ONTOLOGY BASED REPRESENTATIONS OF SIMULATION MODELS FOLLOWING THE PROCESS INTERACTION WORLD VIEW

Gregory A. Silver

Dept. of Computer Science
University of Georgia
Athens, GA 30602-7404, U.S.A.

Lee W. Lacy

College of Engineering and
Computer Science
University of Central Florida
Orlando, FL 32816, U.S.A.

John A. Miller

Dept. of Computer Science
University of Georgia
Athens, GA 30602-7404, U.S.A.

ABSTRACT

The Discrete Event Simulation (DES) process interaction world view describes models that focus on simulated entities that progress through a series of temporally related activities. DES formalisms and vendor approaches for representing DES models serve as a basis for developing an open neutral representation of models that can be encoded into ontologies. This paper reviews world views, formal foundations, and ontologies as background. The process for creating ontologies for the process interaction DES domain is discussed. Next an approach to ontology based simulation model representation is presented and last conclusions and recommendations for future work are provided.

1 INTRODUCTION

Discrete event simulations (DES) are often characterized by their world views. The process interaction world view describes models that focus on simulated entities that progress through a series of temporally related activities.

Formalisms have been developed to describe discrete event simulations. Dozens of popular commercial simulation software packages support DES. DES formalisms and vendor approaches for representing DES models serve as a basis for developing an open neutral representation of models. Ontologies provide a mechanism for formalizing the representation.

Ontology-based representations of simulation models support model interchange which leads to higher quality models that are developed more quickly and at lower cost due to reuse (Lacy and Gerber 2004). An ontology-based representation also enables the use of tools that support the ontology language standard.

In order to provide a basis for ontology development world views, formal foundations, and ontologies as background are reviewed in Section 2. Section 3 discusses creating ontologies for the process interaction DES domain.

An approach to ontology-based simulation model representation is presented in Section 4. Finally, Section 5 provides conclusions and recommendations for future work.

2 BACKGROUND

Discrete event simulations are categorized based on their world view. Various efforts have described DES formalisms. Process interaction (PI) world view formalisms are important because they serve as a basis for developing a PI DES ontology.

2.1 World Views

DES world view development began with simulation programming languages (SPL). According to Lackner (1962) SPLs allowed models to be defined using a set of categories that were identified with a particular view of reality. Lackner believed that modelers used these views when contemplating the system and that a theory of systems and models needed to be developed independently of SPLs. Kiviat's (1969) paper on simulation and programming languages led to a DES taxonomy. Three DES world view definitions (i.e., event scheduling, activity scanning, and process interaction) emerged from the work of Lackner (1962) and Kiviat (1969).

From a static point of view, a model consists of entities and their configurations (e.g., their location and internal state). However, for simulation, much of the work involves modeling dynamics or behavior (i.e., the entities have behaviors). From an abstract modeling perspective there is a simple way to make a distinction between two of the most prominent world views. The essential question is where does "behavior" come from – is it within the entity itself or imposed upon it from outside? One approach to induce behavior is to have the entities react to events. This is easy to code. One needs simply to write event routines to change the location or internal state of affected entities. Although this is straightforward from a programming point of

view, one tends to lose continuity of entity behavior. Therefore, an alternative approach which requires more programming effort, but provides a higher level of abstraction for the modeler, is to make the behavior an intrinsic part of the entity. The former technique corresponds to the *event scheduling world view*, while the latter corresponds to the *process interaction world view.*

The *activity scanning world view* (the third view) (Tocher and Owen 1961) differs from process interaction and event scheduling in that it repeatedly scans at fixed time intervals to check conditions and determine whether an activity is to begin. Since repeatedly scanning to check conditions results in poor runtime performance, the activity scanning approach has been modified into the three-phase-approach. This approach improves runtime performance by combining features of the event scheduling world view with the activity scanning world view (Banks et al. 2001).

A simulation that follows the process interaction world view consists of processes which often can be represented as a single source multi-sink directed graph. Each node within the graph captures a stretch of execution. Control must flow into a node through some incoming edge. It will remain with this node until control flows out. Control may flow out in two ways: (1) A definite delay may be handled by scheduling its next reactivation on the future activation list (FAL) at a particular time in the future, or (2) An indefinite delay may also be used, in which case the process becomes indefinitely suspended until something outside this process happens (e.g., a condition becomes true).

2.2 DES Formalisms

DES formalisms serve as candidate foundations for ontologies. They provide a precise, unambiguous definition, facilitate model transformations/morphisms, and establish a clear categorization.

