Analyzable Enterprise Models Using Ontology

Sagar Sunkle, Vinay Kulkarni and Suman Roychoudhury

Tata Research Development and Design Center
Tata Consultancy Services
54B, Industrial Estate, Hadapsar
Pune, 411013 INDIA
{sagar.sunkle,vinay.vkulkarni,suman.roychoudhury}@tcs.com

Abstract. While a considerable amount of research on enterprise ontologies exists, work showing *how to use ontologies for enterprise architecture (EA) analysis* is scarce. We present our ongoing work on creating analyzable enterprise models using EA-based ontological representation. Our contributions are twofold: first, we show how an existing EA modeling language can be leveraged to create EA ontology and second, we show how EA analyses can be realized using this ontology. Initial results of basic EA change impact analysis suggest that ontology representation facilitates EA analysis prototyping due to right mix of representation and reasoning functionalities.

Keywords: Enterprise Models, Enterprise Architecture, Analysis, Ontology

1 Introduction

From our past experience in delivering 70+ large business-critical enterprise applications using model-driven [1] and product line approaches [2], we have observed that enterprises are getting larger in size and becoming increasingly connected. The cost of incorrect decision in building these systems is becoming prohibitively high in spite of cost benefits of model-driven development [3]. A proper modeling mechanism is needed that provides both core representation abilities to model an enterprise as well as reasoning abilities to conduct analyses on this model. Enterprise ontologies have been used to model enterprises and find answers to common sense questions using deductive capabilities [4–6], but they have been restricted in their use to non-analysis purposes. Our approach is to show that ontological representations can be leveraged effectively for modeling and analyzing enterprises.

For this we create an EA ontology that is based on concepts and entities in ArchiMate [7]. Our specific contribution is the demonstration of how the existing ontology tools can be used to perform EA analysis with default reasoning services, particularly change impact analysis [8] of EA with reference to concepts and relations in this case study using our EA ontology. The main components in our implementation are inference rule execution and exploitation of graph structure of ontology. Our initial implementation of these analyses suggests that ontologies and ontology tools provide the right mix of representation and reasoning abilities for modeling enterprises and quickly prototyping various interesting analyses.

The rest of the paper is structured as follows. In Section 2, we elaborate our motivation and describe our EA ontology based on ArchiMate concepts and relations and how we model the case study using this ontology. In Section 3, we detail a basic change impact analysis with an implementation with ontology tools. We then discuss related work and conclude the paper in Section 4.

2 Ontology-based Representation for EA

A model of enterprise is generally created based on the principles of what is known as *enterprise architecture* (EA). It is defined as the process of translating business vision and strategy into effective enterprise change[1]. We wanted to create model of enterprise as a computational representation of business, IT, and infrastructure dimensions and capture aspects such as structure, behavior, and information etc. across these dimensions. This is illustrated on the left of Figure 1.

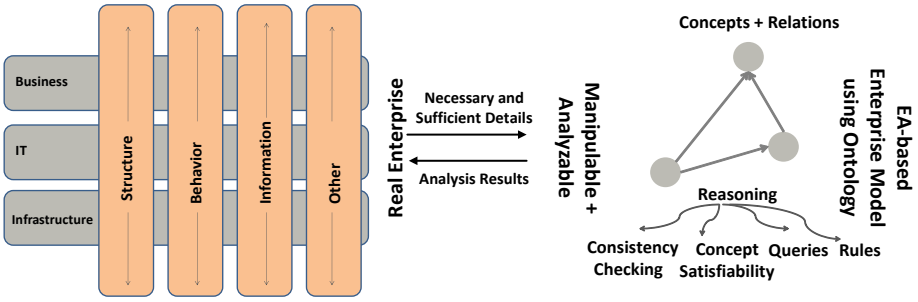

Fig. 1: Machine-processable and Analyzable Enterprise Models using Ontology

2.1 Outline

As a first step in creating such an enterprise model we looked into EA frameworks which assist in the process of creating, maintaining, and leveraging architecture of an enterprise [9], for instance, Zachman Framework, the Open Group Architecture Framework (TOGAF), Federal Architecture Framework (FEA), Gartner, and ArchiMate. Our initial reviews of these frameworks suggested that irrespective of the architectural methodology used by these frameworks, architectural artifacts used in these frameworks are *documents used as reference material* and are *non-machine-processable* [10]. These frameworks lack self assessment mechanism, i.e., what is modeled cannot be checked for consistency but is correct by definition making them *blue-print* frameworks [11]. Experienced enterprise architects and other personnel are supposed to use their judgment in this regard.

