Tool Support for Transformation from an OWL Ontology to an HLA Object Model

Özer ÖZDİKİŞ
OYAK Technology
Teknokent, Galyum Blok, Kat:1,
No:23 ODTU-Ankara TURKEY
+90 535 922 46 40
oozdikis@oytek.com.tr

Umut DURAK
TUBITAK-SAGE
PK.16 06261 Mamak
Ankara TURKEY
+90 312 590 91 76
udurak@sage.tubitak.gov.tr

Halit OĞUZTÜZÜN
Middle East Technical University
Computer Engineering Dept.
Ankara TURKEY
+90 312 210 55 87
oguztuzn@ceng.metu.edu.tr

ABSTRACT
Designing simulation architectures based on domain models is a promising approach. Tools to support transformation of formalized domain models to design models are essential. Ontology languages offer a way of formally specifying the domain knowledge. We adopt a user-guided approach to model transformation, where the source is an OWL ontology and the target is an HLA Object Model, in particular, a federation object model (FOM). This paper presents a flexible transformation tool that enables the user to define transformations in terms of mappings from OWL constructs to HLA Object Model Template (OMT) constructs. The overall objective is to facilitate ontology-based model-driven development in distributed simulation.

Categories and Subject Descriptors
I.6.7 [**Simulation Support Systems**]

General Terms
Design

Keywords
Ontology based simulation, model driven development, High Level Architecture, object models.

1. INTRODUCTION
In the context where distributed simulation architectural design and model driven development meet, the issue of transformation of domain models to platform-specific object models arises. A domain model, which captures knowledge from an area of interest, is an outcome of domain analysis. The approach that is based on the use of model transformations from a domain model to design models of varying levels of detail, and finally to code is known as Model Driven Development (MDD) or Model Driven Engineering (MDE). OMG's Model Driven Architecture (MDA) [10] and ISIS' Model Integrated Computing (MIC) [11] are particular manifestations of MDD/MDE.

Ontologies have recently gained popularity for representing domain knowledge for ease of both human understanding and machine processing [17][6]. Using ontologies as domain models is known as ontology based domain engineering [4]. In applying ontology based domain engineering to simulation development, we envision to derive reusable simulation components and artifacts.

A domain model ideally reflects all the stakeholders' views of the problem area. Further, we hold that tool-supported methodologies are required to bring the simulation developer's point of view into life. Our present focus is on tool support for flexible transformations from a domain ontology, which can be regarded as a representation of a simulation conceptual model [16], into an HLA object model.

OWL is an ontology language [15][7], which enjoys popularity due to "semantic web". When it comes to transforming an available OWL ontology into some target model, the "one size fits all" approach does not work. Because every domain model may require a different transformation procedure depending on the context, data types, conventions, and even the personal preferences of the simulation developer.

The tool provides a user interface to configure mappings from OWL constructs to HLA OMT constructs. Then, mapping definitions are applied on a given OWL ontology (formalizing a domain model), and consequently an HLA Object Model, in the form of an XML document [8], is produced.

1.1 Related Work
France and Rumpe [5] discuss how modeling techniques can be effectively leveraged during software development. Moreover, they note that, "there is a growing realization that MDE requires semantic-based manipulation of models". We believe our work takes some steps along this direction.

Tolk in [19] draws attention of the HLA based distributed simulation community to MDA and points out that employing MDA will enable HLA implementers to improve their products by making better use of the commercial technology.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIMUTools 2010 March 15–19, Torremolinos, Malaga, Spain.
Copyright 2010 ICST, ISBN 78-963-9799-87-5.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8678

Miller and Fishwick [12] identify the potential benefits of ontologies for modeling and simulation. In [20], Tolk also stresses the importance of conceptual data models, which can be parts of ontologies, in simulation development. He argues that in a simulation consisting of several participating systems, ontologies can be used to describe their services and information exchange capabilities to satisfy M&S composability and interoperability.

The work by Rathnam and Paredis [18] also addresses the use of ontologies in constructing HLA-based distributed simulations. In their work, the object models, namely, the FOM of a federation and the SOMs of the federates, are represented as ontologies. The mappings between individual SOMs and the FOM are also represented as an ontology. By means of these mappings, the reusability of existing federates in a new federation is facilitated. That approach requires the user design his ontology specific to HLA standards. In our work the ontology captures the simulation conceptual model in a more abstract way, in that it is not specific to HLA or any other simulation standard. HLA-specific information is provided by the transformations from the ontology to the object model.

