OWL ONTOLOGY TO ECORE METAMODEL TRANSFORMATION FOR DESIGNING A DOMAIN SPECIFIC LANGUAGE TO DEVELOP AVIATION SCENARIOS

Shafagh Jafer
Electrical, Computer, Software and Systems Eng.
Embry-Riddle Aeronautical University
600 S. Clyde Morris Blvd.
Daytona Beach, FL, USA
jafers@erau.edu

Bharvi Chhaya
Electrical, Computer, Software and Systems Eng.
Embry-Riddle Aeronautical University
600 S. Clyde Morris Blvd.
Daytona Beach, FL, USA
chhayab@my.erau.edu

Umut Durak
Institute of Flight Systems
German Aerospace Center (DLR)
Braunschweig, Germany
umut.durak@dlr.de

ABSTRACT

Ontology-driven software development has gained significant interest in domain-specific application development. Ontologies are treated as formal models representing knowledge, which can be used for automatic code generation. Among existing technologies, Eclipse Modeling Framework (EMF) has been widely used to generate domain-specific metamodels and Java code using Ecore. Such robust technologies suggest mapping ontologies to metamodels through series of transformations. In this paper we suggest transforming an ontology model to EMF Ecore metamodel for designing a Domain-Specific Language (DSL). We present the ontology developed for Aviation Scenario Definition Language (ASDL) to capture simulation scenarios in the aviation domain. We demonstrate key terminologies and their definitions in order to generate flight simulation scenarios. The use of Web Ontology Language (OWL) in developing ASDL ontology is discussed and its relation to ASDL Ecore metamodel is presented. This paper finally elaborates the challenges and shortfalls in mapping OWL ontology to Ecore metamodel in EMF.

Keywords: domain-specific, ontology, OWL, Ecore, transformation.

1 INTRODUCTION

A model is a simplified, usually smaller-scale, representation of a system that exhibits all the characteristics of the actual system. A simulation is the manipulation of a system to comprehend its behavior under certain applied conditions. Together, modeling and simulation (M&S) is the discipline for developing a level of understanding of the interaction of the different parts of a system, and of the system as a whole in a swift, accurate and economical manner. However, the execution of any simulation requires a clearly-defined scenario which is being examined. This simulation scenario needs the description of all initial and terminal conditions, significant events that occur within the simulation timeframe and a well-defined environment in addition to the major entities, their capabilities, behavior and interactions over time. In order to define a metamodel that can describe such scenarios, it is essential to have an understanding of the related domain and the key words and phrases associated with it.

In philosophy, the term ontology means a systematic explanation of existence. Gruber defined it for computer science as: “Ontology is an explicit specification of a shared conceptualization” (Gruber 1995). Ontologies are necessary in the software domain in order to enable modeling, sharing and reusing the knowledge of organizations or disciplines. Availability of such ontologies would facilitate their use as models for learning the principles of the domain it describes. A major benefit of using ontologies is their capacity to be extended using new knowledge whenever it is obtained. The Web Ontology Language (Bechhofer 2009) is a language for defining ontologies on the Web. An OWL ontology describes a domain in terms of classes, properties and individuals and may include rich descriptions of the characteristics of those objects (Protégé Home Page 2016).

This paper discusses the gap between the ontology in OWL and the metamodel in EMF during Domain Specific Language (DSL) design and comments on the effort needed to bring them together. Particularly, we are interested in transformation of ontology from OWL to EMF. As the application case we will use the ontology we have built for Aviation Scenario Definition Language (Jafer, et al. 2016) and demonstrate its mapping process from OWL to Ecore metamodel in EMF.

2 BACKGROUND

2.1 Domain-Specific Language

A Domain-Specific Language (DSL) is a custom tailored computer language for a particular application domain. DSL is created to specifically target problems in a specific domain, and stresses upon the main ideas, features, constraints, and characteristics of that domain. DSL enables developers to construct models that are specific to their application. These models are mainly composed of elements and relationships that are verified to be valid for that application. One of the greatest benefits of DSL is allowing non-developers and people who are not experts in the domain to understand the overall design. This is normally supported by allowing graphical modeling, usually in the form of a drag and drop capability to construct models. DSL augmented with model-to-text transformation capabilities directly allows for automatic generation of source code from model.

