Iterchanging Discrete Event Simulationprocess Interaction Modelsusing The Web Ontology Language - Owl

2006

Lee Lacy
University of Central Florida

INTERCHANGING DISCRETE EVENT SIMULATION PROCESS INTERACTION MODELS USING THE WEB ONTOLOGY LANGUAGE - OWL

by

LEE W. LACY
B.S. University of Central Florida, 1985
M.S. University of Central Florida, 1987

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Modeling and Simulation
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2006

Major Professor: Dr. Jose Sepúlveda

ABSTRACT

Discrete event simulation development requires significant investments in time and resources. Descriptions of discrete event simulation models are associated with world views, including the process interaction orientation. Historically, these models have been encoded using high-level programming languages or special purpose, typically vendor-specific, simulation languages. These approaches complicate simulation model reuse and interchange.

The current document-centric World Wide Web is evolving into a Semantic Web that communicates information using ontologies. The Web Ontology Language – OWL, was used to encode a Process Interaction Modeling Ontology for Discrete Event Simulations (PIMODES). The PIMODES ontology was developed using ontology engineering processes. Software was developed to demonstrate the feasibility of interchanging models from commercial simulation packages using PIMODES as an intermediate representation.

The purpose of PIMODES is to provide a vendor-neutral open representation to support model interchange. Model interchange enables reuse and provides an opportunity to improve simulation quality, reduce development costs, and reduce development times.

ACKNOWLEDGMENTS

This research was supported by a very helpful committee that included:

- Dr. Jose A. Sepulveda, Chair
- Dr. Peter Kincaid,
- Dr. Michael D. Proctor,
- Dr. Luis Rabelo, and
- Dr. Charles H. Reilly, III.

I appreciate the consistent support from the corporate leaders and my coworkers at Dynamics Research Corporation (DRC), especially Bruce Harris. Most of all, I appreciate the patience and encouragement of my friends, family, and my biggest encourager – Alex Blanco.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ACD	Activity Cycle Diagram
AD	Activity Diagram
ASM	Abstract State Machines
BML	Battle Management Language
BMPI	Business Process Modeling Initiative
BOM	Base Object Model
BPEL4WS	Business Process Execution Language for Web Services
BPMN	Business Process Modeling Notation
C&P	Characteristics and Performance
CAPS	Computer Aided Programming for Simulation
CFG	Control Flow Graph
CGF	Computer Generated Forces
CMMS	Conceptual Models of the Mission Space
DAML	DARPA Agent Markup Language
DAO	Data Access Objects
DARPA	Defense Advanced Research Projects Agency
DeMO	Discrete-Event Modeling Ontology
DES	Discrete Event Simulation
DESS	Differential Equation System Specification
DEVS	Discrete Event System Specification

DIF	Data Interchange Format
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling Simulation Office
DOAT	DRC Ontology Authoring Tool
DoDAF	Department of Defense Architectural Framework
DRC	Dynamics Research Corporation
DSB	Dynamic Scenario Builder
DTD	Document Type Description
DTSS	Discrete Time System Specification
ECSL	Extended Control and Simulation Language
EU	European Union
FIFO	First In First Out
FOM	Federation Object Model
GPSS	General Purpose Simulation System
GSMP	Generalized Semi-Markov Processes
GUI	Graphical User Interface
HACD	Hierarchical Activity Cycle Diagrams
HLA	High Level Architecture
HTML	HyperText Markup Language
IBM	International Business Machines
IDE	Integrated Development Environment
IDEF	Integration DEFinition
IEEE	Institute of Electrical and Electronics Engineers

KA/KE	Knowledge Acquisition/Knowledge Engineering
KBSI	Knowledge Based Systems, Inc.
KIF	Knowledge Interchange Format
KR	Knowledge Representation
M&S	Modeling and Simulation
MTBF	Mean Time Between Failure
MTTR	Mean Time To Replace
ND	Network Diagrams
NIST	National Institute of Standards and Technology
OIL	Ontology Interface Layer
OMG	Object Management Group
OMML	OpenModel Modeling Language
OMT	Object Model Template
OOD	Object-oriented Design
OOS	OneSAF Objective System
OWL	Web Ontology Language (not a true acronym)
OWL-S	OWL Services
PC	Personal Computer
PDM	Precedence Diagramming Method
PERT	Program Evaluation and Review Technique
PIMODES	Process Interaction Modeling Ontology for Discrete Event Simulation
PN	Petri Net
PNML	Petri Net Markup Language

PSL	Process Specification Language
RDF	Resource Description Framework
RDFS	RDF Schema
SCML	Scenario Markup Language
SCORM	Sharable Content Object Reference Model
SEDRIS	Synthetic Environment Data Representation and Interchange Specification
SH-ACD	Simplified Hierarchical Activity Cycle Diagrams
SIMAN	SIMulation Analysis
SISO	Simulation Interoperability Standards Organization
SLAM	Simulation Language for Alternative Modeling
SMSDL	Simulation Model Specification and Documentation Language
SOM	Simulation Object Model
SRML	Simulation Reference Markup Language
STD	State Transition Diagram
SWRL	Semantic Web Rule Language
TCP/IP	Terminal Control Program/Internet Protocol
TPN	Timed Petri Nets
UI	User Interface
UML	Unified Modeling Language
UOB	Unit Order of Battle
URI	Uniform Resource Identifier
US	United States
VB	Visual Basic

VBA	Visual Basic for Applications
VIMS	Visual Interactive Modeling Systems
W3C	World Wide Web Consortium
WfMC	Workflow Management Coalition
WPDL	Workflow Process Definition Language
WSBPEL	Web Services Business Process Execution Language
WSDL	Web Service Definition Language
WWW	World Wide Web
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XMLS	XML Schema
XMSF	Extensible Modeling and Simulation Framework
XPIM	Extensible Process Interaction Markup
XPoD	XML Populated DIF

1 CHAPTER ONE: INTRODUCTION

The Process Interaction Modeling Ontology for Discrete Event Simulation (PIMODES) research provides a new ontology-based interchange approach for sharing Discrete Event Simulations.

1.1 Subject Problem

Simulation development requires substantial investments in resources. Model developers create discrete event simulations (DES) with a wide variety of software packages and programming languages. These simulations execute models of particular systems/domains. The process interaction world view is a popular method for representing discrete event simulations. Reichenthal and Gustavson (2003) define model sharing as the ability for a simulation system to use models developed for another system. The variety of representations of DES models complicates their reuse. Simulation developers need new solutions that enable simulation model interchange and make simulation development more cost-effective.

Miller and Fishwick (2004) identified the need for an efficient method for linking simulation concepts to support databases, query engines, and human-computer interaction. Knowledge representation and composability technologies provide new opportunities for simulation interchange. The simulation community is adopting new interchange technologies such as XML. Semantically rich languages, such as the Web Ontology Language – OWL, best support interchanging business processes with a global schema (Mendling, De Laborda, & Zdun, 2005).

Knowledge representation, using techniques such as frames and inheritance, is particularly relevant to simulation (Nielsen, 1991). Knowledge representation of simulation models supports composability. Composition using reusable components has been offered as a potential solution. A RAND report (Davis & Anderson, 2003) concluded that "the time is ripe…[for] higher-level representations that would simplify characterization of components".

1.2 Research Purpose

The purpose of this research is to develop a new language, formalized using a Semantic Web ontology, for representing process interaction DES models. An ontology provides a formal specification that supports computer interpretation of conforming descriptions.

1.3 Study Significance

Miller and Fishwick (2004) claim that formally organizing simulation knowledge increases the interoperability, integration, and reuse of simulation artifacts. By representing DES models with a vendor-neutral Web standard, simulation software applications can interchange their internal data models with the standard representation. Seila (2005) states that a standard modeling representation would lead to improved stakeholder communication, aid model verification, improve model documentation, help separate models from software, improve model interchange, promote model component reuse, and support system construction and maintenance. Representing DES models with OWL also exposes modeling information to non-simulation applications that are compliant with Semantic Web standards.

1.4 Chapter Contents

This introductory chapter is followed by Chapter 2 – Literature Review which surveys the related literature. Chapter 3 – Methodology describes the approach taken to performing the research. Chapter 4 – Results documents the resulting products of the research. Chapter 5 – Conclusions provides an analysis of the research results.

2 CHAPTER TWO: LITERATURE REVIEW

A literature review identified existing related work and supported the development of a harmonized view of process interaction DES models.

2.1 Models and Simulations

The Modeling and simulation (M&S) community encompasses many types of technologies and techniques. Clearly specified semantics related to terminology support further discussions and formally define ontology concepts.

2.1.1 Definition / Scope

Figure 1 depicts a high level perspective of the relationships between subject systems, models, and simulations. This concept map differentiates between systems, models, modeling formalisms, modeling languages, simulation software, and simulators. The system is the subject of a simulation. It is described by a model that is encoded using a modeling language. The modeling language expresses model formalisms or representations. Simulation software or a simulator simulates the model. Historically, the modeling language has been very tightly coupled with the simulation software.

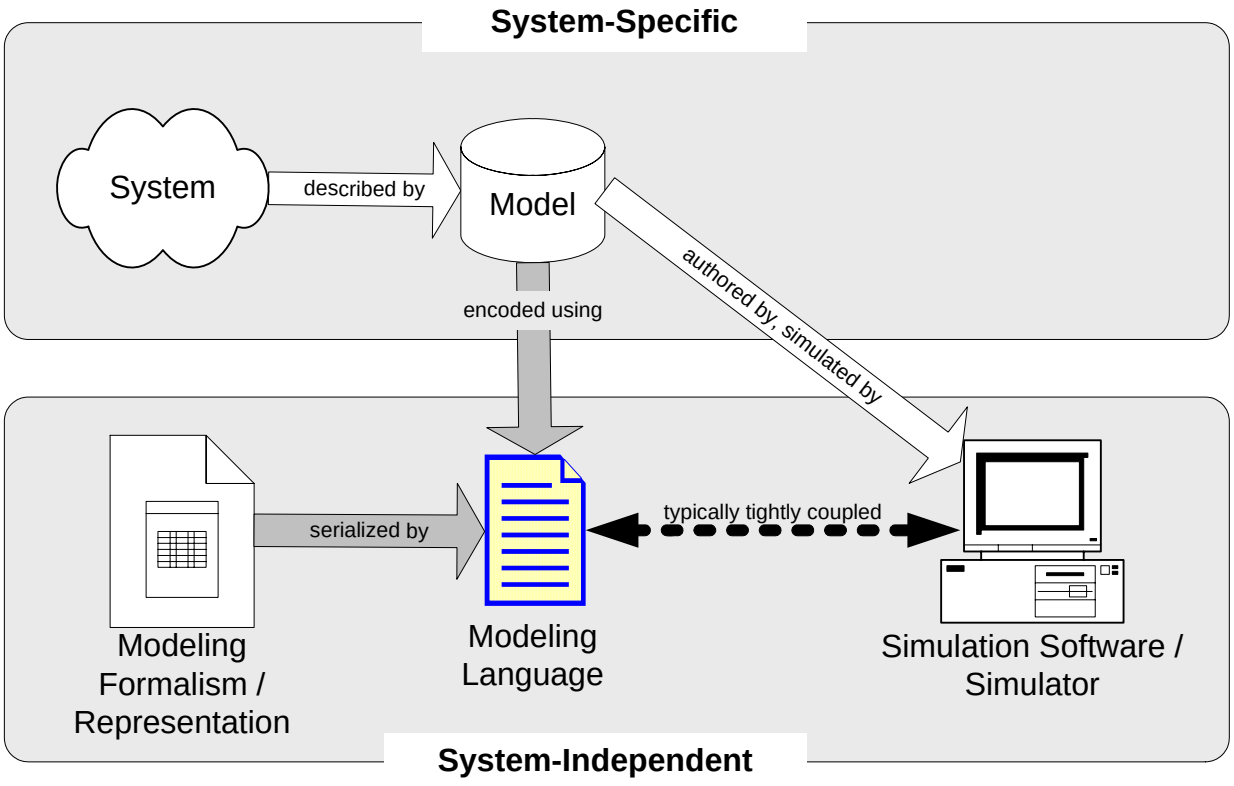

Figure 1. Modeling and Simulation Concept Map

2.1.1.1 Systems

The term "system" is used to refer to a variety of things of interest. Systems typically include interrelated components that perform a function (Cassandras & Lafortune, 1999). Systems can be categorized according to various dimensions:

- Natural vs. man-made,
- Continuous vs. discrete,
- Deterministic vs. stochastic, and
- Open vs. closed.

Systems whose changes occur in finite quanta, or jumps, are discrete systems (Pooch & Wall, 1993). Discrete event systems involve discontinuous changes (events) (Karayankis, 1995). Fishman (1978) defines a discrete system as one in which a phenomenon of interest changes value or state at discrete moments of time rather than continuously with time. The state of a discrete event system changes at only a discrete, but possibly random, set of time points, known as event times (Schriber, 1991). Systems are modeled in order to perform studies or support experiences using simulations.

2.1.1.2 Models

Simulation development methodology includes the modeling phase which includes the model translation step in which a model is prepared and debugged for computer processing. Models serve as surrogates for the subject systems. They enable experimentation and analysis that would be difficult or impossible using the actual system (Cassandras & Lafortune, 1999). Dr. George Box is often quoted for stating "All models are wrong, but some are useful". Models are "wrong" because they are simplified abstractions that fail to clone all aspects of a subject system. However, models are useful when they are sufficiently detailed to support their required use.

Models are encoded using modeling languages. Modeling languages contain statements that support modeling formalisms. Woolfson and Pert (1999) describe models as simplified representations of real objects or physical situations (systems) that serve a particular purpose. Miller and Fishwick (2004) define a model as an approximation of a system that evolves over time. Overstreet and Nance (1985) define a model as an abstraction of a system intended to

replicate some properties of that system. A combination of views is often necessary to adequately represent a system in a model. Liles and Presley (1996) describe a model as a collection of views consisting of a business rule (information) view, activity view, business process view, resource view, and organization view.

There are many varieties of models. Kelton, Sadowski, and Sturrock (2003) and Miller and Fishwick (2004) categorize models based on how they deal with:

- time (static vs. dynamic),
- state (discrete vs. continuous), and
- randomness (deterministic vs. stochastic).

Static models typically represent the allowable paths that objects in a system may follow. Dynamic models describe the behavior of a system over time and enable simulations for analysis (Whitman, Huff, & Presley, 1997). Discrete event models contain a set of state variables / states and a set of events. Page (1994) describes discrete event simulation models as abstract, dynamic, descriptive, and numerical models.

2.1.1.3 Simulations

Simulation is the process of numerically evaluating a system model and estimating variables of interest (Cassandras & Lafortune, 1999). Ball (1996) defines simulation as a technique for imitating the behavior of a situation or system using an analogous model, situation, or apparatus, to gain information more conveniently or to train personnel.

Simulation software executes models. A key distinction must be made between simulation software and the simulation modeling language used to encode the model. Simulation software is often closely tied to particular simulation languages. For example, the Arena®

software package implements the SIMulation Analysis (SIMAN®) simulation language. By abstracting the simulation model from the supporting simulation, simulation descriptions can become simulation-software-independent, enabling the development of abstract simulations (Zeigler, Praehofer, & Kim, 2000).

2.1.2 Types of Simulations

There are many different types of simulations. Specific types of simulations must be explicitly defined in order to scope the associated simulation models being represented. Simulations are categorized in a number of ways including purpose and application. However, the purpose of a simulation does not necessarily affect the way its model is represented.

DES category descriptions evolved as the discipline matured. Ziegler (1976) formalized three approaches to modeling as system specifications:

- Differential Equation System Specification (DESS),
- Discrete Event System Specification (DEVS), and
- Discrete Time System Specification (DTSS).

Nance (1993) describes simulation as an application domain of programming languages that are described as Monte-Carlo, continuous, or discrete event. Pidd (2002) categorizes simulations in terms of:

- time handling,
- stochastic vs. deterministic, and
- discrete vs. continuous.

Similarly, Sulistio, Yeo, & Buyya (2004) and Harrel & Price (2003) describe three primary properties of simulations:

- presence of time (static or dynamic),

- behavior (deterministic or stochastic/probabilistic), and
- basis of value (discrete or continuous).

The focus of this research is representing models for dynamic stochastic discrete simulations.

2.1.3 Discrete Event Simulations

Discrete event simulations represent one subset of simulations. Dependent variables (i.e., state indicators) change discreetly at particular points in time (events) in discrete event simulations (Pooch & Wall, 1993). Page (1994) categorizes discrete event models based on the following characteristics:

- linear vs. nonlinear,
- stable vs. unstable,
- steady-state vs. transient,
- probabilistic (stochastic) vs. deterministic, and
- autonomous vs. nonautonomous.

A variety of theoretical foundations describe discrete event simulations including Zeigler's Systems Theory, the Semi-Markov Processes, and Logic-based Foundation. Zeigler, Praehofer, & Kim (2000) provide a formal definition of DEVS based on systems theory. They define a "classic" DEVS as having:

- a set of input values,
- a set of states,
- a set of output values,
- an internal transition function,
- an external transition function,
- an output function, and
- a mapping of states to positive reals.