Research on formal models for process modeling presents multiple approaches for formalizing semantics for DES. One category of such formal modeling is referred to as process algebra, which is an algebra used for describing and reasoning about systems that are defined in terms of states and transitions. Another approach extends classical system theory to simulation modeling. This second approach led to the development of the Discrete Event Systems Specification (DEVS) formalism (Zeigler 1976), which provides a foundation for discrete event simulation and modeling including the process interaction world view. Cota and Sargent (1992) propose a modification to the process interaction world view, and, in doing so, they rely on the DEVS formalism as a language independent foundation for presenting their work. We briefly review the work of Zeigler as well as the work of Cota and Sargent in the following subsections.

2.2.1 Zeigler

In the 1970s, Zeigler carried out seminal work establishing theoretical foundations for simulation. His most well known contribution was DEVS, which provides a formal framework for discrete event systems using mathematical abstraction. It uses the components $(X, S, Y, \delta_{int}, \delta_{ext}, \lambda, \tau)$ to specify a simulation model (Zeigler 1976). Briefly, X is the set of inputs, S is the set of states, Y is the set of outputs, $\delta_{int}, \delta_{ext}$ are the transition functions, λ is the output function and τ is the time advance function (see the appendix for formal definitions of these elements).

Although less well known, in related work, he has developed a detailed formal framework for process interaction simulations. While some of the elements of the process interaction framework are analogous to the elements in DEVS, we believe it might be awkward to cast it entirely within the classical DEVS framework. The following is a reformulation, with a few simplifications, of Zeigler's work on process interaction, which is presented in (Zeigler 1976):

- $\alpha \in P$, which is a set of processes (or active components) that can take actions to change the state of the system.
- $\gamma \in R$, which is a set of resources (or passive components) that are unable to directly take actions.
- $\beta \in D = P \cup R$, which is a set of components.
- $t_\alpha \in \Re_0^+$, which is the set of nonnegative real numbers representing time.
- $l_\alpha \in L_\alpha$, which is a set labels indicating activation points in a process α.
- $v_\beta \in V_\beta$, which is a set of descriptive variables for a component β.
- $s_\beta = (v_\beta, l_\beta) \in S_\beta$, which is a set of states for component β, if β is passive then l_β is null.
- $IG = \left\{ (\beta_i, \beta_j) \mid \beta_i, \beta_j \in D \right\}$ is the *influences graph* for the model, where the nodes are the components of the system and edges *(β_i, β_j)* indicate which components can influence other components. Either β_i or β_j must be a process. The state of an influencing component *(influencer)* may be used in the computations of influenced component's *(influencee)* condition and activation functions. An influencer's functions may also change the state of influencees.

  for $\alpha \in P$, $IR(\alpha) = \{ \beta \mid (\beta, \alpha) \in IG \}$
- *for $\alpha \in P$, $IE(\alpha) = \{ \beta \mid (\alpha, \beta) \in IG \}$*

  The domains and ranges for process α's condition and activation functions are characterized by the

set of influencers *IR(α)* and the set of influencees *IE(α)*. Note influencers and influencees may be either *processes* (active components) or *resources* (passive components).

- $z_\beta = (s_\beta, t_\beta) \in Z_\beta$, which is a set of total states for component β, if β is passive then t_β will be null.

- $C_\alpha^{l_\alpha} : \underset{\beta \in IR(\alpha)}{\times} Z_\beta \to \{0,1\}$

  This is a condition that is evaluated based on the state values of the *IR(α)*. There is a condition $C_\alpha^{l_\alpha}$ associated with each $l_\alpha \in L_\alpha$.

- $f_\alpha^{l_\alpha} : \underset{\beta \in IR(\alpha) \cup IE(\alpha)}{\times} Z_\beta \to \underset{\beta \in IE(\alpha)}{\times} Z_\beta$

  This is an *activation function* that may take actions to change the state values of the *IE(α)*. There is an activation function associated with each $l_\alpha \in L_\alpha$. It is necessary for the domain of the activation function to contain both *IR(α)* and *IE(α)* rather than just *IR(α)*, since the function often needs access to the state values of *IE(α)* in order to make the desired calculations.

- $\kappa_\alpha^{l_\alpha} = (C_\alpha^{l_\alpha}, f_\alpha^{l_\alpha})$

  This is a *computation segment* of a process α. The combination of all computation segments for a process is analogous to the notion of a transition function in classical DEVS (see appendix). Each label $l_\alpha \in L_\alpha$ points to the first statement of a computation segment $\kappa_\alpha^{l_\alpha}$ in α. When process α is activated at label l_α, function $f_\alpha^{l_\alpha}$ will execute to update the states of the *IE(α)* only if condition $C_\alpha^{l_\alpha}$ evaluates to true.

