Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

FROM DOMAIN ONTOLOGIES TO MODELING ONTOLOGIES TO EXECUTABLE SIMULATION MODELS

Gregory A. Silver
Osama Al-Haj Hassan
John A. Miller

University of Georgia
Athens, GA 30602, U.S.A.

ABSTRACT

Ontologies allow researchers, domain experts, and software agents to share a common understanding of the concepts and relationships of a domain. The past few years have seen the publication of ontologies for a large number of domains. The modeling and simulation community is beginning to see potential for using these ontologies in the modeling process. This paper presents a method for using the knowledge encoded in ontologies to facilitate the development of simulation models. It suggests a technique that establishes relationships between domain ontologies and a modeling ontology and then uses the relationships to instantiate a simulation model as ontology instances. Techniques for translating these instances into XML based markup languages and then into executable models for various software packages are also presented.

1 INTRODUCTION

Many application domains are developing detailed, agreed-upon ontologies to define and document concepts in their domain. For example, the Problem-oriented Medical Records ontology (PMRO) (in the healthcare domain) defines patients, clinical personnel, symptoms, diseases, and treatments (WC3 Health Care and Life Science Group 2006). The Glycomics ontology (GlycO) (in the biology domain) defines glycan (carbohydrate) structures and the items in the biosynthetic pathways that create them (Thomas, Sheth, and York 2006). The premise behind our research is that this knowledge and data can be used productively to bootstrap the creation of simulation models. In this paper, we illustrate how to drive the development of simulation models using ontologies in detail for the healthcare domain and briefly discuss it in the simulation of biosynthetic pathways. The basic strategy is to use information in the application or domain ontologies and map it to a modeling ontology. Through the use of computer aided techniques this information is expanded, refined, and translated step-by-step into effectively executable simulation models.

An overview of ontology driven simulation and its benefits is presented in Section 2 of the paper. In Section 3, we apply our approach to ontology driven simulation to an application scenario to develop a Discrete Event Simulation (DES) model. Section 4 discusses ways in which the Discrete Event Modeling Ontology (DeMO) (Miller, et al. 2004) is used to supply a foundation for ontology driven simulation. In Section 5, we describe the architecture of the Ontology Driven Simulation design tool (ODS) and Section 6 reviews related work. Finally, conclusions and recommendations for future work are discussed in Section 7.

2 ONTOLOGY DRIVEN SIMULATION

2.1 Exploiting Domain Knowledge

Simulation models have been used to simulate processes in many different domains. The simulation modeler must rely on the information provided to him/her by domain experts during the development of the model. Domain experts are a necessary part of the process because they understand the concepts and relationships among the concepts within the domain. In the past few years another source of domain knowledge, known as the domain ontology, has also become available. While domain ontologies cannot replace domain experts, they may be very useful to simulation modelers.

Ontologies provide formal methods for describing the concepts, categories, and relationships within a domain. In addition, ontologies written in languages such as the Web Ontology Language (OWL) (McGuinness and Harmelen 2004) can be processed by machines. Domain ontologies may be particularly helpful to simulation modelers since they can be used to communicate domain information to simulation and modeling tools with limited human intervention. ODS takes advantage of this feature by using software tools to align knowledge resident in domain ontologies with knowledge resident in a modeling

ontology in order to facilitate the creation of simulation models.

In ODS, a tool is used to map concepts from domain ontologies to concepts in a modeling ontology and then create instances of modeling ontology classes to represent a model. Once the ontology instances representing the model have been created additional tools are used to translate the instances into an executable simulation model.

2.2 Benefits of Ontology Driven Simulation

Ontology driven simulation uses ontologies to drive the creation of simulation models and in doing so makes use of an agreed upon set of terms and relationships that are shared by domain experts, modelers, and model development tools. These terms and relationships provide a semantic grounding and structure for the executable model. The domain ontology exists for a specific application area, and its classes and instances determine the types of components that will make up the model. The modeling ontology is developed independently of any specific domain ontology but may rely on general upper ontologies. It provides the concepts and relationships common to the DES modeling world views under which simulation models are built. We might say that the domain ontology provides a universe of discourse for the application domain, while the modeling ontology provides a universe of discourse for discrete event modeling.

A domain ontology, such as a medical records ontology, can provide domain information about patients, diseases, health care workers, and procedures performed on behalf of patients. A DES modeling ontology can provide information about model components, activities, and relationships for the various simulation world views. Information from both ontologies is used in the development of models. Modelers receive the benefit of having a stable set of terms for the domain available through the design tools that they are using. Domain experts and others who make use of the simulation models benefit by having simulations use a common set of terms with which they are familiar.

The relationship between the domain ontologies and the modeling ontology is established during the mapping process, and mapping instances are stored separately from the ontologies. Since the domain ontology is created independently from the modeling ontology, multiple domain ontologies can be mapped to the modeling ontology. Both ontologies can also be reused or modified without directly impacting each other or exsisting simulation models. If models need to be rebuilt based on changes to either ontology, they can be generated as instances of the modeling ontology.