The Aviation Scenario Definition Language (ASDL) has been proposed as a DSL for aviation scenario generation (Jafer, et al. 2016). ASDL aims at providing a standard scenario specification in order to obtain a common mechanism for verifying and executing aviation scenarios, and also enabling effective sharing of scenarios among various simulation environments and allowing for reuse of scenario specifications. Based on DSL design methodologies, ASDL will provide a well-structured definition language to define departure, enroute, re-route, and landing scenarios.

2.2 Ontology

An ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the meaning of terms used in the vocabulary (Yao and Zhang 2009). One of the benefits of using ontologies is their capacity to be easily extended using new knowledge generated by experts so all existing ontologies can be used as a starting point for further development (Hilera and Fernández-Sanz 2010). Among the existing ontology specification frameworks, the Web Ontology Language format (OWL) is most commonly used by the DSL community. OWL enables describing a domain in terms of classes, properties and individuals and may include rich descriptions of the characteristics of those objects (Bechhofer 2009).

Ontology-Driven Software Development (ODSD) has emerged as a significant mechanism in creating domain-specific languages (Čeh, et al. 2011; Pan, et al. 2012), allowing for expressing domain concepts effectively. Ontology provides a quick and simplified description of a DSL, abstracting language’s technically details, while highlighting key terminology and specifics. Once an ontology is built, it is a simple process to generate the language’s metamodel and establish relationships among related concepts. An automated process that takes in DSL’s ontology and generates its corresponding metamodel sounds highly efficient. This has been studied in various development environments including EMF. In the next section, we provide a quick overview of current technologies that support automatic translation of ontology into EMF Ecore metamodels.

2.3 State-of-the-art

A number of research efforts have stressed out the importance of automatic transformation of ontology to DSL metamodel for reducing inconsistency and improving language development efficiency. Particularly, automated transformation of OWL to Ecore has been investigated by the ontology community. Among those, the work presented by Rahmani, Oberle and Dahms (2010) provides a theoretic approach to an adjustable OWL to Ecore transformation. They suggest the transformation as an Eclipse plug-in that would integrate seamlessly with the IDE. Although the authors provide the theoretical foundations of such transformation, the paper does not report on practical use and the work was in fact never implemented. Thus, it is infeasible to comment on the usability and practicality of the suggest technique. OWL2EMF (Stolp 2009) was reported as a graduate research project involving transformation of OWL ontology to an intermediate OwlCat format, an Ecore model generated and manipulated by EMF. The work discussed the transformation algorithm details, however, the implementation (source code) of the project was unavailable, hence could not be utilized by the modeling community and falling short in providing a readily usable OWL to Ecore transformation technology. The EMF4SW (Nijenhuis 2011) eclipse plug-ins provide OWL to Ecore and Ecore to OWL transformations capabilities. However, the effort states the need for tuning the translation rules to adjust for your specific DSL, offering a basic transformation framework as opposed to a rather robust solution that can be used by DSL developers. The EMF4SW plug-in source code is available on GitHub (EMF4SW 2016), however, no contribution has been made to the project since 2012. In contrast, the OWLizer project (OWLizer 2016) attempts to translate Ecore metamodels into OWL ontology.

Given the literature above, there is a clear need for a generic OWL to Ecore plug-in that can be used by any DLS developer without a significant need to customize such framework. In this paper we will comment on the step-by-step process and challenges of developing OWL to Ecore automated transformation solution.

3 ASDL ONTOLOGY

In order to capture aircraft landing details, it is essential to have a definitions reference list that highlights all key terminology as well as procedures and operations that are communicated between the pilot and ATC. The United States' FAA and European SESAR programs provide inclusive glossaries that provide key terminology and concept of operations (Federal Aviation Administration 2012) (SESAR 2015). A review of existing ontologies resulted in one aviation ontology being discovered. However, this described the structural and physical entities of an aircraft, and hence did not provide any useful terms that could be reused. Thus, a new aviation-specific ontology was created for this project, which was further used to develop the model and in the implementation of the scenarios.