- $\tau : \times Z_\beta \to \Re_0^+$

  This is the time advance function that computes the next event time.

- $\alpha = (z_\alpha, \{\kappa_\alpha^{l_\alpha} | l_\alpha \in L_\alpha\})$

  That is, process α consists of its total state as well as all of its computation segments.

- $\eta = (\alpha, l_\alpha, t_\alpha)$

  This is an *activation notice* of a process α. An activation notice is used to represent a process, its reactivation point as indicated by a label, and its reactivation time.

- $FAL = \{\eta\}$

  This is the *future activations list*. The *FAL* is ordered by reactivation time t_α. Note in Zeigler's work (Zeigler 1976) he uses two lists, a *FAL* and a *current activations list (CAL)*.

Given the definitions in this subsection, an algorithm for implementing process interaction simulation is outlined below.

- *INITIALIZATION*: The simulation clock *t* is set to the initial simulation time, the states of the components $\beta \in D$ are set to their initial values and a set of initial activations notices *{η}* are placed on the *FAL*.

- *PROCESSING*: The *FAL* is processed sequentially by removing the first activation notice $\eta = (\alpha, l_\alpha, t_\alpha)$. Notice η is handled by setting the simulation clock $t = t_\alpha$, then going to the computation segment indicated by label l_α in process α (note in practice this might involve coroutine or thread transfers of control). Next, condition $C_\alpha^{l_\alpha}$ attached to label l_α is checked. If this condition function evaluates to true, activation function $f_\alpha^{l_\alpha}$ is applied. Among other things, $f_\alpha^{l_\alpha}$ may schedule notices such as $(\alpha, l_\alpha, t_\alpha = t + delay)$. Processing continues until the *FAL* becomes empty or the clock exceeds some upper bound.

2.2.2 Cota-Sargent

Zeigler's work on the process interaction world view was later modified by Cota and Sargent (1992). In particular, they define a modified process world view which allows preemption of an activity in one process by another process while supporting encapsulation. Process α's activity can be preempted by having another process β cancel (or remove) α's activation notice on the *FAL*. This normally involves modifying α's reactivation time and, in some cases, reactivation point. Preemption implemented in this way violates the encapsulation of the preempted process. Cota and Sargent suggest modifying the process world view to allow $f_\alpha^{l_\alpha}$ to be applied when $C_\alpha^{l_\alpha}$ is true even if the reactivation notice is not at the front of the *FAL* ($t_\alpha > t$ meaning there is still time left in the state). This allows $C_\alpha^{l_\alpha}$ to specify the conditions for applying $f_\alpha^{l_\alpha}$ when the time left in state is greater than zero as well as when the time left in state is zero. The modification allows the information necessary for preempting an activity to exist within the process, thus supporting encapsulation.

2.2.3 Terminology for the Process Interaction World View

The development of a process-oriented model can follow either the *active server* approach, in which the focus is on

the behavior of the resources in the system, or a *transaction flow* approach, in which the focus is on the behavior of simulation entities, referred to as transactions, as they travel through the system (Cota and Sargent 1992). Most of what is proposed in this paper assumes the transaction flow approach as described by Schriber and Brunner (1999). They describe the transaction flow approach by visualizing a system as consisting of discrete units of traffic, sometimes called transactions, that move from point to point in a system while competing with each other for scarce resources.

Vendor-developed simulation packages and tools provide implicit formalisms of PI DES through their implementation of data models for representing simulation models. These packages often provide a graphical user interface to support populating their internal data structures. Many applications share common terminologies and representations of elements for PI DES (e.g., entity creation with random interarrival times). These practical descriptions of PI DES models have been proven through successive use and evolving versions of software.

2.3 Simulation Ontologies

Ontologies (Gruber 1993) are formal descriptions used to describe and categorize concepts and the relationships among concepts within a particular knowledge domain. They can be processed by machines or read by domain experts. Ontologies may be used to share a common understanding of the structure of information, enable reuse of domain knowledge, make domain assumptions explicit, separate domain knowledge from operational knowledge, and analyze domain knowledge (Noy and McGuinness 2001). Such uses facilitate the need for ontologies within the domain of simulation and modeling.

Taxonomies and ontologies allow words and concepts related to domain semantics to be grouped and related to one another in a logical way. While taxonomies for simulation and modeling have existed on paper for some time ontologies for the field are a recent development. They can be used to provide links between the concepts used by various simulation and modeling researchers in a way that can be processed by machines as well as humans. When used in this way ontologies increase the potential for interoperability, integration and reuse of simulation artifacts (Miller et al. 2004).

