both ontologies. This allows for the UO and DSOs to remain unchanged, as no new information is added to them, thus respecting the modularity principle.

### 7.2 Model Analysis

The different reasoning configurations described in section 5 can then be used for the purpose of identifying the dependencies between elements more easily. As such, two `SuperObjectProperty` chains were created and added to the ontology containing the mapping statements with the purpose of modelling dependencies between different elements. A `dependsDown ObjectProperty` was added as a `SuperObjectProperty` of the aggregation, composition, assignment, usage, and realization `ObjectProperties` in the UO, while the `dependsUp ObjectProperty` fills the same

purpose, being an InverseObjectProperty of the former. Since these properties are transitive, a graph of dependencies can be created.

For instance, the question "what are the technological entities supporting the process acquisition of readings?" can be translated into the DL query

```
Thing and
hasLayer some TechnologyLayer and
hasAspect some BehavioralAspect or
hasAspect some ActiveStructuralAspect and
dependsDown value Acquisition_of_readings
```

which uses elements belonging exclusively to the UO, thus being an example of intra-UO reasoning, more precisely, inter-layer-UO reasoning. The results of the execution of this query by the HermiT reasoner can be seen in Figure 10.

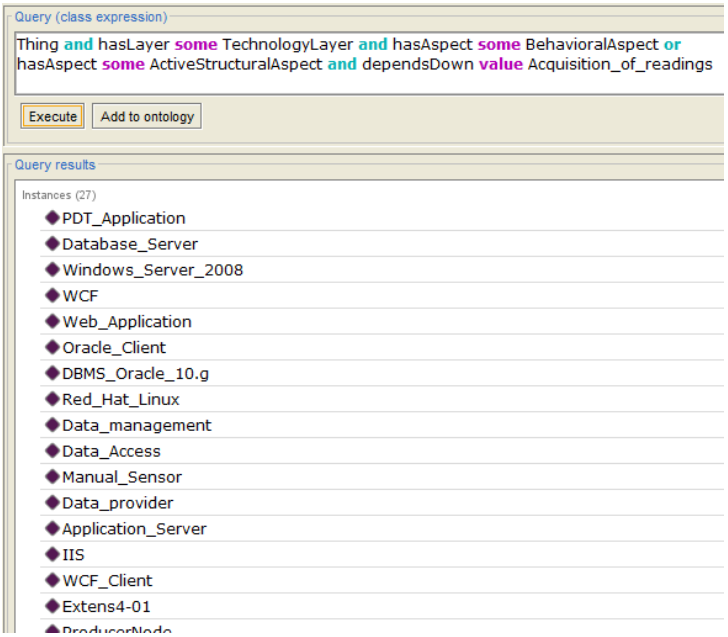


**Figure 10.** Intra-UO reasoning example

Another example, is the question "which ApplicationComponents were responsible for performing the acquisition and transformation of the readings for SensorType Drain?" For being able to answer this question, we need elements either from the UO (i.e., ApplicationComponents) and the sensor DSO (i.e., SensorType). The DL query

```
ApplicationComponent and
dependsUp some (Sensor and hasSensorType value Drain)
```

provides the results that can be seen Figure 11, thus being an example of cross-UO-DSO reasoning.

The execution of these queries will thus highlight the dependencies between the different elements of the infrastructure, and can be used as a valuable tool for decision making. By identifying the dependencies, it is possible to trace the propagation of the changes throughout the architecture. That information can then be used by the organization for decision making. Moreover, since the architecture can be enriched with the addition of new DSOs, other types of semantics might be included on the models, making possible that other kind of decision-making analysis can be performed. For instance, if a DSO that includes runtime data automatically

captured from the environment, the impact might even be automatically quantified. The addressing of the analysis needs of the case stakeholders thus demonstrates that the viewpoint-orientation architecture principle is thus demonstrated in this case study.