[1] `http://www.gartner.com/it-glossary/enterprise-architecture-ea/` Gartner IT Glossary

In contrast, representing enterprise models using ontology provides both machine-processability and a number of reasoning services including consistency checking. It is possible therefore to make application of EA frameworks and techniques more or less person-independent. These models could be used to conduct various analyses of the real enterprise that they capture and utilize the results by possibly making changes to and/or improving the real enterprise. This is shown on the right of Figure 1.

2.2 EA Ontology

To create our enterprise ontology, we refer to ArchiMate metamodels of business, IT, and infrastructure dimensions [12]. Note that in ArchiMate parlance what we referred to as dimensions is called layers and IT layer is referred to as Application layer. We chose ArchiMate as the EA framework to refer to because its set of core concepts and relations provide good starting points for all that we intend to model in an enterprise.

Basic Elements of EA Ontology We retained the distinction made in ArchiMate around business, application, and infrastructure layers as well as structural (active and passive) and behavioral concepts in the EA ontology. Active structural concepts signify actors, behavioral concepts signify tasks and passive structural concepts signify objects acted upon. Relations are also distinguished as being structural and behavioral but the same set of relations occur across all three layers as indicated by generic metamodel of ArchiMate [12]. Although *Interface* and *Service* are structural and behavioral concepts respectively, they are not so distinguished in the ontology to retain special meaning ascribed to them in the generic metamodel of ArchiMate. Accordingly, *BusinessService* is defined as "*Class: BusinessService _ EquivalentTo: BusinessLayer and Service _ SubClassOf: usedBy some BusinessAgent, usedBy some BusinessCoreBehavior*", where a *BusinessAgent* signifies either a *BusinessActor*, *BusinessRole*, or *BusinessCollaboration* and *BusinessCoreBehavior* is either a *BusinessFunction*, a *BusinessProcess*, or a *BusinessInteraction*. All other concepts are defined similarly.

Case Study and Instantiated Model Archisurance is an enterprise architecture and modeling case study referred to in [7, 8, 13]. It concerns a recent merger of three insurance companies dealing in homeowners' and travel insurance, auto insurance, and legal expense insurance formed to take advantage of synergies between three organizations. Key business functions of Archisurance include customer relations, claims, finance, and document processing among others. Main concerns in this merger are integration and alignment for the new companies' business processes and applications. EA analysis approaches could be used to address these issues to some extent. For instance, change impact analysis [8] can be conducted to find out how different entities in Archisurance affect each other due to new integration efforts. We show how this can be achieved in the next section.

Ontology Tools We used the open source ontology editor Protégé[2] to build our EA ontology and to instantiate Archisurance model. We use Apache Jena[3] semantic web application framework for API-based access to ontologies specified in web ontology language (OWL). To programmatically invoke reasoner services, we use Pellet[4] API.

[2] `http://protege.stanford.edu/` Protégé Ontology Editor
[3] `http://jena.apache.org/` Apache Jena
[4] `http://clarkparsia.com/pellet/features` Pellet Reasoner

For querying as well as executing rules over Archisurance model, we rely on SPARQL[5] which is a query language for resource description framework (RDF - Serialization format for OWL ontologies).

3 Change Impact Analysis for EA

Change impact analysis for EA is concerned with computing the effects of change in any part of an enterprise on the rest of the enterprise [8]. For instance, changes in an enterprise's strategy can have multiple significant consequences in all three layers of an enterprise including business processes, organization structure, data management and technical infrastructure creating *ripple effects*. The basic use of change impact analysis is to find out what would happen if a change occurs before it actually happens. This is particularly relevant in the integration effort in Archisurance, as indicators of what can be integrated and what needs to remain same as before [8].

The basic idea of change impact analysis in EA is centered around a set of heuristic rules based on the nature of relations that connect concepts. Such heuristic rules take the form of '*If there is a relation of kind X between concepts A and B, then when A is deleted/modified B needs to be deleted/modified/is going to be dangled.*', where X is an ArchiMate relation [8]. Table 1 shows the heuristic rules for various ArchiMate relations. Note that EA change impact analysis may neither be restricted to this particular type of change impact computation, nor it is our intent to demonstrate how good this type of analysis is compared to other methods of change impact computation or to formalize change impact analysis for EA. We demonstrate in the following how our ontological representation facilitates change impact analysis and makes it easier to examine the ripple effect described in [8].