Glynn (1989) describes a particular type of stochastic process – Generalized Semi-Markov Processes (GSMP). GSMP descriptions involve precise mathematical descriptions to formally define a discrete event system. Radiya and Sargent (1994) define a logic-based foundation for discrete event models and simulation by defining terms from a logician's perspective, defining a Discrete Event Logic, and by describing a simulation algorithm for processing models described using the logic. Theoretical foundations are useful for formally defining classes of simulations, but have limited utility for representing models.

2.1.4 Discrete Event Simulation World Views

Discrete event simulations are typically associated with a particular world view. Nance and Sargent (2002) describe the history of DES world views in the 1960s. Zeigler, Praehofer, and Kim (2000) describe world views as simulation strategies that are realized in simulation languages and systems. World views, also known as conceptual frameworks, categorize approaches for representing and executing the logic in simulation models. DES languages are typically aligned with specific world views (Cota & Sargent, 1992).

DES literature associates the world views with events, activities, and processes. Pidd (2002) differentiates the classic views by contrasting them to the three phase approach that he recommends. The event-based world view focuses on events that occur and event-associated code. The activity-based world view focuses on conditional statements that specify the initiation of activities. The process interaction world view considers the complete lifecycle of an entity as it progresses through a process.

2.1.4.1 Event-based Approach

Events are the points in time when something triggers a change to the state of a system. The atomic components of event-based models are event routines (Pidd, 2002). Event routines are collections of programming language statements that describe the potential results (logical consequences) of an event.

The event-based approach focuses on the events that instantaneously transform a system's state and/or schedule future events (Miller & Fishwick, 2004). The event-based approach is also referred to as:

- Event approach (Pidd, 1984),
- Event scheduling (Banks & Carson, 1986) (Cota & Sargent,1992) (Schruben, 1983) (Trick, 2005) (Cassandras & Lafortune, 1999) (Zeigler, Praehofer, & Kim, 2000), and
- Event orientation (Pooch & Wall,1993).

Events trigger discontinuous changes in a system's state. Event types are typically associated with a procedure in a programming language. Events can schedule other events to be simulated at later times or cancel events that have already been scheduled to occur. An event procedure may change the state of the data objects that are used to represent the state of the system, and may use instructions for scheduling and canceling events.

Modelers using the event-based approach define the types of events that can occur and the causal relationships between events (Cota & Sargent, 1992). The event scheduling approach enables users to prepare a system description by concentrating on the moments in time when state changes occur. In event-oriented models, all events are prescheduled and are not activated by global state conditions (Zeigler, Praehofer, & Kim, 2000).

2.1.4.2 Activity-based Approach

Activities and their preconditions (triggers) are the focus of the activity-based world view (Miller & Fishwick, 2004). Activities are described with preconditions and actions similar to rule-based programming languages (Balci, Bertelrud, Esterbrook, and Nance, 1998). An activity's conditions must be satisfied for an activity's operations to be scheduled and performed. Activities have associated start events and end events. The activity-based approach is a state-based approach to modeling (Balci, Bertelrud, Esterbrook, and Nance, 1998). In this approach, events can be based on conditions (contingency tests) (Zeigler, Praehofer, & Kim 2000). The activity-based approach is also referred to as:

- Activity scanning (Zeigler, Praehofer, & Kim, 2000) (Banks & Carson, 1986) (Cota & Sargent,1992) (Schruben, 1983) (Trick, 2005) (Pooch & Wall, 1993) (Balci, Bertelrud, Esterbrook, & Nance, 1998),
- Activity approach (Pidd, 1984), and
- Two-phased approach (Balci, Bertelrud, Esterbrook, & Nance, 1998).

An activity-based model is described by defining the types of events that can occur and their causal relationships (Cota & Sargent, 1992). Modelers can also define contingent events that occur when a stated condition is met. Modeling formalisms used to describe the activity-based approach models include:

- Activity Cycle Diagrams (ACD),
- Petri Nets (PN),
- activity wheel charts, and
- activity lifecycle diagrams (Miller & Fishwick, 2004) (Schruben, 1983).

ACDs are supported by tools such as Computer Aided Programming for Simulation (CAPS) (Clemenston, 1986)

2.1.4.3 Process Interaction Approaches

The process interaction world view can be considered a combination (hybrid) of the activity-based and event-based approaches (Zeigler, Praehofer, & Kim, 2000). The process interaction approach focuses on processes and the entities that flow through the process and interact with resources (Banks & Carson, 1985) (Miller & Fishwick, 2004). The process interaction approach is also referred to as the process orientation world view (Pooch & Wall, 1993) (Trick, 2005). A process is the sequence of operation that an entity passes through during its life in the system (Pidd, 2002). Processes are sequences of events or sequences of activities (Cota & Sargent, 1992). Processes describe the behavior of entities that flow through a system (Miller & Fishwick, 2004). Processes are typically represented by control flow diagrams that describe the sequence of processes that each entity proceeds through during its lifecycle. Entities move through a system and consequently through time. Entities sometimes encounter impediments to progress and are delayed.

Process oriented simulations represent a large class of DES that involve resource contention (Cassandras & Lafortune, 1999). Entities undergo a sequence of events separated by time intervals as they flow through the DES. Entities either receive resource services or wait for resources. Processes are described for each type of entity.

The process interaction world view is considered to be a natural way to describe models (Franta and May, 1977). The process interaction approach requires a modeler to describe the flow of each entity through the system by defining a set of processes, entities, and resources (Cota & Sargent, 1992). Since entities move through their lifecycle, people often visualize

entities moving through a system by anthropomorphizing and drawing analogies to construct their own mental models of the modeled system.

Activity cycle diagrams can be used to describe entity processes (Pidd, 2002). Additional modeling formalisms used to describe the process interaction world view include Petri Nets (Miller & Fishwick, 2004), Control Flow Graphs (CFGs) (Cota & Sargent, 1992), Activity Diagrams (AD), and Network Diagrams (ND). Software that support the simulation of process interaction models include: GPSS, SIMPL/1, SIMSCRIPT II.5, SIMULA.

Cota and Sargent (1992) proposed a modification to the traditional process interaction world view to support modularity and encapsulation. Zeigler, Praehofer, and Kim, (2000) describe two sub-views of the process interaction world view that are associated with an emphasis on resources or entities.

The business world's focus on modeling business processes and the Web services community's focus on modeling processes provide potential opportunities for leveraging standard descriptions of processes.

2.1.4.4 Non-Classical Approaches

In addition to the three commonly described "classical" approaches, there are additional world views employed for performing discrete event simulation. Pidd (2002) describes the Three-Phase approach as a more efficient variant of Activity-Scanning or a hybrid of Activity-Scanning and Event-Scheduling.

2.1.4.5 World View Description Summary

Although the event-based approach is the most computationally efficient of the three classical world views, the process interaction approach is closer to most people's mental model. The activity-based approach is less efficient than the event scheduling approach because it requires frequent evaluation of conditions that would not be evaluated with event scheduling. The process interaction approach is more efficient than the activity-scanning world view. However, it is less efficient than the event-based approach. The process interaction approach has been popularized by the abundance of available easy-to-use tools.

2.2 Model Representation

Pidd (2002) describes the representations of models as model logic. The term model is more commonly used. Discrete event simulations models typically result from a development process that involves the use of simulation software with underlying languages that represent a particular modeling formalism or representation approach.

2.2.1 Process Interaction Concepts

The process interaction approach focuses on entities and process descriptions. Process interaction models describe the lifecycle of objects that move through and interact with system processes (Balci, Bertelrud, Esterbrook, & Nance, 1998). The main components of a process oriented simulation are entities, attributes, process functions, resources, and queues (Cassandras & Lafortune, 1999). Simulated processes typically have associated software procedures. Procedures associated with delays suspend execution for an interval of time.

2.2.2 Model Development Process

The model development approach typically involves using authoring software to create a model that is executed by a simulation engine which produces statistical and/or animation results. A key step in the development of a simulation is the encoding of the model using a simulation programming language. Simulation models can be expressed using a high-level programming language or described as data for execution by a data-driven simulation system (Ball, 1996).

There is an important distinction between simulations and the models they execute. Zeigler, Praehofer, & Kim (2000) point out that separating the model from the simulation provides a number of benefits including:

- portability and interoperability by executing a formalized model using multiple simulators and
- the ability to develop and verify simulation algorithms for executing the model formalisms.

Pidd (2002) describes many of the modern software packages as visual interactive modeling systems (VIMS) (e.g., Witness, ProModel, Micro Saint). Most VIMS use a network as their underlying generic model with entities flowing through the network from node to node. From a user's perspective, a model is normally encoded with software that has an underlying model language that supports a particular formalism.

2.2.3 Simulation Software Implementation Approaches

A variety of approaches are used for encoding simulations with programming languages. Both general purpose and special-purpose simulation languages are used. General purpose

languages were used for implementing simulations before special purpose languages were developed. General purpose languages continue to be used due to cost and complexity issues (Pidd, 2002). Although reuse can be achieved with general purpose programming languages, most contemporary models are developed using simulation packages. Schriber (1991) points out that no single modeling language works well for all situations. A variety of modeling languages exist to support various applications of simulation.

2.2.3.1 Simulation Language Categories

Just as Integrated Development Environments (IDEs) are now used to develop software applications, VIMS are increasingly popular for developing simulations. VIMS have either an explicit or implicit simulation language underlying them.

Kreutzer (1986) differentiates between low-level languages that are optimized for computers and high-level languages that are closer to a human's thinking processes. He describes several categories of simulation-specific languages as:

- packaged and precompiled program libraries (e.g., SIMPAS, SIMPL/1, SLAM, GASP, SIMAN),
- application-oriented general purpose languages (e.g., SIMULA, SIMSCRIPT),
- scenario languages / application-oriented language extensions (e.g., GPSS, DEMOS), and
- declarative languages.

Modeling methodologies include network representations, process concept, and the entity-attribute-set approach. Network representations are often used to describe DES models. Implementations of network representations include GPSS and activity-cycle-based languages (Overstreet & Nance, 1985). Pidd (2002) classifies simulation software approaches into the following categories:

- programming approaches in general purpose languages,
- programming approaches in simulation languages,
- block-structured systems, and
- visual interactive modeling systems (VIMS).

2.2.3.2 Visual Interactive Modeling Systems (VIMS)

DES models can be described with simulation languages and visual simulation software packages. DES models can be represented by simulation languages (e.g., SLAM, Extended Control and Simulation Language, SIMAN) (Ball, 1996). Simulation languages provide versatility in describing models. However, encoding models using a simulation language can be a complex process. Alternatively, simulation software packages (e.g., Witness, Simul8, Micro Saint, Automod, ProModel, Taylor II) can be used to describe DES models (Pidd, 2002). The advantage of VIMS is that they speed up the development process (Ball, 1996). Some packages (e.g., Arena) employ an alternative hybrid approach, providing the flexibility of a programming or simulation language and the productivity of a VIMS.

VIMS can be considered simulation systems with graphical representations. Similarly, some graphical representation tools (e.g., ProcessCharter) have simulation capabilities. It is difficult to construct an exhaustive list of DES simulation packages and languages as software continues to be developed and evolved. However, identifying the key features of representative and popular languages and packages is helpful for categorizing purposes. VIMS typically persist their models as datafiles, allowing them to be more easily exchanged.

2.2.3.3 Object-Oriented Simulation languages

Object-oriented techniques are sometimes used to represent simulation models. Many object-oriented simulation techniques can be traced back to the SIMULA programming language from the 1960s. Simulation packages that employ object-oriented techniques include Simple++. Object-oriented techniques can be used to describe entities in a DES (Pidd, 2002). Benefits of this approach include the ability to extend existing definitions through inheritance.

2.2.3.4 Agent Based Simulation

Agent based simulations model intelligent, autonomous entities (agents) as they interact to attain some goal in their environment (Dubiel & Tsimhoni, 2005). Although the focus is on entities within the simulation, the models tend to be more activity-based rather than process based. The behavior of the entities is typically described with state transition diagrams rather than the control flow diagrams typically used to describe entities with the process world view. AnyLogic is an example of an agent based simulation package (Dubiel & Tsimhoni, 2005). AnyLogic's discrete modeling framework includes statecharts, timers, and events to simulate object behavior. AnyLogic includes its Enterprise library that implements activities as active objects that treat entities as messages.

2.2.4 Process Interaction Modeling Software Packages

Simulation software packages support either an explicit or implicit simulation language underlying their application for representing simulation models. Several vendors provide simulation software packages that support the development of process interaction simulations.

Swain (2003) identified over forty software tools that support DES. The simulation software market is very fluid and new products continue to become available. Therefore, surveys are quickly out of date. An important aspect of surveys is the identification of tools that belong to classes of software to be supported by interchange mechanisms. The following sections describe some well known DES software packages.

2.2.4.1 Arena Software Package

Arena® is a software package used for graphically describing SIMAN models. Arena uses hierarchical flow chart models that include graphical objects (icons) called modules (Banks & Carson, 1996). Arena icons are connected in a flowchart to represent entity flow.

Arena uses an object-oriented design for graphically developing models (Markovitch & Profozich, 1996). Arena modeling constructs, called modules, are grouped into templates for arrangement into hierarchical model diagrams (Law & Kelton, 2000). Module specifications are authored using dialog boxes and spreadsheet-style forms. Arena's modules represent types of data and commands within the software. These modules effectively represent a vendor-specific simulation language.

Arena provides integration with Visio, Active X interfaces, Data Access Objects (DAO) interfaces, and Visual Basic for Applications (VBA) to extend the tool's capabilities (Bapat & Swets, 2000).

2.2.4.2 AutoMod

The AutoMod simulation package is focused on manufacturing and material handling systems. Templates are used for representing common entities and resources. A simulation programming language is also available (Banks, 2001). AutoMod models can describe process systems that contain complex logic to control the flow of materials, messages, resource contention, or wait times (Rohrer, 2000). Automod has general programming features including the specification of processes, resources, loads, queues, and variables (Banks & Carson, 1996). AutoMod processes are described in terms of traffic limits, input connections, output connections, and itineraries. AutoMod resources are described in terms of their capacity, processing time, Mean Time Between Failure (MTBF), and Mean Time To Replace (MTTR). Schriber (2001) maps generic discrete event simulation terms to the concepts used in AutoMod.

2.2.4.3 ProModel

ProModel provides manufacturing-oriented modeling elements and rule-based decision logic (Banks, 2001). It is a simulation tool used for modeling manufacturing and service systems (Harrell, Ghosh, and Bowden, 2000). ProModel elements include parts/entities, locations, resources, path nets, routing/processing logic, and arrivals. Systems are modeled in ProModel by selecting modeling elements and modifying appropriate parameters (Harrell and Price, 2000) (Harrell and Price, 2003). ProModel variants (with different graphics libraries) are available for the medical domain (MedModel) and service domain (ServiceModel). ProModel constructs have been mapped to the NIST shop model interchange format (Harward, 2005).

2.2.4.4 Witness

WITNESS is a simulation software package oriented towards manufacturing. WITNESS models are based on template elements that are combined into a designer element for reuse (Banks, 2001).

2.2.4.5 ProcessModel

The ProcessModel® software package provides a graphical user interface to define and execute simulation models called process models. Process models are flow diagrams that can include objects representing process elements and connections depicting element relationships (ProcessModel, 1999). ProcessModel object types include entities, activities, storages, and resources. ProcessModel connection types include entity arrivals, entity routings, resource assignments, and order signals.

2.2.4.6 SIMPROCESS

SIMPROCESS is a process modeling tool whose models are described with processes, resources, and entities (flow objects) (Swegles, 1997). SIMPROCESS models can be simulated using an event-driven approach.

2.2.4.7 Software Package Summary

Reichenthal and Gustavson (2003) identified a common architecture employed by many process simulation tools. The software in these systems can be viewed as having three layers to

their architectures. The first layer provides a Graphical User Interface (GUI) for building the simulations. The next layer contains the process simulation domain objects. The third layer provides the discrete event simulation engine, storage, and communication. Most contemporary DES simulation software packages supporting the process interaction world view share the following characteristics:

- Personal Computer (PC)-based,
- Graphical user interface with "drag and drop" modeling features,
- Support for hierarchical models,
- Support for evolutionary model optimization,
- Process flow depictions of models, and
- Proprietary file formats used for encoding models.

2.2.5 Process Interaction Modeling Languages

Process interaction modeling languages explicitly or implicitly underlie the tools used by modelers. Certain simulation language support particular simulation world views (Fishman, 1978). This section describes some of the DES modeling languages that support the process interaction world view. The following sections describe sample languages.

2.2.5.1 GPSS/H

One of the earliest simulation languages is General Purpose Simulation System (GPSS). GPSS/H is the contemporary version of the language which was originally released by IBM in 1961 (Schriber, 1991). In GPSS/H, a system is considered to be a collection of inter-related elements that work together to achieve a stated objective (Schriber, 1991).

