A Modular Ontology for the Enterprise Architecture Domain

Marzieh Bakhshandeh[†],Gonçalo Antunes [†], Rudolf Mayer[‡], José Borbinha[*][†], and Artur Caetano[*][†]
{marzieh.bakhshandeh, goncalo.antunes, jose.borbinha, artur.caetano}@ist.utl.pt, rmayer@sba-research.at
[*]Instituto Superior Técnico, Technical University of Lisbon, Portugal
[†]INESC-ID - Information Systems Group, Lisbon, Portugal
[‡]Secure Business Austria, Vienna, Austria

Abstract—**Enterprise architecture supports the analysis, design and engineering of business-oriented systems through multiple views. Each view expresses the elements and relationships of a system from the perspective of specific system concerns relevant to one or more of its stakeholders. As a result, each view needs to expressed in the architecture description language that best suits its concerns. Therefore, an enterprise architecture may be described using a set of different languages. However, current enterprise architecture modelling languages display two issues in this setting. First, they lack mechanisms to integrate multiple architecture description languages. This issue hinders the specification of views using different languages. Second, enterprise architecture models lack quantitative analysis support. This paper describes an ontology-based approach in order to have a modular ontology for the enterprise architecture domain, to specify and integrate multiple architecture modelling languages and to analyse the resulting integrated models. The approach relies on transformations between an upper-domain ontology based on the ArchiMate language and on a set of domain-specific ontologies to deal with the specific architecture modelling languages. The resulting models are quantifiable in the sense they provide the means to assess the consistency of the enterprise architecture models and to analyse their structure. The applicability of the approach is shown through a case study and the correctness of the ontology is shown by a set of competency questions.**

Keywords*—*enterprise architecture, ontology, integration, modular ontology, ArchiMate, OWL.

I. Introduction

Enterprise Architecture is a set of descriptive representations relevant for describing an enterprise so that it can realize management requirements and be maintained over the period of its useful life [1]. According to the ISO/IEC/IEEE 42010 standard [2], the usage of multiple views is fundamental to describe an architecture of a system. A system has multiple stakeholders, each with specific interests on the system. An architecture description should aggregate multiple views, materialized in a set of models, that are formulated according to viewpoints expressing the concerns of the stakeholders of the system-of-interest [15]. Stakeholders play a fundamental role in the development of enterprise architecture, as they have interest in reflecting their concerns on the architecture of the system. As such, enterprise architects need to conceive views from the viewpoint of the stakeholders and to address their concerns and requirements. However, the existence of semantic gaps between architects and stakeholders may produce conceptual misalignments which can negatively affect the architecture [3]. Technical level evaluations are sometimes performed, but are still not enough for addressing the issue. Hence, an evaluation method should be in place so that the concerns of the different stakeholders and the requirements that they expect to be fulfilled are all answered.

Enterprise architecture covers a broad range of aspects, from the infrastructure layer, through software applications running on top of the infrastructure, to business processes supported by these applications. Within each of these layers and also the relations between the layers analysis techniques can be applied, which require detailed models with semantically rich definitions as input [4].

This paper applies ontologies and model transformation for the specification and integration of enterprise architecture models in order to have a Modular Ontology for the Enterprise Architecture Domain. Modular Ontology refers to a methodological principle in ontology engineering. Modularization [5] in itself is a generic concept that is intuitively understood as referring to a situation where simultaneously a thing (e.g. an ontology) exists as a whole but can also be seen as a set of parts (the modules). Hence, in the perspective of ontologies a module is a sub-ontology that can be connected to other sub-ontologies by integration and would be able to interact among each other. This ontological modularity strategy will help to reduce the complexity of designing and to facilitate ontology reasoning, development, and integration. This paper describes an ontological modularity approach to specify and integrate multiple architecture modelling languages and to analyse the resulting integrated models. The concepts are described on an upper-domain ontology (UDO) containing the fundamental concepts of an enterprise architecture, which can be assumed as a module. Where, each set of specific concerns are represented using domain-specific ontologies that would also assume to be modules, that are linked to the other module by integration. Logical reasoning is applied to check the inconsistencies on the models and for inferring different dependencies between different elements of enterprise models, which can be used for analysing the impact of changes on architectural elements. Furthermore, this approach contributes in filling in the semantic gap between stakeholders and IT architect by integrating domain-specific ontologies.

The outline of this paper is as follows. Section II describes the related work. Section III presents the ontological modularity approach and the architecture of the proposal for model architecture analysis. In section IV the proposal architecture is applied to enterprise architecture. In section V, a validation through a case study has been done to prove that the proposal is applicable and satisfiable in enterprise architecture modelling languages. Finally VI presents our conclusions and discusses future work.