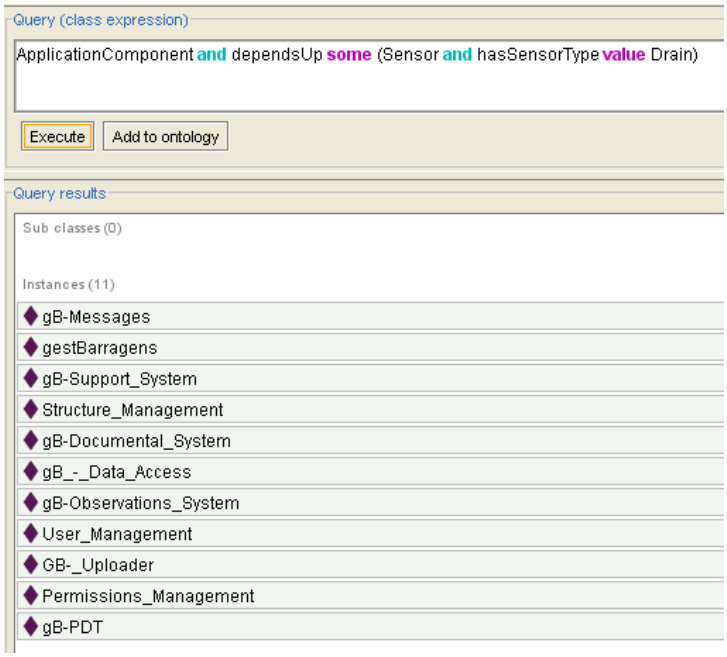


**Figure 11.** Cross-UO-DSO reasoning example

## 8 Conclusions

This work raised three research questions which respectively focused in model representation, model support for analysis, and model analysis techniques. The use of ontologies for the representation of enterprise architecture models was hypothesized as an answer to those questions. Therefore, and in order to test the hypothesis, an ontology-based approach to the representation and analysis of enterprise architecture models was proposed. It consisted of an extensible architecture, based on architecture principles for enforcing meta-model coherence, model conformance, analysis capabilities and ontological integration of stakeholder viewpoints.

The proposal was then implemented and demonstrated in a case study that, on the one hand, addressed RQ1 and RQ2 by showing the extensibility offered by the approach and, on the other hand, addressed RQ2 and RQ3 by showing the analysis capabilities brought by the approach. The implementation of the proposal and the application to a case study also made clear some limitations that should be addressed in the future, namely:

- The analysis necessities of organizations might imply the addition of many DSOs, which might increase the complexity of the overall enterprise architecture and harden the task of managing the different ontologies.
- The necessity for an adequate tool support to manage the integrated ontologies, including the addition of new DSOs and their population.
- The lack of an adequate integrated visualization of the ontologies and of the analysis outcomes, which would benefit of an integrated graphical representation.

Nonetheless, it is considered that this proposal has relevant impact in the communities of enterprise architecture and ontology engineering, providing a bridge between the two fields. The enterprise architecture community benefits from having models that, besides documenting the organization, can enhance decision making. On the other hand, the ontology engineering community benefits from the application of the body of knowledge and techniques in solving concrete organizational problems.

In line with the above, future work will follow several lines of work, which include: applying the approach to new case studies, highlighting the different analysis capabilities; adding tool support through the creation of tools that facilitate the management and population of the ontologies; and the creation of a frontend to visualize the ontologies and analysis results in a graphical fashion. Another possible line of work is the creation of automatic data extraction and ontology population tools to feed the ontologies with real data, which can facilitate the process of creating models of the organization or even be used for checking the conformance of the models to reality.

## Acknowledgements

This work was supported by national funds through FCT - Fundação para a Ciência e a Tecnologia, under project PEstOE/EEI/LA0021/2013 and the grant (SFRH/BD/69121/2010) to Gonçalo Antunes, by COMET K1, FFG - Austrian Research Promotion Agency, by the Vienna Science and Technology Fund (WWTF) through project ICT12-046 (BenchmarkDP), and by the European Commission under the 7th Framework Programme for research and technological development and demonstration activities (FP7/2007-2013) under grant agreement no. 269940 (TIMBUS project).