In a previous study [14], we achieved to provide tool support for user-guided model transformation from ontology to the object oriented design for the simulation software, in the form of a UML class diagram. This present effort is built upon the premise that the domain knowledge that is standardized in the form of a common ontology can be utilized to derive a representation of the information shared among the participants in a distributed simulation.

1.2 Background

This effort builds up a weak analog to the levels of abstraction that are identified in OMG's MDA [10] while developing a tool support for model driven simulation development. MDA presents the abstraction levels of system development as follows: The computation independent viewpoint as the first abstraction level, focuses on the environment in which the system of interest will operate in and on the required features of the systems [5]. The platform independent viewpoint focuses on the aspects of system features that are not likely to change from one platform to another. We expect ontologies to possess both computation independent or platform independent viewpoints depending on their design purpose and content. The next step is platform specific viewpoint which is regarded as the last level of abstraction before the executable code. It specifies how that system utilizes a particular platform.

The idea is that domain knowledge which is captured at a conceptual level will be used to generate the models towards the executable assets as automated as possible utilizing model transformation practices. This effort tries to build a tool to allow the simulation engineer to guide the transformation from conceptual model which is represented by an OWL ontology, to an asset towards executable code, which is an Object Model. Object Models are regarded as HLA specific interface models which then can also be transformed to Federation Design Data [21].

Our transformation process can be located in reference to the four-layer metamodeling hierarchy of OMG's Meta Object Facility (MOF) [13]. MOF is defined as the extensible model driven integration framework for defining, manipulating, and integrating metadata and data in a platform independent manner. It provides a meta-metamodel at the top level, which is called M3 layer of the four-layer metamodeling hierarchy. Any M3-layer meta-metamodel can be used to define more specific metamodels at M2 layer, such as the HLA Object MetaModel (HOMM). A fully-fledged metamodel for the HLA Object Model is provided as a part of the Federation Architecture Metamodel (FAMM) [21]. We have used a scaled down version of HOMM. FAMM employs metaGME, the meta-metamodel provided in GME, a (meta)modeling environment supporting MIC. Object Models conforming to the HOMM (thus, to the HLA OMT standard) are at layer M1. Finally, the M0 layer includes the objects and interactions created during federation execution as instantiations of the Object Model.

A specific ontology can be viewed as conforming to a metamodel (which plays the role of a grammar for an ontology language, such as OWL). Our ontology modeling hierarchy is based on Eclipse Metamodel Framework (EMF) [2]. The meta-metamodel at M3 layer in EMF is called Ecore. IBM implemented an Integrated Ontology Development Toolkit (IODT) for ontology driven development built on EMF [9]. IODT includes a library called EMF Ontology Definition Metamodel (EODM) which is an implementation of the OMG's Ontology Definition Metamodel [3]. EODM has OWL parsing, serialization, and reasoning features.

In the transformation, we have an ontology model as our source and the HLA Object Model as our target. Our approach facilitates the definition of the mappings between the EODM and HOMM at M2 layer. These mappings are applied to a given OWL ontology to generate an HLA Object Model. The modeling layers which are used in this transformation are shown in Figure 1.

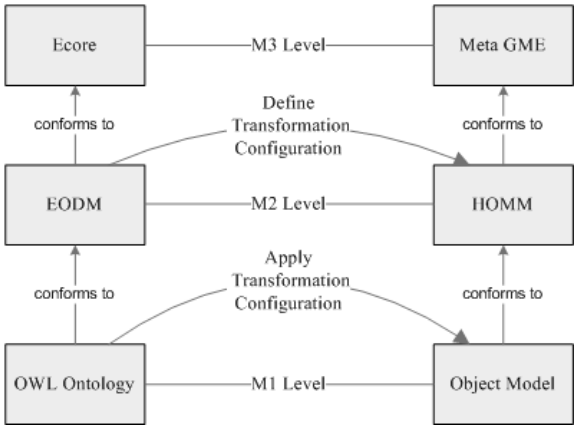

Figure 1. Relations between the modeling layers

In the following sections, available OWL constructs of the source and OMT constructs of the target are introduced. The tool that was developed to configure the mapping from source to target is presented. Finally, our ongoing work on a case study and future work are commented upon.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8678

2. THE OWL-TO-OBJECT MODEL TRANSFORMATION

Our tool lets the user configure the transformations as appropriate. A transformation configuration is composed of mapping groups, mappings and constraints.