The ontology consists of two parts: keywords that describe the physical model and operation of flights, and words that describe key communication between the control tower and pilots. This section lists majority of these keywords along with their definitions and use. A complete ASDL ontology can be accessed online (ASDL Ontology 2016). Once over 100 keywords were identified, the primarily used terms were added to a basic ontology created using Protégé (Alatrish 2013), which saves them in OWL format. The Web Ontology Language (Bechhofer 2009) is a language for defining ontologies on the Web. An OWL ontology describes a domain in terms of classes, properties and individuals and may include rich descriptions of the characteristics of those objects (Protégé Home Page 2016).

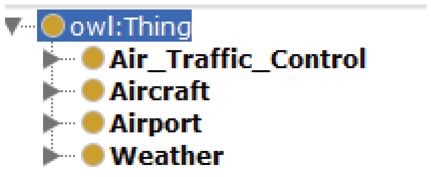

Figure 1: High-Level View of ASDL Ontology.

An ontology focuses mainly on classes which describe the concepts of the domain. It follows a hierarchical model where subclasses are all necessarily a part of the superclass (Noy and McGuinness 2001). The ASDL ontology has four base classes: Air_Traffic_Control, Aircraft, Airport and Weather. This can be seen in Figure 1 below. All these terms have been defined in Table 1.

Table 1: Definition of terms in base class of ASDL Ontology.

Term	Definition
Air Traffic Control	A service operated by appropriate authority to promote the safe, orderly and expeditious flow of air traffic.
Aircraft	Any machine that can derive support in the atmosphere from the reactions of the air other than the reactions of the air against the earth's surface.
Airport	An area on land or water that is used or intended to be used for the landing and takeoff of aircraft and includes its buildings and facilities, if any.
Weather	The state of the atmosphere at a place and time as regards heat, dryness, sunshine, wind, rain, etc.

As shown in Figure 2, the ATC class includes a Controller and various key terms that need to be used in conversation. The main part of this ontology involves the aircraft and its properties. Figure 3 shows the subclasses of the Aircraft class.

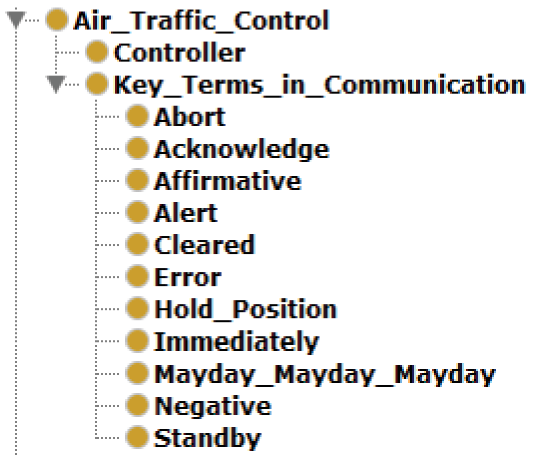

Figure 2: Elements present in ATC Class.