Semantic Web (Berners-Lee et al. 2001) research and development has led to the creation of machine-processable languages such as XML and RDF allowing meaningful descriptions of Web resources to be encoded into ontologies. The Web Ontology Language – OWL has recently been developed as a standard for encoding ontologies (McGuinness and Harmelen 2004; Lacy 2005a).

Ontologies have been proffered as a solution for modeling and simulation challenges including domain descrip-

tions and simulation development/composition (Lacy 2001). Ontologies have also been identified as a potential solution for data mapping across simulated representations (Blais and Lacy 2004). The activities performed on entities in a PI DES are related to the dynamic behaviors performed on simulated computer-generated forces (CGF) entities. Although CGF entities are usually more complex than PI DES entities, they both undergo state changes that must be described in the simulation models. Ontologies have been developed for representing CGF behaviors to support simulation reuse (Lacy and Gerber 2004).

In addition, there is ongoing work to integrate ontologies into modeling and simulation frameworks and to use them in the development formal methods for simulation and modeling. Knowledge Based System Inc's Framework for Adaptive Modeling and Simulation (FAMOS) is using ontologies for the purpose of ontology driven simulation and the University of Georgia's Discrete Event Model Ontology (DeMO) exists as a test bed for exploring issues in ontology development and formal methods for simulation and modeling (Miller et al. 2004).

Simulation models are typically developed using SPLs, while formalisms exist to provide a basis for the specification of the models. We take elements from DES formalisms and organize them into an ontology in order to provide a foundation for ontology-based model representations.

3 PROCESS INTERACTION ONTOLOGIES

Researchers have implemented a variety of approaches for developing PI DES ontologies. Much of the challenge of developing an ontology involves minimizing the compromises in integrating knowledge from related sub-domains (i.e., keep complexity under control, but do not muddy or dilute the definitions of concepts for the sake of unity). The approach taken by DeMO is to provide a comprehensive ontology for discrete-event simulation with parts for each of the DES world views (Fishwick and Miller 2004). An alternative approach is taken by Lacy (2005b, 2005c) in which a complete ontology is built specifically for the process interaction world view. A third approach would be to develop an ontology for discrete-event simulation based on a unified framework such as DEVS. Time will tell the subset of the these three approaches that will be useful.

3.1 PI DES Concepts

The first step in developing a formal ontology is often to identify the primary concepts for the subject domain and their relationships. Figure 1 identifies categories of common high-level concepts associated with process interaction world view simulation models.

It is prudent to examine the major formal frameworks that have been developed for the process interaction world

view. We have given an overview of two of the most detailed formal frameworks in the background section (Section 2). We now discuss the development of process oriented simulation ontologies.

Two research groups rely on Figure 1 as a common core for the development of two ontologies. The first is called Process Interaction Modeling Ontology for Discrete Event Simulations (PIMODES) and the second is called DeMO. The intention of PIMODES is to create a complete ontology focused on the process interaction world view, while the purpose of DeMO is to provide a comprehensive ontology for discrete event model interaction of which the process interaction world view is an important component.

3.2 PIMODES

PIMODES is being developed to support the interchange of simulation models.

The PIMODES ontology includes sets of classes for basic concepts (e.g., entity type, entity attribute, variable, location), activities (e.g., creation, branching, processing, disposition), and control flow relationships of activities described with flowcharts. Ontological properties help define class attributes with OWL data type properties for simple values and OWL object properties to relate instances of classes.

PIMODES provides a formal open neutral interchange format that is encoded using the Web Ontology Language – OWL. OWL uses the Resources Description Framework (RDF) encoded using XML. The ontology is heavily influenced by implicit formalisms from popular software packages and standard terminology from the Workflow Management Coalition.

PIMODES serves as a Data Interchange Format (DIF) for DES models that are described using the process interaction world view. Historically, simulation application vendors have used proprietary methods for representing their models. However, they often provide programmatic access to their internal data structures or export the contents of their data structures to accessible file formats. Portions of models developed in Arena, ProcessModel, AnyLogic, and ProModel have been translated to and from PIMODES instance data files.