II. Related Work

A number of different approaches for evaluating EA models have been described in the past few years. In [6], and [7], the authors have implemented a tool for analysing EA models, which guides the creation of enterprise information system scenarios in the form of enterprise architecture models and generates quantitative assessments of the scenarios as they evolve. In [8], the authors propose an uniform approach for capturing quality attribute requirements and analysing system and software architecture. However, such approach did not considered the business architecture[9].

In [10], the author presents an approach for quantitative analysis of layered, service-based enterprise architecture models, which can be used as an analysis framework where existing methods for detailed performance analysis,based on, e.g., querying models, Petri nets or simulation, can be plugged in. In [11], a semantic model is defined by the author, which is formal interpretation of the symbolic model which provides a formal approach to the design of architectural description languages and a general mathematical foundation for the use of formal methods in enterprise architectures analysis.

Some works have dealt with the adoption of an ontology-based approach for Enterprise model analysis. In [12], the author presents a core ontology for Business Process Analysis (BPA). The ontology builds upon a Time Ontology and is structured around the process, resource, and object perspectives as typically adopted when analysing business processes. The ontology has been extended and validated by means of an Events Ontology.

Ontologies have been used for the analysis of modelling languages. In [13], the author presents a framework which verifies how clear and expressive a modelling language is, by focusing on its notation. The proposed framework is based on the construction of an ontology to describe the conceptual domain of discourse. This ontology is then used as a type of ‘mirror’ for the modeling language, i.e. for verifying how well this modelling language is able to represent the concepts and relations represented in the ontology [14]. This foundational ontology has been used with success in a set of Enterprise Modelling approaches such as ARIS framework [15], and in [16] which provides a semantic foundation for role-related concepts in enterprise modelling.

III. Architecture

An ontology describes a domain model by associating meaning to its terms and relations. According to Gruber[17] a formal ontology is an ontology with structure which tries to provide a domain and application independent view on reality and remain consistent with increasing content and also context independence. In particular, formalized enterprise models can support new methods and tools for various fields such as business model design, business strategy, information systems alignment and enterprise architecture. Therefore, different concerns of different stakeholders which have different vocabularies can be integrated by the use of two kind of ontologies: a domain ontology that will describe the sources of a problem domain, and an Enterprise ontology, that will present the fundamental concepts of enterprise architecture. The integrated ontology will be able to answer more comprehensive set of questions about organizations, and will also able us to analyse and evaluate the models more precisely by reasoning the stakeholders questions and concerns in the integrated ontology.

The ontological modularity architecture is grounded on the following concepts:

- Upper domain ontology (UDO).
- Domain-specific ontology (DSO).
- Ontology integration.

these concepts will be explained in detail, in the next subsections.

A. Upper-domain Ontology

This kind of ontology tries to describe the concepts generally and in a high level among the domains. It can also facilitate the semantic interoperability between a set of domain specific ontologies, which are built under the upper ontology. The UDO has the capability to provide the fundamental concepts upon the domain-specific ontologies, which will provide extensibility for adding more domain-specific ontologies. These concepts span the domain of enterprise architecture. As such, the UDO represents a minimum set of concepts pertaining to enterprise architecture,but yet comprehensive enough for describing the majority of the cases. The UDO is designated domain-independent since it does not address specific domain-dependent concerns.

B. Domain Specific Ontologies

These ontologies, specialised in detail a set of concepts introduced in the upper ontology. Domain ontologies are able to capture the knowledge of specific domains by defining the reusable vocabularies in the domain concepts and the relationships among these concepts [18]. So a domain-specific ontology (DSO) represents a domain-specific language that addresses a particular set of concerns. For example, a Software Licensing DSO would describe the concepts required to model the universe of licenses, and may include concepts that cover licensing models, licensing agreement, copyrights, license types (e.g. free software, open source), etc. The UDO will generalises a set of DSOs. Each DSO should be designed with the minimum set of concepts required to describe a given domain. The model should also be easily extended so that an additional DSO is added to the model without affecting the existing DSO.

C. Ontology Integration

Ontology integration is the process of building new ontologies by finding common concepts between two (or more) ontologies, which is categorized by three basic processes:

- Ontology Mapping which, 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 Alignment which, 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 Merging which, is the process of building a new ontology by merging several ontologies into a single one, that will create a will create a more general ontology about a subject.

In the proposed architecture, the ontology mapping process has been selected in order to deal with the combination of the different ontologies in such a way that the overall UDO is consistent and able to address the domains covered by each ontology. In the simplest case, each DSO needs to be mapped with some of the core concepts represented in the UDO. Several DSO can be also be integrated in order to add more expressive power to specific domains, which will help the architecture to be able to represent the domain concepts without ambiguity. This entails defining the minimum set of types and relationships to describe a domain. For instance, Figure 1 depicts a scenario where a DSO integrates with two more specific DSOs, this approach facilitates layering multiple DSO according to the modeling needs.