A mapping group is a collection of mappings from some specific OWL constructs to some OMT constructs. In other words, a mapping group includes the specification of source OWL constructs and mappings to apply on these constructs. Source constructs can be OWL classes or OWL properties. The user can define constraints on the source OWL constructs, so that the mappings in the mapping group are applied only on the desired subset of source constructs.

Following the description of the source constructs, the user must specify, in terms of mappings, how to use them in the transformation. Depending on the source-target combinations, the user can define four types of mappings in a mapping group: to Object Class, to Attribute, to Interaction Class and to Parameter. A mapping is the prescription of how the target OMT construct should be built using the source OWL construct.

Figure 2 shows a screenshot from the tool. It enables the user to define several mapping groups with several mappings inside. The boxes labeled OC, IC, AT and PR represent four different mapping types regarding OMT constructs.

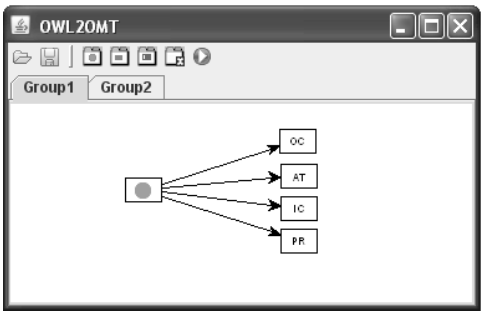

Figure 2. Overview of the UI

Constraints can be defined to restrict the entities to be evaluated in a specific mapping group or mapping. Constraints are actually condition-value pairs applied on an OWL construct. While evaluating an OWL construct in a mapping group or mapping, these condition-value pairs are used to check if that construct is selected and should be processed. As an example, if the user wants to define some mappings on a specific subset of source OWL objects, he must define a new mapping group, then define a constraint for that mapping group to specify the interested OWL objects, and finally define his mappings in that mapping group so that they are applied only on the specified source OWL objects.

The last step of the transformation is the validation of the resulting model. Since this transformation is a user-guided transformation, there may be inconsistencies, for example, a reference to a Dimension that actually does not exist in the object model. Details of validation are explained in the forthcoming sections.

2.1 Available OWL Constructs at the Source

An OWL ontology involves Classes, Object Properties, and Datatype Properties. A Class has a name and possibly super classes. A Class may be defined as an intersection of, union of or complement of other classes. Further, a class may have different types of restrictions, namely, MinCardinalityRestriction, MaxCardinalityRestriction, CardinalityRestriction, AllValuesFromRestriction, SomeValuesFromRestriction and HasValueRestriction. These restrictions define the values (Restriction.Value) that the Class must take for a Property (Restriction.Property). A Class may also be an enumeration class, which includes the list of individuals that are the members of the class. A Property has a name, domain and range information, and possibly super properties. Our tool lets the user use definitions of Classes, Object Properties, and Datatype Properties as a source for the mappings to OMT.

2.2 Available OMT Constructs at the Target

According to the IEEE 1516 Standard [8], an OMT model consists of the definitions of Object Classes and their Attributes, Interaction Classes and their Parameters, Dimensions, Datatypes, Transportations, Switches, Time, Synchronizations and User Supplied Tags. Our primary concern here is the creation of Object Classes, Attributes, Interaction Classes and Parameters. These constructs may have references to Datatypes, Dimensions and Transformations and if these referenced constructs do not exist in the target Object Model, new Datatype, Dimension and Transportation definitions will be introduced with default properties through the transformation process. Details of these OMT constructs are expected to be edited with an OMT Editor by the user after the transformation. Switch, Time, User Supplied Tags and Synchronization definitions are ignored in this process.

The class hierarchy for Object Classes and Interaction Classes are represented by *subClassOf* properties in our OMT model. While serializing the model into FOM file, these *subClassOf* properties are replaced with the nested class definitions.