GPSS/H models are described as a sequence of events, separated by lapses in time, which describe how "objects" flow through a system resembling the structure of a flowchart of the

system being modeled. (Crain, 1997). Complex GPSS/H models require procedural and text-based programming code to supplement the visual model built using the iconic approach (Henriksen and Crain, 2000).

GPSS/H supports the description of process oriented simulation models. GPSS/H models are described with files of "block" statements that can be expressed graphically with block diagrams that portray each statement as an icon connected to related statement icons using arrows (Crain, 1997). The GPSS/H language is based on over 60 types of "blocks" that have associated graphical representations (Cassandras & Lafortune, 1999).

GPSS/H models entities (called units of traffic in GPSS/H) that compete for resources (Schriber, 1991). Entities moving through a GPSS/H model are referred to as units of traffic and transactions. Transactions move from block to block along the one-way paths in the block diagram. Each block represents an action to be performed whenever a transaction enters a block. Blocks can have associated labels, an operation keyword (e.g., "Generate"), and most have one or more operands.

GPSS requires modelers to envision transactions (entities) flowing around a network (Pidd, 2002). The nodes of the network represent transaction delay points. GPSS facilities are the permanent entities that represent resources required by transactions.

2.2.5.2 Micro Saint

Micro Saint models are represented with flowchart diagrams that describe networks of tasks. Task networks represent a sequence of tasks that simulation entities flow through (Pidd, 2004). The diagrams support branching logic, sorted queues, and conditional task execution

(Banks, 2001). Micro Saint's task-based approach allows users to specify preconditions, and actions to take based on the beginning, ending, or launching of a task (Pidd, 2002).

2.2.5.3 SIMAN

The SIMulation Analysis (SIMAN) simulation language supports the description of DES models (Pegden, Shannon, & Sadowski, 1995). SIMAN is used to define the logical and physical components of a system. Standard features in SIMAN include the description of resources, queues, process logic, and system data (Banks, 1996). Processes are represented with SIMAN blocks that have associated graphical representations that are combined to create block diagrams (Cassandras & Lafortune, 1999).

SIMAN models are described in the model frame file and the experimental frame file (Davis & Pegden, 1988). The model frame contains the simulation program that describes the logical interaction of the simulation's entities. SIMAN models were originally described using block diagram flowgraphs that sequenced blocks (Pegden, 1983). Block types were associated with different functions and were described by their operands.

SIMAN is a block-structured language and SIMAN programs are listings of blocks with associated parameters (Pidd, 2002). SIMAN models can be entered using block and element statements (Banks, 1996).

Arena is a software package that supports the execution of SIMAN models. Arena makes it possible to use SIMAN as part of a VIMS (Pidd, 2002). Some research has looked at the viability of reposing SIMAN models on the Web (Guru, Savory, & Williams, 2000).

2.2.5.4 SLAM / Visual SLAM

The Simulation Language for Alternative Modeling (SLAM) supports process oriented and event-scheduling world views (Pritsker, O'Reilly, & LaVal, 1999). SLAM models can be described graphically with network diagrams that have "nodes" and "branches" (Cassandras & Lafortune, 1999). SLAM II process orientation models are represented using network models of a process (Pritsker, 1986). The diagrams consist of nodes and branches that represent elements such as queues, servers, and decision points. Entities flow through the network model when it is simulated.

Visual SLAM models are described with network or flow diagrams that graphically present the flow of entities through a system. Visual SLAM networks have nodes where processing is performed. The nodes are connected by activities that define entity routines and associated time requirements for performing the operations. Statements are the input associated with graphic Visual SLAM models.

The network diagrams/models are converted into statements by the AweSim software. Visual SLAM models can be executed using the AweSIM simulation problem-solving environment (O'Reilly, 2002).

2.2.5.5 DES Process Interaction Language Summary

Model interchange can be enabled by effectively defining a superset language of all the systems whose interchange is desired. Process interaction model representations share certain functionality:

- creation of entities,

- branching,
- manipulation of entity attributes, and
- elimination of entities.

They also typically have a high-level graphical programming metaphor and a general purpose scripting language for more flexibility/control.

2.2.6 Process Representations

Simulation languages are based on modeling formalisms or representation approaches. Graphical representations support the visualization of models for a certain segment of users to whom a "picture is worth a thousand words". Graphical representation involves associating icons with statement types and representing control flow with arcs and nodes. Graphical representations of systems are closer to users' mental models – resulting in more efficient manipulation and better detection of errors (Nielsen, 1991).

The graphical representations of simulation languages are often serialized into textual statements that are used to interchange model datafiles. They are used to represent:

- business process representations,
- military process representations,
- general purpose software
- Web-services representations, and
- process interaction discrete event simulation models,

Oscarsson and Moris (2002) identify several criterions that should be supported by a model representation approach:

- neutral notation,
- generic notation,
- recognized notation,
- user friendly,
- descriptive in several levels, and

- supported by in-house competence.

The following sections describe methods for visualizing simulation models using graphical representations.

2.2.6.1 Business Process Representation / Process Modeling

Business process and workflows are directly related to the process interaction world view in DES. Business processes are described with a variety of languages and associated graphical representations. Menzel and Gruninger (2001) describe process modeling as the linguistic, diagrammatic, or numerical representation of patterns of activities (processes). Business process representation/modeling approaches include:

- task networks,
- Business Process Modeling Notation (BPMN),
- Workflow Process Definition Language (WPDL),
- Process Specification Language (PSL),
- process specification graphs, and
- block diagrams.

A task network is a collection of nodes and paths that represent the flow of work (Belanger, 1994). Examples of task networks include Program Evaluation and Review Technique (PERT) charts and the Precedence Diagramming Method (PDM). Task network modeling can be used to extend function and task analyses to support predictive models of human performance (Laughery, 1998).

The Business Process Modeling Notation (BPMN) was developed by the Business Process Modeling Initiative (BMPI). BPMN notation supports pools/lanes, events/activities,

sequence/message flows, and model message/control (Nainani, 2005). One of the stated purposes of BPMN is to support the simulation of process models.

The Workflow Management Coalition (WfMC) has specified a textual grammar for interchanging process definitions called the Workflow Process Definition Language (WPDL) (WfMC, 1999). The WfMC standardized many of the process oriented terms that apply to process interaction DES.

The Process Specification Language (PSL) is a language for describing processes (Menzel and Gruninger, 2001). PSL is an interchange format designed to help exchange process information automatically among a wide variety of applications including process modeling tools. PSL is defined with first-order logic using the Knowledge Interchange Format (KIF). The specification formalizes the "Activity" concept that represents behavior specifications and the "Occurrence" concept that represents a runtime execution of an "Activity" (Bock and Gruninger, 2005). PSL process concepts have been mapped to XML and objects used in processes can be represented using the Resource Description Framework (RDF) (Lubell, 2001).

Process specifications can be graphically represented with process specification graphs (Menzel & Gruninger, 2001). A process specification graph is a directed graph that makes the graphical structure of a process description in a description's component declarations explicit.

Block diagrams and process networks use flowchart diagrams that show the movement of entities through various system operations (Praehofer & Pree, 1993). The purpose of block-structured systems is to enable non-programmers to develop discrete event simulation models using flowcharting symbols (Pidd, 2002).

2.2.6.2 Military Operations Representations

The military community is a large consumer of simulation technology. Military activities can be considered a special type of business process. Military users employ a variety of techniques for representing military activities including operational templates, the Battle Management Language, and the IDEF family of standards.

Military operations can be described with operations templates. These templates have three views: temporal, spatial, and informational (Joint Warfighting Center, 1997). The temporal view provides a graphical representation of the sequencing of activities. The spatial view shows the geographic locations of entities. The informational view shows how information is input by activities that create outputs used by other activities. Multiple views or perspectives are often necessary for describing processes.

Another method for describing military operations is with the Battle Management Language (BML) (Hieb, Pullent, Sudnikovich, & Tolk, 2004) (Carey, Kleiner, Hieb, & Brown, 2002a) (Carey, Kleiner, Hieb, & Brown, 2002b). BML defines a consistent language for representing military tasks, actions, and missions. Computer-generated forces (CGFs) are used in military simulations to represent opposing and flanking forces (Pew & Maver, 1998). Various efforts have focused on standardizing the descriptions of CGF behaviors. Fineberg (1995) developed a taxonomy of verbs for use in standardizing and organizing CGF behavior descriptions.

The US Air Force developed a set of Integration DEFinition (IDEF) methods for describing perspectives of enterprises (Whitman, Huff, & Presley, 1997). The Integrated DEFinition (IDEF) methodology is a family of standard methods originally intended for use in

systems engineering (Hanrahan, 1995). IDEF0 is a functional modeling technique used for modeling business functions and activities that support functional/activity modeling. IDEF1X is used to describe data models. The IDEF2 method was intended for dynamic modeling, but has been supplanted by commercial simulation tool approaches. IDEF3 supports process description capture (Mayer, Menzel, & Mayer, 1991). IDEF3 has been used as a vendor-neutral process language to demonstrate interchanging process information between discrete event simulation models, scheduling models, and cost models (Benjamin, Akella, Malek, & Fernandes, 2005). Both the IDEF0 and the IDEF3 approaches utilize decomposition which supports the description of hierarchical models. IDEF3 captures relationships between situations and events (KBSI, 2005). The IDEF5 ontology capture method was developed for collecting knowledge about physical and conceptual objects and their associations (Liles & Presley, 1996).

2.2.6.3 Graphical Representations of Software Applications

A variety of graphical representations are used to describe software design. The United States military has defined a collection of artifacts in their DoD Architectural Framework (DoDAF). Several techniques (e.g., DoDAF, military operations views, IDEF) recognize the need for multiple perspectives to provide a complete view of a system. Kreutzer (1986) points out that an advantage of a graphical representation of a simulation model is the emphasis on structural connectivity and symmetry. He also states that graphical representations provide a rich syntax for visually defining concepts such as links, flows, and direction. Graphical representations used to describe software include state transition diagrams and the Unified Modeling Language.

Some software behaviors can be described using state transition diagrams (STDs). STDs identify states and the conditions that result in transitions to new states. There is a subtle difference between an entity's state (the value of one or more of the entity's attributes) and the sequence of process steps that an entity proceeds through during its lifetime.

The Unified Modeling Language (UML) is a collection of Object Management Group (OMG) standards that are used to represent software designs. UML 2.0 activity diagrams (ADs) can be used to represent processes. The UML version 2 (UML 2) activity models follow traditional control and data flow approaches (Bock, 2003). Activities are behaviors that are factored into actions (Pilone, 2005).

UML has been proposed for representing simulation conceptual models and Knowledge Acquisition/Knowledge Engineering (KA/KE) artifacts (Risner, Porter, Lacy, O'Brien, & Kollmorgen,1998). Research has been conducted on automatically transforming UML-specified software designs into simulations (Arief & Speirs, 2000). Research has also been conducted on translating UML models into Abstract State Machines (ASMs) that can be simulated (Cavarra, Riccobene, & Scandurra, 2004).

The semantics of UML 2.0 activities appear to support the control flow behavior provided by Petri-nets (Storrle, 2005). Although UML 2.0 ADs can be used to describe processes, there are some expressiveness issues (Russell, van der Aalst, ter Hofstede, & Wohed, 2006) (Vitolins & Kalnins, 2005). UML class diagrams can also be augmented by color-coded archetypes. UML models can be interchanged using the XML Metadata Interchange (XMI) standard.

2.2.6.4 Web Services Representations

Representation schemes have been developed to describe the processes supported by Web services, a specific type of software. Web services representation languages include:

- Business Process Execution Language for Web Services (BPEL4WS),
- OWL Services (OWL-S), and
- Web Service Definition Language (WSDL).

The Business Process Execution Language for Web Services (BPEL4WS) is a language for describing business processes (Andrews et al, 2003). BPEL4WS business processes describe the flow and sequence of tasks and the data they share. BPEL4WS is a workflow language that can be used for process modeling. BPEL4WS's model and grammar are used to formally specify business process and business interaction protocols. The language is being evolved into the Web Services Business Process Execution Language (WS-BPEL).

OWL-S is the collection of Web standards that describe OWL ontologies designed to support Web Services. OWL-S provides constructs for describing Web services' properties and capabilities to facilitate the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring. OWL-S is described using OWL ontologies. The OWL-S ontology is used to describe what the service provides clients, how it is used, and how interactions occur. The Web service use description is supported by a process model that is captured by the OWL-S ServiceModel ontology. The OWL-S process model was designed to support simulations of Web services and its developers claim that it is a superset of the constructs typically found in process modeling and workflow languages (Sycara, Martin, McGuinness, McIlraith, & Paolucci, 2004). OWL-S process models could support the automatic verification of Web services through simulation (Ankolekar,

Paolucci, & Sycara, 2004). Business process modeling formalisms have been successfully mapped to OWL-S (Guo, Chen-Burger, & Robertson, 2004). OWL-S has also been used as an upper ontology for services in order to describe military missions and tasks (Mili & Ghanekar, 2005).

The Web Service Definition Language (WSDL) is an XML format for describing the public interfaces of Web services. XLANG is an extension of WSDL that provides a notation for the specification of message exchange behavior among participating Web services (Thatte, 2001). It describes both the model of an orchestration of services as well as collaboration contracts between orchestrations.

2.2.6.5 Process Interaction Modeling Representations

A variety of graphical representation techniques have been developed for describing process interaction models for simulation. These techniques include:

- SIMULA Activity Diagrams,
- Control flow graphs,
- Petri nets,
- Activity cycle diagrams, and
- Process Network diagrams.

Miller and Fishwick (2004) describe activity diagrams as graphs with well-defined functional nodes (e.g., start, terminate, delay, engage resource, and release resource). SIMULA activity diagrams depict the flow of entities and resources through a modeled system.

Cota and Sargent (1992) and Cota, Fritz, and Sargent (1994) describe control flow graphs as a graphical representation of process behavior. Control flow graphs represent models as directed graphs with nodes depicting model states and edges depicting event transitions. Control

flow graph vertices represent possible control states. Arrows leaving a state represent a guard and identify the next event. Control flow messages are sent to channels instead of directly to processes. Channels act as First In First Out (FIFO) queues for messages.

Petri nets are defined by specifying the Petri net graph/structure and adjoining the graph with an initial state, marked state, and a transition labeling function (Cassandras & Lafortune, 1999). A simple Petri net is a graph with place vertex labels and instantaneous vertex labels (Schruben, 1992). Miller and Fishwick (2004) describe Petri Nets (PN) as graphs with transition and place nodes. Arcs connect the nodes. Transitions "fire" if sufficient tokens populate each input place. Timed Petri Nets (TPNs) have delays associated with their transitions. Petri nets can support the process interaction world view (Miller & Fishwick, 2004). Petri nets explicitly represent DES transition functions (Cassandras & Lafortune, 1999). Petri nets can be formally defined and used with similarly formally defined DEVS (Bobeanu, Kerckoffs, and Van Landeghem, 2004). Bobeanu, Kerckoffs, and Van Landeghem, (2004) describe a systematic approach for implementing discrete event systems using Petri nets.

Activity cycle diagrams (ACDs) are primarily associated with activity scanning, but can also support the process interaction world view (Miller & Fishwick, 2004). ACDs can model entity interactions (Pidd, 2002). An ACD describes the progression of activity and queue states that entities pass through (Clementson, 1986). Miller and Fishwick (2004) describe ACDs as graphs with activity (active state) nodes and wait (dead state) nodes connected by arcs. ACDs depict the lifecycles of interacting entities flowing through a system. ACDs can be considered an extension of Petri Nets (Clementson, 1986). Hierarchical Activity Cycle Diagrams (HACDs) are variants of ACDs (Odhabi, Paul, & Macredie, 1998). A simplified version of HACDs have

been defined - called Simplified Hierarchical ACD (SH-ACD with associated icons for graphical representations (Odhabi, Paul, & Macredie, 1998).

Process network modeling is a popular approach for modeling discrete event systems (Schruben, 1992). Miller and Fishwick (2004) describe network (or block) diagrams (in the simulation context) as a class of diagrams similar to activity diagrams, but with more types of nodes corresponding to their associated languages' primitives. The associated languages include GPSS, SLAM, and SIMAN.

Common themes emerge from reviewing the various representations. Most of these representations use graphical representations of nodes that are related with arcs to indicate control flow.

2.2.7 Formal DES Semantics

Formal definitions are required to support the explicit semantics of a graphical representation. Static model representations provide potential for migrating representation features into simulation model descriptions (Whitman, Huff, & Presley, 1997). A variety of formalisms have been developed to represent discrete event models. Ziegler, Praehofer, and Kim, 2000) provide a formal description of various types of discrete event simulations. His approach is mathematically complete, but difficult for modelers to relate to. Zeigler's formalism has been extended by others (Barros 1995). Narain (1991) defined an axiomatic basis for general discrete event modeling.