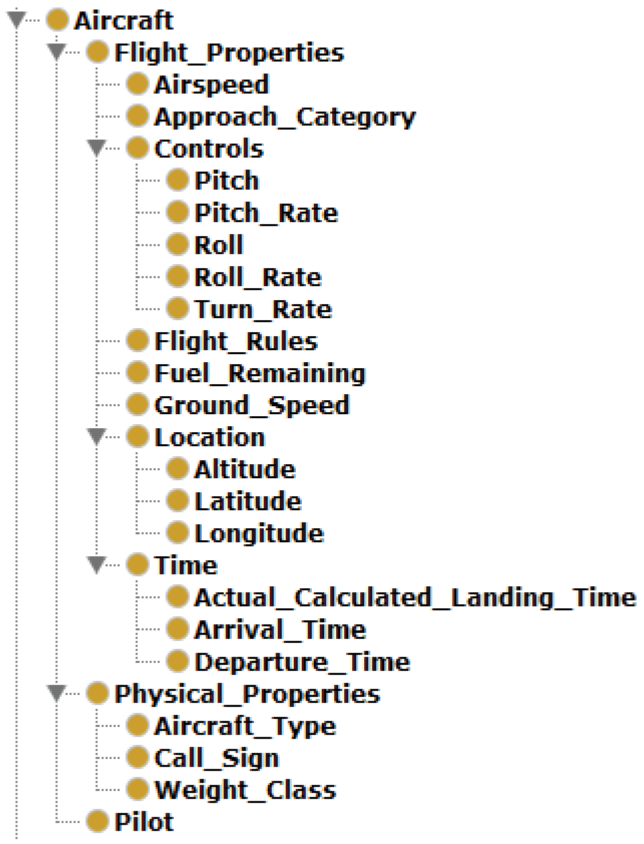

Figure 3: Elements present in Aircraft Class.

The Flight_Properties subclass describes the rules (IFR or VFR) that govern the flight, the speed of the aircraft, the fuel remaining and has three other subclasses: controls (pitch, roll and turn rates), location (altitude, latitude, longitude) and time (arrival time, departure time and ACLT). The physical properties subclass contains the call sign, type of aircraft and its weight class.

The Airport class includes an identifier, the details of terminals and gates present in the airport, its elevation as well as runway details. Each runway's information also includes its heading. These can be seen below in Figure 4. Figure 5 shows the items present in the Weather class. This includes the cross-wind component (in case of crosswind landings), the temperature, dew point, sky condition, visibility, wake turbulence and wind shear as relevant to the scenario.

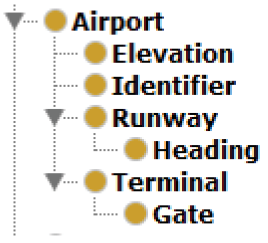

Figure 4: Ontology Elements of Airport Class.

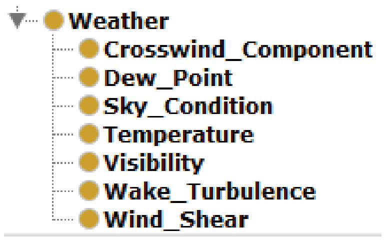

Figure 5: Ontology Elements of Weather Class.

Using Protégé, instances can also be created of all these classes having the requisite properties. There is a large scope for addition of various related items to the ontology; this is only a basic framework that lists the main items that are used in this model. A more comprehensive list of definitions is available online at (ASDL Ontology 2016).

4 ONTOLOGY TO METAMODEL TRANSFORMATION

Eclipse Modeling Framework (EMF) is "a modeling framework and code generation facility for building tools and other applications based on a structured data model" (Eclipse 2016). Once a model specification has been described, EMF provides tools to produce a set of Java classes for the model, along with a set of adapter classes which enable viewing and editing of the model. The first step is to have a design of the structure of the data which includes all data items and the relationships between them. This can then be defined in EMF in the Ecore format, which is basically a subset of UML Class diagrams. This is the metamodel, which describes the structure of the model. A metamodel can further be used to generate a model, which is a concrete instance of this structured data. The Ecore file allows users to define the following elements for the model: (1) EClass: a class with zero or more attributes and references, (2) EAttribute: an attribute of the class which has a name and a type, (3) EReference: an association between two classes, and (4) EDataType: the data type of an attribute.

The framework for ASDL was developed in two main stages, as follows:

- **Stage 1:** An aviation scenario metamodel was developed in order to capture the necessary characteristics of a flight. This drew upon the ontology developed in the first part of the project in order to define these attributes.
- **Stage 2:** The aviation metamodel was integrated with the BOM metamodel in order to define scenarios with specific aviation-related properties.

The steps followed for this were obtained from the framework provided by Durak et al. (2014): (1) define the classes required to accurately represent the model, (2) determine the attributes used to describe the classes, (3) define the structure and relationships between the classes, (4) create an Ecore model based on the entities identified, (5) integrate this model of aviation entities into the BOM framework, (6) generate Java code for the model, and (7) create a runtime instance and use it to define and edit an aviation scenario. The first four steps here were part of Stage 1 of the project and the next three were part of Stage 2. It can be seen from the description above that the first three steps can be completed by creating a well-defined ontology for the domain. Producing an ontology was thus a crucial step towards being able to execute Stage 1 of this project.

