IT Business Standards as an Ontology Domain

Adam Czarnecki and Cezary Orłowski

Gdańsk University of Technology, Faculty of Management and Economics,
Department of Information Technology Management,
ul.Narutowicza 11/12, 80-233 Gdańsk, Poland
{Adam.Czarnecki,Cezary.Orlowski}@zie.pg.gda.pl

Abstract. The aim of this paper is to report a selection of Semantic Web aspects pertaining to ontology development activities in the domain of the IT business standard (TOGAF 9) such as formulating competency questions, conceptualization of the domain, resolution of the source knowledge deficiency and applying common design patterns in the OWL formalization. Authors also try to determine target groups that may benefit from such ontology models.

Keywords: enterprise architecture, information technology, ontology engineering, OWL, Semantic Web, standards, TOGAF.

1 Introduction

The old engineering joke says that the best thing about standards is that there are so many to choose from. But when a standard is chosen to be deployed, this deployment should conform to the guidelines provided in order to maintain compatibility. If not, we may have multiple implementations of the same norm that cannot be aligned.

Strict technical norms generally leave a narrow or no margin of interpretation and their scope, requirements, processes and outputs are well defined. But there is a range of IT business standards that operate on both sides of the borderline that separates technical and social systems. Their deployment usually relies on the context and often requires some tailoring before they can be applied in the organization. Also the standard description provided in documents may not be as precise as in the pure technical specification. Both aforementioned conditions may lead to the inconsistency of the solution that is based on that standard. The third issue one should bear in mind is that although standards are reviewed in order to eliminate all errors and discrepancies, there is still a margin of faults that may have been overlooked in the current revision.

There are numerous methods of enforcing the textual description of the standard with more formal structured information such as tables, figures and diagrams to mention a few. Authors of this paper would like to discuss the potential applicability of ontologies developed in the Web Ontology Language as one of the methods for assuring the consistency of standards. This approach would be considered from two perspectives: the standard originator and organization that is implementing the given standard.

P. Jędrzejowicz et al. (Eds.): ICCCI 2011, Part I, LNCS 6922, pp. 582–591, 2011.

Authors have chosen The Open Group Architecture Framework version 9 (TOGAF) as an example of the standard that combines business (social) and technical threads. To narrow the scope of the ontology domain, core content metamodel—a part of TOGAF standard that defines a formal structure of terms within enterprise architecture development method—has been selected. The paper focuses on the part of the ontology engineering process (however does not address any methodology in particular), i.e. on the set of activities: knowledge acquisition, conceptualization and formalization. This process serves as a case study for discussing the degree of the effort that must be put into reflecting the knowledge on the standard in the ontology and what benefits—if any—can be drawn from this project.

The intended readers of this paper are people involved in two kinds of processes related to standards: their devising and deploying. Authors would also like to notice that some earlier remarks on ontologies as tools for the IT management standards support were outlined in [2].

The first part of this paper briefly outlines TOGAF and its content metamodel. Then the activities in ontology engineering are described: competency questions formulation, domain knowledge acquisition based on TOGAF core content metamodel specification and the actual development of the ontology in OWL (which is tool-independent but in fact Protégé 4.0.2 application was used). To sum up, at the end of the paper conclusions are drawn and future works are outlined.

Due to the limited length of this paper a substantial part of the work (two figures and two tables) were excluded from the main text and published online as external references: [3], [7], [8] and [9]. Authors strongly suggest looking up to those links.

2 What is TOGAF

The Open Group Architecture Framework document specifies a detailed method and a set of supporting tools for developing enterprise architecture. The enterprise architecture (EA) can be described (attributive definition) as the organizing logic for key business process and IT capabilities reflecting the integration and standardization requirements of the firm's operating model [10]. The objective definition denotes a formal description of the phenomena given in the attributive definition. There is also a functional approach to the EA which indicates tasks and skills essential for managing the subject of the attributive definition [4].

TOGAF in its 9th version (later in this paper referred to as TOGAF 9 or TOGAF) addresses four main architectural domains:

1. Business,
2. Data,
3. Application,
4. Technology,

and there are six main parts of the framework:

1. Architecture Development Method (ADM)—a description of phases of the iterative cycle.
2. ADM guidelines and techniques—applying iteration to the ADM, its usage at the different enterprise levels.

3. Architecture content framework—specifies the content metamodel, artifacts, deliverables and building blocks.
4. Enterprise Continuum and tools—a specific repository of architectures and solutions.
5. Reference models—foundation architecture and integrated information infrastructure.
6. Architecture capability framework—reference material on how to manage organization structures, processes, roles, responsibilities and skills.