2.3 Interchanging Simulation Information

Simulations have associated data that can be considered part of a model or it's associated experimental frame. This data is often interchanged between simulation systems and a variety of techniques have been developed to support simulation data interchange. Most simulation interoperability research has focused on the runtime interchange of information to support distributed interactive simulations for the military.

2.3.1 Simulation Information Interchange Motivation and Requirements

The motivation for data interchange and interoperability includes the desire for improved system quality, reduced development cycles, and reduced development costs. System quality can be improved by reusing validated models and data. Cycle times and development costs can be reduced by reusing existing information rather than generating new information. An early system that automatically generated SIMAN models from facility planning software was motivated by the desire to reduce model development time and improve model quality (Ingalls, 1986).

2.3.2 Simulation Information Representation

Information must be represented to support interchange. Sheehan (2001) points out the need for common semantics and syntax for interchanging simulation data. He states that canonical representations are the most useful and have the most structural syntax maturity as well as semantic content control (see Figure 2).

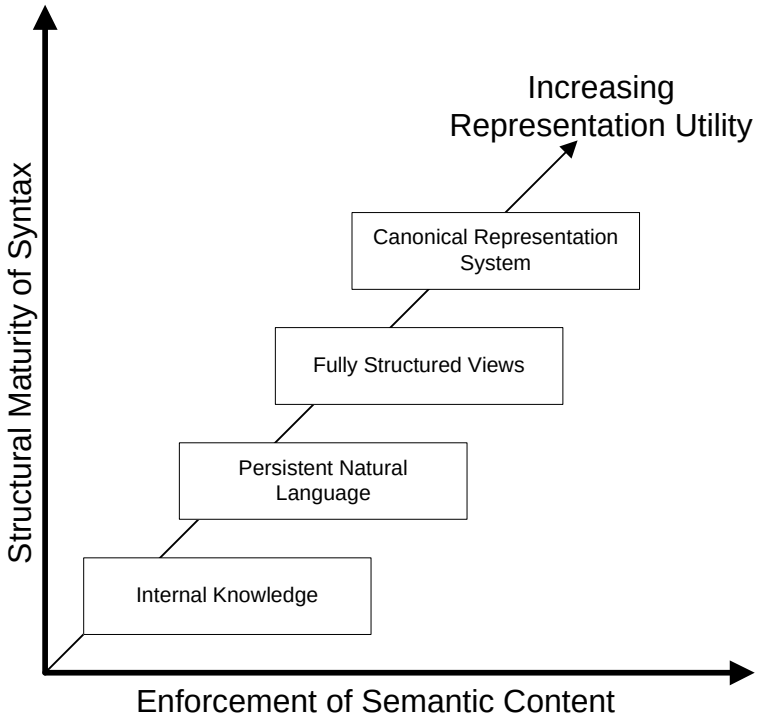

Figure 2. Semantic Control of Representations

Describing quality data interfaces requires proper data representation support. Knowledge Representation (KR) applies theories and techniques from the fields of logic, ontology, and computation (Sowa, 2000).

The military has invested significant resources to research the development of Conceptual Models of the Mission Space (CMMS). CMMS are simulation-implementation-independent descriptions of processes, entities, and the environment (Sheehan, Prosser, Conley, Stone, Yentz, & Morrow, 1998). While much of the focus has been on models in simulations, formal methods of describing the data used by simulations has also been investigated (Roberts, 1991).

2.3.3 Simulation Data Interchange Formats

The use of an intermediate neutral data interchange format (DIF) reduces the number of interfaces (and associated converters) between N systems from $O(N^2-N)$ to O(2N-1) (Benjamin, Akella, Malek, & Fernandes, 2005). Leveraging DIFs requires legacy systems to generate a DIF "view" of the system's data model. One approach is to directly create a DIF view. Another approach is to create an XML view of the legacy data store and then convert from the legacy systems associated XML format to the DIF format.

Simulation DIFs define how data will be exchanged between applications (Gravitz, Sheehan, and McLean, 1999). A DIF is a formal specification of the structure and format of data interchanged between producers and consumers of data. DIFs should define the syntax and semantics of the interchanged data (Sheehan, 2001).

National Institute of Standards and Technology (NIST) researchers developed libraries of formal, neutral models of simulation components (Son, Jones, and Wysk, 2000, 2003). Express is a model specification language (Schenck and Wilson, 1994). The XML Metadata Interchange (XMI) specification can be used for interchanging models described using the Unified Modeling Language (UML).

Competing definitions have been proffered for the concept of conceptual models (Lacy, Randolph, Harris, Youngblood, Sheehan, Might, & Metz, 2001). Simulation conceptual models can be defined as a developer's method of translating modeling requirements into a detailed design framework for a simulation (Pace, 2001). The development of conceptual models is a key phase in the development of a simulation (Lacy & O'Brien, 1997) (Risner, Porter, Lacy, O'Brien, & Kollmorgen, 1998).

The idea of representing simulation models in a formal manner is not a new concept. Overstreet & Nance (1985) described the concept of a formal Simulation Model Specification and Documentation Language (SMSDL). They recognized the reduction in modeling costs and the improvement in quality that could result from interposing an intermediate form between a conceptual model and an executable representation of the model.

2.3.4 XML Simulation DIFs

Some vendors (e.g., XJ Technologies) have recognized the benefits of XML for data interchange and are using it for natively representing their simulation models (Filippov, 2003). Technologies such as the High Level Architecture (HLA) include interoperability standards. However, interoperability is also important in an off-line mode. Neutral, open standards are needed to define the syntax for interchanging data during the development of simulations. These simulation interchange requirements led to the use of the Extensible Markup Language (XML) (Lacy & Tuttle, 1998).

Domain-specific XML Data Interchange Formats (DIFs) support information representations that are platform and machine-independent (Miller & Fishwick, 2004). A DIF's XML element name can be based on concepts formalized in a domain ontology (Miller & Fishwick, 2004).

Research has been performed to demonstrate the use of XML for interchanging data to support discrete event simulations (Harrison, Maynard, & Pollak, 2004). NIST has developed the Shop Data Model neutral file format for interchanging information that supports discrete

event simulations of manufacturing type facilities. The format has been mapped to ProModel's internal structures to determine compatibility (Harward, 2005).

XML has become a widely-popular metalanguage for defining file formats. An early use of XML was as a mechanism for interchanging simulation data (Lacy & Tuttle, 1998) (Gravitz, Sheehan, & McLean, 1999). Early work was performed involving developing XML DIFs for simulation scenario data (Lacy, Stone, and Dugone, 1999a) and CMMS information (Lacy, Stone, and Dugone, 1999b). Examples of XML simulation DIFs include scenario DIFs and equipment characteristics and performance data (Lacy & Dugone, 2001a) (Lacy, Dugone, and Youngren, 2001). Examples of XML military simulation DIFs include the Unit Order of Battle (UOB) DIF and the CMMS DIF (Gravitz, Sheehan, & McLean, 1999). XML has been proposed for describing the behaviors of computer-generated forces in military simulations (Lacy, Stone, & Dugone, 2001) (Lacy & Dugone, 2000b). The U.S. Army's OneSAF Objective System (OOS) uses XML extensively for interchanging simulation data including composable behaviors (DaCosta, 2002) (DaCosta, Lucas, Outar, & Helton, 2003). The XML instance files that conform to a DIF are sometimes referred to as XML Populated DIFs (XPoDs).

In addition to domain-specific XML DIFs, simulation DIFs associated with simulation techniques have been developed. XML-based model interchange formats have been developed for sharing Petri Nets including the Petri Net Markup Language (PNML) (Syrjakow, Syrjakow, & Szczerbicka, 2002). The OpenModel Modeling Language (OMML) is an XML-based model interchange format for representing behavioral models (Hall and Zisman, 2004a, 2004b). OMML is a procedural language for expressing functionality in terms of function/object theories. The Defense Modeling and Simulation Office (DMSO) considered an ontological alternative to XML DIFs for equipment descriptions (Lacy, 2001).

2.3.5 XML-based Simulation Interoperability Standards

One challenge in developing XML DIFs for simulation data interoperability has been the standardization process. The Simulation Interoperability Standards Organization (SISO) has led the development of various XML-based standards to support simulation interoperability.

Data Interchange Formats are used to interchange a variety of data associated with the High Level Architecture (HLA) approach to distributed simulation. The HLA Object Model Template (OMT) Specification (IEEE P1516.2) specifies the objects, attributes, interactions, and parameters that are required for an HLA Simulation Object Model (SOM). The OMT Data Interchange Format (DIF) is an XML simulation DIF that structures HLA OMT descriptions for use by automated tools (Hobbs, 2003).

A scenario narrative ontology was used to create an XML grammar called the Scenario Markup Language (SCML). XML documents described with SCML are called hyperscenarios (Hobbs, 2003). The High Level Architecture (HLA) Dynamic Scenario Builder (DSB) research effort promoted the use of XML for interchanging scenario data (Lacy, Stone, & Dugone, 1999a).

The Extensible Modeling and Simulation Framework (XMSF) is a composable set of standards, profiles and recommended practices for Web-based modeling & simulation (M&S) (Brutzmann, Zyda, Pullen, & Morse, 2002). XMSF leverages Web technologies to extend systems interoperability by enabling simulations to interact over highly distributed networks. XMSF includes Web, internet and XML technologies for open interoperability in M&S. Ontologies represent one of the XMSF functional requirements categories. An ontology for sharing discrete event simulations could be one of the XMSF standards. The XMSF group

identified RDF and the DARPA Agent Markup Language (DAML) (which later evolved into OWL) as semantic representations of particular interest.

The Simulation Interoperability Standards Organization (SISO) is developing a standard based on the Simulation Reference Markup Language (SRML). SRML was developed to describe the structure and behavior of simulations models using XML and was documented in a W3C note (Reichenthal, 2002). A process modeling case study demonstrated SRML features (Reichenthal, 2004). SRML provides a format for representing the behavior of encapsulated Base Object Models (BOMs). BOMs represent reusable simulation interaction patterns and components that support the description of HLA SOMs and Federation Object Models (FOMs). The SRML XML schema defines object-oriented elements for implementing identity, modularity, classes, associations, behavior, communication, inheritance, polymorphism, and extensibility. SRML could serve as the basis for a more formal description of simulation models (Fishwick & Miller, 2004). Alternatively, SRML could be upgraded with new information representation technologies and extended to formally support its description mechanisms for DES models (Lacy, 2006).

2.4 OWL Ontological Representations of Simulation Information

While XML DIFs addressed the syntax aspect of simulation data interchange, the challenge of semantic representation remained. This challenge was very similar to the challenge of the HTML-based current World Wide Web (WWW). A new set of technologies is enabling the evolution of the current Web into a Semantic Web. The Semantic Web is empowered by formal ontologies that are encoded using the Web Ontology Language – OWL.

2.4.1 Current Web

The Semantic Web represents a new evolution of the current Web. The current Web is dominated by files encoded with the Hypertext Markup Language (HTML). It supports human readers with Web browsers (e.g., Internet Explorer). Standardizing protocols (e.g., TCP/IP) and languages such as HTML and the Extensible Markup Language (XML) enabled the Web by supporting interoperability at various levels of the Web's layered network architecture. This approach is convenient for human consumption, but difficult for computers to process. However, the current Web provides insufficient structure to support efficient computer processing of content. Computers require structured information to support efficient unambiguous interpretation. Semantic Web techniques provide explicit descriptions of information's semantics.

2.4.2 Ontologies

Ontologies provide a shared and common understanding of a domain to facilitate knowledge sharing and reuse (Fensel, 1998). Ontologies explicitly describe the semantics of compliant information. Gruber (1993) succinctly defines an ontology as a "formal specification of a conceptualization." Computer scientists typically use the term to describe references to formal descriptions of a domain in order to support knowledge sharing and reuse. Ontologies in computer science describe information sources with collections of terms and their relationships. McGuiness (2002) described a spectrum of methods for supporting knowledge representation in terms of their sophistication. She identifies ontological representations as those that have:

- A finite controlled (extensible) vocabulary,

- An unambiguous interpretation of classes and term relationships,
- Strict hierarchical subclass relationships between classes, and
- An ability to support inferencing.

Daconta, Obrst, and Smith (2003) presents a similar continuum of "smart data" whose positive extreme is described by XML ontologies and automated reasoning. Ontologies support a common understanding by humans and software agents of the information associated with the domain. Common understandings help reduce misinterpretation of information.

Semantic Web ontologies encoded using OWL provide a means to define classes, properties, individuals, and relationships between them. An OWL ontology can be defined as a "web-distributed vocabulary of declarative formalisms describing a model of a domain" (Lacy, 2005). Just as a simulation model represents a system, an ontology is an abstraction of a domain.

Ontologies support information sharing by formally communicating a common understanding of a domain with expressive statements that provide explicit declarations of semantics. OWL-compliant software can interpret ontologies and accurately manipulate the information. Information sharing requires the use of a common language and access to the information (syntax). Applications must also have a common semantic understanding of the information for effective reuse.

Semantics formally describe terms and their relationships which support computer understanding and reduce ambiguity. Each DES language has its own semantics even though they often share concepts. There are different types of ontologies. Fensel (1998) categorizes ontologies as:

- domain ontologies,
- metadata ontologies,
- generic / common sense ontologies,
- representational ontologies, and
- method/task ontologies.

An ontology to represent discrete event simulation models would be considered a representational ontology.

Considerable emphasis in simulation development has been placed on encoding a simulation model for a particular simulation software package. However, the focus should be on encoding shareable conceptual models. Successful use of ontologies requires encoding ontologies using a language, marking up compliant instances, and using software that commits to the ontologies.

Ontologies are encoded using formal ontology languages so that software can parse them and use their explicit semantics to interpret compliant information instances. The ontologies are described using formal vocabularies of terms and their relationships. A variety of formal languages are used to encode ontologies. IDEF5, the Ontology Description Capture standard, was developed to represent ontological information as part of the IDEF family of standards. However, it never achieved the widespread use and maturity of other IDEF standards. The Web Ontology Language – OWL was developed to support the World Wide Web Consortium (W3C) concept of the Semantic Web.

2.4.3 Semantic Web

The Semantic Web is the next evolution of the World Wide Web that supports automated processing of structured information (Berners-Lee, 1999). Berners-Lee, Hendler, and Lasilla (2001) described their concept of the Semantic Web as a new form of Web content that is meaningful to computers and that will unleash a revolution of new possibilities. With the Semantic Web, the emphasis shifts from proprietary data formats to “smart data” that is

machine-processable using a neutral open representation formats based on XML (Daconta, Obrst, and Smith, 2003).

Just as with the current Web, information on the Semantic Web is marked up according to a particular language, is distributed across servers, and can be accessed by software that understands the mark up language. Unlike the current Web, Semantic Web applications are able to leverage ontologies to perform more advanced features with structured information.

Berners-Lee based his Semantic Web concept on the current Web which was enabled by the protocols he developed to support interoperability (Berners-Lee, 1999). Berners-Lee published his Semantic Web road map to document his vision for a Web of machine-understandable data, represented as Web resources (Berners-Lee, 1998).

Semantic Web technology is suitable for applications involving well understood domains, heterogeneous information sources, and information interchange requirements (Lacy, 2005). Structured information representations enable the Semantic Web with explicit semantics defined by ontologies. The W3C's concept of the Semantic Web relies on information marked up in a computer-understandable manner using the Web Ontology Language – OWL.

2.4.4 OWL

The Defense Advanced Research Products Agency (DARPA) created the DARPA Agent Markup Language (DAML) as part of its Semantic Web research effort. European Union (EU) researchers developed the Ontology Interface Layer (OIL). A joint EU/US Committee on Agent Markup Languages merged many concepts from OIL with DAML to create the DAML+OIL language (McGuiness, Fikes, Hendler, and Stein, 2002). The World Wide Web Consortium

(W3C) Web Ontology Group evolved DAML+OIL into the Web Ontology Language – OWL, which was released in February 2004. The language is documented in:

- an overview document (McGuiness & van Harmelen, 2004),
- a language guide (Smith, Welty, & McGuiness, 2004),
- a language reference (Dean, Schreiber, van Harmelen, Hendler, Horrocks, McGuiness, , Patel-Schneider, & Stein, 2004),
- test cases (Carroll & DeRoo, 2004), and
- a Semantics and Abstract Syntax document (Hayes, Horrocks, & Patel-Schneider, 2004).

The Web Ontology Language – OWL was developed for defining ontologies and associated individual data. Knowledge representation technologies (e.g., frame-based reasoning systems, Description Logics) influenced OWL's development. OWL statements, also called assertions, describe classes, properties, and individuals. Assertions can be stated within individual ontologies or in combinations of multiple joined ontologies. Additional facts can be derived or logically entailed using inferencing. OWL, like XML, provides an open standard for information representation. This allows compliant software to manipulate information without having to have domain-specific knowledge (Meeks, Aviles, & Lacy, 2004).