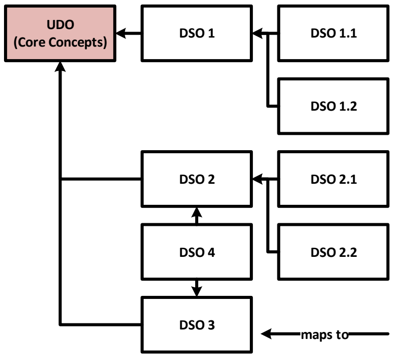


Fig. 1. UDO-DSO integration Architecture

The ontology mapping described above makes use of model transformation to relate a DSO to the UDO or to relate multiple DSOs. Model transformation entails defining a mapping strategy from a source model to a destination model [13], [19].This mapping may create different types of representational deficiencies. The basic assumption is that any deviation from a 1:1 relationship between the constructs in the target representation model and the corresponding constructs in the source modeling language leads to a situation of representational deficiency in the language, potentially causing confusion for its users. Two principal evaluation criteria may be studied: ontological completeness and ontological clarity. The study of ontological completeness is the analysis of the extent to which a modeling language has a deficit of constructs mapping to the set of constructs proposed in the Bunge-Wand-Weber (BWW) representation model [20]. The study of ontological clarity involves the analysis of the extent to which the modeling language constructs are deemed overloaded (i.e. they map to two or more constructs in the BWW model), redundant (i.e. two or more language constructs map to the same construct in the BWW model), or excess (i.e. they map to none of the constructs in the BWW model).

The ontology architecture is designed to adhere to the principle of high cohesion and low coupling. High cohesion means that each architectural module deals only with a set of related domain-specific concerns. Low coupling means that the number of dependencies between architectural modules is designed to be minimum. Together, these two properties promote modularization along with the ability to incremental extend the architectural modules. The ontology architecture comprises an UDO. This upper ontology is able to provide a high-level description of a system and to support inference around the core structure, behaviour and consistency.

A DSO can be related to the high-level concepts of the UDO. Creating these relationships implies transforming the concepts and relationships of the DSO to the concepts and relationships of the UDO. This transformation process is straightforward when there is a one-to-one relationship or map between the concepts of the DSO and the UDO. As a result of this approach, each DSO relates to the UDO through one map. A DSO can also be mapped to another DSO through a map. Each map is actually an ontology that specifies the transformation rules from the source to the target ontology.

D. Reasoning

One of the advantages of having a formal ontology is that it can be processed with logical reasoning mechanism. By using a Description Logic reasoner services, such as consistency checking, inferring dependencies,completeness of models it will be possible to validate the correctness of an ontology. It can be assumed that the upper ontology comprises three layers, i.e., business, application and technology. The classes and properties of each DSO are mapped to the classes and properties of the UDO. This allows for the following reasoning configurations:

- UDO reasoning which, inference is exclusively based on the UDO concepts. Considering the three layers of the UDO, two options exist:
  (1) Intra-layer UDO reasoning, when inference is limited to the concepts of just on the UDO layers.
  (2) Inter-layer UDO reasoning, when inference concepts related to two or three UDO layers.
- DSO reasoning which is, similar to UDO reasoning but inference is exclusively based on a single DSO. Whenever transformations between two or more DSO

exist, then reasoning may span several DSO without interfering with the UDO.

- UDO-DSO reasoning which, inference is based on the UDO concepts plus the concepts of one or more DSO. Requires a transformation map between each UDO-DSO pair. In this case, the resulting reasoning may span more than one UDO layer, depending on the transformation map.

Figure 2 shows the following reasoning configurations.

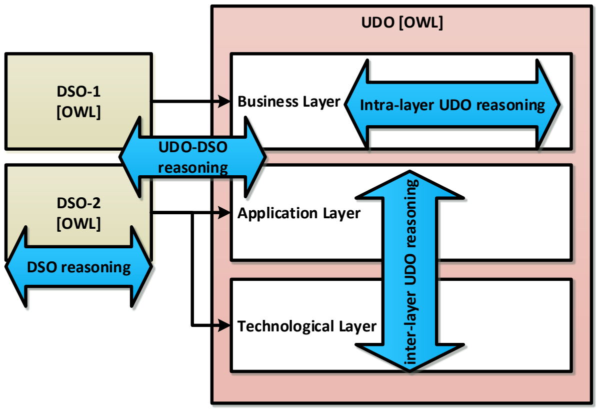


Fig. 2. Reasoning configurations

IV. Application to Enterprise Architecture

The ArchiMate 2.0 language [21] was decided to be used as the core of the the UDO. ArchiMate is a standard from The Open Group [1] that covers the domain of enterprise architecture. This language includes a minimum set of concepts and relationships and the framework includes a minimum set of layers and aspects to be able to model the majority of cases. The motivation to select the ArchiMate language as the core of the UDO is that provides a high-level of abstraction, is concern-oriented and viewpoint-orientated and, was designed with extensibility in mind.