Once all keywords had been identified and defined, the metamodel could be created in EMF. All the main classes were already arranged hierarchically within the ontology, so a direct correlation was made between the elements present in the ontology and those needed in the metamodel. All top-level elements within the ontology were defined as classes in the metamodel and those at the lowest level in the ontology were defined as attributes of their parent classes. References were added manually based on the order of elements in the ontology. The various elements described in Section 3 can be seen mapped on to the final ASDL metamodel as shown in Figure 6. Comparing this to Figures 2-5, it can be observed that all the classes and the attributes within the final metamodel have been defined earlier in the ontology and their references and connections to other elements have been preserved during this transformation. To compare the two, it can be seen that Weather is modeled as a class in Figure 6 and has as its attributes all the child elements that are seen in Figure 5.

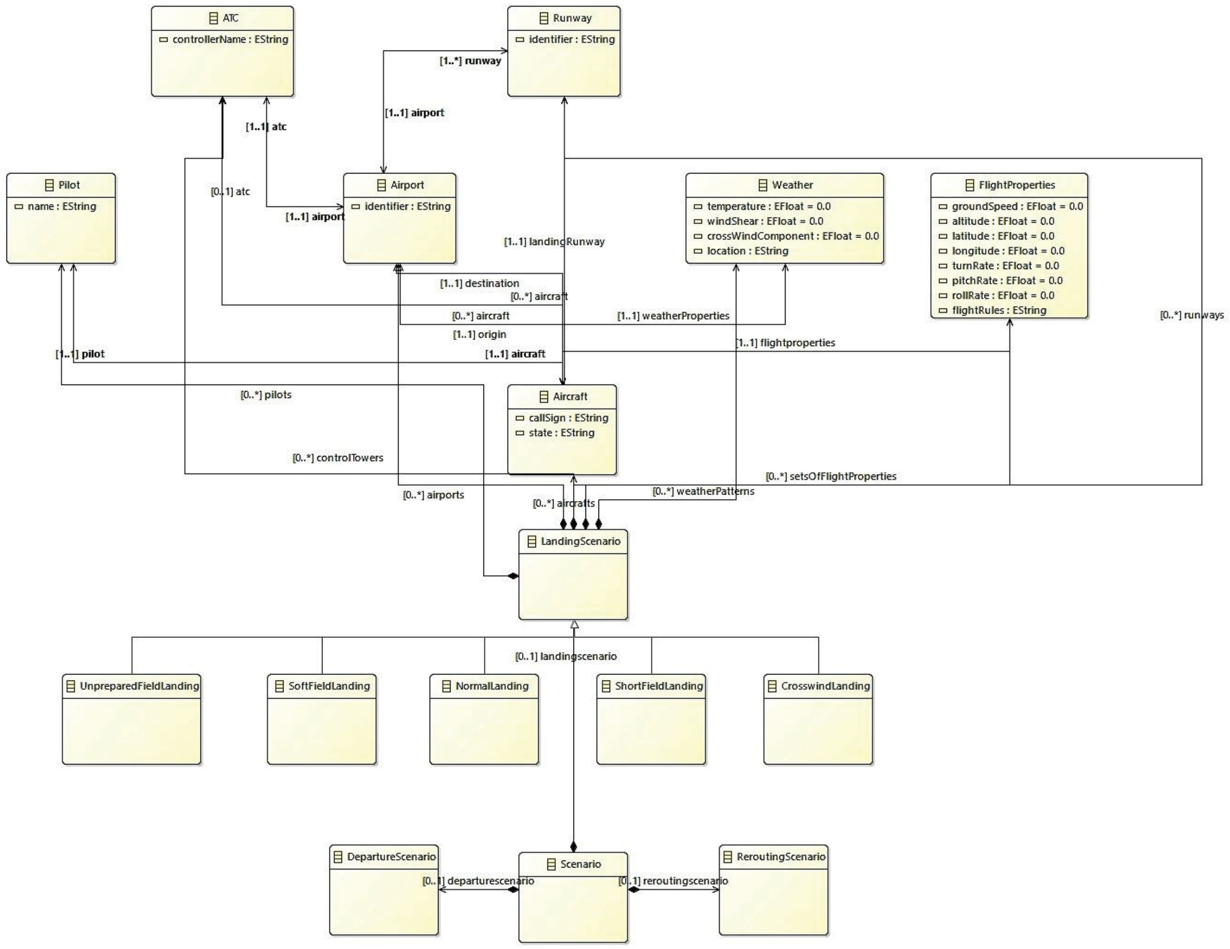

Figure 6: ASDL Metamodel defined in EMF.

In this case, references were added manually for all elements as there were no common rules defined for this process. However, this would affect scalability greatly as it may not be feasible to create these individually for all elements in a larger metamodel. While the connection between subclasses and superclasses can be determined easily, it is suggested