## References

[1] M. Lankhorst, Enterprise Architecture at Work: Modeling, Communication, and Analysis. Springer, 2005.

[2] J. A. Zachman, “Enterprise Architecture: The issue of the century,” Database Programming and Design, vol. 10, no. 3, 1997, pp. 44–53.

[3] J. Schekkerman, How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. Trafford Publishing, 2006.

[4] The Open Group, TOGAF 9 - the open group architecture framework version 9, 2009.

[5] The Open Group, ArchiMate 2.0 Specification, 2012.

[6] J. Saat, U. Franke, R. Lagerstrom, and M. Ekstedt, “Enterprise Architecture meta models for IT/business alignment situations,” in 2010 14th IEEE International Enterprise Distributed Object Computing Conference, 2010.

[7] S. Buckl, F. Matthes, and C. M. Schweda, “Conceptual models for cross-cutting aspects in Enterprise Architecture modeling,” in 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2010, pp. 245-252. Available: http://dx.doi.org/10.1109/EDOCW.2010.18

[8] S. Buckl, C. M. Schweda, and F. Matthes, “A design theory nexus for situational Enterprise Architecture management,” in 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2010, pp. 3-8. Available: http://dx.doi.org/10.1109/EDOCW.2010.27

[9] A.Caetano, C. Pereira, and P. Sousa, “Generating multiple consistent views from business process models," in Lecture Notes in Business Information Processing, Research and Practical Issues of Enterprise Information Systems, Springer-Verlag, Berlin, Heidelberg, 2011.

[10] M. Galster, “Dependencies, traceability and consistency in software architecture: towards a view-based perspective,” in ECSA ’11 Proceedings of the 5th European Conference on Software Architecture: Companion Volume, 2011. Available: http://dx.doi.org/10.1145/2031759.2031761

[11] J. Romero, J. I. Jaen, and A. Vallencillo, “Realizing correspondences in multi-viewpoint specifications,” in Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference, 2009, pp. 163-172. Available: http://dx.doi.org/10.1109/EDOC.2009.23

[12] R. France and B. Rumpe, “Model-driven development of complex software: A research roadmap,” in Future of Software Engineering 2007 (FOSE ’07), 2007, pp. 37-54. Available: http://dx.doi.org/10.1109/FOSE.2007.14

[13] M. Bjekovic, E. Proper, and J.-S. Sottel, “Towards a coherent enterprise modelling landscape,” in Sandkuhl, K. (ed.), Emerging Topics in the Practice of Enterprise Modeling : short paper proceedings of 5th IFIP WG8.1 Working Conference on the Practice of Enterprise Modeling, Rostock, Germany, November 7-8, 2012, 2012.

[14] J. P. A. Almeida, M. E. Iacob, and P. van Eck, “Requirements traceability in model-driven development: Applying model and transformation conformance,” Inf Syst Front, vol. 9, 2007, pp. 327–342. Available: http://dx.doi.org/10.1007/s10796-007-9038-3

[15] M. Buschle, P. Johnson, and K. Shahzad, "The Enterprise Architecture Analysis Tool - Support for the Predictive, Probabilistic Architecture Modeling Framework", in the Proceedings of the 19th Americas Conference on Information Systems, Chicago, Illinois, August 14-17, 2013.