ArchiMate helps stakeholders to design, assess and communicate the consequences of decisions and changes within and between business domains[4]. In order to achieve this, ArchiMate provides a set of viewpoints that can be used to accommodate different concerns. The viewpoints act as filters on the model and are used to highlight different aspects that matter to different stakeholders. Some viewpoints display intra-layer concepts and dependencies, while others display cross layer concepts and relationships. The framework organizes the modeling language in a three by three matrix: the lines capture the enterprise layers, i.e., business, application, and technology, and the columns capture cross layer aspects: active structure, which contains entities capable of performing behavior; behavior, which contains elements defined as units of activity performed by one or more active structure elements; and passive structure, which contains objects on which behavior is performed , therefore, it can be considered an Upper Domain Ontology in the setting of enterprise architecture. Therefore, the ArchiMate language meta-model was converted to an ontology representation as the UDO, so that inference can be applied to its models. The transformation process uses (1) an OWL representation of the ArchiMate meta-model and (2) OWL representations of ArchiMate models.

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

The ArchiMate meta-model representation results from an analysis of the its concepts, relationships and constraints, as a result, a set of transformation rules maps each element from the ArchiMate meta-model into the corresponding OWL representation. Each concept is transformed into an OWL Class. After the concepts and relationships are represented as classes and object properties, the ontology constraints still need to be included, these constraints are required so that derived relationships can be correctly inferred through logical reasoners. Relationship cardinalities were also added to make the ontology compliant with ArchiMate meta-model. The resulting core ontology is extended through a set of DSOs tailored to address explicit modeling concerns.

Next, an analysis of the mapping process between ArchiMate meta-model and models to OWL was performed, from two important criteria: completeness and clarity. This analysis is based on the ontological analysis method as described by Bunge-Wand-Weber (BWW) representation model [20]. From the perspective of the completeness criteria, all concepts from the ArchiMate can be mapped to OWL concepts. Therefore, the mapping is complete. This means that any ArchiMate concept can always be mapped to OWL concepts. Next for the Excess criteria, all of the main elements of OWL classes, properties, instances of classes, and relationships between these instances for mapping from ArchiMate to OWL were used. Therefore the main elements of OWL have no excess concepts in term of mapping from ArchiMate to OWL. In some cases, different ArchiMate concepts map into the same OWL concept. This situation is known as overload. For example, this situation occurs in the case of junction, Instances which both will be represented as individuals in OWL, and also access types and relation will both be represented as object properties. None of the overloaded concepts had cause problems. Finally, there are no situations, that ArchiMate has more than one concept suitable to represent a single OWL concept. This situation is known as redundancy.

After that three DSO, were selected to integrate with the current ArchiMate UDO. The transformation map between the concepts and relationships of the three DSO and the UDO was produced. This transformation map is an ontology that defines the rules that relate each DSO concept to the relevant DIO concept. Figure 3 shows the following relationship between UDO, DSO and transformation maps in the context model architecture. Each relationship indicates the mapping of concepts from a source to a target ontology.

A. *ArchiMate Upper Domain Ontology*

ArchiMate itself is grounded in the entity-relation paradigm, providing specialization of these generic concepts into en-

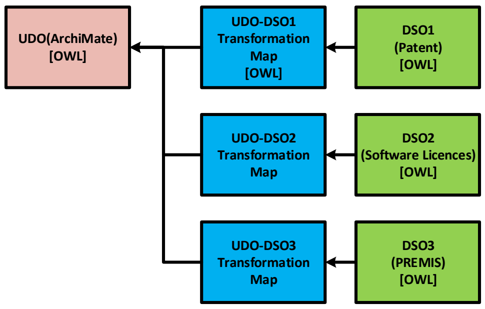
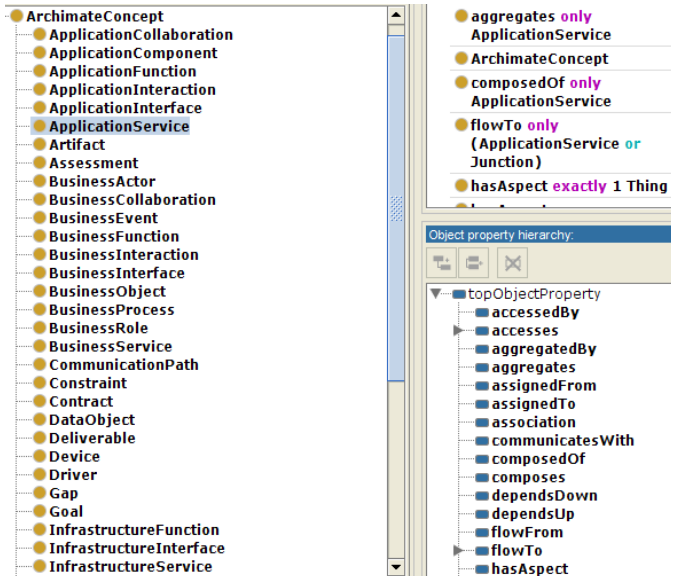