The development of a simulation model which accurately represents the processes being modeled is a challenging task. Information about the environment and processes is typically collected by discussions with domain experts, observing the processes, reading documents and learning terminology. An ODS tool suite can help model developers in these early phases of model design as well as facilitate greater automation in the later phases of model development.

Given a particular problem domain, an ODS tool suite could be used in the following fashion. It would allow the modeler to select ontologies relevant to the domain and then select a subset of concepts that can be applied to the problem. Typically, the modeler would like to have simultaneous views of two ontologies, one for the application domain and one for the modeling paradigm. The modeler would first browse to find the relevant portions of the domain ontology, and then browse the modeling ontology looking for a suitable simulation world view. Choice of world view is influenced by the application domain (specifics of the domain ontology) as well as the modelers experience with different world views. The DeMO ontology provides the following four world views: state oriented, event oriented, activity oriented, and process oriented.

After choosing a world view, the modeler may go on to select a particular modeling technique such as Stochastic Petri Nets. Once these selections are made, the more difficult task of matching and mapping come into play (Kalfoglau and Schorlemmer 2003). A correspondence between classes and properties in each ontology needs to be established. Typically, the classes will not be equivalent; therefore, mappings will need to be provided.

Once a correspondence between the domain and modeling ontologies has been created, detailed knowledge needs to be transferred from the application domain to the model development process. This boils down to taking instances from the domain ontology and transforming them into instances of the modeling ontology. The key question regarding the merit of this approach is how much knowledge can be transferred from the domain ontologies to the modeling ontology. Briefly, the following types of information may be available in the domain ontology: entities, resources, activities, etc. We address this question in more detail in Section 5.1.

Ontology driven simulation makes it possible to represent simulation models for various simulation world views and even models which may be targeted for a wide variety of different simulation engines in a common format which can be browsed by various tools. Since models can be represented as a collection of modeling ontology instances, it will be possible to create a repository of models which can be browsed using currently available Semantic Web tools. Furthermore, since the instances which make up the models have names that are grounded in particular domain ontologies, tools can be used to browse for models within a particular domain. These ca-

pabilities make it possible to create Web based model repositories which can be organized by application domain or by simulation world view. Terms from domain ontologies can be used to search for models within a particular domain, while terms from the modeling ontology can be used to search for models conforming to a particular simulation world view.

These model repositories could also be populated using reverse engineering. Legacy simulation models developed using proprietary packages may be reverse engineered into modeling ontology instances (Lacy 2006). This would allow models developed using various tools to be represented in a common format, become available to a larger audience, support translation from one package to another, and be converted to another simulation world view. In general, support for interoperability will be a key advantage of using techniques like ODS.

3 APPLICATION SCENARIO

This Section discusses how ODS takes place by using an example from the domain of clinical medicine. The domain ontology used for the example is the Problem-oriented Medical Records Ontology (PMRO). The example is a simulation for a hospital emergency department. The table below shows some of the concepts that were needed to simulate the emergency department using process interaction. The PMRO classes that were used to represent emergency department concepts and the Discrete Event Modeling Ontology (DeMO) classes to which the PMRO classes were mapped are also shown.

Table 1: Classes Used in Simulation Example

Concept	PMRO Class	DeMO Class
Patient	Patient	Entity
Triage	Screening	Activity
Examination	Clinical-Exam	Activity
Physician	Clinician	Resource

Each of the PMRO classes in the simulation were mapped to DeMO classes so that DeMO ontology instances could be used to represent the simulation model. The ontology instances were then translated into an XML based markup language which was used to generate an executable simulation model. The details of this process are covered in Sections 4 and 5.

4 THE DEMO APPROACH TO ODS

4.1 Review of DeMO

The Discrete Event Modeling Ontology (DeMO) is a general ontology for discrete event simulation (DES) and includes state oriented, activity oriented, event oriented, and process oriented models. Development of DeMO began in 2003 (Miller, Baramidze, Fishwick, and Sheth 2004; Fishwick and Miller 2004; Miller and Baramidze 2005; Silver, Lacy, Miller 2006) to explore issues and challenges in developing ontologies for Modeling and Simulation (M&S). It is focused on discrete events models in which state changes discretely over time due to the occurrence of events. The OWL language is used to define over 80 classes. Figures 1, 2, and 3 illustrate visualizations created by OntoViz (Sintek 2003) showing the DeMO classes and their relationships. The ontology is divided into four main parts: ModelConcept, DeModel, ModelComponent and ModelMechanism. The principal component, DeModel, is itself divided into four top-level classes each representing a modeling paradigm. The first is based upon state based modeling, a modeling paradigm used throughout computer science. The other three are based on the most popular simulation world views. These classes, StateOrientedModel, ActivityOrientedModel, EventOrientedModel, and ProcessOrientedModel form the foundation for all of the modeling techniques supported in DeMO. In its current form, DeMO does not address much of the modeling and simulation domain including continuous models, statistical modeling, output analysis, random variates.