The idea of creating an ontology based on the TOGAF is not novel. SOA Working Group has created TOGAF 8 Ontology Draft [6] that captured most of the enterprise architecture artifacts and other objects and relations between them. It is important to add that TOGAF 8 was not equipped with a content metamodel description.

3 TOGAF Content Metamodel Overview

The TOGAF document is quite voluminous (700+ pages not including other referenced papers). TOGAF's Chapter 3 introduces 90 essential definitions as a prerequisite to the main part of the framework and there are 93 supplemental terms given in the Appendix A.

To define a formal structure for these terms and to ensure their consistency within the Architecture Development Method (ADM) content metamodel has been introduced in the TOGAF 9 document. The content metamodel defines a set of entities that allow architectural concepts to be captured, stored, filtered, queried, and represented in a way that supports consistency, completeness, and traceability [5, p. 373].

The TOGAF content metamodel is divided into the core metamodel which provides a minimum set of architectural content to support traceability across artifacts and metamodel extensions that provide concepts to support more specific or more in-depth modeling. In this paper the focus is set on the core content metamodel as a domain of the ontology.

4 Ontology Development

In following subsections the process of ontology development in the TOGAF core content metamodel domain has been described. For the sake of clarity of the example and because of the research being still in progress, the scope of the domain addresses the following parts of the content metamodel specification:

- Core content metamodel entities,
- Relationships between core content metamodel entities,
- Content metamodel objects (a list and a hierarchy but without relationships).

Among elements that have been omitted in this paper there are:

- ADM phases,
- Core architecture artifacts,

- Content metamodel extensions,
- Content metamodel attributes.

In the progress of further research the scope of the ontology should be broaden to encompass all concepts in TOGAF content metamodel.

The OWL DL formalization of the domain was implemented using Protégé 4.0.2 application. Authors have provided some examples of the OWL syntax in the text to better illustrate certain ontology constructs.

4.1 Competency Questions

The aim, scope and depth of the ontology depend on the knowledge it is supposed to store and conclude. The software engineering discipline has a requirements modeling technique of 'use cases' that a system should respond to. When specifying ontology a knowledge worker has a tool of 'competency questions'—a list of issues that ontology should (must) reply to.

For the domain of TOGAF core content metamodel the following competency questions were formulated:

- What are the core metamodel entities?
- What is the structure of the core metamodel entities?
- How core metamodel entities relate to each other?
- What is the type of the individual that is described using object property assertion?

These questions were chosen to illustrate the process of ontology development and can easily be expanded by adding other elements of the core content metamodel and its extensions to the domain.

4.2 Core Content Metamodel Concepts

The TOGAF core content metamodel is the backbone (see: Fig. 34-1 in [5, p. 368]) of the content metamodel. The list of its concepts can be found on multiple pages of TOGAF9's Chapter 34, e.g.:

- A list of core terms titled 'Core Metamodel Entities' on page 369,
- A Figure 34-2 titled 'Core Entities and their Relationships' on page 371,
- A Figure 34-5 titled 'Detailed Representation of the Content Metamodel' on page 375,
- A Figure 34-6 featuring a UML-like class diagram titled 'Entities and Relationships Present within the Core Content Metamodel' on page 376 (see: Fig. 1),
- A Figure 34-7 titled 'Content Metamodel with Extensions' on page 378,
- A Figure 34-8 titled 'Relationships between Entities in the Full Metamodel' on page 379,
- A table that lists content metamodel objects on pp. 393–396,
- A table with content metamodel attributes on pp. 396–406,
- A table of metamodel relationships on pp. 406–409.

This abundance of sources can be used for cross-checking of the list of core metamodel concepts that are to be stored as classes in the OWL ontology. This, however, reveals some ambiguity between sets of terms that need to be somehow resolved in the ontology (see: [8]). There are only four core metamodel concepts that appear through the mentioned sources and keep their names unchanged. These are: 'Actor', 'Data Entity', 'Function' and 'Role'. Other concepts need preprocessing activities that are described below.