Listing 1.1: INSERT Query Form for composedOf relation in SPARQL

```
INSERT
    { ?b :Deleted true . } # Then b is deleted as well.
WHERE
    {
    # If a is composed of b
    ?a :composedOf ?b .
    # and a is deleted
    ?a :Deleted true .
    };
```

EA Change Impact Analysis with Ontology A rule can be specified in SPARQL using the CONSTRUCT query form. CONSTRUCT generates new facts based on existing facts that match patterns specified in the WHERE clause. The facts generated by CONSTRUCT are nevertheless not updated in the base ontology. For this we can use the INSERT query form.

Listing 1.1 shows that when a is composed of b and a is deleted, then so should be b. This SPARQL query is in *Terp* format which is a combination of Turtle and Manchester syntax for RDF serialization[6]. The WHERE clause specifies *if* part and CONSTRUCT

[5] http://www.w3.org/TR/rdf-sparql-query/ SPARQL RDF Query Language
[6] www.w3.org/2007/02/turtle/primer/ Turtle Syntax for SPARQL

Table 1: Heuristic Rules Capturing Change in EA

Relation **X**	When Concept **A** **<X>** Concept **B**	Notes
accesses	*A.Deleted >> #*	Generally, A is a behavioral concept accessing data object B
	A.Modified >> B.Modified	To maintain integrity of model, B ***may*** need to be modified
	B.Deleted >> A.Dangled	Signal to enterprise architect to adjust model
	B.Modified >> A.Modified	The way A accesses B ***may*** need to be modified
assignedTo	*A.Deleted/Modified >> B.Dangled*	Deleting/modifying A may result in dangled B, for which enterprise architect needs to be signaled
	B.Deleted >> #	
	B.Modified >> A.Modified	
usedBy	*A.Deleted >> B.Dangled*	B is generally declared to the environment. If A is deleted, B cannot use it anymore; A should be replaced by something that will satisfy B's requirement
	A.Modified >> B.Modified	
	B.Deleted >> #	
	B.Modified >> A.Modified	
realises	***A.Deleted >> B.Deleted***	B is generally a logical entity while a concrete entity A realizes it
	A.Modified >> B.Modified	
	B.Deleted >> #	
	B.Modified >> A.Modified	
triggers	*A.Modified/B.Modified >> #*	Since B starts *after* A, they are isolated and changes in either do not affect the other
	A.Deleted >> B.Dangled	
	B.Deleted>> #	If after deleting A, there is no trigger left for B, enterprise architect needs to be signaled
composedOf	***A.Deleted >> B.Deleted***	B cannot exist without A
	A.Modified >> B.Modified	A's modification may need modifying B; similarly change in B may require change in A
	B.Deleted >> A.Modified	
	B.Modified >> A.Modified	

>> - Implies, **#** - Don't Care

clause specifies the *then* part of a rule. Unlike CONSTRUCT, INSERT actually updates the boolean data type property *Deleted* to true. Note that similar to INSERT there is a query form called DELETE, but we do not want to delete anything from the underlying ontology, only indicate using a boolean flag that a concept is deleted.

Executing such rules over an ontology is achieved by loading the ontology (.owl file created, say in Protégé) as an Apache Jena ontology model via Pellet reasoner factory. Then, a GraphStore is created with this model which acts as a container for graphs of triples to be updated in the underlying ontology. The INSERT rule for instance can then be executed over this ontology and the updated ontology is returned via updated GraphStore. When talking about updating ontology, we are referring only to individuals rather than classes. Only the model is updated, not the metamodel.

Implementation Results To see the effect of executing such rules over the representation of entire enterprise, we refer the reader to Figure 2 which shows a snapshot of Archisurance with concepts that are related to the *ApplicationComponent* **HomeNAwayPolicyAdministration**. We wish to know what would happen if this concept is deleted. As shown in Table 1, deleting concepts leads to deleting concepts they are related to when relations are *realises* and *composedOf*. Deletion of a concept is treated as a trigger for a change ripple.