UDO(ArchiMate) [OWL]
UDO-DSO1 Transformation Map [OWL]
DSO1 (Patent) [OWL]
UDO-DSO2 Transformation Map
DSO2 (Software Licences) [OWL]
UDO-DSO3 Transformation Map
DSO3 (PREMIS) [OWL]

Fig. 3. Relationship between UDO, DSO and transformation maps in the architecture. Each relationship indicates the mapping of concepts from a source to a target ontology

terprise architecture concepts and also into domain-specific concepts. The OWL format of the ArchiMate UDO, was imported to an ontology editor called Protege [2] figure 4 depicts an OWL representation of the UDO.


```
ArchimateConcept
    ApplicationCollaboration
    ApplicationComponent
    ApplicationFunction
    ApplicationInteraction
    ApplicationInterface
    ApplicationService
    Artifact
    Assessment
    BusinessActor
    BusinessCollaboration
    BusinessEvent
    BusinessFunction
    BusinessInteraction
    BusinessInterface
    BusinessObject
    BusinessProcess
    BusinessRole
    BusinessService
    CommunicationPath
    Constraint
    Contract
    DataObject
    Deliverable
    Device
    Driver
    Gap
    Goal
    InfrastructureFunction
    InfrastructureInterface
    InfrastructureService
```

```
aggregates only
ApplicationService
ArchimateConcept
composedOf only
ApplicationService
flowTo only
(ApplicationService or
Junction)
hasAspect exactly 1 Thing
```

```
Object property hierarchy:
topObjectProperty
    accessedBy
    accesses
    aggregatedBy
    aggregates
    assignedFrom
    assignedTo
    association
    communicatesWith
    composedOf
    composes
    dependsDown
    dependsUp
    flowFrom
    flowTo
    hasAspect
```

Fig. 4. OWL representation of the ArchiMate(UDO)

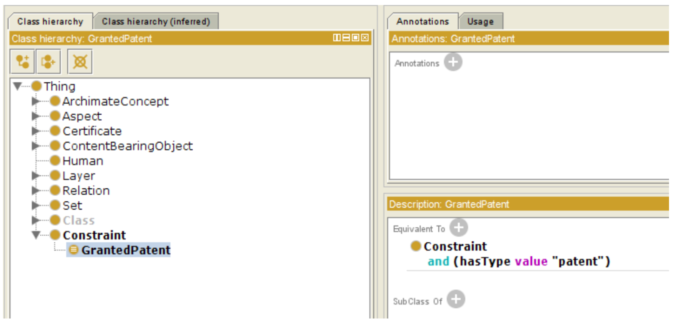
The first column shows the classes of the ArchiMate(UDO), the second column shows the constraints of the classes and column the relationships.

B. Domain Specific Ontologies

In order to test the proposed architecture through a specific case study, three DSOs were selected to integrate with the UDO, the Patent, Software Licenses and PREMIS DSO. In the approach of extending the DSOs, it was decided to reuse the existing domain specific ontologies whenever available and fit. This allows for a greater interoperability of UDO and decreases the overall risks in engineering an ontology for each domain from scratch.

The Patent DSO covers aspects on patents, e.g. who is the owner of a specific patent, what the patent covers, or when it was granted. Patents also imply a restriction on how a software, hardware or method can be used. The most suitable candidate patent DSO that was identified was from the PATExpert project[3].