that relationships among classes at the same level be specified within the ontology so that a rule can be created for direct translation to a different metamodel. As discussed in the future work section of this paper, automating this process is a work in progress and we are investigating feasible approaches to utilize rules and automated transformations.

5 AUTOMATED MAPPING OF OWL TO METAMODEL

5.1 Automated Transformation

Eclipse plug-ins are Java libraries that are run within Eclipse and can use Eclipse Integrated Development Environment (IDE) to extend Eclipse's functionality in many different ways. In general, a plug-in in Eclipse is a component that provides a certain type of service within the context of the Eclipse workbench. The automation of the mapping process from ontology to metamodel can be accomplished by creating an Eclipse plug-in that can read the Ontology files in OWL/XML format and convert them into Ecore objects using a set of established rules as has been shown in Figure 7. The transformation process itself could take place in several steps as suggested in Figure 8.

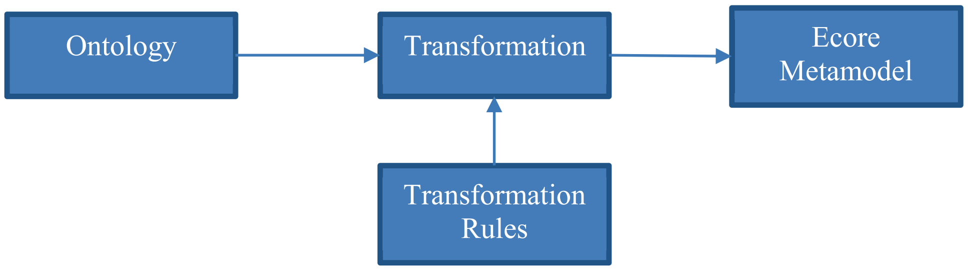

Figure 7: Transformation from Ontology to Metamodel.

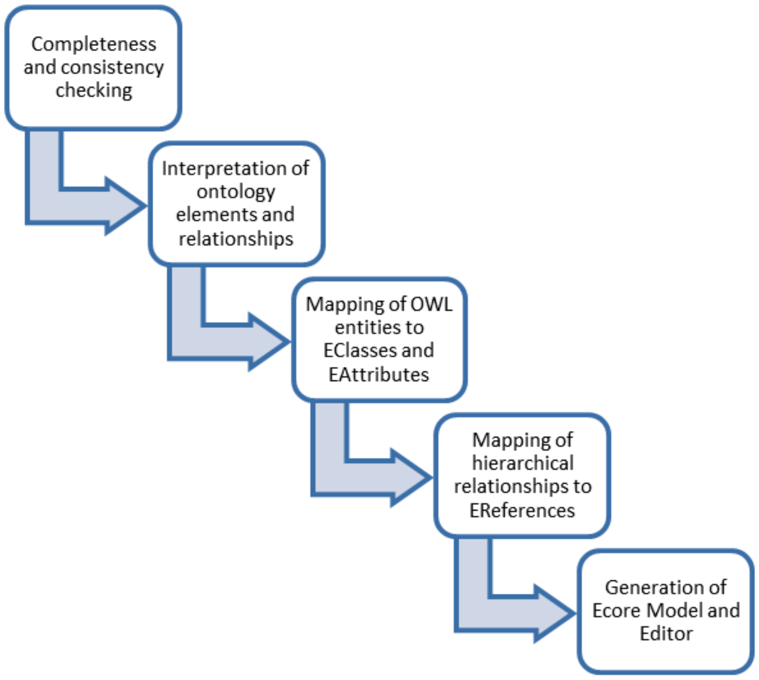
Completeness and consistency checking
Interpretation of ontology elements and relationships
Mapping of OWL entities to EClasses and EAttributes
Mapping of hierarchical relationships to EReferences
Generation of Ecore Model and Editor

Figure 8: Expansion of Transformation Block.