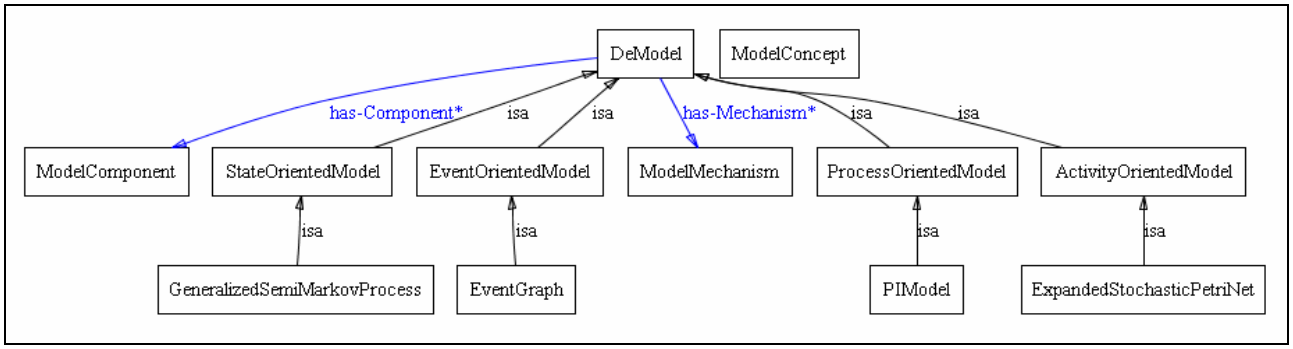

Figure 1: DeMO Top Level Classes

4.2 Using DeMO for ODS

Two of the top level classes of the DeMO DeModel are currently used for ODS. The portion of the model represented by the ActivityOrientedModel class, as seen in Figure 2, is used to create Petri Net models. The instances used to represent these models can be transformed by the ODS tool suite into the Petri Net Markup Language (PNML) (Jungel, Kindler, and Weber 2000) for execution on Petri Net simulators like the one found in the Petri Net Kernel (PNK) (Kindler and Weber 2001).

The PIModel subclass of the ProcessOrientedModel top level class, as seen in Figure 3, is used by the ODS tool suite to develop Process Interaction Models. The models may be translated to Extensible Process Interaction Markup Language (XPIML) (Silver, et al. 2006) and then into executable models for simulation software packages. The details of creating executable models will be discussed in Section 5.

5 ARCHITECTURE AND IMPLEMENTATION

In order to support ODS, we have developed an architecture that gives a smooth way to refine knowledge from domain ontologies to the point of creating executable simulation models. Because of the complexity, our approach consists of multiple steps as shown in Figure 4. The figure uses the following conventions: rectangles represent documents, ovals represent process steps or programs, and rounded rectangles represent the ultimate goal of simulating the generated models. The blue text (left side of the diagram) in the model represents an emergency room simulation using a medical records domain ontology and the text in green (right side of the diagram) represents a glycan biosynthetic pathway simulation using the GlycO Ontology. These steps are explained in greater detail in the sub-sections below.

5.1 ODS Design Tool Suite

One issue addressed by the ODS design suite is the task of assembling the knowledge necessary to create a simulation model. This knowledge can come from a variety of sources, one of which is the domain ontology. The ontology mapping component of the suite provides a modeler with the capability to browse a domain ontology and the DeMO ontology simultaneously and create a correspondence between selected concepts in the ontologies. For example, in Figure 5 the Patient class of the PMRO ontology is mapped to the Entity class of the DeMO ontology.

Other sources of information include databases, logs, spreadsheets, and other simulation models. These same sources can also provide information about resources used in, and cost associated with clinical procedures. The data generated by prior simulation studies is another source of information. Medical personnel and information systems use terms to categorize the data stored in the sources mentioned above. If these terms originated from the domain ontology used for the simulation study, then communication among those involved in the study and the collection of data for the study will be easier since a single set of concepts is being used by all of the information sources.