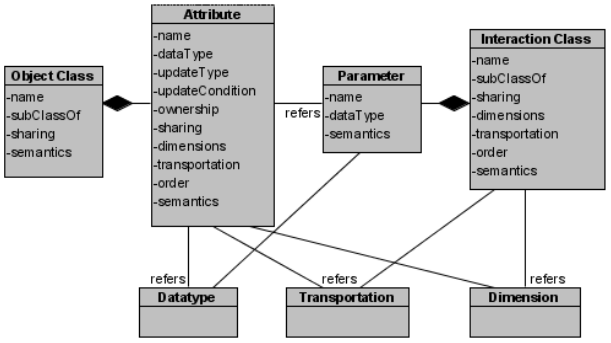

Figure 3. Target OMT Constructs

Figure 3 shows the relationships between the OMT objects and their properties handled during a transformation. Some properties are allowed to take a value from a predefined value set (*sharing* property of an Object Class can be set to "Publish", "Subscribe", "PublishSubscribe" or "Neither"), some can take any String (like the *name* of an Object Class) and others must refer to an existing OMT object in the model (*dataType* property of an Attribute must be the name of an existing Datatype object).

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8678

2.3 Mappings from OWL to OMT

Our tool provides an interface to the user to define mappings between the available OWL constructs at the source and the OMT constructs at the target, which were defined above. Mappings can be classified into 4 types regarding the target constructs. These are

- Mappings to Object Classes
- Mappings to Attributes
- Mappings to Interaction Classes
- Mappings to Parameters

Our tool lets a mapping read the values of the properties of above OMT constructs from the ontology. The user can also fix the values in the transformation configuration. For instance, he can either say "The value for *subClassOf* property of an Object Class will be taken from the name of the super class of corresponding OWL Class" or "Attributes of Object Classes whose name begin with 'X' will have *datatype* 'HLAboolean'".

The definitions of these 4 OMT objects may have references to Datatype, Dimension and Transportation definitions that do not exist in target Object Model by default. These cases are handled by adding the definitions for these new OMT constructs with their default properties. For example, if the transformation results in an Interaction Class with its *transportation* 'X', a Transportation object with *name* 'X' is added to the resulting Object Model. Details of this Transportation instance are supposed to be configured later manually by the user.

During the transformation process, mappings to Object Classes and Interaction Classes are resolved first. As will be explained in the following sections, new Attributes and Parameters are introduced to the target model during the resolution of mappings to Object Classes and Interaction Classes, respectively. Mappings to Attributes and Parameters are used to set their further properties like *datatype* or *sharing*.

2.3.1 Mappings to OMT Object Classes

This mapping type enables the user to define new Object Classes in the target model. The *name* of the Object Class is taken from the name of the OWL Class/Property in the source. While creating new Object Classes, user can also configure mappings for the properties of these Object Classes. In other words, he can configure how to set the *subClassOf*, *sharing* and *semantics* of the related Object Classes. Each Object Class can have at most one super class, and the name of this superclass is represented in a *subClassOf* property in the target model. The name of the super class can be taken from the source ontology constructs depending on the mapping configured by the user as shown in Figure 4.

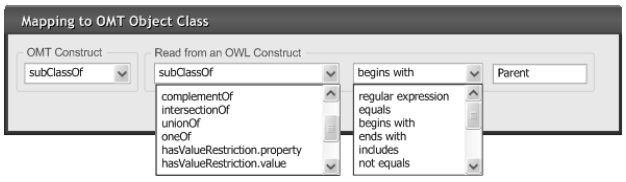

Figure 4. Object Class Mapping for subclass relationship

As an example, the configuration in Figure 4 will be evaluated as follows: OWL Classes/Properties in the input ontology are traversed one by one according to the constraint definitions for the mapping group which includes this mapping. For each valid OWL Class/Property in the source, an Object Class with same *name* is created in the target Object Model. Moreover, if an OWL Class/Property named "C" is defined to be the subclass/subproperty of OWL Class/Property named "ParentC", then Object Class "C" will be the subclass of Object Class "ParentC" in the resulting Object Model.