3.3 PIModel of DeMO

In the past, natural language has been used to provide semantics for modeling and simulation, but these semantics are not machine processable or searchable. Multiple groups are working to overcome this limitation by creating ontologies for modeling and simulation. The Discrete Event Model Ontology (Miller et al. 2004; Miller and Baramidze 2005), known as DeMO, uses the XML-based ontology language OWL to define a general ontology for state oriented, activity oriented, event oriented, and process ori-

ented models. DeMO defines several upper level formalisms that are used as root classes for all of its other modeling formalisms.
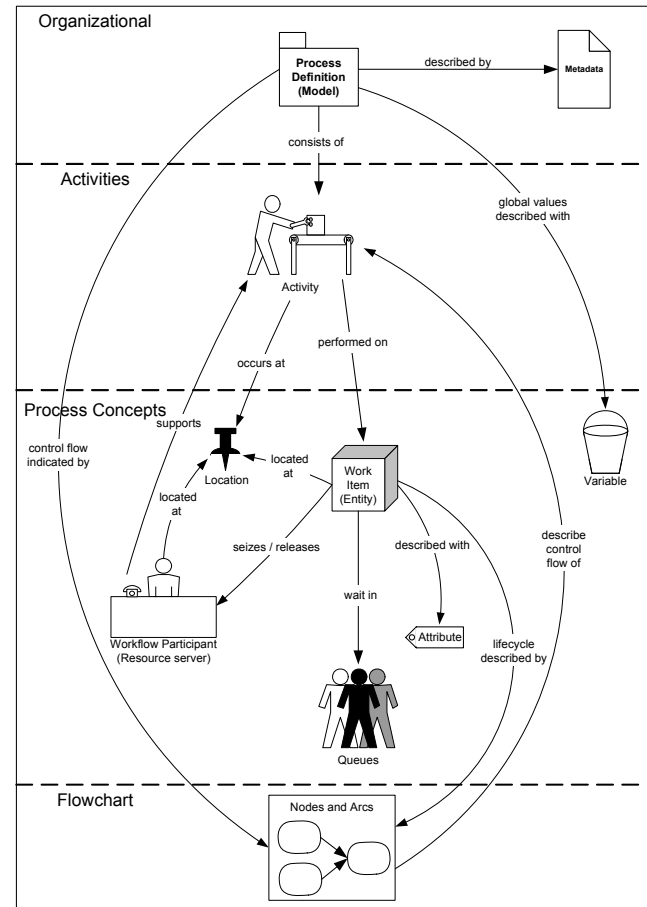

Figure 1: Model Representation Concepts in Process Interaction World View

We have extended the process oriented portion of DeMO (PIModel) to better support ontology based representations of simulation models. Figure 2 contains a graphical representation of the DeMO PIModel ontology. Models can be represented as instances of the DeMO PIModel using OWL or as instances of the Process Interaction Modeling Language (XPIM) which will be introduced in Section 4.

4 ONTOLOGY BASED MODEL REPRESENTATION

Model specifications can be represented as instances of a DES ontology defined using OWL. Models can be represented as RDF/XML files that conform to the PIMODES ontology. Translation software has demonstrated that PIMODES model instance files can be converted to/from

popular simulation software packages. Models can also be described as instances of the DeMO PIModel.
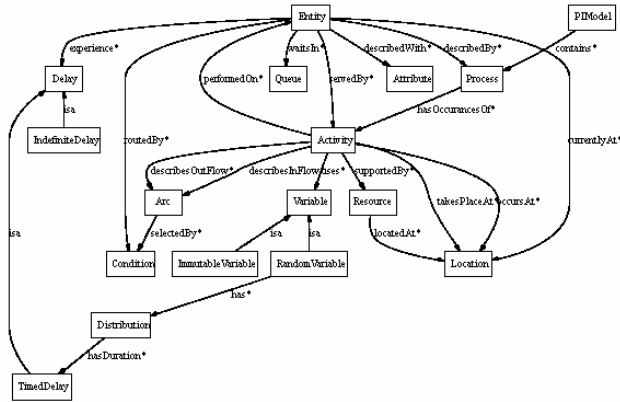

Figure 2: DeMO PIModel Ontology

4.1 Simulation Markup Languages as XML Dialects

Markup languages are languages that allow text along with information about the format, presentation, or meaning of the text to be placed in the same document. The most common example is HTML which allows a document to contain text and tags, used to describe information about the presentation of the text. While HTML is concerned with allowing specific types of applications, usually Web browsers, to display the information embedded in a document, another more general purpose markup language called XML can be used to describe text for almost any purpose. XML is most often used to describe other more specialized markup languages. This is accomplished by providing tools for defining document types or schemata. Once a schema has been defined XML documents conforming to the schema can be created. In order to conform to a schema an XML document must use the structure and vocabulary specified by the schema.