PATExpert defined a suite of ontologies that describe patent documents, covering aspects such as the structure of documents and content they provide. An important class in the ontology is the PatentDocument, including the subclasses PatentPublication and GrantedPatent. For the initial version of the mapping, a simple approach was selected that considers the GrantedPatentDocument class to be a specialised version of a Constraint class in the UDO, which is indicated in the UDO by the property hasType:Patent. Figure 5 depicts an a screenshot of the mapping between the two ontologies.



```
Class hierarchy: GrantedPatent
Thing
    ArchimateConcept
    Aspect
    Certificate
    ContentBearingObject
    Human
    Layer
    Relation
    Set
    Class
    Constraint
        GrantedPatent
```

```
Annotations: GrantedPatent
Annotations

Description: GrantedPatent
Equivalent To
    Constraint
     and (hasType value "patent")
SubClass Of
```

Fig. 5. Mapping of the GrantedPatent class to the Constraint class of the UDO

The Software Licenses DSO deals with software components and their relations to each other. These relations describe which other software components are required to run a certain software, or which software is conflicting. The license in this case is a specific kind of contract that grants certain rights to the license taker regarding the usage of the software, e.g. as a component his own applications use. It defines for example whether the customer can get access to the source code, modify it, redistribute the software, etc. The most suitable candidate Software Licenses DSO that was identified was from the Software Ontology (SWO)[4].

The SWO is an ontology for describing software tools, their types, tasks, versions, provenance and associated data. SWO was originated in a project between the European Bio informatics Institute and the University of Manchester. The ontology models two important concepts software licenses, and License clauses. License clauses define properties and restrictions on what can be done with the software, e.g. whether redistribution is allowed, and in what form (with or

[2]http://protege.stanford.edu/

[3]http://cordis.europa.eu/ist/kct/patexpert_synopsis.htm

[4]http://theswo.sourceforge.net/

without notice), or whether there is a restriction on the number of users that can use the software. Software licenses are then a composition of these clauses. The ontology mapping is relative straightforward, and allows both a software license and a License clause to be specified as a subclass of a constraint class in the UDO. This way, it can take profit from the predefined standard licenses in case one uses such a license, but can easily define their own license as a composition of clauses.

The PREMIS DSO[5] deals with storing provenance information in the context of digital long-term preservation which includes five important concepts for digital preservation purposes: Intellectual Entities, Object, Event, Agent, Rights. For the ontology mapping, intellectual entity was mapped to the business object in UDO, the object was mapped to the data object, and the agent was mapped to the business actor.

V. Case Study

ArchiSurance has been selected as a case study to analyse the current ontology. ArchiSurance [21], is a fictitious insurance company used throughout the ArchiMate 2.0 specification as a case study to illustrate modeling of strategy, business, applications and technology. The example was converted to the OWL to be represented as the instances of the ArchiMate(UDO), this example in this section is used to illustrate the capabilities of reasoning, by validating the correctness of the ontology. A set of predefined competency questions were used in order to validate ontology.

According to [22] the competency questions should be defined in a stratified manner, with higher level questions requiring the solution of lower level questions However, in this work we elaborated a single-level list of questions since our purpose is to validate that we are able of answering a more comprehensive set of questions that interrelate a greater number of concept rather than the ability of answering more complex questions.

The set of competency questions defined to validate the integrated ontology is composed by the following questions:

- What Business Services are used by the Customer Business Role?
- What Business Processes are used by the Customer?
- What ArchiMate concepts belong to the Application Layer?
- What ArchiMate concepts are Behavioural Aspects?
- What are the licenses required to execute software application?

Figure 6 below represents a view that shows the Business Services that are used by a specific Business Role and the Business Processes that realize each of the aforementioned Business Services. In this particular example the Customer business role uses four business services that in turn are realized by a total of four business processes (e.g. the Insurance Application Service is realized by the Close Contract process).

Figure 7 and Figure 8 shows the query result, Note that the Customer does not directly use any business process. The resulting relationship is a derived relationship because

[5]http://premisontologypublic.pbworks.com/w/page/45987067/FrontPage/

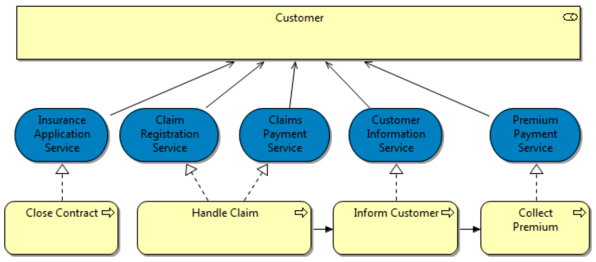
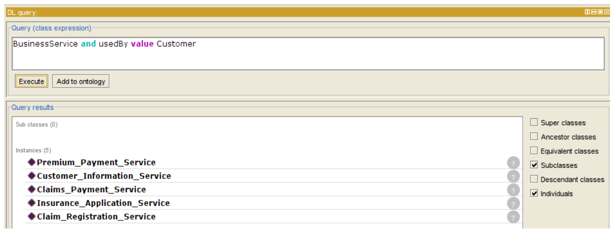


Fig. 6. Archisurance ArchiMate example: service realization viewpoint



```
BusinessService and usedBy value Customer
```

```
Sub classes (0)

Instances (5)
  ◆Premium_Payment_Service
  ◆Customer_Information_Service
  ◆Claims_Payment_Service
  ◆Insurance_Application_Service
  ◆Claim_Registration_Service
```

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☑ Subclasses
☐ Descendant classes
☑ Individuals

Fig. 7. What Business Services are used by the Customer Business Role?

the customer is transitively related to a business process through a business service. This example shows how derived dependencies can be inferred from models.