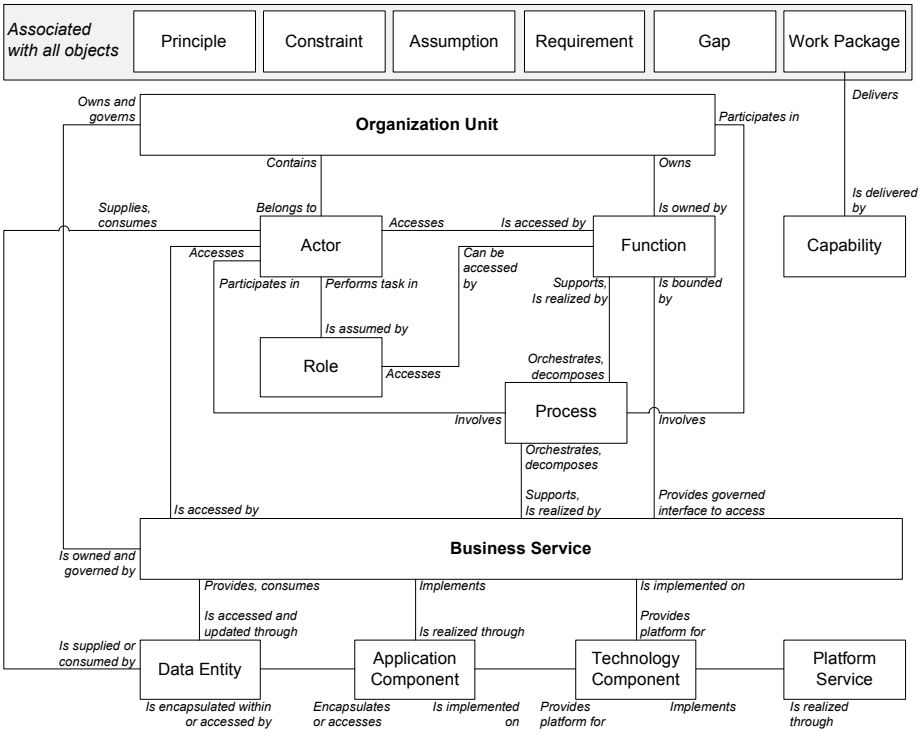

Fig. 1. Entities and Relationships present within the Core Content Metamodel [5, p. 376]

Concept Categories and Multiple Inheritance. One of the problems the TOGAF 9 ontology modeler meets is associated with the entity called an 'Application Component'. It can be found on the core metamodel concepts list but later in the document this object splits into two: 'Logical Application Component' and 'Physical Application Component'. The first one is marked as a part of the core content, the latter—a part of the infrastructure consolidation extension. All three entities are listed as metamodel objects, which includes core and extensions entities. The proposed solution to this problem would be to make 'Logical Application Component' and 'Physical Application Component' be both subclasses of the 'Application Component'. The 'Application Component' would be a subclass of the 'Content Metamodel Object' class and 'Logical Application Component' would get a 'Core Metamodel Entity' as its superclass. The OWL syntax (in XML serialization) of these relations is presented below:

```
<owl:Class rdf:about="#Application_Component">
  <rdfs:subClassOf
rdf:resource="#Content_Metamodel_Object"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Logical_Application_Component">
  <rdfs:subClassOf
rdf:resource="#Application_Component"/>
  <rdfs:subClassOf
rdf:resource="#Core_Metamodel_Entity"/>
</owl:Class>
<owl:Class rdf:about="#Physical_Application_Component">
  <rdfs:subClassOf
rdf:resource="#Application_Component"/>
</owl:Class>
```

The OWL allows a class to have multiple superclasses, so there is no contradiction in the presented solution. A very similar situation to the given above applies to the set of 'Technology Component', 'Logical Technology Component' and 'Physical Technology Component' and has been resolved in the same manner.

Class Equivalence. The next problem encountered was the concept that had different names across the metamodel specification. In the presented example domain this applies to the pair of 'Organization' and 'Organization Unit'. There are two constructs in the OWL language that can support the ontology engineer in stating that two classes represent the same concept: `owl:sameAs` and `owl:equivalentClass`. The first property links two individuals and can only be used in OWL Full to denote the identity of two classes. The second property, `owl:equivalentClass`, can be used for classes defined in the OWL DL and states that two classes are both of the same type (they are subclasses of each other which means that they share same sets of individuals).

Same pattern applies in the given domain to 'Business Service' and 'Service' classes. It may be misleading that there are three concepts present in the TOGAF core content metamodel: 'Business Service', 'Platform Service' and 'Service'—the latter is not a superclass for Business and Platform Service but it is an abbreviated name of Business Service as one can conclude from cross-checking tables and diagrams in the TOGAF document.

Concepts Reduction. The example in this paper is limited to the domain of the core content metamodel in TOGAF 9 and therefore the list of [8] terms must be cleared. There are at least four forms of formality in the content metamodel description:

- Plain text,
- Lists (mostly bulleted, not numbered),
- Tables,
- Diagrams.