To actually affect change ripples, we have to execute INSERT for all relations as enumerated in Table 1 in one iteration. The iterations continue, until no new nodes (concepts) in the GraphStore have their *Deleted* property updated to true. This is shown in Listing 1.2. All the *Update strings are essentially INSERT queries similar to Listing 1.1. Once the iterations stop, we get all nodes (concepts) that are deleted (i.e., need

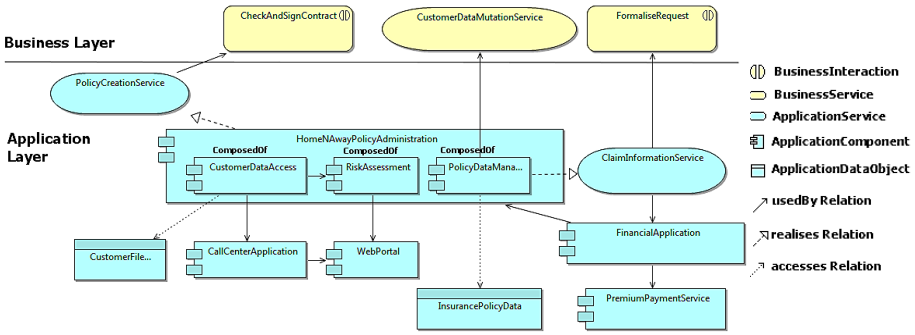

Fig. 2: A Snapshot of Business and Application Layer Concepts in Archisurance

to be deleted) or have been dangled (i.e., are potentially dangling) due to deletion of **HomeNAwayPolicyAdministration**.

Listing 1.2: Ripple Effect Computation due to Deletion of a Concept

```
1  public void affectRipples(String startConcept) {
2    ...
3    while (rippleOut){
4      UpdateAction.parseExecute(prefix + accessUpdate , graphStore) ;
5      UpdateAction.parseExecute(prefix + assignedToUpdate , graphStore) ;
6      UpdateAction.parseExecute(prefix + usedByUpdate , graphStore) ;
7      // also execute usedBy, realises, triggers, & composedOf over graphStore
8      ...
9      resultsNumNodes = com.clarkparsia.pellet.sparqldl.jena.
           SparqlDLExecutionFactory.
10      create(numNodesUpdated, model ).execSelect();
11     while (resultsNumNodes.hasNext()) {
12       QuerySolution row= resultsNumNodes.next();
13       RDFNode concept= row.get("updatedConcepts");
14       ... // Collect concepts that changed
15     }
16     ... // Continue until no new concepts change
17   }
18 }
```

Upon executing, we find that concepts that are deleted are {**PolicyDataManagement, CustomerDataAccess, RiskAssessment, ClaimInformationService, PolicyCreationService**}. Similarly concepts whose relations could now be potentially dangling are {**FinancialApplication, CheckAndSignContract, WebPortal, CallCenterApplication, FormaliseRequest, CustomerDataMutationService**}. Note that effects of this deletion reach from the application layer to the business layer due to relations between affected concepts. Deletion of **HomeNAwayPolicyAdministration** results in making the relations of all business layer concepts shown in Figure 2 to be potentially dangling. Only the *accesses* relations to concepts {**CustomerFileData, InsurancePolicyData**} and *usedBy* relation to {**PremiumPaymentService**} remain unaffected.

It is possible in this way to represent effects of any change specified in Table 1. Since we do not actually delete a concept in the ontology, we do not have to order the updates.

For instance, **RiskAssessment** application component is both deleted (because **HomeNAwayPolicyAdministration** is deleted) and dangled (because **CustomerDataAccess** is *usedBy* **RiskAssessment** and **CustomerDataAccess** is deleted because **HomeNAwayPolicyAdministration** is deleted). We simply indicate that both deletion and dangling is possible for the concept **RiskAssessment**.

With this implementation infrastructure, it is easily possible to see which concepts of a given kind in a given layer are most important and any change to these should be treated with care. It is also possible to go from a coarser level of changes (i.e., deletion or modification or concepts) to finer levels where value of a specific property changes for a given concept leading to similar changes in properties of other concepts. The point we want to stress is that with the ontological representation and rule execution, variants of change can be easily conceptualized and tested for impact analysis in EA.

4 Related Work and Conclusion

Previous ontology approaches targeted specific aspects of enterprises rather than taking a holistic view of them, for instance activities and resources [4,14], tasks and workflows [6] in organization [15], and strategy and marketing [5] etc. Our basic motivation for enterprise modeling is that point views are insufficient to tackle rising complexities. This is where EA frameworks come into play. While we based our ontology on EA framework and modeling language ArchiMate, it is equally possible to do the same using other frameworks. While ontological representation makes it straight forward to apply structural and behavioral analyses, further investigations are needed to conduct EA analyses such as capturing strategic intentions of actors in an enterprise [16] and scenario playing for enterprises with system dynamics [17].