The other property of an Object Class is *sharing*. *Sharing* can either be taken from an OWL construct in the source or set to one of the possible values in a choice list. Figure 5 shows the configuration panel to define how to set the *sharing* property of related Object Classes.

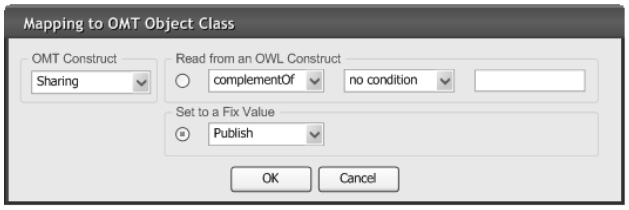

Figure 5. Object Class Mapping for Sharing

Similar to the *sharing* property, *semantics* can either be taken from an OWL construct in the source or set to some fix value. Figure 6 shows an example, which also illustrates the constraint definitions. The mapping configuration in Figure 6 is processed as follows: if the OWL Class in the source has an OWL HasValueRestriction definition on an OWL Property named "description", the value of this specific Restriction will be set as the *semantics* of the corresponding Object Class.

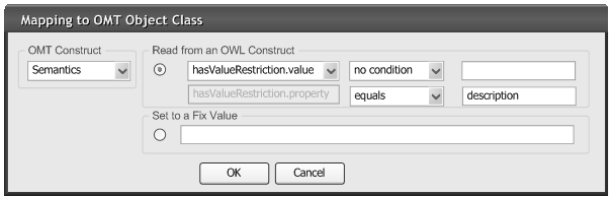

Figure 6. Object Class Mapping for Semantics

Object Classes can have Attributes. In the mapping configuration for Object Class, user can define where to get the attribute names for the corresponding Object Classes. If the user configures the mapping for the attribute names, new Attribute objects with desired names are generated in the target model. In this same mapping, user can also set the properties of Attribute objects, i.e. *dataType*, *updateType*, *updateCondition*, *ownership*, *sharing*, *dimensions*, *transportation*, *order* and *semantics* as shown in Figure 7. However our tool does not allow getting the values of Attribute properties from the source OWL constructs in this mapping panel, instead their values are set to some desired fix values. If the user wants to get Attribute property values from the OWL ontology, he has to define a new "Mapping to Attributes" as explained in the following section. The required attribute mapping flexibility is provided in that mapping type.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8678

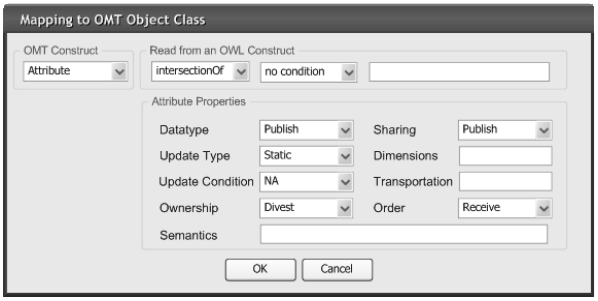

Figure 7. Object Class Mapping for Attributes

2.3.2 Mappings to OMT Attributes

There may be cases where some Class or Property in the ontology defines an Attribute with its properties. This mapping type is used to set the properties of Attributes which were created during the configuration of "Mappings to Object Classes". The transformation for this type of mapping works as follows: OWL Classes/Properties in the input ontology are traversed one by one according to the constraint definitions for the mapping group. For each selected OWL Class/Property named "C", an Attribute with name "C" is searched in the target model. For each Attribute named "C", the OWL construct defined in the mapping is used to feed the property values for this Attribute. Figure 8 shows a case where the *datatype* of defined Attributes are taken from the property name of the MaxCardinalityRestrictions defined for the corresponding OWL Class.

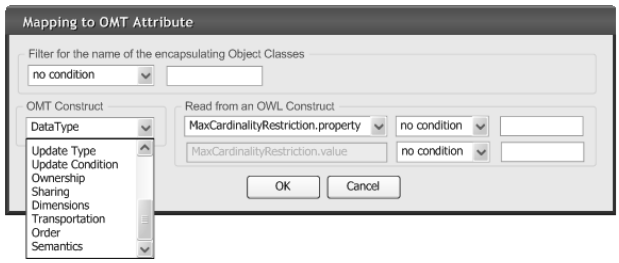

Figure 8. Attribute Mapping for possible Attribute properties

The precondition for this mapping is that the correct Attributes have already been defined for desired Object Classes. This is achieved by the Attribute configurations in Object Class mappings. Thus, "Mapping to Attributes" is just a matter of feeding the property values to the previously defined Attributes of Object Classes. One additional feature in this mapping is that the name of the owner Object Class can be bound to a constraint in the mapping.