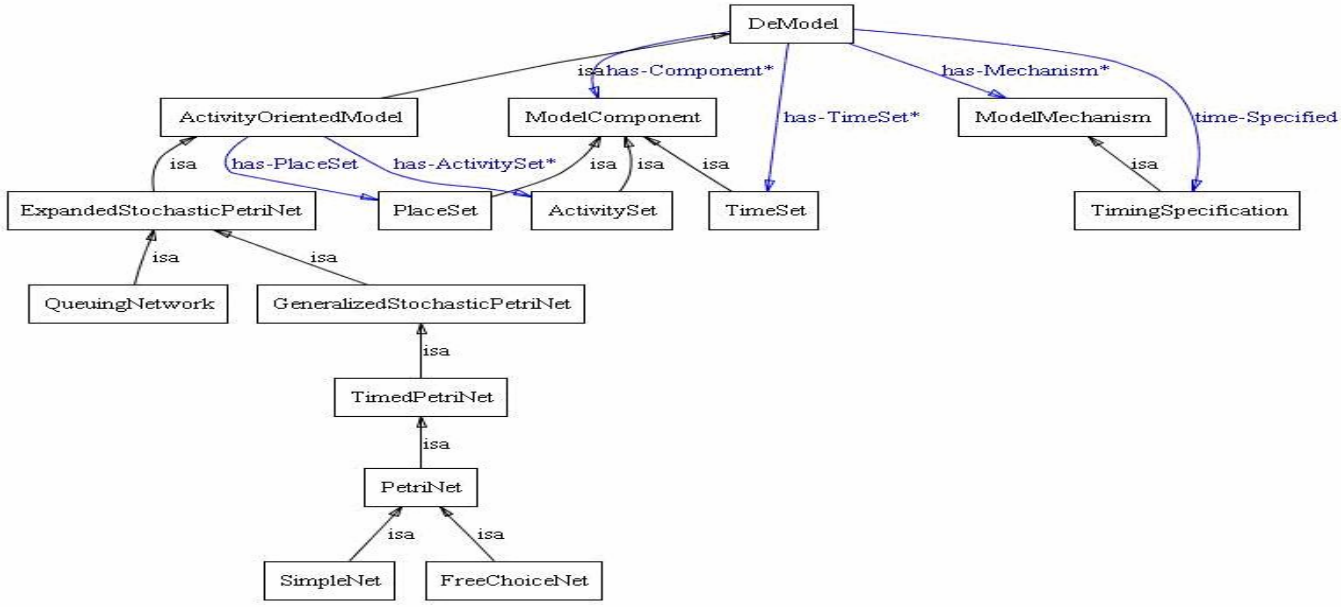

Figure 2: DeMO ActivityOrientedModel

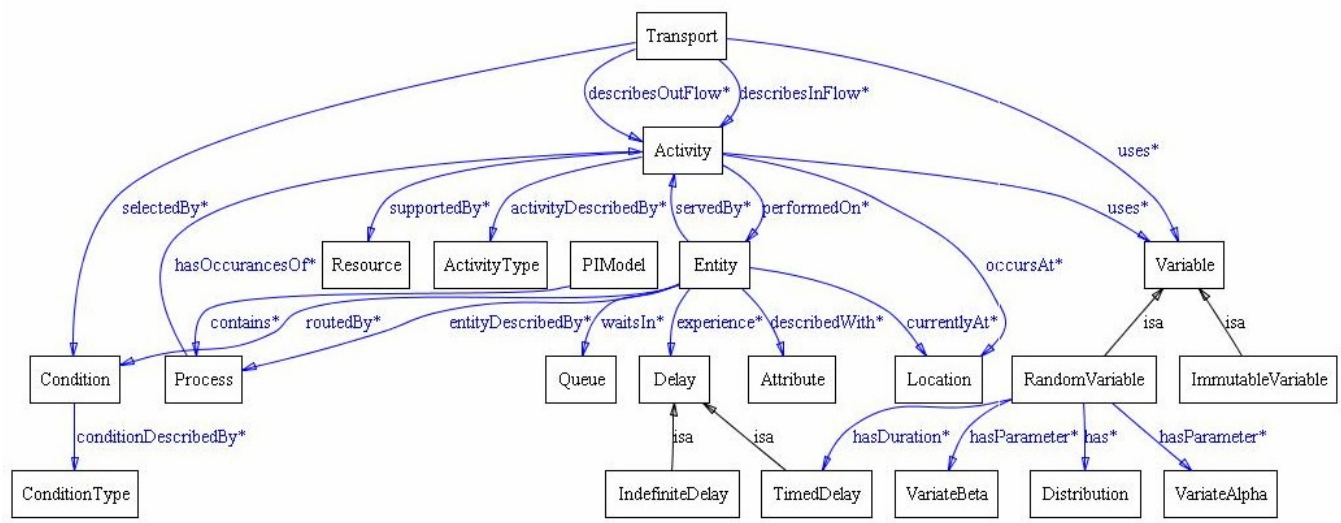

Figure 3: DeMO PIModel

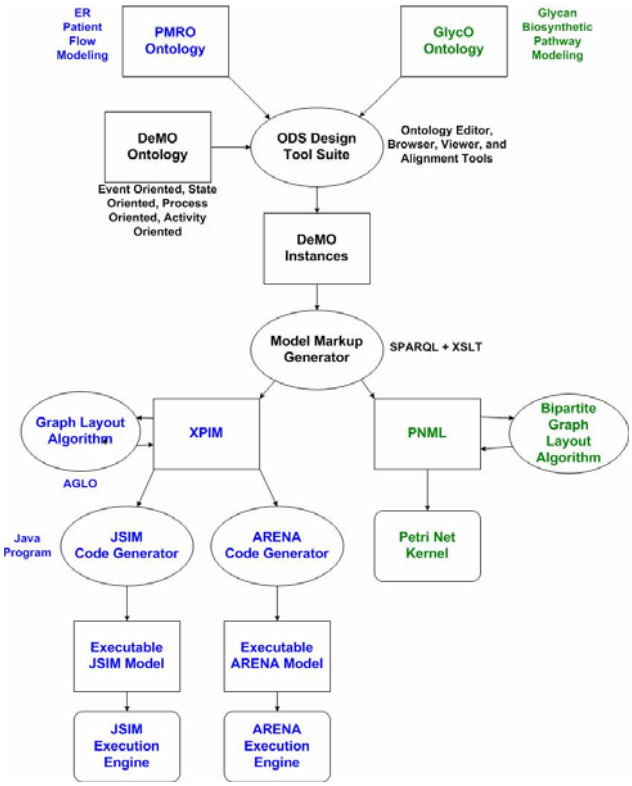

Figure 4: ODS Architecture

If the domain ontology is populated with instances, it may also serve as an additional source of information for the model. The classes and properties in an ontology define the terms and relationships of a domain, and in doing so they act as a schema, but the usefulness of the ontology is increased if it is populated with instances. In the medical records ontology mentioned earlier, a clinician might be described as a person who plays a clinical role. In this case, the Clinician class is a subclass of the Person class. If the ontology is populated, ODS will be able to gather information from the domain ontology about the existence of various clinician resources available for the processes being modeled.

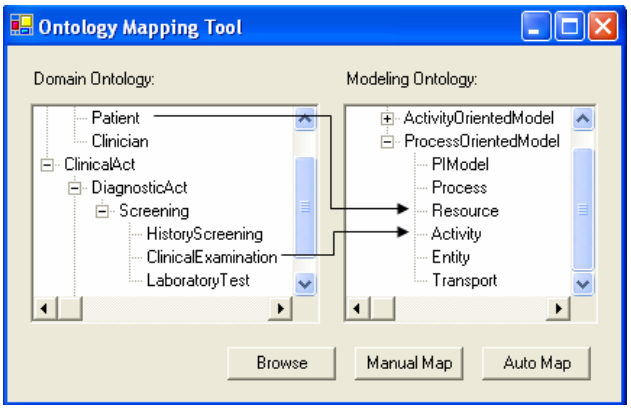

Figure 5: ODS Mapping Tool

A simulation model should contain enough information to accurately represent the processes being simulated. This may include the activities involved in the processes, resources used in the processes, the entities involved in the processes, the routes taken by the entities. It may also include statistical data such as mean arrival times, service times, and their standard deviations. Some of this information can be gathered from the domain ontology schema, while other portions may be retrieved from the instances of the populated ontology, external data sources, domain experts, or the person developing the model.

When simulation models are developed using popular software packages, the reusability of the model is very limited for several reasons: (1) There is no formal way of specifying an agreed upon domain of discourse for the application domain of the process being modeled. If modelers want to select components from one model to be used in the development of another model, they must be able to determine the intended function of the selected components. Unless both models reference a commonly agreed upon set of definitions, this is a difficult task. For example, one model for a hospital emergency department may refer to a particular resource as a physician, and another may refer to a component that performs the same function as a doctor; (2) The modeling domain suffers from a similar problem. The same term may mean different things in different simulation software packages. For example, a component that is an activity in one software package may be called a process another package. ODS overcomes these problems by allowing commonly agreed upon ontologies for both the application domain and the modeling world view to be used as the basis upon which model components are constructed; (3) There is no commonly agreed upon format for representing and storing models. For example, models developed using ARENA are stored in one format and models developed in JSIM are stored in a different format. ODS overcomes this problem by representing models as DeMO ontology instances which can be translated into executable simulation models for various software packages.