[16] T. R. Gruber, "A translation approach to portable ontology specifications," Knowledge Acquisition, vol. 5, 1993, pp. 199-220. Available: http://dx.doi.org/10.1006/knac.1993.1008

[17] D. Kang, J. Lee, S. Choi, and K. Kwangsoo, “An ontology-based Enterprise Architecture,” Expert Systems with Applications, vol. 37, 2010, pp. 1456–1464. Available: http://dx.doi.org/10.1016/j.eswa.2009.09.052 http://dx.doi.org/10.1016/j.eswa.2009.06.073

[18] G. Wagner, “Ontologies and rules for enterprise modeling and simulation,” in 2011 15th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2011, pp. 385-394. Available: http://dx.doi.org/10.1109/EDOCW.2011.68

[19] C. L. B. Azevedo, J. P. A. Almeida, M. van Sinderen, D. Quartel, and G. Guizzardi, “An ontology-based semantics for the motivation extension to archimate,” in 2011 15th IEEE International Enterprise Distributed Object Computing Conference, 2011, pp. 25-34. Available: http://dx.doi.org/10.1109/EDOC.2011.29

[20] J. P. A. Almeida and G. Guizzardi, “An ontological analysis of the notion of community in the RM-ODP enterprise language,” Computer Standards & Interfaces, vol. 35, no. 3, 2013, pp. 257 – 268. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548912000402 http://dx.doi.org/10.1016/j.csi.2012.01.007

[21] H. H. Hoang, J. J. Jung, and C. P. Tran, “Ontology-based approaches for cross-enterprise collaboration: a literature review on semantic business process management,” Enterprise Information Systems, 2013, pp. 1–17. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/17517575.2013.767382 http://dx.doi.org/10.1080/17517575.2013.767382

[22] P. S. Santos Jr., J. P. A. Almeida, and G. Guizzardi, “An ontology based semantic foundation for organizational structure modeling in the Aris method,” in 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2010.

[23] R. Fischer, S. Aier, and R. Winter, "A federated approach to enterprise architecture model maintenance", In Enterprise Modeling and Information Systems Architectures, Vol. 2, Issue, 2, 2007, pp. 14 - 22.

[24] H. Kühn, F. Bayer, S. Junginger, and D. Karagiannis, "Enterprise Model Integration", In Proceedings of the 4th International Conference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, 2003.

[25] M. Lankhorst, "Enterprise architecture modeling - the issue of integration", in Advanced Engineering Informations, vol. 18, Issue 4, 2006, pp. 205-216.

[26] H. Jonkers, M-E Jacob, M. Lankhorst, and P. Strating, "Integration and Analysis of Functional and Non-Functional Aspect in Model-Driven E-Service Development", In Proceedings of the 2005 9th IEEE International EDOC Enterprise Computing Conference (EDOC '05), 2005.

[27] F. Arbab, F. de Boer, M. Bonsangue, M. Lankhorst, H. A. Proper, and L. van der Torre, "Integrating Architectural Models: Symbolic, Semantic and Subjective Models in Enterprise Architecture", In Enterprise Modelling and Information Systems Architectures, Vol. 2, Issue 1, 2007, pp. 40-57.

[28] S. Zivkovic, H. Kühn, and D. Karagiannis, "Facilitate modeling using method integration: An approach uisng mapping and integration", In Proceedings of the 15th European Conference in Information Systems, St. Gallen, Switzerland, 2007.

[29] S. Kurpjuweit and R. Winter, "Concern-oriented Business Architecture Engineering", In Proceedings of SAC '09, Honululu, Hawaii, USA, 2009. Available: http://dx.doi.org/10.1145/1529282.1529339

[30] E. Grandy, C. Feltus, and E. Dubois, "Conceptual Integration of Enterprise Architecture Management and Security Risk Management", In Proceedings of the 2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2013. Available: http://dx.doi.org/10.1109/EDOCW.2013.19

[31] A.Zimmermann, M. Pretz, G. Zimmermann, D. G. Firesmith, I. Petrov, and E. El-Sheikh, "Towards Service-oriented Enterprise Architectures for Big Data Applications in the Cloud", In Proceedings of the 2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2013. Available: http://dx.doi.org/10.1109/EDOCW.2013.21

[32] K. Breitman, M. A. Casanova, and W. Truszkowski, Semantic web: concepts, technologies and applications, Springer, 2007.

[33] A.Gómez-Pérez and R. Benjamins, "Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods", Proceedings of IJCAI-99 Workshop on Ontologies and Problem Solving Methods (KRR5), Stockholm, Sweden, 1999.