2.3.3 Mappings to OMT Interaction Classes

The mappings for Interaction Classes and Parameters are similar to the mappings for Object Classes and Attributes. With this mapping type, new Interaction Classes are added to the target model. Properties of the Interaction Classes, namely *subClassOf, sharing, dimensions, transportation, order, and semantics* are also configured by either using the source OWL constructs or selecting values from choice lists. Moreover, if the Interaction Class needs to have Parameters, user can configure this mapping to create Parameters with desired names for the corresponding Interaction Classes. In this mapping window, user can also choose a *dataType* for the Parameters from a choice list. If the *datatype* for the Parameters are to be taken from some OWL constructs, the user has to configure a "Mapping to Parameters".

2.3.4 Mappings to OMT Parameters

Just like for the "Mappings for Attributes", this mapping requires that the Parameters are already defined in the target model through the performance of "Mappings for Interaction Classes". This mapping type enables the user to set the OWL constructs to feed the *datatype* and *semantics* for desired Parameters. During the execution of this mapping, each OWL class/property in the source ontology is traversed and if a Parameter with the same OWL class/property name is found, its *datatype and semantics* are set with the OWL construct according to the mapping configuration. Moreover user can define a constraint on the name of the encapsulating Interaction Class to apply this mapping.

2.4 Validation of the Model

The last step of the transformation is Object Model validation. The reason for this step is that the resulting model may not always be consistent. Especially if the model constructs are to be taken dynamically from the source ontology, the values set to these constructs may not be in the allowed range or referred objects may not exist in the target object model. The validation checks our tool currently applies include the following:

- **Enumerations:** Some OMT constructs (namely *sharing, updateType, ownership* and *order*) may get only some restricted specific values. For example, if the transformation sets the *sharing* property of an Object Class to a value other than "Publish", "Subscribe", "PublishSubscribe" or "Neither", this would not be a valid FOM.
- **Class hierarchy:** An Object Class or an Interaction Class cannot have multiple super classes.
- **Uniqueness:** There cannot be two OMT constructs of the same type with the same name.
- **New OMT constructs:** Transformation may result in references to Datatype, Transportation or Dimension objects which do not exist in FOM. These constructs are introduced with default property values.
- **Dependent properties:** The value of a property may depend on the value of another property of an OMT construct. For example, if *dataType* of an Attribute is "NA", its *updateType*, *updateCondition* and *dimensions* must also be "NA" and a *transportation* and *order* must be specified for that Attribute.

3. DISCUSSION

The specific contribution of our work is a tool for ontology based simulation design. We introduce a user-guided transformation process to bridge the gap between the domain modeling and simulation software modeling realms with minimum loss of information and maximum simplicity. By mapping the OWL and HLA Object Model constructs on a user interface in a point-and-click fashion, the knowledge captured in an ontology is automatically transformed into a FOM. Once the transformations are defined, subsequent updates to ontology, in so far as they do

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8678

not disturb the existing input-output patterns, are reflected to the target FOM without further user involvement. This FOM can be then used in an HLA simulation environment without any need for OWL knowledge.

An ongoing case study attempts to generate a FOM from the Trajectory Simulation Ontology (TSONT) [1]. The FOM will be for a federation involving simulation of some munition trajectories. TSONT essentially captures the trajectory simulation domain knowledge, including mathematical models, and specifies the functionality required to carry out a simulation. Currently, the generation of FOM is manual. Our goal is to let the user configure a transformation to automatically generate the same FOM.

Current tool design aims at generation of four main OMT constructs, namely Object Classes, Attributes, Interaction Classes and Parameters. As explained in Section 2.2, new Datatypes, Transportations and Dimensions are created with default properties if necessary. In future releases of our tool, we are planning to introduce new mapping types to let the user set the properties of Datatypes, Transformations and Dimensions using the information in the source ontology.