As stated earlier, ODS creates a correspondence between classes in the domain ontology and the modeling ontology. This correspondence is not ontology mapping in the typical sense, where like classes in two ontologies are mapped to indicate that a class in one ontology is the same as a class in another ontology. ODS creates a correspondence between classes to indicate that a class in the domain ontology can serve as a instance of an object represented by a class in the modeling ontology. For example, the clinical-examination class in the POMR ontology may be mapped to the activity class in DeMO to indicate that a clinical-examination may serve as an activity in a model. Determining the correspondence between classes can be a difficult task since, for example, a nurse object might represent the activity performed by a nurse in one simulation model; while in another simulation model a nurse object might be a resource associated with a clinical examination activity. Many of the decisions associated with the creation of correspondences between classes are dependent upon the perspective of the modeler rather than the design of the ODS tool suite.

Given the fact that the creation of correspondences is dependent upon the modeler's perspective, the ODS tool suite cannot ensure their validity. Instead of attempting to do so the tool assists the modeler by finding inconsistencies within the correspondences. If, for example, nurse and physician are both subclasses of clinician in the POMR ontology but nurse is mapped to the DeMO entity class and physician is mapped to the DeMO resource class, the ODS design tool suite will indicate that an inconsistency exists within the correspondences since a clinician is considered to be a model entity in one place and a model resource in another.

5.2 Model Markup Generator

Simulation models represented as modeling ontology instances can be transformed directly into executable models (Lacy 2006) or indirectly into an XML markup language such as XPIML or PNML. Process Interaction models in XPIML are described as a collection of processes which are themselves described as a collection of activities.

The model markup generator of the ODS design tool suite currently supports the creation of activity oriented models in the form of Petri Nets and process oriented models for process interaction simulation. The base class used to represent a Petri Net model is the DeMO ExtendedStochasticPetriNet class, and the base class for process interaction models is the DeMO PIModel class. The model markup generator searches the populated DeMO ontology and displays instances of these classes on the markup language generator (see Figure 6). When an ExtendedStochasticPetri instance is selected for generation, PNML will be generated and when a PIModel instance is selected, XPIML will be generated.