OWL's developers designed language features in layers that build on open W3C Web standards. Tim Berners-Lee defined an initial layered architecture view of the Semantic Web (Berners-Lee, 2000). Various alternative views of Semantic Web technology layers have since been developed.

Figure 3 presents an alternative layered conceptual view of Semantic Web technologies from an OWL perspective (Lacy, 2005). The layers are not strict layers in the networking model sense, but do illustrate extensions of features since each layer depends on the layers beneath and uses their features to provide its capability. The implementation layer at the top of the figure supports specific applications. The logical layer supports formal semantics and reasoning using

OWL. The Resource Description Framework (RDF) Schema (RDFS) language is used to define vocabularies using the ontological primitives layer. RDFS and individuals are specified using RDF, which provides the basic relational layer with its consistent approach for using XML and XML Schema (XMLS) datatypes in the transport/syntax layer. The symbolic/reference layer uses Uniform Resource Identifiers (URIs) and XML namespaces.

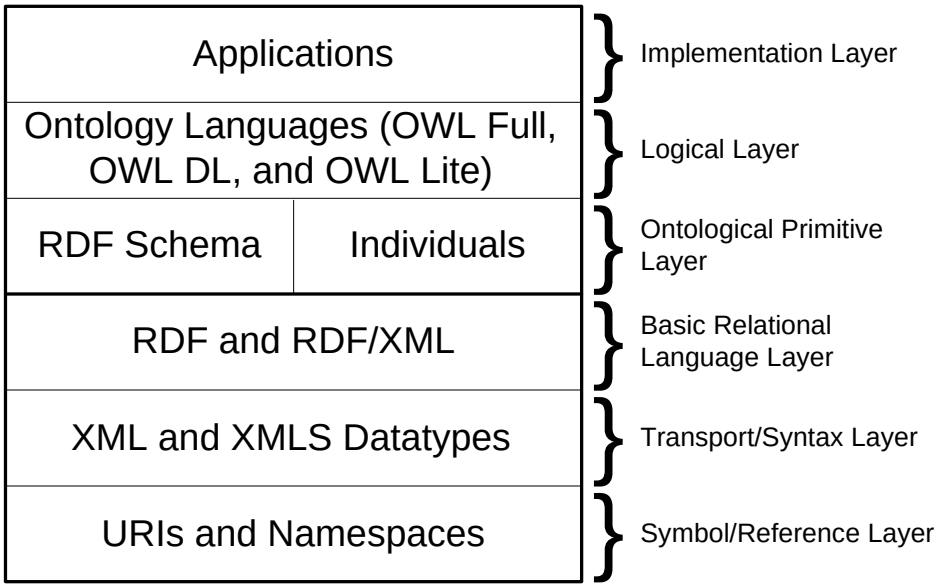

Figure 3. Semantic Web Technology Layers

Additional layers may be required to provide sufficient expressiveness. Fishwick (2004) states that complex ontologies suitable for modeling and simulation will require a combination of OWL and the Semantic Web Rule Language (SWRL).

2.4.5 Ontology Engineering Processes

The successful use of an ontology depends on the quality of the ontology engineered to support a Semantic Web application. Ontology engineering processes are often based on software engineering processes and share concepts with DIF development.

2.4.5.1 DIF Development Process

Ontology development is also similar to DIF development activities. Steps used by the military for defining DIFs are:

- Identify the need for a DIF and a strategy for managing the DIF,
- Develop a logical data model (schema) and specify use cases,
- Determine and build the physical representation of the DIF,
- Determine DIF definition style and build physical DIF,
- Package the DIF,
- Post and review DIF, and
- Publish and maintain the DIF (Gravitz, Sheehan, and McLean, 1999).

Lacy & Dugone (2000a) described a simulation DIF development process that included:

- Defining data requirements,
- Developing logical data models,
- Evolving data models into XML Document Type Descriptions (DTDs),
- Demonstrating and testing the XML DTDs, and
- Documenting and evolving the resulting standard.

2.4.5.2 Ontology Engineering

OWL ontologies are best developed using mature documented ontology engineering techniques. Gomez-Perez, Fernandez-Lopez, and Corcho (2004) surveyed several approaches to ontology development. Many of the development processes appear to share an approach involving specification, conceptualization, formalization, and implementation phases.

Ontologies should be developed by leveraging existing domain knowledge (Lacy, 2005). Noy and McGuiness (2001) recommend the following ontology development steps:

- Determine the domain and scope of the ontology,
- Consider reusing existing ontologies,
- Enumerate important terms in the ontology,
- Define the classes and the class hierarchy,
- Define the properties of classes – slots,
- Define the facets of the slots, and
- Create instances.

2.4.6 Simulation Ontologies

Lacy and Dugone (2000a) and Lacy and Gerber (2004) identified OWL's predecessor language – DAML as part of a potential emerging ontology solution for interchanging simulation information. Blais and Lacy (2004) describe the potential for Semantic Web technologies to support the M&S domain by dramatically improving composability of functional capabilities and the interoperability of systems. Ontologies can be used for a variety of modeling and simulation information representations including static authoritative domain descriptions, simulation development and composition, dynamic data representation, and CGF behaviors (Lacy, Stone, and Dugone, 2001). Ontologies have been used to define several ontologies to support simulation-related applications including simulation objects, CGF behaviors, and discrete event models.

OWL has been used to describe military equipment in support of distributed interactive simulations. The taxonomy of equipment described in the Distributed Interactive Simulation (DIS) enumeration document was evolved into an OWL ontology that can be used to map information from other military equipment databases to simulation applications (Lacy, 2004).

The rube™ project is using OWL to encode an ontology that describes the geometry and dynamics knowledge of objects in a simulated air battle scene (Fishwick, 2004).

Separating the description of CGF behaviors out of code and into data has been a goal for some time (Lacy and Henninger, 2003b). Early efforts used XML, but more recently, OWL has been used to demonstrate how simulated behaviors could be represented (Gerber and Lacy, 2004a) (Gerber and Lacy, 2004b).

OWL has been used to encode the Discrete Event Modeling Ontology (DeMO) (Miller & Fishwick, 2004) (Fishwick, 2004) (Miller & Baramidze, 2005). The DeMO prototype was developed to support an investigation of ontology development issues. DeMO's goal was to formally define foundational concepts for extension by future ontologies. Process interaction models in DeMO are represented by the "Process-Oriented Model" class. Specific classes (e.g., "GPSS Block Diagram") are then defined as subclasses to the "Process-Oriented Model" class. DeMO has been extended to support the Process Interaction world view using OWL and with an XML Schema language called the Extensible Process Interaction Markup (XPIM) language (Miller, Silver, & Lacy, 2006).

2.5 Representing DES Models with Ontologies

Process interaction DES models can be represented and interchanged with the help of ontologies. Developing a formal ontology for representing DES process interaction models effectively involves defining a new simulation language. The use of OWL for defining an ontology for DES has been recommended (Seila, 2005) and theorized (Lacy, 2001) (Fishwick & Miller, 2004). Advantages of representing DES models with ontologies include making models

processable by Semantic Web-compliant software. A standard for sharing DES models should include an ontology, an XML-based interchange language, and a graphical representation (Seila, 2005). The design should also leverage existing formalisms and take into consideration existing tools.

2.6 Background Literature Summary

Discrete event simulations represent a commonly used type of simulation. The process interaction world view is a popular paradigm for representing discrete event simulations. Ideas for representing process interaction world view models can be derived from a variety of techniques associated with modeling processes, describing software, describing Web services, simulation languages, and simulation software. Simulation-related information can be interchanged using a variety of techniques including XML-based Data Interchange Formats. The Web Ontology Language – OWL can also be used to interchange simulation data and models. An OWL ontology can be developed to describe process interaction world view models for discrete event simulations.

3 CHAPTER THREE: METHODOLOGY

PIMODES research focused on developing a new open language for describing process interaction Discrete Event Simulation models. An OWL ontology formalized the new language. Software tools supported the execution of documented procedures for conducting the PIMODES research process. The research has known limitations associated with the concept and the implementation approach.

3.1 Instrumentation

A variety of software tools supported PIMODES research activities. Software-supported activities included reviewing legacy simulation applications, exchanging data, developing the PIMODES ontology, and developing software. The software tools used are listed in Table 1 along with the vendors of the tools and the role of the tools in PIMODES research.

Table 1. Software Tools Employed in PIMODES Research

Category	Tool	Vendor	Purpose / Role
Commercial Discrete Event Simulation software packages	Arena®	Rockwell Automation	Authoring sample Arena models, model representation analysis
	ProcessModel®	ProcessModel	Authoring sample ProcessModel models, model representation analysis
	AnyLogic™	XY Logic	Authoring sample AnyLogic models, model representation analysis
	ProModel®	ProModel Corporation	Authoring sample ProModel models, model representation analysis
Commercial file format manipulation	Access™	Microsoft	Reviewing Arena export file
	Microsoft® Office Excel 2003	Microsoft	Reviewing and accessing ProcessModel export files
Ontology design	Microsoft® Visio®	Microsoft	Drawing UML-style class diagrams of the ontology design
Ontology encoding	SemanticWorks™	Altova	Editing and validating ontologies.
	Protégé	Stanford University	Creating and editing ontologies
	DOAT	DRC	Creating ontologies from a database structure
Instance file encoding	XMLSpy®	Altova	Validating instance files
Ontology output formatting	DumpOnt	BBN	Formatted presentation of ontologies
Software Development	Visual Basic®	Microsoft	Exporting data from Arena and converting ProcessModel data from MS Excel to RDF/XML
Documentation	Microsoft® Word 2003	Microsoft	Performing word processing
	Acrobat®	Adobe	Translating document formats
	Visio®	Microsoft	Editing figures
Web Site Development	Microsoft Front Page®	Microsoft	Developing the support website

3.2 Procedures

The PIMODES research followed a structured systems engineering approach that was flexible enough to allow for innovation and creativity throughout the process. A goal of the process was to support repeatability, verification, validation, and extensibility by documenting the process and the results.

The research process included research planning, a literature search, the development of the PIMODES ontology, a demonstration of the ontology's use, and the documentation of research results. An IDEF0 activity model of the research process is shown in Figure 4. Although the model suggests a waterfall approach, the process actually involved spiral development with iterative refinement. The following sections describe each activity.

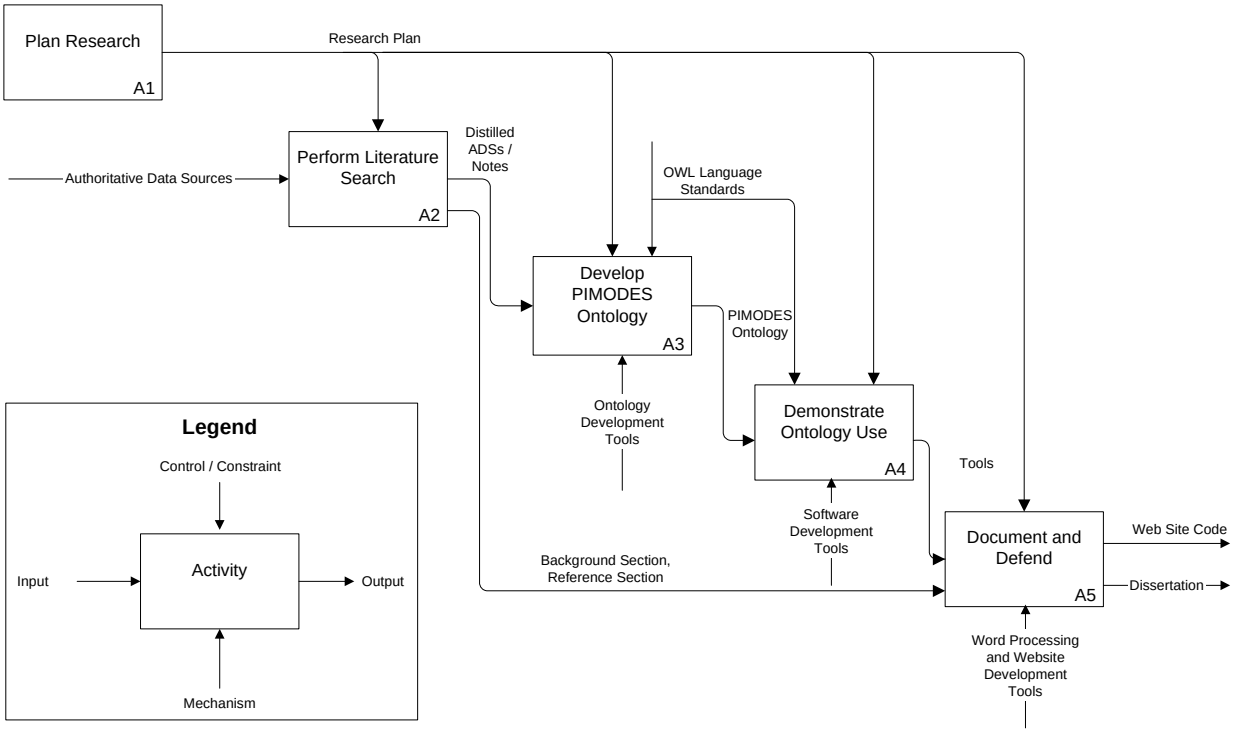

Figure 4. PIMODES Research Activity Model

3.2.1 Research Planning

The planning phase of the research included describing the research process with the activity model presented above. A product of the planning phase is a Research Plan that details the steps to be performed and the products to be produced. Identified tasks should be detailed by identifying their inputs and outputs, their relationships to other tasks, and a schedule for completing them with milestone delivery dates for the artifacts they produce.

3.2.2 Literature Search

The literature search for this effort was highly influenced by DES papers presented at the Winter Simulation Conference, books on DES, and documentation for popular DES software applications. Prior work was reviewed in the subjects of discrete event simulations, process metamodels, process languages, discrete event simulations, and simulation software. Key information gleaned from this phase was used to design the PIMODES ontology.

3.2.3 PIMODES Ontology Development

The focus of the research was the development of the Process Interaction Modeling Ontology for Discrete Event Simulation (PIMODES) language - formalized by an OWL ontology. Reviewed literature and legacy applications heavily shaped the ontology design. The PIMODES ontology development effort was similar to a software development effort. Ontologies, like software, are best developed with documented, repeatable, and mature processes. The PIMODES ontology development process steps are depicted in the activity

model in Figure 5. The diagram represents an expanded view of the "A3" activity described in the overall activity model shown in Figure 4 above.

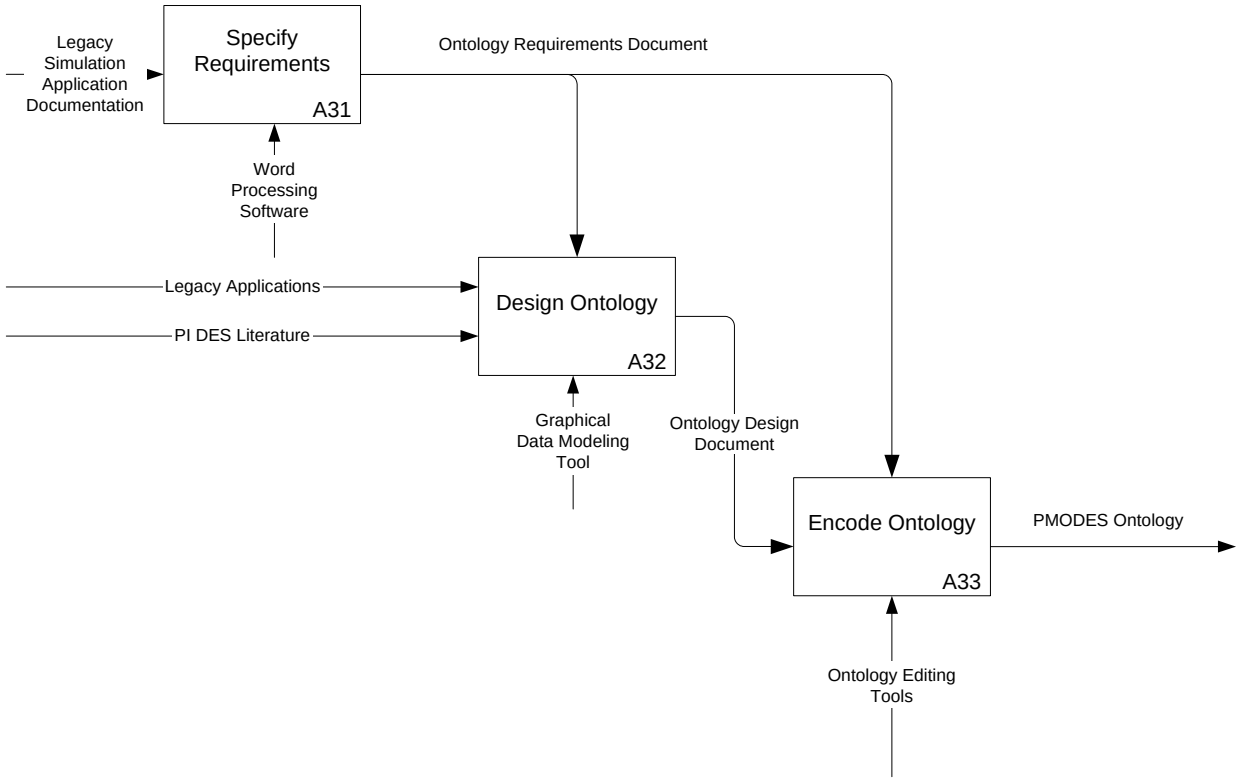

Figure 5. PIMODES Ontology Development Activities

PIMODES ontology development activities included specifying requirements, designing the ontology, and encoding the ontology. Ontology development involves an evolution and coagulation of granular ambiguous domain concepts into formal specific encoded formalisms (see Figure 6).