They have been listed here in the order of the ascending formality level. In general, the more formal the description is, the less interpretation is needed. So, when selecting candidates for classes in the ontology, diagrams (with UML classes and relations) and tables were mostly used. The final set of core metamodel entities is depicted on [7] (subclasses of the 'Core Metamodel Entity' class).

4.3 Content Metamodel Objects

All classes that are members of the 'Core Metamodel Entity' class are also members of the broader 'Content Metamodel Object' class, so although the 'Core Metamodel Entity' is not mentioned to be a formal content metamodel object, it has been declared to be the subclass of the 'Content Metamodel Object' class as shown on the [7] figure. It allows automatic classification of any future core metamodel entities to be also content metamodel objects. If it would be unwanted to have 'Core Metamodel Entity' class as a subclass of the 'Content Metamodel Object' one would have to assert that every subclass of the 'Core Metamodel Entity' is also a subclass of the 'Content Metamodel Object'.

4.4 Metamodel Relationships

The section 34.7 of the TOGAF 9 specification contains a table of metamodel relationships with following columns:

- Source Object,
- Target Object,
- Name,
- Extension Module.

First two columns list objects that have been modeled in the ontology as subclasses of the 'Content Metamodel Class'. The 'Name' column lists (mostly) relationships that were depicted on the fig. 34-8 in [5, p.379]. It corresponds well with the 'subject–predicate–object' triple that can be modeled in the OWL as a 'class–object property–class'. But before names of the relationships will become object properties in the ontology, the above mentioned table should be preprocessed in order to:

- Select only relationships between core metamodel entities (as it narrows the scope to the given in this paper),
- Cross-check the consistency of relationships provided by a mentioned table and diagrams,
- Discover pairs of relationships that can be modeled as inversed object properties,
- Discover other types of object properties, e.g. functional, symmetric or transitive.

The output of the preprocessing activities is presented in [9]. If there was an inverse relationship between objects, the inverse relationship is not repeated in the second row with a switched source and target object.

The cross-checking of sources of knowledge of the core metamodel relationships has revealed the inconsistence between them. For instance, metamodel UML-like

diagrams (see: Fig. 1) mark the existence of associations between 'Organization Unit' and 'Process' entities while the section 34.7 of the TOGAF document—which lists metamodel relationships (see: [5, pp. 406–409])—lacks this information. None of the relationships pointing from the class to itself (such as 'Decomposes') were presented on any diagram. There are also distinct differences between core and full metamodel diagrams—in this case the full metamodel diagram relationships were accepted as they were coherent with the aforementioned table.

Relationships Categorization. There are 13 pairs of direct and inverse relationships with some appearing more than once in the metamodel. This includes 5 identified opportunities to create inverse relationships that were not present in TOGAF but may contribute to content metamodel. Most of these pair could be easily derived from the simple lookup: if 'Object1–Relationship1–Object2' and 'Object2–Relationship2–Object1', then Relationship1 and Relationship2 are inverse to each other. This, however, fails when one considers 'Service' and 'Data Entity' objects (see: [9]) when neither of two relationships directing from 'Service' to 'Data Entity' corresponds with the reversed relationship.

The OWL also supports symmetric object properties, where one relationship is direct and inverse at the same time. The candidate for the symmetric property is the 'Communicates with' relationship.

Modeling Relationships in the OWL. The first-class objects in the OWL that are suitable for storing relationships are object properties. The question arises how to model the triple of Object1–Relationship–Object2.

One of the approaches to the relationship modeling would suggest creating a triple of two individuals and one object property between them, i.e.:

```
<owl:ObjectPropertyrdf:about="#Delivers"/>
<owl:Thingrdf:about="#Capability"/>
<owl:Thingrdf:about="#Work_Package">
<Delivers rdf:resource="#Capability"/>
</owl:Thing>
```

As it is shown in the aforementioned code 'Capability' and 'Work Package' concepts are defined as individuals, not classes, due to the OWL DL limitations. To preserve concepts as classes an ontology engineer can use a number of design patterns. One of them is to set the domain and the range of the object property. This approach can be applied when the object property is and will be exclusively used to link entities of the given domain and range. If we had in the same ontology a domain–object property–range declaration that would state: 'Programmer'–'Delivers'–'Source Code', the reasoner would infer that a 'Programmer' is-a 'Work Package' and the 'Source Code' is-a 'Capability'. And in fact, there is a number of relationships in the considered TOGAF domain that have the same name but links different objects, i.e. 'Communicates with', 'Consumes', 'Decomposes', 'Is performed by', 'Is realized by', 'Orchestrates', 'Supplies' and 'Supports'. To resolve this name collision each of the mentioned relationships should became a superproperty of object properties that

link specific classes. At the time of writing this paper this task has not yet been completed, but the example for the Actor–Decomposes–Actor triple can be demonstrated is action: the definition of the 'Actor decomposes Actor' object property allows the reasoner to classify any individual that has an assertion that includes this property as a type of 'Actor' as shown on the referenced figure [3].