[34] D. Fensel, D. McGuiness, E. Schulten, W. K. Ng, G. P. Lim, and G. Yan, "Ontologies and electronic commerce," Intelligent Systems, IEEE, vol. 16, no. 1, 2001, pp. 8-14. Available: http://dx.doi.org/10.1109/MIS.2001.1183337

[35] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," Knowledge engineering review, vol. 11, no. 2, 1996, pp. 93-136. Available: http://dx.doi.org/10.1017/S0269888900007797

[36] N. Guarino, "Formal Ontology in Information Systems", Proceedings of the First International Conference (FIOS'98), June 6-8, Trento, Italy, vol. 46, IOS press, 1998, pp. 3-15.

[37] D. L. McGuinness, "Ontologies come of age," Spinning the semantic web: bringing the World Wide Web to its full potential, p. 171, MIT Press, 2005.

[38] B. C. Grau, I. Horricks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "OWL 2: The Next Step for OWL," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 6, 2008, pp. 309-322. Available: http://dx.doi.org/10.1016/j.websem.2008.05.001

[39] H. S. Pinto, A. Goméz-Pérez, and J. P. Martins, "Some issues on ontology integration", In Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends, 1999.

[40] J. Euzenat and P. Shvaiko, Ontology matching, Springer, 2007.

[41] J. Davies, R. Studer and P. Warren, Semantic Web Technologies: Trends and Research in Ontology-based Systems, Wiley, 2006. Available: http://dx.doi.org/10.1002/047003033X

[42] Y. K. Hooi, M. F. Hassan and A. M. Shariff, "A Survey on Ontology Mapping Techniques," in Advanced in Computer Science and its Applications, Springer, 2014, pp. 829-836. Available: http://dx.doi.org/10.1007/978-3-642-41674-3_118

[43] Y. Kalfoglou and M. Schorlemmer, "Ontology mapping: the state of the art," The knowledge engineering review, vol. 18, no. 1, 2003, pp. 1-31. http://dx.doi.org/10.1017/S0269888903000651

[44] K. Kotis, G. A. Vouros and K. Stergiou, "Towards automatic merging of domain ontologies: The HCONE-merge approach," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 4, no. 1, 2006, pp. 60-79. Available: http://dx.doi.org/10.1016/j.websem.2005.09.004

[45] S. Amrouch and S. Mostefai, "Survey on the literature of ontology mapping, alignment and merging," in Information Technology and e-Services (ICITeS), 2012 International Conference on, 2012, pp. 1-5. Available: http://dx.doi.org/10.1109/ICITeS.2012.6216651

[46] P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou and S. Tessaris, “D2. 2.1 Specification of a common framework for characterizing alignment,” 2004.

[47] S. M. Falconer, N. F. Noy and M.-A. D. Storey, “Ontology Mapping - a User Survey” in OM, 2007.

[48] M. Lenzerini, D. Milano, and A. Poggi, “Ontology representation & reasoning,” Universit di Roma La Sapienza, Roma, Italy, Tech. Rep. NoE InterOp (IST-508011), 2004.

[49] O. Corcho, M. Fernandez-López and A. Gómez-Pérez, “Ontological engineering: principles, methods, tools and languages,” in Ontologies for software engineering and software technology, Springer, 2006, pp. 1-48. http://dx.doi.org/10.1007/3-540-34518-3_1

[50] F. Baader, I. Horrocks and U. Sattler, “Description logics,” Foundations of Artificial Intelligence, vol. 3, 2008, pp. 135-179. http://dx.doi.org/10.1016/S1574-6526(07)03003-9

[51] D. Greefhorst and E. Proper, Architecture Principles: The Cornerstones of Enterprise Architecture. Springer, 2011. Available: http://dx.doi.org/10.1007/978-3-642-20279-7

[52] ISO/IEC/IEEE 42010:2011 - Systems and Software Engineering - Architecture Description, International Organization for Standardization, International Electrotechnical Commission and Institute of Electrical and Electronic Engineers Std.