In order for the ontology to be the only source of data used to create the metamodel, it is essential that it is both accurate in representing the domain and consistent in its definition. A completeness and consistency check is necessary in order to be confident about the accuracy of the resulting metamodel. Thus, this would be the first step within the transformation process, and in case a validation check fails, the rest of the transformation will not be able to execute.

Once the ontology has been accepted as a valid input, the next step would be to interpret it. Within this interpretation phase, elements will be separated based on their types and their relationships. A record of parent elements and all their child elements will be kept along with how they are related. This relationship is generally of the subsumption type, which is also known as *is-a* and is observed between superclasses and their subclasses, or of the mereology type, which is also known as *part-of* and is observed when different elements combine together to form a composite element.

After interpreting the ontology, the chief portion of the transformation needs to occur. This is the mapping of the OWL elements to the EMF elements in order to define the metamodel. For this part, transformation rules would be needed within the plug-in. A transformation rule describes a component of an existing metamodel and its corresponding component in the newer metamodel. It explains how all these elements can be mapped to each other. These rules form the basis of the entire model transformation and require flexibility and should be editable as needed on a case-by-case basis. These necessitate an understanding of the syntax and semantics of both the source and target metamodel languages (Sendall and Kozaczynski 2003). To an extent, these are also dependent upon the metamodel itself as each item may not be related to others in the same way in different situations. In order to make the plug-in more generic, different sets of rules will need to be defined and the user can select the one to be used for each particular transformation. For example, in the case of ASDL and other languages with a similarly-defined ontology, the rules would be that each parent element should also be an EClass in the metamodel. All elements at the lowest level of hierarchy would be EAttributes within their parent classes.

After defining all the elements and as many relationships as available within the ontology, the Ecore metamodel can be generated using the tools available in Eclipse and would be available for graphical editing in EMF.

5.2 Challenges and Shortfalls

A major difficulty in creating such a plug-in is that apart from the relationship between parent and child elements, the references between various elements cannot be modeled very specifically within an Ontology. The hierarchical definition in subclass-superclass form and containment relationships can be modeled easily, but the interactions between elements at the same level (subclasses of the same superclass) cannot be defined clearly within the ontology. Hence, even when this plug-in is created, it would not completely automate the creation of a metamodel as references would have to be defined manually.

Another challenge exists within the context of the design of this tool. It is difficult to think about all possible transformation rules when creating the plug-in, so it is likely that the tool will need to be refined continuously as different transformations are attempted in order to capture all the nuances of both the ontology language and EMF.

6 CONCLUSION AND FUTURE REMARKS

Ontology-driven software development has gained significant interest in domain-specific application development. The creation of an ontology is the first step towards the design of a DSL as it helps define the classes, attributes and relationships which describe the domain. Right now, the ontology and DSL metamodels are separate entities which are connected only by the domain they describe. The ontology needs to be manually translated into a metamodel that defines the properties of the domain. A plug-in that can automate this process and convert the ontology into a basic metamodel definition in Ecore could save countless hours of repetitive work. Such a plug-in has been suggested here as an alternative to manual interpretation of ontologies and design of a DSL metamodel. The creation of this plug-in would be the next step for this project in order to automate the process for future use. Some shortfalls of this idea include the manual context required in understanding relationships between various elements and the various different ways in which elements can be mapped from one format to another. At the moment, this can be done in a limited way with some manual input and would require customization of the transformation rules based on the application.

REFERENCES

Alatrish, Emhimed. 2013. "Comparison Some of Ontology Editors." *Management Information Systems* 8 (2): 018-024.