We have defined a markup language to describe discrete event simulation models using the process interaction world view. The language, Extensible Process Interaction Markup (XPIM), allows instances of model specifications to be defined in XML. The markup language was defined using XML schema, as seen in Figure 3, and instances can be transformed into Java for processing by the JSIM simulation engine (Nair, et al. 1996).

4.2 Models as a Collection of OWL Instances

When a model has been defined as DeMO PIModel instances the concepts used to define the model are standardized as specified by the ontology. Since a common set of terms, concepts and relationships are described in DeMO the model specification may be understood and processed by a variety of tools. Humans with access to DeMO have

the potential to read and understand the specification as does software written to make use of DeMO.

Figure 3: XPIM Schema

Describing models using DeMO has the advantage of a semantic grounding in a DES ontology. The components that make up a model are instances of particular DeMO PIModel classes and can be referred to, queried, stored, etc. based on their type.

Once described as DeMO instances models can be transformed into a modeling markup language. DeMO based Petri Net models have been transformed to PNML (Billington, et al. 2003) and DeMO based PI models have been transformed into XPIM as described later in this paper. The process for transforming DeMO PIModel instances into XPIM model specifications has been automated using the Extensible Stylesheet Language Transformation (XSLT) (Clark 1999). The XSLT document used for the transformation contains template rules for transforming DeMO PIModel instances into XPIM instances. This process is described in Section 4.3.

4.3 Transforming Ontological Instances into Markups

Process Interaction models in XPIM are described as a collection of processes, which are themselves described as a collection of activities. Each process within the model is a directed graph with nodes representing activities and arcs indicating the control flow between activities. The primary elements of the XPIM schema are matched with classes defined in the DeMO PIModel. Figure 4 shows an example of two XPIM elements that match PIModel classes.

As mentioned earlier, PI models can also be expressed as OWL instances of DeMO PIModel classes. Since XPIM has a semantic grounding in the DeMO PIModel, PIModel instances can be transformed into XPIM instances using

XSLT. An XSLT stylesheet containing template rules for transforming instances of PIModel classes into XPIM elements is executed by an XSLT processor to create an XPIM instance document. Figure 5 shows an example of a DeMO PIModel instance for a loan application activity. The *LoanApplicaton* instance of the *Activity* class in the example has properties that describe the flow of control into and out of the activity and the resource which supports the activity. Each of the properties of the *LoanApplication* instance are also instances of DeMO PIModel classes, and the *LoanApplication* instance is part of a collection of *Activity* class instances used to describe a *Process* class instance.
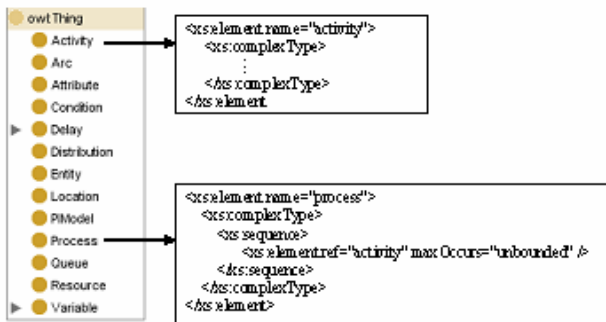

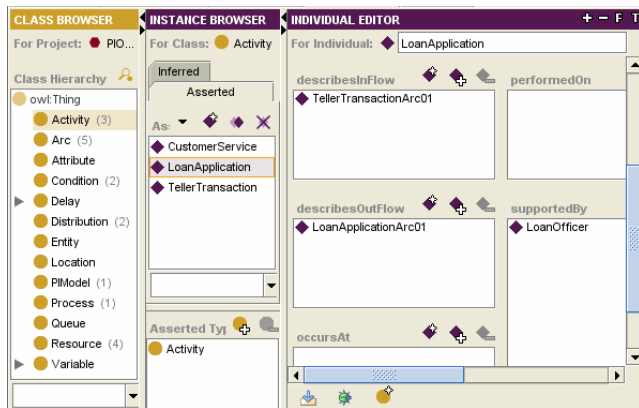Figure 4: XPIM Elements Matched to PIModel Classes

Figure 5: DeMO PIModel Instance

The instances seen in Figure 5 can be transformed into XPIM elements as shown in Figure 6. The *LoanApplication* instance of Figure 5 maps to an XPIM *activity* element, with the name of the instance mapping to the *activityid* attribute of the XPIM element. The instances that represent the activity's inflow and outflow arcs map to XPIM *arc* elements, while the *Resource* instance, *LoanOfficer*, maps to an XPIM *resource* attribute of the *LoanApplicaton* element.