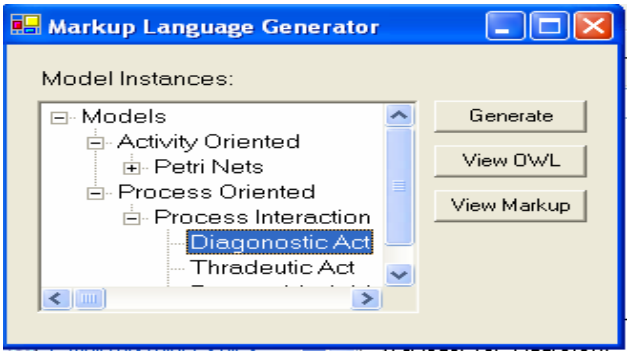

Figure 6: ODS Markup Language Generator

DeMO PIModels are made up of process instances. Each process is stored as a directed graph with nodes represented as activity instances and edges represented as transport instances. Figure 7 shows an activity instance in OWL and Figure 8 shows the XPIML that was generated for this instance.

Since XPIML may be viewed as having a grounding in the DeMO PIModel, PIModel instances can be transformed into XPIML instances using Jena (McBride 2002), SPARQL (Prud'hommeaux and Seaborne 2007), and Java

IO. ExtendedStochasticPetriNet instances are transformed into PNML using a similar process. When translating PIModel instances to XPIML, the ODS design tool suite uses Java to execute a series of SPARQL queries against OWL PIModel instances. It then parses the results in order to write the XPIML output.

```
<Activity rdf:ID="ClinicalExamination">
 <uses>
  <RandomVariable rdf:ID="ExaminationDist">
   <hasBetaParameter rdf:resource="200.0"/>
   <has rdf:resource="#Uniform"/>
   <hasAlphaParameter rdf:resource="300.0"/>
  </RandomVariable>
 </uses>
 <uses>
  <RandomVariable rdf:ID="ExaminationCostDist">
   <hasBetaParameter rdf:resource="300.0"/>
   <has rdf:resource="#Uniform"/>
   <hasAlphaParameter rdf:resource="100.0"/>
  </RandomVariable>
 </uses>
 <activityDescribedBy rdf:resource="#Facility"/>
 <numberResources rdf:resource="1"/>
</Activity>
```

Figure 7: OWL Process Interaction Activity Instance

Our decision to use a combination of Java and SPARQL to implement the translation of models into XPIML arose from the need to accommodate the hierarchical structure XPIML. Each XPIML model element may have several nested process elements, each process element may have several nested activity elements, and each activity element may have several transports and other parameters. Java programs are used to execute a SPARQL query for each level of nesting (model, process, activity, etc.). The query result sets are kept in Java data structures and accessed as needed. Using a procedural language such as Java allowed us to create independent query results for each level of nesting and move back and forth between levels as necessary during the parsing of the results sets.

It is possible to have SPARQL return query results as XML. When this is done, languages designed for use with XML, such as XSLT (Clark 1999) may be used to transform the query results. Future plans include modifying the model markup generator to transform the SPARQL query results into XPIM using XSLT instead of Java.

```
<activity activityid="ClinicalExamination" activitytype="Facility"
      caption="Examination" ">
      <location x="331" y="314" />
      <costdist distributiontype="Uniform" alpha="100.0"
            beta="300.0" stream="0" />
      <servicedist distributiontype="Uniform" alpha="300.0"
            beta="200.0" stream="0" />
</activity>
```

Figure 8: XPIML Activity Instance

5.3 Simulation Code Generator

In ODS, models represented as ontology instances can be translated into an intermediate markup language or directly into executable models for target simulation packages. At this time, the ODS design tool does not support translation of ontology instance directly into executable models. Instances must first be translated into a markup language and then into executable models. The simulation code generator of the ODS design tool performs the translation of markup language based models into executable models. DeMO PIModel instances that have been translated into XPIML can be translated by the tool into executable models for either JSIM (Nair, Miller, and Zhang 1996) or ARENA.

The simulation code generator of the ODS design tool suite takes the XPIML representation of the model as input and produces and executable simulation model. When an executable JSIM model is being generated, the code generator reads and parses the XPIML file then outputs Java classes which represent the simulation model. These classes can be compiled and executed using the JSIM simulation environment.

6 RELATED WORK

In the early days of DES, individual programs were written for each simulation application. These programs were written using general purpose programming languages which had no specific support for simulation. This began to change in the early 1960s with the introduction of the first Simulation Programming Languages (SPLs). The General Simulation Program (GSP) was developed by Tochar in 1958 (Tochar and Owen 1960) and was followed by several other SPLs, such as CSL, GPSS, GASP, and SIMULA (Buxton and Laski 1963; Kiviat 1963; Wexelblatt 1980; Nygaard and Dahl 1981), within the next few years.