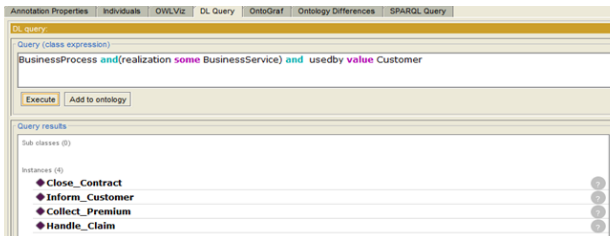
In Figure 9 and Figure 10 which is exclusively based on the UDO concepts, the concepts of a specific layer and a specific aspect in ArchiMate(UDO) is shown.The inference is exclusively based on the UDO concepts.

A UDO-DSO reasoning, which is based on the UDO concepts plus the concepts of the Software Licenses DSO is shown in Figure11 . Software licenses, is a concept,that is not possible to model in the UDO with the desired level of detail. But, it can be captured in an increased level of detail through the use of the Licenses DSO.

VI. Conclusion and Future Work

In this paper, a modular Ontology for the enterprise architecture domain was presented. Ontology technique were applied, to be used for the analysis and design of enterprise architecture models, specifically for inconsistency checking on the models and dependency analysis between the different elements of the enterprise architecture. To this end, a model architecture was proposed which includes a core enterprise ontology named Upper domain ontology that will present the fundamental concepts of enterprise architecture, and could be extended to Domain Specific Ontologies, that describes in detail a set of concepts introduced in the upper ontology and then in order to integrate the UDO and DSOs a mapping strategy has to be define between them. Next, to implement and evaluate the proposal architecture, the ArchiMate 2.0 was converted to the OWL language.

The resulting artefact is the UDO. A set of DSO were selected along with the corresponding transformation maps to test the mapping technique and to evaluate the UDO. This proposed architecture was validated by a set of competency



DL query:

Query (class expression)

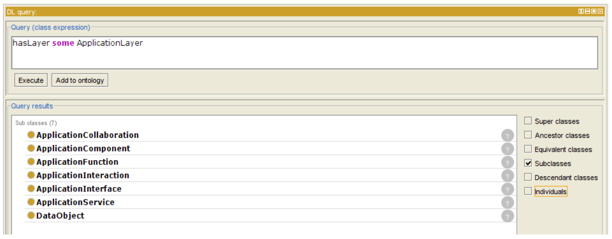
```
BusinessProcess and(realization some BusinessService) and  usedby value Customer
```

Query results

Sub classes (0)

Instances (4)
- Close_Contract
- Inform_Customer
- Collect_Premium
- Handle_Claim

Fig. 8. What Business Processes are used by the Customer?



DL query:

Query (class expression)

```
hasLayer some ApplicationLayer
```

Query results

Sub classes (7)
- ApplicationCollaboration
- ApplicationComponent
- ApplicationFunction
- ApplicationInteraction
- ApplicationInterface
- ApplicationService
- DataObject

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☑ Subclasses
☐ Descendant classes
☐ Individuals

Fig. 9. What ArchiMate concepts belong to the Application Layer?

questions,which can act as benchmarks for any ontology. This validation was processed with reasoning methods. The reasoning was performed on different types of analysis on the UDO and DSO for checking the consistency and dependencies between different elements of the architecture models. Our formal integrated enterprise ontology approach was able to successfully answer all the competency questions and facilitate the analysing of Enterprise architecture models. In further research, the integrated ontology will be extended with more DSOs. New scenarios and more rules and constrains will be defined on the core ontology, and also will be evaluated with different types of description logic reasoners.