Figure 6: XPIM Instance

While DeMO PIModel instances can be translated into XPIM instances that are suitable for further translation into JSIM model specifications, (see Figure 7), our goal is to provide an ontology and a markup language that will provide general support for ontology driven simulation model development. Ontology driven development of JSIM model specifications is simply a first step in the process.

Figure 7 illustrates the process that we are currently using for ontology driven development of DeMO PIModel based simulation models. Models can be created as DeMO PIModel instances, be transformed into XPIM and then to JSIM model specifications which can be executed via the JSIM simulation engine. The transformation of XPIM instances into JSIM model specifications is accomplished using our XPIM to JSIM Translator.
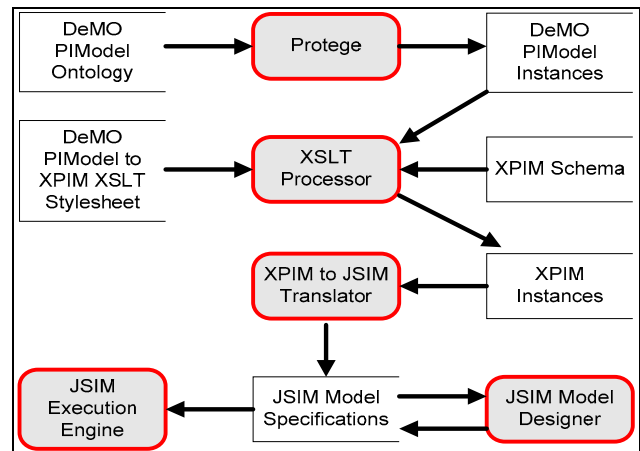

Figure 7: Ontology Driven Model Development Using the DeMO PIModel

5 CONCLUSIONS AND FUTURE WORK

This paper presents PI DES modeling ontologies developed by two different research groups. Both were devel-

oped using OWL, but each group takes a different approach for the development of their ontology.

PIMODES is intended to be a complete ontology focused specifically on the process interaction world view and is being developed to support the interchange of simulation models. Models can be represented as PIMODES instances and PIMODES instances have been converted to/from popular simulation software packages. DeMO is intended to be a comprehensive DES ontology with parts for each of the DES world views. The process interaction portion of DeMO (PIModel) has been extended to better support ontology-based representations of simulation models. PI DES models can also be represented as DeMO PI-Model instances and can be transformed into XPIM instances using XSLT. XPIM instances can then be transformed into executable JSIM simulation specifications.

Areas to consider for future work include the following: (1) Research on ontology-based representations of simulation models could compare and contrast the various approaches for representing PI DES models with ontologies. (2) DeMO PIModel instances and JSIM simulation specifications can both be created and edited using graphical design tools but no design tool exists for creating and editing XPIM instances. A graphical design tool suitable for editing XPIM should be developed so that model specifications can be created or refined in each of the languages supported by the DeMO PIModel. (3) The XPIM to JSIM Translator needs to be enhanced to support the translation of XPIM instances into languages used by other simulation software packages.

APPENDIX

Classical DEVS: DEVS provides a formal framework for discrete event systems using mathematical abstraction. It uses the components $(X, S, Y, \delta_{int}, \delta_{ext}, \lambda, \tau)$ to specify a model. The easiest way to understand DEVS is to focus on the transition functions, both internal (δ_{int}) and external (δ_{ext}). Let x at time t_x be the first input once the system is in state s since time t_s, then the next state s' is given by whichever of the transition occurs first.

$$s' = \begin{cases} \delta_{int}(s) & \text{if } \tau(s) < t_x - t_s, \\ \delta_{ext}(s, t_x - t_s, x) & \text{otherwise} \end{cases}$$

The specification includes the definition of three sets and four functions:

- X is a set of inputs.
- S is a set of states.
- Y is a set of outputs.

- $\tau : S \to \Re_0^+$ is a *time advance function* which computes the time that the model stays in state s given no external transitions (events) occur in the time interval $[t_s, t_s + \tau(s)]$.
- $\delta_{int} : S \to S$ is an *internal transition function* which determines the next state $s' = \delta_{int}(s)$, provided no external events occur in the time interval $[t_s, t_s + \tau(s)]$.
- $\delta_{ext} : S \times \Re_0^+ \times X \to S$ is an *external transition function* which determines the next state $s' = \delta_{ext}(s, t_x - t_s, x)$ as a response to an external event, provided this event occurs before an internal transition changes the state.
- $\lambda : S \to Y$ is an *output function* which determines the output, $y = \lambda(s)$. The output is generated just before the internal transition changes the state (external transition functions can only produce output indirectly by triggering internal transitions).