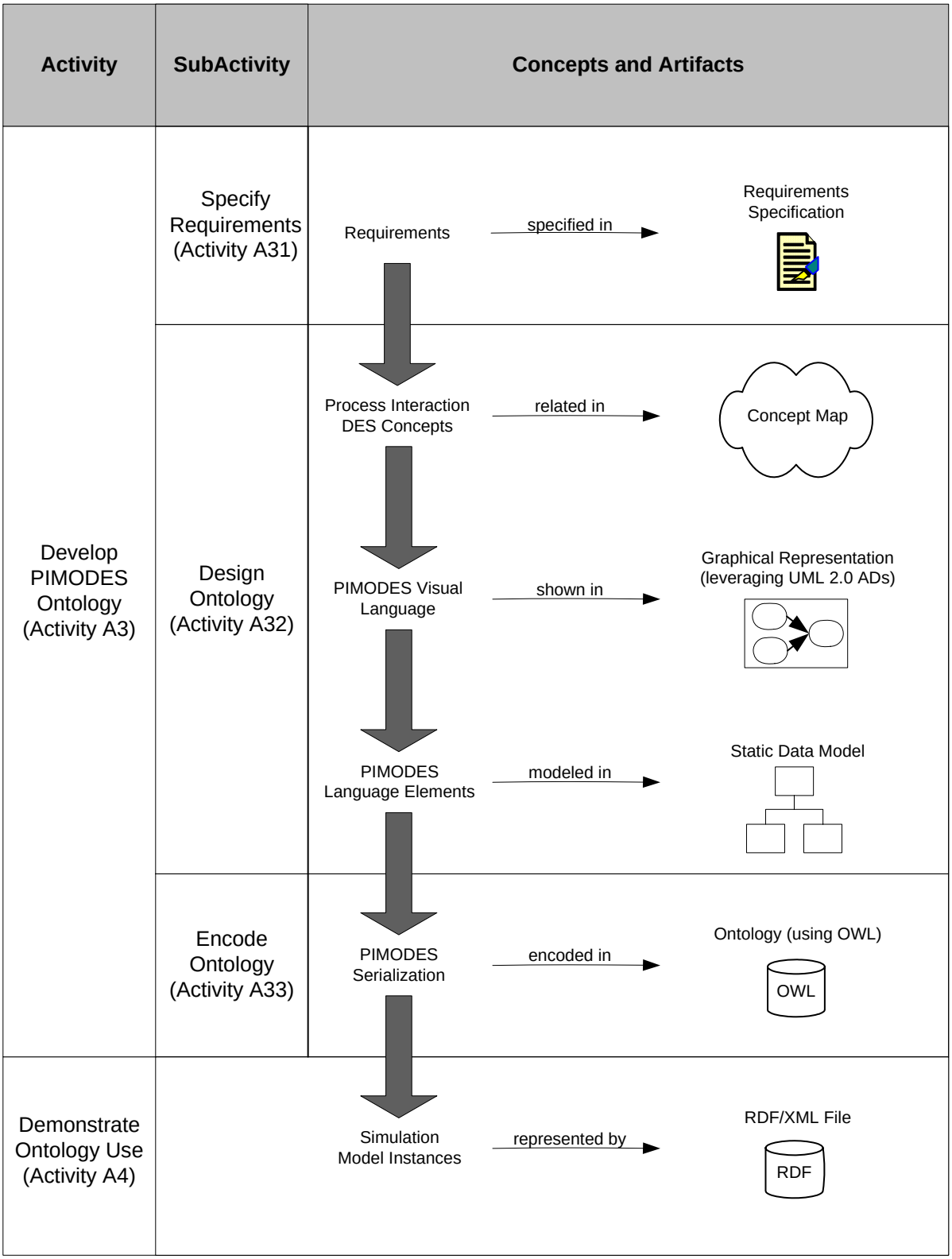

Figure 6. Ontology Concept Evolution

3.2.3.1 Specifying Requirements

The requirements analysis phase scoped the ontology and focused the effort. A comprehensive set of requirements for the research effort must specify requirements for legacy data support, the objective ontology, and the demonstration translation software. Unique requirement identifiers for each requirement help support traceability.

3.2.3.2 Designing the Ontology

Key ontology design steps included identifying harmonized DES concepts, specifying a visualization language, and identifying language elements. Ontology requirements scope the domain and help identify process interaction DES concepts. The design should document the harmonization of legacy application model representation approaches and trace back to specified requirements. The PIMODES concepts were identified using a harmonization process that considered approaches implemented by legacy applications and incorporated widely-adopted concepts from process interaction DES literature. Concept maps helped relate key concepts from the domain. The subject domain of the PIMODES ontology is a language that describes temporally related activities. Therefore, a graphical representation was a key related element. PIMODES concepts and graphical representations evolved into ontology classes with associated properties. Supported information was modeled in a static class diagram that considered object-oriented techniques (e.g., generalization).

3.2.3.3 Encoding the Ontology

The static data model from the ontology design was encoded using OWL. OWL ontology encodings must be represented in compliant datafiles that are computer readable. The preferred format for OWL files is RDF/XML. OWL ontologies can be encoded directly with a text editor, or through the use of ontology editing tools. The encoded ontological elements should include comments. A database tool (DOAT) was used to manage the class and property descriptions and automatically generate the complex OWL RDF/XML syntax.

3.2.4 Ontology Testing and Use Demonstrations

Prototype software was developed to demonstrate the feasibility of translating models from legacy simulation products into the PIMODES format (and vice versa). The software was designed, coded, and tested.

3.2.4.1 Demonstration Translation Software Design

A key software design goal was modularity to support software extensions for additional legacy applications in the future. The design process evolved requirements into a high level design. The high level design formed the basis for designing detailed data mappings describing translations of the semantics of simulation model descriptions. The modular software design helped associate support for legacy applications with specific portions of code. The high level design of the software focused on the dataflow between legacy representations and the PIMODES representation.

A detailed design process generated data models and mapping descriptions using object-oriented design principles. Data representations and translation functions were allocated to object-oriented software classes and methods. Legacy application model representation approaches were compared and contrasted with the harmonized PIMODES representation to identify required conversions. The detailed design defined mappings between the legacy application representation and the PIMODES representation. Object-oriented surrogate classes were designed to simplify the code by temporarily storing model data from legacy applications and PIMODES ontology classes during the translation process. The software design traces back to requirements to help verify the completeness of supported model data as well as the functionality of the translation software.

3.2.4.2 Software Coding

The detailed software design has evolved into software code by writing Visual Basic .NET code using Microsoft Visual Studio. Object-oriented classes were encoded as classes with supporting methods. Legacy application data access mechanisms constrained the interface code. Access routines imported legacy application data files or data objects in memory. Software routines for legacy support and PIMODES were modularized using class libraries. Mappings were implemented with assignment statements and conversion functions. The software code was traced back to requirements using comments that reference requirement numbers. A simple user interface was required to allow users to identify source and destination filenames and formats. Error messages were defined to identify unsupported items to the user during the translation process.

3.2.4.3 Demonstration and Testing Model Development and Experimentation

A set of sample models was needed to test and demonstrate the features of the translation software. Demonstration models representing sample common process interaction DES problems were developed to demonstrate interchange. Testing models exhaustively employed legacy and PIMODES language constructs to verify translations.

Conceptual models were needed for each type of demonstration model to support model designs. A version of each model was needed for each supported legacy simulation application in order to compare representations.

Investigative models were also needed to determine how various applications stored their data and to test the translation software for completeness. A demonstration script was needed to make it easy for others to repeat demonstration results.

Model translation verification involved several steps. First, the conversion process needed to execute smoothly without warnings or errors being generated. Next, the syntax of the resulting file needed to be checked. Models translated in the PIMODES format were checked in SemanticWorks with automated ties to the PIMODES ontology to verify consistency with the ontology. Models translated from PIMODES into legacy application formats were opened using the target tool to ensure that the generated files were valid. Tools such as SemanticWorks can use the PIMODES ontology to automatically identify any syntactic or semantic errors in the instance file. Lastly, a manual process verified that all of the model content was translated according to the design of the software.

3.2.5 Research Artifact Documentation

Results of the research need to be shared with other researchers and potential adopters. Research artifacts including the ontology, documentation, and source code must be hosted on a public website to provide access to the widest possible audience.

3.3 Limitations

Certain assumptions and decisions were made during the research process that resulted in limitations. PIMODES research limitations were primarily associated with the concept of developing a standard process interaction DES language and with decisions made regarding the research approach.

3.3.1 Concept Limitations

The concept of developing a standard process interaction DES language has limitations involving the approach of developing a universal language, lack of accepted formalisms for process interaction DES, dependencies on simulation application vendors, and the long term requirements of simulation developers to adopt the ontology as a standard.

3.3.1.1 Universal Language Development Approach Concept Limitation

Kreutzer (1986) states that the development of a universal simulation language that could support model interchange is impractical and unrealistic. At one extreme, lossless conversion to and from a standard language is impossible without a true superset of all support languages'

constructs. Developing a common process interaction DES language is similar to the DoD-initiated and partially-funded Sharable Content Object Reference Model (SCORM) and Synthetic Environment Data Representation and Interchange Specification (SEDRIS) standards that support interoperability of interactive multimedia instruction and terrain databases respectively. These standardization efforts attempt to support interoperability with open neutral languages and have been lengthy and expensive. As noted above, Shriber (1991) points out that although modeling languages are often based on common principles, mastering a single language does not enable a person to apply simulation to all situations. This statement could be used to argue that a DES process interaction ontology (language) will not support all models.

3.3.1.2 Process Interaction DES Formalism Concept Limitation

PIMODES is not tied to a widely accepted formalism for process interaction DES because a widely accepted authoritative exhaustive list of ` process interaction DES concepts does not yet exist. Other popular DES world views (e.g., event-based, activity-based) have formalisms associated with them that are more mature compared to the process interaction DES world view. However, the PIMODES ontology does support common key features of process interaction world view models.

3.3.1.3 Vendor Dependencies Concept Limitation

A limitation of developing a standard language is its dependency on simulation software application vendors. The PIMODES concept requires applications to expose and populate their internal model data. Translation software must provide programmatic access or import/export

formats to read and write model data. PIMODES translation requires common / mappable semantics. If an application does not support a concept, a disconnect will occur. For example, ProcessModel does not provide a direct method for specifying the maximum number of entities to create in its arrival routing connection, a feature provided in Arena's "Create" flowchart module.

3.3.1.4 Adoption Concept Limitation

Long term success of the PIMODES concept requires adoption by simulation application developers. One purpose of PIMODES was to encourage adoption of a neutral interchange format. Successful standards typically originate from recognized benefits rather than being mandated. Vendors must see advantages to competing on the user interface and execution portions of their products and cede control of their data representations. Without legacy applications adopting PIMODES as a natively supported format, translation issues will arise as vendors continue to update their applications and their model data representations. At a minimum, adopters of the concept must provide access to their internal data. Optimally, adopters would read and write the PIMODES format natively.

3.3.2 Approach Limitation

Some limitations are associated with the specific approach taken in this research effort to implement the PIMODES concept. Approach limitations are related to the scope of the effort, ontology design, the choice of the ontology language, and the design and coding of the translation software.

3.3.2.1 Scope Approach Limitation

The developed ontology only supports one type of DES model – those described using the process interaction world view. Within process interaction DES models, only a subset of language elements are supported. More complex concepts (e.g., Arena's transporters and conveyors) could be added in future versions.

3.3.2.2 Ontology Design Approach Limitation

Harmonization choices and design decisions were made regarding model information representation. Ontology design decisions were made regarding the use of OWL language features. For example, many of the properties were defined as functional properties. An OWL functional property can only have one value associated with a particular instance. A determination was made for each ontology property regarding which OWL property features should be employed. A normalization process similar to database normalization process might result in a more efficient but less human understandable version of the ontology.

3.3.2.3 Ontology Language Choice Approach Limitation

The developed ontology is encoded using OWL. The PIMODES ontology is therefore limited to the expressiveness of OWL. OWL was selected because it is the W3C standard for representing ontologies. Other ontology languages exist and there are some critics of OWL. Critics cite issues such as the lack of support for rules in the current version of OWL. Models often contain conditions that are naturally expressed as rules. The Semantic Web Rules Language (SWRL) is under development and may eventually support this aspect. Some critics

of Semantic Web technology claim that interoperability is still being performed at the syntactic and not semantic level (Butler, 2006).

3.3.2.4 Translation Software Design and Coding Approach Limitations

Some limitations exist because of how the demonstration translation software was designed and coded. The demonstration application is PC-based and cannot be easily implemented as a Web application. Also, the coding for the translation demonstration software uses Microsoft Visual Basic which limits portability compared to languages such as Java. The PIMODES demonstration software was developed as a PC application because it needed to easily interface with ActiveX components that were installed along with their associated application on a particular PC. An installed PC application is less portable than a Java-based application or a Web-services approach. The demonstration software loads all model data into memory before processing. This approach limits the size of supported models.

4 CHAPTER FOUR: RESULTS

PIMODES research resulted in artifacts identified as outputs from the activity diagrams shown above in the Methodology section. The following subsections describe each of the resulting artifacts.

4.1 Research Plan

A research plan was developed that identified the research activities that were performed. The plan helped to organize the effort and keep the research focused on the objectives. All phases of the research were heavily influenced by the choice of legacy simulation applications. Arena and ProModel were selected because of their large installed base as evidenced by references in Winter Simulation Conference papers. AnyLogic represents simulation applications employing object-oriented techniques. ProcessModel was selected as an example of low-cost DES software.

4.2 Requirements Specification

The requirements specification identifies the information that the ontology must represent, the functionality of the demonstration translation software, and the content to be hosted on the support website. The requirements were iteratively refined throughout effort to describe the “as built” effort. The requirements document is titled the PIMODES Research Artifacts Requirements Specification and is provided online at:

http://www.opendes.org/PIMODES/Artifacts/PIMODES Requirements.pdf. Table 2 summarizes the requirements for supporting legacy applications and PIMODES model data.

Table 2. Model Information Support Requirements Summary

	Arena	**ProcessModel**	**AnyLogic**	**ProModel**	**PIMODES**
Process Concepts	Entity (Type) Queue Resource Variable (Entity) Attribute	Entity Resource	Variable	Entity Resource Location Attribute Variable	Entity Type Entity Attribute Queue Resource Variable Location
Activities	Assign Create Decide Dispose Process (includes Delay)	Activity Arrival	Source Sink Queue SelectOutput Delay Resource SeizeQ Release ProcessQ	Processing Arrivals	Creation Assignment Resource Interaction Delay Branching Disposition Queue
Control Flow	Connections	Connections	Port References Connections	Routing	Flowchart Nodes and Arcs

4.3 Ontology Design Document

The ontology design document describes the object-oriented design of the PIMODES ontology. The document describes legacy application methods for representing model data. The document shows how concepts from various legacy applications were harmonized into the PIMODES design. The ontology design document also describes a graphical representation language that leverages the visualization representation associated with UML 2.0 Activity Diagrams. The ontology design document is titled the PIMODES Ontology Design Document

and is provided online at: http://www.opendes.org/PIMODES/Artifacts/PIMODES Ontology Design.pdf.

4.3.1 Legacy Model Representation Analysis

Several popular legacy applications were analyzed to determine their process interaction DES approach for representing models and interchanging model data. Each legacy application analyzed (i.e., Arena, ProcessModel, AnyLogic, and ProModel) uses different concepts to represent process interaction DES, uses a different GUI for authoring internal data structures, and employs different techniques for importing and exporting model data. The disparate approaches shared key concepts for representing process interaction DES models that were harmonized into a single representation.

4.3.2 Harmonized Concepts

A concept map helps to graphically represent subject concepts and their relationships. The PIMODES concept map is shown in Figure 7. The activity concept is decomposed to show the relationship of activity concepts to process concepts (see Figure 8). In an effort to simplify translations, composite commands (e.g., Arena's "Process" flowchart module that performs resource interaction, queuing, and delays) were split into sequential chains of equivalent atomic commands. Many of these single action statements were reminiscent of SIMAN commands. A tradeoff exists between composite operations that are convenient for the author to use and atomic operations that simplify interchange.