Acknowledgments

This work was supported by national funds through FCT - Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2011 and the grant (SFRH/BD/69121/2010) to Gonçalo Antunes, by COMET K1, FFG - Austrian Research Promotion Agency, 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

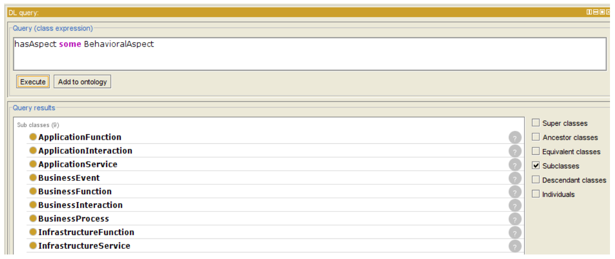
[1] J. A. Zachman, "Enterprise architecture: The issue of the century," *Database Programming and Design*, vol. 10, no. 3, pp. 44–53, 1997.

[2] ISO, "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, 2011.

[3] D. E. Jenz, "Business process ontologies: Speeding up business process implementation," *Jenz & Partner GmbH*, 2003.

[4] M. Lankhorst, *Enterprise architecture at work: Modelling, communication and analysis*. Springer, 2009.

[5] H. Stuckenschmidt, C. Parent, and S. Spaccapietra, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. Springer, 2009, vol. 5445.



DL query:

Query (class expression)

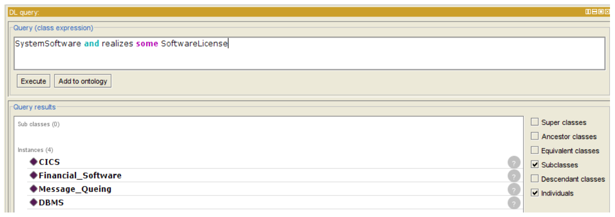
```
hasAspect some BehavioralAspect
```

Query results

Sub classes (9)
- ApplicationFunction
- ApplicationInteraction
- ApplicationService
- BusinessEvent
- BusinessFunction
- BusinessInteraction
- BusinessProcess
- InfrastructureFunction
- InfrastructureService

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☑ Subclasses
☐ Descendant classes
☐ Individuals

Fig. 10. What ArchiMate concepts are Behavioural Aspects?



DL query:

Query (class expression)

```
SystemSoftware and realizes some SoftwareLicense
```

Query results

Sub classes (0)

Instances (4)
- CICS
- Financial_Software
- Message_Queing
- DBMS

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☑ Subclasses
☐ Descendant classes
☑ Individuals

Fig. 11. What are the licenses required to execute software application?

[6] P. Johnson, E. Johansson, T. Sommestad, and J. Ullberg, "A tool for enterprise architecture analysis," in *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*. IEEE, 2007, pp. 142–142.

[7] M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A tool for enterprise architecture analysis using the prm formalism," in *Information Systems Evolution*. Springer, 2011, pp. 108–121.

[8] A. SoS, "A uniform approach for system of systems architecture evaluation," 2009.

[9] J. Klein and M. J. Gagliardi, "A workshop on analysis and evaluation of enterprise architectures," 2010.

[10] M.-E. Iacob and H. Jonkers, "Quantitative analysis of enterprise architectures," in *Interoperability of Enterprise Software and Applications*. Springer, 2006, pp. 239–252.

[11] F. De Boer, M. Bonsangue, J. Jacob, A. Stam, and L. Van der Torre, "Enterprise architecture analysis with xml," in *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*. IEEE, 2005, pp. 222b–222b.

[12] C. Pedrinaci, J. Domingue, and A. K. A. de Medeiros, "A core ontology for business process analysis," in *The Semantic Web: Research and Applications*. Springer, 2008, pp. 49–64.

[13] G. Guizzardi, *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, 2005.

[14] R. Guizzardi, X. Franch, and G. Guizzardi, "Applying a foundational ontology to analyze means-end links in the i* framework," in *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*. IEEE, 2012, pp. 1–11.

[15] P. S. Santos Jr, J. P. A. Almeida, and G. Guizzardi, "An ontology-based semantic foundation for aris epcs," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 124–130.

[16] J. P. A. Almeida, G. Guizzardi, and P. S. Santos Jr, "Applying and extending a semantic foundation for role-related concepts in enterprise modelling," *Enterprise Information Systems*, vol. 3, no. 3, pp. 253–277, 2009.

[17] T. R. Gruber *et al.*, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[18] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda, "Task ontology for reuse of problem solving knowledge," *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, pp. 46–57, 1995.

[19] P. Green, M. Indulska, and M. Rosemann, "A reference methodology for conducting ontological analyses," 2004.

[20] M. Bunge, *Treatise on Basic Philosophy: Volume 5: Epistemology & Methodology I: Exploring the World*. Springer, 1983.

[21] M. Iacob, H. Jonkers, M. Lankhorst, E. Proper, and D. Quartel, "Archimate 2.0 specification: The open group," 2012.

[22] M. Fox and M. Gruninger, "Enterprise modeling," *AI magazine*, vol. 19, no. 3, p. 109, 1998.