REFERENCES

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2001. *Discrete Event System Simulation*. 3rd ed. Upper Saddle River, New Jersey: Prentice Hall.

Berners-Lee, T., J. Hendler, O. Lassila. 2001. The Semantic Web. *Scientific American* May 2001: 28-37.

Billington, J., S. Christensen, K. van Hee, E. Kindler, L. Petrucci, O. Kummer, C. Stehno, and M. Weber. 2003. The Petri Net Markup Language: Concepts, Technology, and Tools. In *Proceedings of the International Conference on Applications and Theory of Petri Nets*, 483-505. Eindhoven, Netherlands.

Blais, C. and L. W. Lacy. 2004. Semantic Web: Implications for Modeling and Simulation System Interoperability. In *Proceedings of the Fall 2004 Simulation Interoperability Workshop*. Orlando, Florida.

Clark, J. 1999. XSL Transformations (XSLT) Version 1.0. Available at <http://www.w3.org/TR/xslt/> [Accessed June 8, 2006].

Cota, B. A. and G. S. Sargent. 1992. A Modification of the Process Interaction World View. *ACM Transactions on Modeling and Computer Simulation*. (2):2, 109-129.

Fishwick, P. A. and J. A. Miller. 2004. Ontologies for Modeling and Simulation: Issues and Approaches. In *Proceedings of the 2005 Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Gruber, G.R. 1995. Toward Principles for the Design of Ontologies Used in Knowledge Sharing. *International Journal of Human Computer Studies*. Vol. 45: 907-928.

Kiviat, P. J. 1969. Digital computer simulation: Computer programming languages. RAND Memo. RM-5883-PR, RAND Corporation, Santa Monica, California.

Lackner, M. R. 1962. Toward a General Simulation Capability. In *Proceedings of the AFIPS Spring Joint Computer Conference*, 1-14. San Francisco, California.

Lacy, L. W. 2001. Representing Computer Generated Forces Behaviors Using eXtensible Markup Language (XML) Techniques. In *Proceedings of the 10th Conference on Computer Generated Forces*. Norfolk, Virginia.

Lacy, L. W. 2005a. *OWL: Representing Information Using the Web Ontology Language*. Victoria, British Columbia: Trafford Publishing.

Lacy, L. W. 2005b. PIMODES. *2005 Winter Simulation Conference (Doctoral Symposium)*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Lacy, L. W. 2005c. PIMODES. *Fourth International Semantic Web Conference Doctoral Symposium*. Galway, Ireland.

Lacy, L. W. and W. J. Gerber. 2004. Potential Modeling and Simulation Applications of the Web Ontology Language – OWL. In *Proceedings of the Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

McGuinness, D. L. and F. Harmelen. 2004. OWL Web Ontology Language Overview. Available at <http://www.w3.org/TR/owl-features/> [Accessed June 8, 2006].

Miller, J. A. and G. Baramidze. 2005. Simulation and the Semantic Web. In *Proceedings of the 2005 Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

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.

Nair, R., J. A. Miller, and Z. Zhang. 1996. A Java-Based Query Driven Simulation Environment. In *Proceedings of the 1996 Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Noy, N. F. and D. L. McGuinness. 2001. Technical Report KSL-01-05, Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory, Stanford California.

Schriber, T. J. and D. T. Brunner. 1999. Inside Simulation Software: How it works and why it matters. In *Proceedings of the 1999 Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Tocher, K. D. and D. G. Owen. 1961. The Automatic Programming of Simulations. In *Proceedings of The Second International Conference on Operations Research*, 50-67.

Zeigler, B. P. 1976. *Theory of Modeling and Simulation*. New York: John Wiley and Sons.

AUTHOR BIOGRAPHIES

GREGORY A. SILVER is a Ph.D. 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.

LEE LACY is a Ph.D. candidate at the University of Central Florida. He also heads the Modeling, Simulation, and Training Business Unit at Dynamics Research Corporation (DRC) and is on the Board of Directors for the National Center for Simulation (NCS).

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 and workflow as well as parallel and distributed systems. Dr. Miller received a B.S. degree in Applied Mathematics from Northwestern University in 1980 and M.S. and Ph.D. degrees 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 100 technical papers in the areas of database, simulation and workflow. He is an Associate Editor for the ACM Transactions on Modeling and Computer Simulation and IEEE Transactions on Systems, Man and Cybernetics.