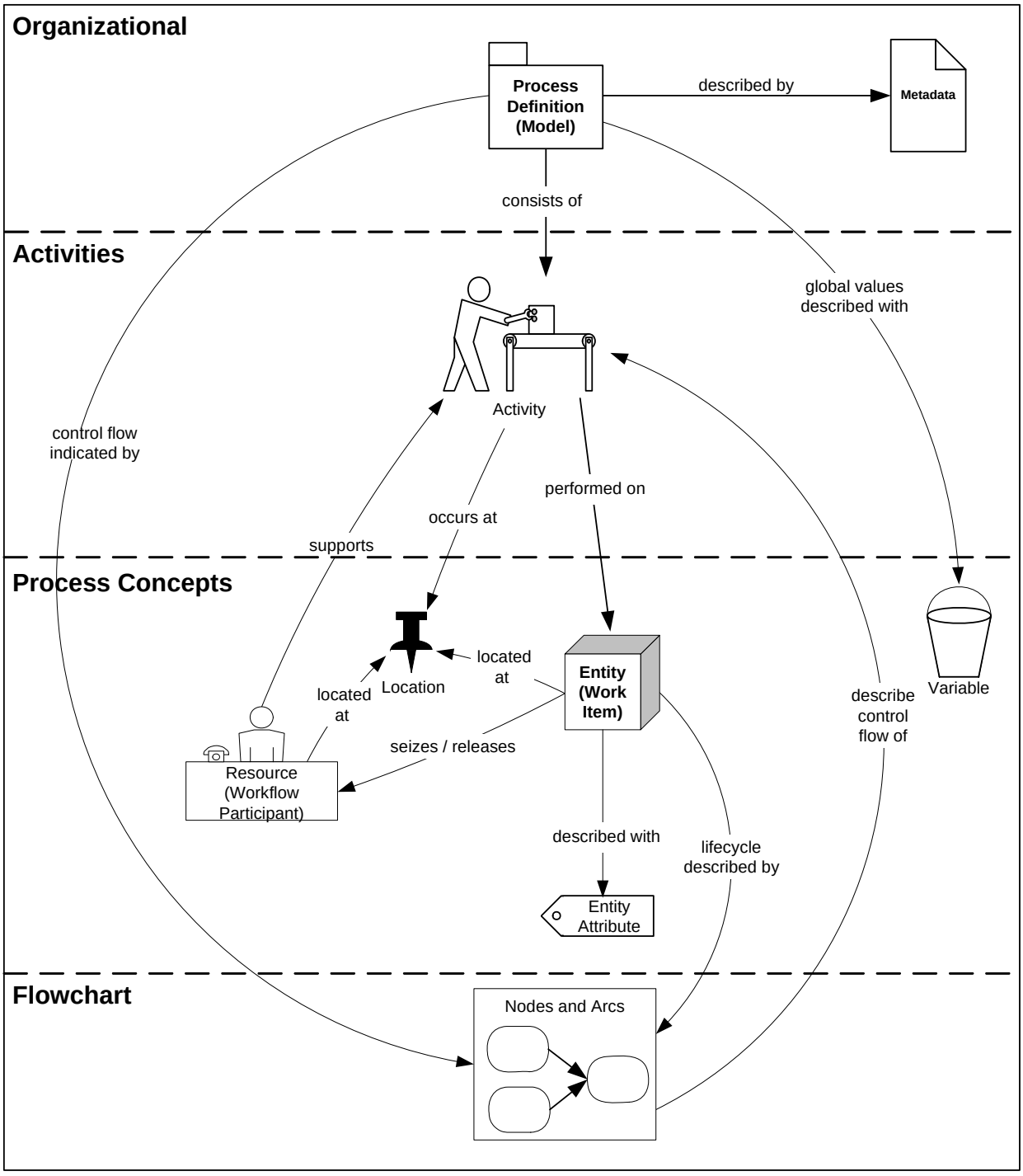

Figure 7. PIMODES Concept Map

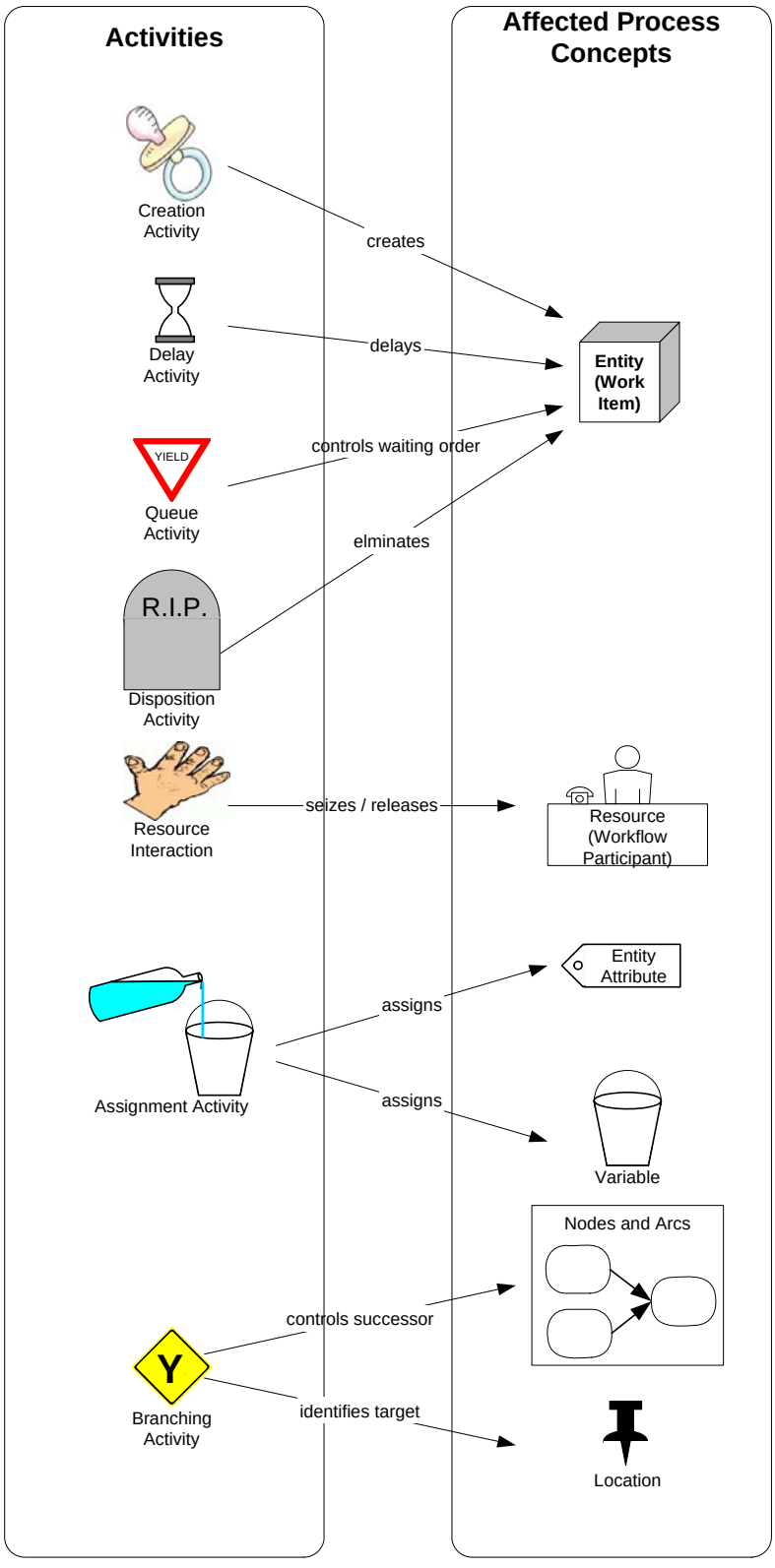

Figure 8. Activity Influence on Process Concepts

4.3.3 Objective PIMODES Ontology Description

The harmonized concepts were evolved into an object-oriented description. Clear semantics are required to unambiguously define concepts in an ontology. A static data model was developed for representing the PIMODES design. The data model was expressed using modified IDEF1X/UML static class diagrams to describe ontology classes and properties. The static object model for classes with associated properties was detailed sufficiently for encoding into an OWL ontology. An overview class diagram is presented in Figure 9.

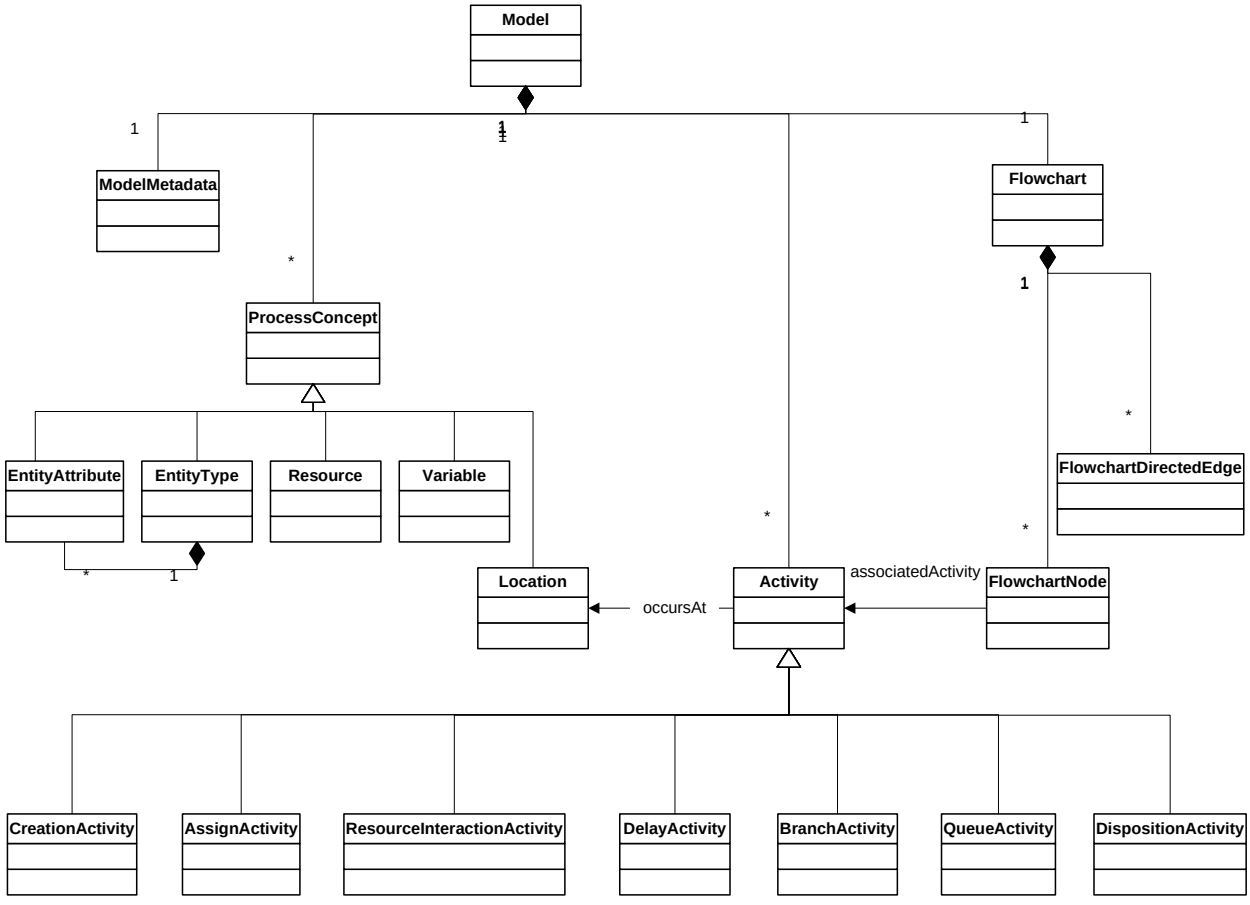

Figure 9. PIMODES Ontology Class Diagram

4.3.4 Legacy Application Support

The harmonized ontology design's support for legacy applications varied. One metric for determining coverage would be the percentage of a legacy application's constructs that are supported by the ontology. However, since some constructs are rarely used in practice, a better metric would relate the level of support to the frequency of use (e.g., creation of entities occurs in almost every model). A detailed analysis of PIMODES ontology support for legacy constructs is provided in the design document.

4.3.5 Graphical Representations

A graphical representation was specified for representing PIMODES models. PIMODES flowcharts can be represented using an enhanced UML 2.0 activity diagram approach. Standard UML diagrams were enhanced by relating activities to resources to indicate resource interaction. This approach is similar to resource assignment representations in ProcessModel.

4.4 PIMODES Ontology Description Report

The PIMODES ontology was encoded using the DOAT tool. The PIMODES Ontology Description Report provides DOAT table views, DumpOnt listings, and the RDF/XML code for the PIMODES ontology. The document is provided at http://www.opendes.org/PIMODES/Artifacts/PIMODES Ontology Description.pdf.

4.5 Translation Software Design

The translation software design document describes the design of the demonstration translation software. It includes high level design diagrams and detailed mapping tables and is provided at http://www.opendes.org/PIMODES/Artifacts/PIMODES Translation SW Design.pdf. The overview dataflow diagram is presented in Figure 10.

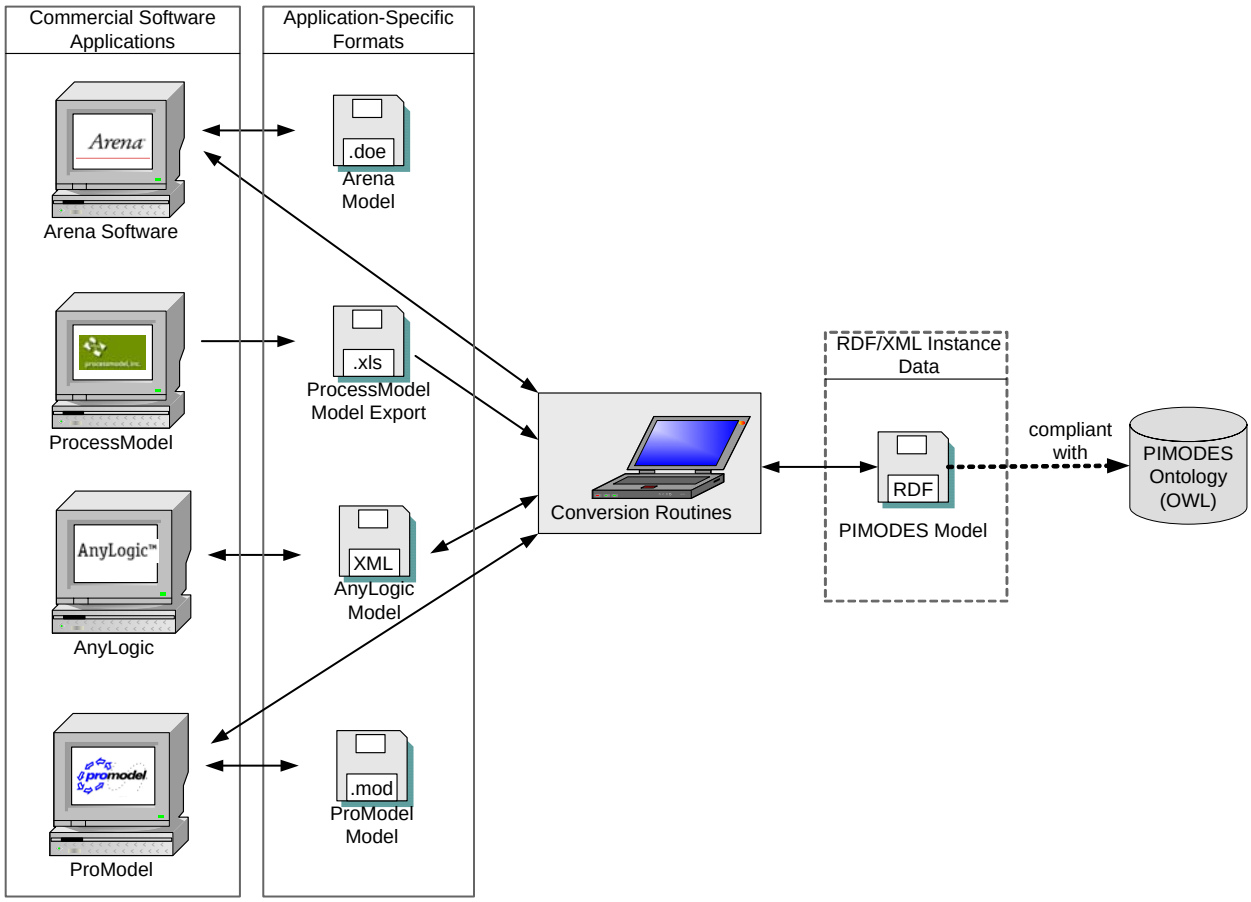

Figure 10. Translation Software Design

The design document includes detailed mapping information and identifies limitations of the translation process. Besides conceptual (semantic) differences (e.g., no explicit location support

in AnyLogic), some losses occur in the translation process due to the complexity of implementing the mappings.