One could also use other OWL modeling technique: class equivalence of the existential restriction as a necessary and sufficient condition (definition) of the class. More on this subject can be found in [1].

4.5 Before the Next Iteration

During the process described in section 4 of this paper, an ontology consisting of 81 classes and 39 object properties has been developed. As it was mentioned earlier in the text, this ontology requires further work in developing and structuring more complex object properties and domain–range definitions.

The time required to build this ontology was 3–4 days including knowledge acquisition, conceptualization and formalization. The estimated time needed to finish the phase of core content relationship modeling is 0.5–1 day. Number of days needed to fully model TOGAF content metamodel is hard to assess but can range from 2 to 4 weeks for a single ontology engineer.

5 Summary and Future Works

The ontology development process described in this paper has shown that the domain of core content metamodel of TOGAF can be represented in the formal and computable language—OWL. This language allows for relatively easy knowledge classification based on existing reasoning software. It also allows detecting certain inconsistencies within the framework documentation. The next contribution of ontology is promoting sharing knowledge which collective computational intelligence can benefit from. The question arises whether the effort put in the ontology development brings benefits that could not be achieved in shorter time, with fewer resources engaged and with at least the same quality.

The results of applying ontology to a very narrow scope of the only one standard do not constitute a representative sample to draw general conclusions. However this experiment sheds some light on the path of the future research—the team involved in standard's development may use ontology as a 'back-office' model to share a common vocabulary and understanding of the standard among the people involved in that collaboration. The parallel development of the ontology in that body could support the process of the standard elaboration and revision phases.

The second party that could benefit from the ontological model of the standard is organizations that intend to implement such standard. Although further studies are needed, the current experience of authors suggests that such effort of developing ontology in the domain of the given IT standard on the client-side is resource-consuming and requires skills that are rare on the job market. Therefore the economic efficiency of such activities would be low. But if the standard body would provide ontology of the standard—as it was suggested in the preceding paragraph—along with

the interface that would allow answering competency questions formulated by the client organization, the efficiency of the ontology use might increase. Authors of this paper would like to consider it as a 'working hypothesis' that needs further examination.

To conclude, IT business standards can be a promising array of domains of applying ontologies that may give these standards a sound formal semantic (logic) shape that is based on the Semantic Web architecture. It allows verification and validation of the model presented in the given standard. Yet, the degree in which such ontology can support the standards' bodies and users should be further studied.

References

1. Allemang, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufman, Burlington (2008)
2. Czarnecki, A., Orłowski, C.: Ontology as a Tool for the IT Management Standards Support. In: Jędrzejowicz, P., Nguyen, N.T., Howlet, R.J., Jain, L.C. (eds.) KES-AMSTA 2010. LNCS(LNAI), vol. 6071, pp. 330–339. Springer, Heidelberg (2010)
3. Reasoner Classification in Protégé Based on the Domain and Range Declaration (Appendix to this paper), http://www.zie.pg.gda.pl/zzti/iccci2011/fig3.tif
4. Sobczak, A.: Modele i metamodele w architekturze korporacyjnej. In: Proceedings to The Organizations Support Systems Conference, Akademia Ekonomiczna w Katowicach, Katowice (2008) (in Polish), http://www.swo.ae.katowice.pl/_pdf/396.pdf
5. The Open Group Architecture Framework, Version 9. The Open Group (2009)
6. TOGAF 8 Ontology draft. SOA Working Group, http://www.opengroup.org/projects/soa-ontology/doc.tpl?gdid=11367
7. TOGAF Content Metamodel Objects (Appendix to this paper), http://www.zie.pg.gda.pl/zzti/iccci2011/fig2.tif
8. TOGAF Core Metamodel Concepts According to Different Sources (Appendix to this paper), http://www.zie.pg.gda.pl/zzti/iccci2011/tab1.htm
9. TOGAF Core Metamodel Relationships (Appendix to this paper), http://www.zie.pg.gda.pl/zzti/iccci2011/tab2.htm
10. Weill, P.: Innovating with Information Systems: What do the most agile firms in the world do? Presentation at Sixth e-Business Conference, Barcelona (March 27, 2007)