2016. *ASDL Ontology.* https://github.com/ASDL-prj/Ontology.

Bechhofer, Sean. 2009. "OWL: Web Ontology Language." In *Encyclopedia of Database Systems*, by Ling Liu and M. Tamer Özsu, 2008-2009. Boston: Springer US.

Čeh, Ines, Matej Črepinšek, Tomaž Kosar, and Marjan Mernik. 2011. "Ontology driven development of domain-specific languages." *Computer Science and Information Systems 2* 317-342.

Durak, Umut, Okan Topcu, Robert Siegfried, and Halit Oguztuzun. 2014. "Scenario Development: A Model-Driven Engineering Perspective." *Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH), 2014 International Conference on.* IEEE. 117-124.

Eclipse. 2016. *Eclipse Modeling Project.* https://eclipse.org/modeling/emf/.

2016. *EMF4SW.* https://github.com/ghillairet/emf4sw.

Federal Aviation Administration. 2012. *AFS Flight Program Flight Operations Manual.* Federal Aviation Administration.

Gruber, Thomas R. 1995. "Toward principles for the design of ontologies used for knowledge sharing?" *International journal of human-computer studies* 43 (5): 907-928.

Hilera, José, and Luis Fernández-Sanz. 2010. "Developing Domain-ontologies to Improve Sofware Engineering Knowledge." *International Conference on Software Engineering Advances.*

Jafer, Shafagh, Bharvi Chhaya, Umut Durak, and Torsten Gerlach. 2016. "Formal Scenario Definition Language for Aviation: Aircraft Landing Case Study." *AIAA Modeling and Simulation Technologies Conference.* 3521.

Nijenhuis, Christiaan Frank. 2011. "Automatic generation of graphical domain ontology editors."

Noy, Natalya F., and Deborah L. McGuinness. 2001. "Ontology development 101: A guide to creating your first ontology."

2016. *OWLizer.* http://st.inf.tu-dresden.de/owlizer/.

Pan, Jeff Z., Steffen Staab, Uwe Aßmann, Jürgen Ebert, and Yuting Zhao, . 2012. *Ontology-driver software development.* Springer Science & Business Media.

2016. *Protégé Home Page.* http://protege.stanford.edu/.

Rahmani, Tirdad, Daniel Oberle, and Marco Dahms. 2010. "An adjustable transformation from owl to ecore." *International Conference on Model Driven Engineering Languages and Systems.* Springer Berlin Heidelberg. 243-257.

Sendall, Shane, and Wojtek Kozaczynski. 2003. "Model transformation: The heart and soul of model-driven software development." *IEEE software* 20 (5): 42-45.

SESAR. 2015. *EUROCONTROL ATM Lexicon.* Single European Sky ATM Research.

Stolp, D. G. A. 2009. "Using ontology mapping to automate derivation of transformations for data integration."

Yao, Zhong, and Quan Zhang. 2009. "Protégé-Based Ontology Knowledge Representation for MIS Courses ." *International Conference on Web Information Systems and Mining.*

AUTHOR BIOGRAPHIES

SHAFAGH JAFER is an Assistant Professor of Software Engineering at Embry-Riddle Aeronautical University. She holds a PhD in Computer Systems Engineering from Carleton University in Canada. Her research interest include simulation-based engineering, mission-critical software systems, and model-based development. Her email address is jafers@erau.edu.

BHARVI CHHAYA is a Ph.D. candidate in Electrical Engineering and Computer Science at Embry-Riddle Aeronautical University. She received her Master of Software Engineering and Bachelor of Science in Aerospace Engineering degrees from Embry-Riddle. Her doctoral research focuses on software modeling, domain-specific language development, and formal modeling and simulation. Her email address is chhayab@my.erau.edu.

UMUT DURAK is a Research Scientist at Institute of Flight Systems of German Aerospace Center (DLR). He holds a PhD in Mechanical Engineering from Middle East Technical University in Turkey. His research interests include model-driven approaches applied to engineering of simulation systems and modeling and simulation based engineering of flight systems. His email address is umut.durak@dlr.de.