EA frameworks provide holistic treatment of enterprise systems but lack machine-processability, modeling assessment, and analyzability. Ontologies help in addressing these issues in general and we showed in this paper how current ontology tools can be utilized in concert to create machine-processable enterprise models based on ArchiMate EA framework. We also showed that various existing EA analyses that are based on the nature of concepts and relations can be readily prototyped with this infrastructure. The same advantages can be obtained if EA ontology was based on any other EA framework. Transferring the results of EA analysis to actual enterprise still requires human intervention and automating this to the maximum extent possible constitutes part of our ongoing work. Yet we believe that using ontologies to address these issues as shown in this paper takes a small step in that direction.

References

1. Kulkarni, V., Reddy, S., Rajbhoj, A.: Scaling Up Model Driven Engineering - Experience and Lessons Learnt. In Petriu, D.C., Rouquette, N., Haugen, Ø., eds.: MoDELS (2). Volume 6395 of Lecture Notes in Computer Science., Springer (2010) 331–345
2. Kulkarni, V., Barat, S., Roychoudhury, S.: Towards Business Application Product Lines. [18] 285–301
3. Sunkle, S., Kulkarni, V.: Cost Estimation For Model-driven Engineering. [18] 659–675

4. Fox, M.S.: The TOVE Project Towards a Common-sense Model of the Enterprise. In: Proceedings of the 5th international conference on Industrial and engineering applications of artificial intelligence and expert systems. IEA/AIE '92, London, UK, UK, Springer-Verlag (1992) 25–34
5. Uschold, M., King, M., House, R., Moralee, S., Zorgios, Y.: The Enterprise Ontology. The Knowledge Engineering Review **13** (1998) 31–89
6. Fraser, J., Tate, A., Bridge, S.: The Enterprise Tool Set - An Open Enterprise Architecture (1995)
7. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer (2005)
8. de Boer, F.S., Bonsangue, M.M., Groenewegen, L., Stam, A., Stevens, S., van der Torre, L.W.N.: Change Impact Analysis Of Enterprise Architectures. In Zhang, D., Khoshgoftaar, T.M., Shyu, M.L., eds.: IRI, IEEE Systems, Man, and Cybernetics Society (2005) 177–181
9. IEEE: Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Std 1471-2000 (2000)
10. Kulkarni, V., Roychoudhury, S., Sunkle, S., Clark, T., Barn, B.: Modeling and Enterprises - The Past, the Present, and the Future. In: MODELSWARD'13. (2013) Accepted.
11. Wagter, R., Proper, E., Witte, D.: A Practice-based Framework For Enterprise Coherence. In Proper, E., Gaaloul, K., Harmsen, F., Wrycza, S., eds.: PRET. Volume 120 of Lecture Notes in Business Information Processing., Springer (2012) 77–95
12. Haren, V., Publishing, V.H.: ArchiMate 2. 0 Specification. Van Haren Publishing Series. Bernan Assoc (2012)
13. Jonkers, H., Band, I., Quartel, D.: Archisurance Case Study. The Open Group Case Study (Document Number Y121) (January 2012)
14. Gruninger, M., Fox, M.S.: An Activity Ontology For Enterprise Modelling. `http://www.eil.utoronto.ca/tove/active/active.html` (June 1994)
15. Fox, M.S., Barbuceanu, M., Gruninger, M.: An Organisation Ontology For Enterprise Modelling: Preliminary Concepts For Linking Structure And Behaviour. In: Proceedings of the 4th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'95). WET-ICE '95, Washington, DC, USA, IEEE Computer Society (1995) 71–
16. Yu, E.S.K., Strohmaier, M., Deng, X.: Exploring Intentional Modeling and Analysis for Enterprise Architecture. In: Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC) Workshops. (2006) 32
17. Sterman, J.D.: Business Dynamics: Systems Thinking and Modeling for a Complex World. Irwin/McGraw-Hill (2000)
18. France, R.B., Kazmeier, J., Breu, R., Atkinson, C., eds.: Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings. In France, R.B., Kazmeier, J., Breu, R., Atkinson, C., eds.: MoDELS. Volume 7590 of Lecture Notes in Computer Science., Springer (2012)