Early SPLs required model developers to deal with programming language syntax and other implementation details not directly related to model behavior. Recognition of this situation was one of the primary factors that lead to a shift in focus from simulation program development to simulation model development in the 1970s (Nance 1983; Page 1994). Since this shift, the modeling and simulation community has seen the introduction of several model development environments that allow models to be represented at higher levels using specific conceptual frameworks (or world-views). Many factors, such as advances in computer hardware, software, and networks have influenced how model development environments have evolved over the past several years (Nance and Sargent 2002).

One factor that is currently influencing the evolution of model development environments is the Semantic Web

(Fishwick and Miller 2004; Miller and Baramidze 2005). The Semantic Web improves discovery, integration, and reuse of Web resources by providing meaningful and machine-processable descriptions of Web resources. These descriptions of Web resources are typically made available via ontologies encoded in Semantic Web languages such as OWL. Work related of ontologies in modeling and simulation is currently underway by several groups. Lacy (2006) developed the Process Interaction Modeling Ontology for Discrete Event Simulations (PIMODES) to focus specifically on the process interaction world view and support the interchange of simulation models between commercial simulation packages. Benjamin et al. (2005) present an ontology driven framework for process oriented applications, including process interaction simulations. The framework is an attempt to overcome the problem of semantic inaccessibility where the semantic intentions of the application developers and the semantic rules of the application are not available to other entities within an organization. Turnitsa and Tolk (2005) describe what is required for an ontology to be sufficiently complete to serve as a reference for simulation interoperability and they also propose a method for evaluating an ontology to determine it completeness.

7 CONCLUSIONS AND FUTURE WORK

We presented an ontology driven approach to the development of DES models. The ODS design tool suite was created in order to support this approach. The tool provides facilities for: (1) mapping concepts from domain ontologies to a modeling ontology in order to represent models as ontology instances; (2) translating ontology instances to an intermediate XML markup language; and (3) generating executable simulation models from markup language representations of models. The benefits of this approach are: (1) model developers and users have an agreed upon set of terms and concepts available throughout the development and use of a simulation model. This facilitates good communication between all of the entities involved in a simulation study and reduces the incidence of ambiguities; (2) DES models targeted for various simulation software packages or developed using various simulation world views can be represented in a common language; (3) Since models can be represented in a Semantic Web enabled language, the possibility exists for Web based DES model repositories, and (4) since much of the knowledge needed to develop a simulation model exists in the domain and the modeling ontologies, ODS is able to speed up model development by allowing the developer to assemble ontology components to create a model.

Possibilities for future work include the following:

- creating an complete online demonstration of the Ontology Driven Simulation design tool suite (www2.ac.edu/faculty/gsilver/ODS.html);
- developing Web based repositories for DES models represented as DeMO ontology instances;
- refining the process for simulating biosynthetic pathways using the GlycO and DeMO ontologies.

REFERENCES

Buxton, J., N. and J. G. Lanski. 1962. Control and Simulation Language. *Computer Journal.* Volume 5: 194-199.

Clark, J. 1999. *XSL Transformations (XSLT) Version 1.0* [online]. Available via < http://www.w3.org/TR/xslt/ > [Accessed June 26, 2007].

Fishwick, P. A. and J. A. Miller. 2004. Ontologies for Modeling and Simulation: Issues and Approaches, *Proceedings of the 2004 Winter Simulation Conference (WSC'04),* 259-264 . Washington, DC.

Jungel, M. E., E. Kindler, and M. Weber. 2000. The Petri Net Markup Language, *Workshop Algorithmen und Werkzeuge für Petrinetze*. Koblenz, Germany.

Kalfoglau,, Y. and M. Schorlemmer. 2003. Ontology Mapping: The State of the Art. *The Knowledge Engineering Review*. Volume 18 Issue 1: 1-31. Cambridge: Cambridge University Press.

Kiviat, P. J. 1963. GASP – a General Activity Simulation Program. Applied Research laboratory, United States Steel Corporation, Monroeville, Pennsylvania.

Kindler, E. and M. Weber. 2001. The Petri Net Kernel - An Infrastructure for Building Petri Net Tools. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 3 No. 4: 486-497. Berlin: Springer.

Lacy, L. W. and W. J. Gerber. 2004. Potential Modeling and Simulation Applications of the Web Ontology Language – OWL. In *Proceedings of the 2004 Winter Simulation Conference*. Washington D. C.

Lacy, L. W. 2006. *Interchanging Discrete-Event Simulation Process-Interaction Models using the Web Ontology Language – OWL*. Unpublished PhD Dissertation, Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, Florida.

McBride, B. 2002. Jena: A Semantic Web Toolkit. *Internet Computing*, Volume 6 No. 6: 55-59. Los Alamitos, CA: IEEE Computer Society.

McGuinness, D. L. and F. Harmelen. 2004. *OWL Web Ontology Language Overview* [online]. Available via < http://www.w3.org/TR/owl-features/ > [Accessed June 5, 2007].

Miller, J. A. and G. Baramidze. 2005. Simulation and the Semantic Web. In *Proceedings of the 2005 Winter Simulation Conference*, 2371-2377. Orlando, Florida.