It is desirable for the tool to be able to read the source data from multiple ontology files. Multiple ontologies may account for multiple domains involved in a complicated federation scenario. These data sources may need to be combined and interpreted to generate a federation object model.

There are some structural differences between an ontology and an HLA Object Model. For example, an OWL class may have multiple superclasses, while an Object Class in FOM may have only one superclass. If the user configures the mappings which somehow result in an Object Class with multiple superclasses, only the last superclass assignment becomes effective.

4. REFERENCES

[1] Durak,U., Oguztuzun,H., and İder,K. 2006 An Ontology for Trajectory Simulation. Proceedings of the 38th Winter Simulation Conference, Monterey, CA, USA

[2] EMF, Eclipse Modeling Framework Project, http://www.eclipse.org/modeling/emf/?project=emf

[3] EODM, EMF Ontology Definition Metamodel, http://www.eclipse.org/modeling/mdt/?project=eodm#eodm

[4] Falbo, R.A., Guizzardi, G., and Duarte, K.C. 2002 An Ontological Approach to Domain Engineering. International Conference on Software Engineering and Knowledge Enginnering, Ischia, Italy.

[5] France, R., and Rumpe, B. 2007 Model-driven Development of Complex Software: A Research Roadmap. Proceedings of the Conference on Future of Software Engineering, (May 23-25, 2007), p.37-54.

[6] Hesse, W. 2005 Ontologies in the Software Engineering Process. EAI 2005 - Proceedings of the Workshop on Enterprise Application Integration, Berlin.

[7] Horridge, M., Knublauch, H. , Rector, A., Stevens, R., and Wroe, C. 2004 A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, The University of Manchester, Stanford University

[8] IEEE Std 1516.2-2000, 2001, IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - object model template (OMT) specification. http://ieeexplore.ieee.org/xpl/standardstoc.jsp?isnumber=19791

[9] IODT, Integrated Ontology Development Toolkit, http://www.alphaworks.ibm.com/tech/semanticstk

[10] Kleppe, A., Bast, W. and Warmer, J. B. 2003. MDA Explained, the Model Driven Architecture: The Model Driven Architecture: Practice and Promise. 2nd Ed. Addison-Wesley, Boston.

[11] MIC, Model Integrated Computing, http://www.isis.vanderbilt.edu/research/MIC

[12] Miller, J.A., and Fishwick, P.A. 2004 Investigating Ontologies for Simulation Modeling. Proceedings of the 37th Annual Simulation Symposium (ANSS'04), Arlington, VA, USA.

[13] MOF, Meta-Object Facility, http://www.omg.org/mda/specs.htm#MOF

[14] Ozdikis, O., Durak, U. and Oguztuzun,H. 2009 User Guided Transformation for Ontology Based Simulation Design. 2009 Summer Computer Simulation Conference, Istanbul, Turkey.

[15] OWL Web Ontology Language Overview, http://www.w3c.org/TR/2004/REC-owl-features-20040210

[16] Pace, D.K . 2000 Ideas About Simulation Conceptual Model Development. John Hopkins APL Technical Digest, 21, 3.

[17] Prieto-Diaz, R. 1990 Domain Analysis: An Introduction. ACM SIGSOFT Software Engineering Notes, ACM Press.

[18] Rathnam, T., and Paredis, C.J.J. 2004 Developing federation object models using Ontologies. Proceedings of the 2004 Winter Simulation Conference, Washington, DC, USA

[19] Tolk, A. 2002 Avoiding another Green Elephent – A Proposal for the Next Generation HLA based on the Model Driven Architecture. 2002 Fall Simulation Interoperability Workshop, Orlando, FL, USA.

[20] Tolk,A., and Turnitsa,C.D. 2007 Conceptual modeling of information exchange requirements based on ontological means. Proceedings of the 39th Winter Simulation Conference, Washington D.C.

[21] Topçu, O., Adak, M. and Oğuztüzün, H. 2008 A metamodel for federation architectures. ACM Transactions on Modeling and Computer Simulation, 18,3, (July 2008), 10:1-10:29. DOI=http://doi.acm.org/10.1145/1371574.1371576

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8678