[53] G. Kramler, G. Kappel, T. Reiter, E. Kapsammer, W. Retschitzegger, and W. Schwinger, "Towards a semantic infrastructure supporting model-based tool integration", In Proceedings of the 2006 International Workshop on Global Integrated Model Management (GaMMa '06), 2006, pp. 43-46. Available: http://dx.doi.org/10.1145/1138304.1138314

[54] P. Johnson and M. Ekstedt, Enterprise Architecture: Models and Analyses for Information Systems Decision Making. Lightning Source Incorporated, 2007. [Online]. Available: http://books.google.pt/books?id=2LdxPQAACAAJ

[55] M. Buschle, J. Ullberg, U. Franke, R. Lagerstrom, and T. Sommestad, “A tool for Enterprise Architecture analysis using the PRM formalism,” in CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers, 2010, pp.108-121. Available: http://dx.doi.org/10.1007/978-3-642-17722-4_8

[56] T. Binz, F. Leymann, A. Nowak, and D. Schumm, “Improving the manageability of enterprise topologies through segmentation, graph transformation, and analysis strategies,” in 2012 16th IEEE International Enterprise Distributed Object Computing Conference (EDOC), 2012. Available: http://dx.doi.org/10.1109/EDOC.2012.17

[57] S. Buckl, M. Buschle, P. Johnson, F. Matthes, and C. M. Schweda, "A meta-language for Enterprise Architecture analysis", In Proceedings of the 16th International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2011), London, United Kingdom, 2011, pp. 511-525. Available: http://dx.doi.org/10.1007/978-3-642-21759-3_37

[58] R. A. Martin, E. L. Robertson, and J. A. Springer, "Architectural Principles for Enterprise Frameworks: Guidance for Interoperability", In Proceedings on the International Conference on Enterprise Integration Modelling and Technology 2004 (ICEIMT 2004), Toronto, Canada, 2005, pp. 79-91. Available: http://dx.doi.org/10.1007/0-387-29766-9_7

[59] G. Guizzardi, “Ontological foundations for structural conceptual models,” Ph.D. dissertation, University of Twente, Enschede, The Netherlands, 2005.

[60] M. Rosemann, P. Green, and M. Indulska, “A reference methodology for conducting ontological analyses,” in Proceedings of the 23rd International Conference on Conceptual Modelling (ER 2004), 2004, pp. 110-121. Available: http://dx.doi.org/10.1007/978-3-540-30464-7_10

[61] M. A. Bunge, Treatise on Basic Philosophy Volume 3: Ontology I - The Furniture of the World. Kluwer Academic Publishers, 1977.

[62] R. Weber, Ontological Foundations of Information Systems, Coopers and Lybrand and the Accounting Association of Australia and New Zealand, Melbourne, Australia, 1997.

[63] D. Tsarkov and I. Horrocks, “FaCT++ description logic reasoner: System description,” in Automated reasoning, Springer, 2006, pp. 292-297. Available: http://dx.doi.org/10.1007/11814771_26

[64] B. Motik, R. Shearer, and I. Horrocks, “Hypertableau reasoning for description logics,” Journal of Artificial Intelligence Research, vol. 36, no. 1, 2009, pp. 165-228. Available: http://dx.doi.org/10.1613/jair.2811

[65] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner,” Web Semantics: science, services and agents on the World Wide Web, vol. 5, no. 2, 2007, pp. 51-53. Available: http://dx.doi.org/10.1016/j.websem.2007.03.004

[66] E. Thomas, J. Z. Pan, and Y. Ren, “TrOWL: Tractable OWL 2 reasoning infrastructure,” in The Semantic Web: Research and Applications, Springer, 2010, pp. 431-435. Available: http://dx.doi.org/10.1007/978-3-642-13489-0_38

[67] W3C, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)", W3C Recommendation, 2012.