Miller, J. A., G. Baramidze, and P. A. Fishwick. 2004. Investigating Ontologies for Simulation and Modeling. In *Proceedings of the 37th Annual Simulation Symposium*, 55-71. Arlington, Virginia.

Miller, J. A., P. A. Fishwick, G. Baramidze, A. P. Sheth and G. Silver. 2006. Ontologies for Modeling and Simulation: An Extensible Framework. Technical Report No. UGA-CS-LSDIS-TR-06-011, Department of Computer Science, University of Georgia, Athens, Georgia.

Nair, R., J. A. Miller, and Z. Zhang. 1996. A Java-Based Query Driven Simulation Environment. In *Proceedings of the 1996 Winter Simulation Conference*, 786-793. Coronado, California.

Nance, R. E. 1983. A Tutorial View of Simulation Model Development. In *Proceedings of the 1983 Winter Simulation Conference*, 325-331. Arlington, Virginia.

Nance, R. E. 2002. Perspective on the Evolution of Simulation. *Operations Research*. Volume 50, Number 1: 161-172. Hanover, Maryland: INFORMS.

Nygaard, K. and O. J. Dahl. 1981. The Development of the SIMULA Languages. In *History of Programming Languages*, 439-493. New York: Academic Press.

Page, E.. H. 1994. *Simulation Modeling Methodology: Principles and Etiology of Decision Support.* Unpublished PhD Dissertation, Department of Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Perakath, B. C., V. A. Kumar, and R. Fernandes. 2005. An Ontology Driven Framework for Process-oriented Applications, *Proceedings of the 2005 Winter Simulation Conference (WSC'05),* 2355-2363. Orlando, Florida.

Prud'hommeaux, E. and A Seaborne. 2007. *SPARQL Query Language for RDF W3C Candidate Recommendation 14 June 2007* [online]. Available via <http://www.w3.org/TR/rdf-sparql-query> [accessed June 24, 2007].

Silver, G. A., L. W. Lacy, and J. A. Miller. 2006. Ontology Based Representations of Simulation Models Following the Process Interaction World View, *Proceedings of the 2006 Winter Simulation Conference (WSC'06),* 1168-1176. Monterey, California.

Sintek, M. 2003. *Ontoviz Tab: Visualizing Protege Ontologies* [online]. Available via <http://protege.stanford.edu/plugins/ontoviz/ontoviz.html> [accessed July 15, 2007].

Thomas, C., J., A. P Sheth,, and W. S. York. 2006. Modular Ontology Design Using Canonical Building Blocks in the Biochemistry Domain. *In Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*. Fairfax, Virginia: IOS Press.

Turnitsa, C. and A. Tolk. 2005. Evaluation of the C2IEDM as an Interoperability-Enabling Ontology. In *Proceedings of the European Simulation Interoperability Workshop*. Toulouse, France.

WC3 Health Care and Life Science Group (HCLS). 2006. *A Problem-Oriented Medical Records Ontology* [online]. Available via <http://esw.w3.org/topic/HCLS/POMROntology> [accessed June 5, 2007].

Wexelblatt, R. L. 1981. *History of Programming Languages*. New York: Academic Press.

AUTHOR BIOGRAPHIES

GREGORY A. SILVER is a PhD student in the Computer Science Department at the University of Georgia. He is also a Computer Information Systems Instructor at Anderson University. Mr. Silver received his M.S. Degree in Computer Information Systems from Georgia State University in 1996. His research interests include modeling and simulation, Web services, and distributed systems.

OSAMA M. AL-HAJ HASSAN is a PhD student in the Computer Science Department at the University of Georgia. Mr. He received his M. Sc. Degree in Computer Science from New York Institute of Technology in 2004. Mr. Al-Haj Hassan's research interests include distributed systems, databases, and simulation.

JOHN A. MILLER is a Professor of Computer Science at the University of Georgia and has also been the Graduate Coordinator for the department for 9 years. His research interests include database systems, simulation, bioinformatics and Web services. Dr. Miller received the B.S. degree in Applied Mathematics from Northwestern University in 1980 and the M.S. and Ph.D. in Information and Computer Science from the Georgia Institute of Technology in 1982 and 1986, respectively. During his undergraduate education, he worked as a programmer at the Princeton Plasma Physics Laboratory. Dr. Miller is the author of over 125 technical papers in the areas of database, simulation, bioinformatics and Web services. He has been active in the organizational structures of research conferences in all these areas. He has served in positions from Track Coordinator to Publications Chair to General Chair of the following conferences: Annual Simulation Symposium (ANSS), Winter Simulation Conference (WSC), Workshop on Research Issues in Data Engineering (RIDE), NSF Workshop on Workflow and Process Automation in Information Systems, and Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE). He is an Associ-

ate Editor for *ACM Transactions on Modeling and Computer Simulation* and *IEEE Transactions on Systems, Man and Cybernetics* as well as an Editorial Board Member for *Journal of Simulation* and *International Journal of Simulation and Process Modelling*. In addition, he has been a Guest Editor for the *International Journal in Computer Simulation* and *IEEE Potentials*.