4.6 Translation Software Code

The software coding effort resulted in a set of object-oriented Visual Basic projects. The PIMODES Translation Software Description Report is provided at: http://www.opendes.org/PIMODES/Artifacts/PIMODES Code.pdf. It describes the structure of the translation software class libraries. The document also shows traceability back to requirements.

4.7 Demonstration and Test Models Report

Demonstration and testing models were developed. Versions of these models included conceptual models, legacy application file formats, and native PIMODES versions. The models are described in the PIMODES Demonstration and Test Models Report which is provided at: http://www.opendes.org/PIMODES/Artifacts/PIMODES Models.pdf. An airport model was developed to demonstrate common queuing elements. An inventory model was developed to demonstrate mathematically-oriented features. Exhaustive testing models were developed for ensuring the completeness of the translation process. Based on the three models and the seven translation directions, a set of twenty one experiments were executed (see Table 3).

Table 3. Model List

Model Type	Source	Destination	Experiment #
Airport	Arena	PIMODES	1
	ProcessModel	PIMODES	2
	AnyLogic	PIMODES	3
	ProModel	PIMODES	4
	PIMODES	Arena	5
		AnyLogic	6
		ProModel	7
Inventory	Arena	PIMODES	8
	ProcessModel	PIMODES	9
	AnyLogic	PIMODES	10
	ProModel	PIMODES	11
	PIMODES	Arena	12
		AnyLogic	13
		ProModel	14
Exhaustive	Arena	PIMODES	15
	ProcessModel	PIMODES	16
	AnyLogic	PIMODES	17
	ProModel	PIMODES	18
	PIMODES	Arena	19
		AnyLogic	20
		ProModel	21

The following sections describe the results observed from performing the 21 translations with the demonstration translation software.

4.7.1 Arena to PIMODES Results

Arena does not appear to expose a differentiation of true and false connections from the Decide flowchart module. The information appears to be contained in “private” operands. Copying and then pasting modules in Arena results in the loss of unique identifiers due to Arena’s duplication of information in the copying process.

4.7.2 ProcessModel to PIMODES Results

ProcessModel does not expose connection arcs in its .xls export file. Therefore, PIMODES flowchart arcs cannot be automatically generated from ProcessModel. ProcessModel does not expose its resource assignments in its .xls export file. Therefore, PIMODES Resource Interaction activities cannot be automatically generated from ProcessModel. ProcessModel does not expose the "firstTime" attribute associated with periodic arrivals in its .xls export file. Therefore, the associated property in the PIMODES Creation Activity cannot be automatically generated from ProcessModel.

ProcessModel allows for different "firstTime" and "interarrivalTime" units of measure (time) associated with periodic arrivals in its .xls export file. However, only one unit of measure can be specified in a PIMODES Creation Activity.

4.7.3 AnyLogic to PIMODES Results

No loss of information was observed in the translation from AnyLogic to PIMODES.

4.7.4 ProModel to PIMODES Results

ProModel allows for different types of processing to be applied to different entity types. This is specified by associating an entity type with the Processing table instructions. However, the translation software currently assumes that the same processing logic applies to all entity types at the specified location.

ProModel allows for assignments to be made to both variables and entity attributes within the process operation of a Processing table record. However, the translation software currently assumes that all assignments are made to variables.

ProModel allows for branching to specified with multiple records in the Processing table's Routing subtable. However, the translation software assumes that only two-way branches occur and that their destinations are specified in consecutive records. The translation software also assumes that the sum of percentage conditions is 100%.

4.7.5 PIMODES to Arena Results

Arena data modules for entity types, resources, and queues are automatically generated when related flowchart modules are specified. Therefore, the translation software does not map PIMODES information to these items. However, this can cause a problem if the PIMODES version of the model contains Queue activities with different queue types because Arena queues can only have a single queue type. Since PIMODES Queue activities are not directly translated to Arena, the associated connections to and from Queue activities are lost. Since Arena does not support a concept of locations, the PIMODES Location information is lost.

4.7.6 PIMODES to AnyLogic Results

Since AnyLogic does not support a concept of locations, the PIMODES Location information is lost. The current version of the translation software does not construct the connections to Resource objects.

4.7.7 PIMODES to ProModel Results

The current version of the translation software does not construct the Processing table's Routing subtable. This results in a loss of control flow specification.

4.7.8 Experimentation Results Summary

The demonstration and testing models only contain supported items. Therefore, no issues of scope arose during the translations. However, observed problems were associated with issues of:

- syntax (data exposure from legacy applications),
- semantics (PIMODES ontology support), and
- automated conversions (translation software design and code).

4.8 Demonstration Script

The translation software user interface (see Figure 11) is fairly simple and intuitive to use. However a demonstration script was developed to ensure repeatability. The script provides step-by-step instructions for executing the translation software with one of the demonstration models. The script is provide at http://www.opendes.org/PIMODES/Artifacts/PIMODES Demo Script.pdf.

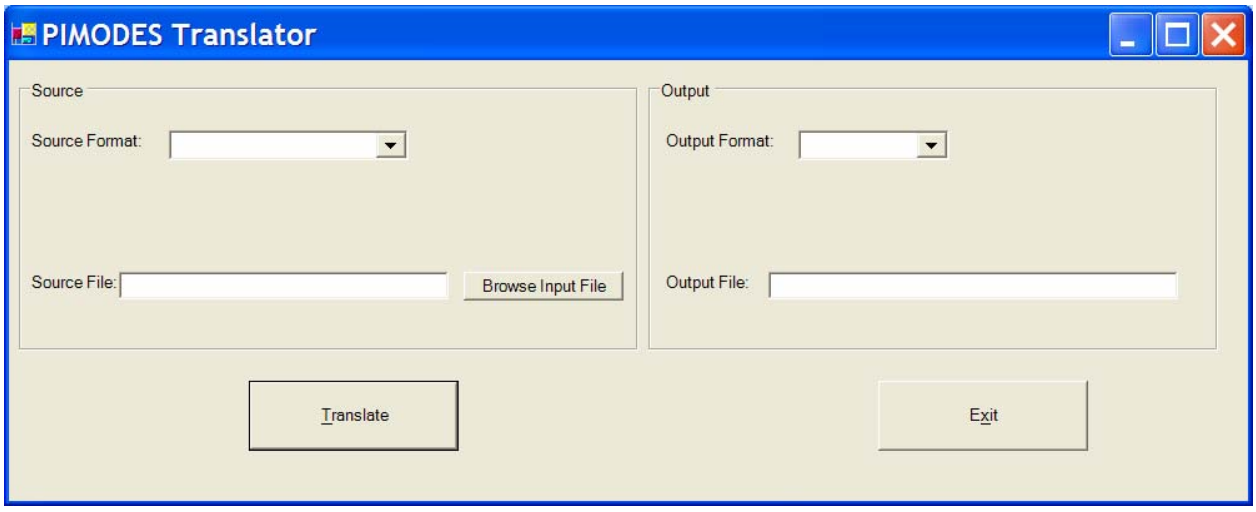

Figure 11. PIMODES Translation Software User Interface

4.9 Web Site

The project support website is available at: http://www.opendes.org/PIMODES/. The site provides information about the effort, the PIMODES ontology, the demonstration translation software, and softcopies of the research artifacts.

4.10 Results Artifact Summary

In addition to the dissertation text, the work products identified in Table 4 were generated as a result of the research. Each of the artifacts is provided on the project website.

Table 4. Artifact Summary

Document Title / Hyperlink	Format
PIMODES Research Artifacts Requirements Specification / http://www.opendes.org/PIMODES/Artifacts/PIMODES Requirements.pdf	Microsoft Word
PIMODES Ontology Design Document / http://www.opendes.org/PIMODES/Artifacts/PIMODES Ontology Design.pdf	Microsoft Word, Visio
PIMODES Ontology Description Report / http://www.opendes.org/PIMODES/Artifacts/PIMODES Ontology Description.pdf	Microsoft Access DB Tool, OWL output
PIMODES Translation Software Design Document / http://www.opendes.org/PIMODES/Artifacts/PIMODES Translation SW Design.pdf	Microsoft Word, Visio UML dataflow diagrams
PIMODES Translation Software Description Report / http://www.opendes.org/PIMODES/Artifacts/PIMODES Code.pdf	ASP.NET, XSLT
PIMODES Demonstration and Test Models / http://www.opendes.org/PIMODES/Artifacts/PIMODES Models.pdf	Microsoft Word, Visio
PIMODES Demonstration Script / http://www.opendes.org/PIMODES/Artifacts/PIMODES Demo Script.pdf	Microsoft Word

5 CHAPTER FIVE: CONCLUSIONS

This chapter presents research conclusions, offers recommendations to the problem investigated, discusses implications for future studies, and addresses the implications of the results.

5.1 Research Conclusions

The PIMODES research demonstrated that a process interaction DES ontology can be developed and that compliant models can be interchanged between legacy applications using automated translation software.

5.1.1 Process Interaction DES Ontology Development

The PIMODES research demonstrated that a process interaction DES ontology could be developed. DES concepts from literature and legacy applications were harmonized into a concept map that provided the basis for an ontology design. The ontology design was encoded into an OWL ontology file. The PIMODES ontology provides formal computer-parseable descriptions of process interaction DES concepts with OWL class and property specifications. The use of software technology including Semantic Web technology provides benefits over a simple text file approach. For example, entity types in a model can be distributed on the Web, identified with a URI reference, and associated with a domain ontology. Ontology-enabled software can enforce constraints on conforming instance files, leading to early detection of errors.

The PIMODES ontology refines and formalizes key process interaction DES concepts. PIMODES requires explicit individual start and end nodes in the control flow flowcharts. This serves to remove ambiguity and helps determine model completion status. PIMODES also defines a formal relationship between the activity, flowchart node, and location concepts. PIMODES formally defines the concept of an "Entity Type" which is often confused with the concept of the associated entities that are instances of the "Entity Type".

5.1.2 Legacy Application Model Interchange Feasibility

The PIMODES research also demonstrated that popular legacy application process interaction DES models can be interchanged using the PIMODES ontology. Significant portions of models authored using legacy applications were translated to the PIMODES format with translation software. PIMODES model data was also translated into legacy formats. The interchange experiments showed that:

- Legacy application process interaction DES model representations share a high degree of commonality,
- Vendors must expose their model data to allow for interchange, and
- Software can be developed to automate the translation of model data to and from the PIMODES format.

5.1.2.1 Legacy Model Representation Approach Commonality

Extensive semantic commonality exists in legacy process interaction DES languages. This is due to common objectives and a common heritage (family tree) of process interaction DES concepts. Similarities also resulted from the selection of the four primary legacy applications to review.

Although there are considerable similarities, there are also differences. For example, the ProModel location concept is more important for shop floor models but practically useless in describing processes performed by distributed Web services in a Service Oriented Architecture. Software applications also differ in their use of composite operations (e.g., Arena's "Process" flowchart module) for authoring convenience.

Older simulation applications tend to follow a functional approach while many newer applications (e.g., AnyLogic) use an object-oriented approach. This difference may mirror the trends in the broader software industry which has migrated from functional programming to object-oriented programming. In an object-oriented approach, activities are treated as objects (manipulators) and entities are represented as messages between them.

This research effort demonstrated that models from various legacy simulation applications can be interchanged. The loss of data associated with these conversions varies. The loss depends largely on how applications expose their model data and the semantics of how the information represents the application's particular perspective of the process interaction world view of discrete event simulation.

5.1.2.2 Vendors Must Expose Their Model Data to Allow Interchange

Legacy applications store model data in their internal data structures. A major feature of most commercial packages is their authoring GUI that populates their model data structures. However, vendor model representations are often overly coupled with their user interfaces, resulting in interchange challenges. Model interchange requires legacy applications to expose

their data and import new data. The easiest method of interchange would be to use PIMODES natively. The next simplest approach is to use XML - the approach used by AnyLogic.

5.1.2.3 Translation Software Feasibility

This effort demonstrated that the development of automated translation software is feasible. The software is constrained by vendors' support for importing and exporting model data. The demonstration translation software shows that models developed with existing popular applications can be interchanged with each other. Model developers can now begin to share models. The translation software can be extended to support additional simulation packages.

5.2 Recommendations

PIMODES research results should be leveraged to improve the state of the process interaction DES practice. PIMODES should be promoted as a "strawman" for a process interaction DES model representation interchange standard. Tool vendors should be educated on the benefits of providing a native PIMODES view of their internal model representations. Users should be told about the benefits of model reuse and the technology options for interchanging models. The theoretical descriptions of process semantics should be investigated and discussed to develop a consensus formalism that could serve as the basis for an ontology.

5.3 Implications for Future Studies

PIMODES research results can support a variety of future research studies. Additional work could investigate the scope and design of the PIMODES ontology as well as software applications to leverage the ontology. Research could also be performed regarding the aggregation and dispersion of model components using PIMODES.

5.3.1 Ontology Scope

The PIMODES requirements document specifies model data that must be supported by the ontology. New requirements could be added to the PIMODES requirement specification to support additional applications. Support for additional languages (e.g., FlexSim, GoldSim, Gensys G2) would help validate the PIMODES ontology and lead to improvements.

The PIMODES ontology provides a foundation for ontology-based simulations. The PIMODES ontology could be connected to upper ontologies to support semantic joins with other ontologies. Additional process interaction DES concepts could be adopted to PIMODES such as supporting hierarchical models, and including additional process concepts (e.g., schedules, sets). PIMODES currently supports only a single "flat" level of process steps. However, complex models require hierarchical models and PIMODES could be extended to support them. PIMODES could be incorporated into the SCORM family of standards for interchange training simulations. As a common language, PIMODES enables testing benchmarking by allowing for direct comparisons of models authored with various simulation software packages.

5.3.2 Ontology Design

A variety of designs could result from the PIMODES ontology requirements. Alternative designs could be developed. A significant model reuse challenge is the variety of approaches for representing mathematical expressions, especially distribution functions. The commonality of expression representations should be addressed, perhaps by investigating the use of MathML. Similarly, rules could be used to formalize the expression of conditions.

A variety of graphical representations are used to describe processes. A formal ontology of diagrams could help differentiate between methods for representing control flow. An associated graphical language could be investigated. UML 2.0 ADs are insufficient because of the need to describe resource requirement associations and branching logic.

5.3.3 Software Application Development

New software can now be developed to support process interaction DES users. Open source software initiatives would be consistent with the open nature of the PIMODES ontology. Authoring software could edit PIMODES models as a native file format. Other software could execute PIMODES models. Eventually, new technologies such as Web services could support the format. A model editor could be developed that uses PIMODES as its native data format. The editor would help users visualize and manipulate translated data.

The PIMODES focus is on describing a model. However, a small amount of additional data could describe an experiment. Such data would include replication restrictions and other data to support simulation execution. Software could execute the PIMODES models and provide output statistics to users. This would avoid the need for translating to/from legacy simulation

software application model formats. A popular trend in software development is the use of Web services. Web services could be developed to provide PIMODES model authoring and execution using software distributed on the Web. This approach would be consistent with the distributed model capabilities enabled by the PIMODES ontology and the web-ready features of OWL.

5.3.4 Aggregation and Dispersion

PIMODES could be used to support the aggregation and dispersion of model contents. A model author might want to assemble a new model from portions of existing models that are represented with various languages. The existing models could be converted to the PIMODES format and then portions of interest could be extracted from the converted models and assembled into a new PIMODES model.

Another use case could involve splitting a PIMODES model into components that are then converted to other formats for execution by various simulation applications. In this way, the best features of different packages could be used to simulate specific portions of the original model.

5.4 Implications of the Results

The development of the PIMODES ontology represents a new opportunity to share DES models. Many common operations and concepts are supported. Therefore, a great deal of legacy model content can be interchanged using the ontology.

The PIMODES ontology represents a new open process interaction DES language, formalized with an OWL ontology, for interchanging process interaction DES models. Researchers can extend the PIMODES language with additional concepts and activities.

The PIMODES ontology enables a new ontology-based approach to process interaction DES model interchange that supports reuse. Ultimately, model development can be better, faster, and cheaper through the reuse enabled by using PIMODES.

LIST OF REFERENCES

Andrews, T. et al (2003). *Business Process Execution Language for Web Services, Version 1.1*. Retrieved March 18, 2006 from http://xml.coverpages.org/BPELv11-May052003Final.pdf.

Ankolekar, A., Paolucci, M., & Sycara, K. (2004). Spinning the OWL-S Process Model Toward the Verification of the OWL-S Process Models. *Proceedings of the Semantic Web Services Workshop at the Third International Semantic Web Conference*.

Arief, L. B., Speirs, N. A. (2000). A UML Tool for an Automatic Generation of Simulation Programs. *Proceedings of the Second International Workshop on Software and Performance*.

Balci, O., Bertelrud, A. I., Esterbrook, C. M., & Nance, R. E. (1998). Visual Simulation Environment. *Proceedings of the 1998 Winter Simulation Conference*.

Ball, P. (1996). Introduction to Discrete Event Simulation. Proceedings of the Second DYCOMANS workshop on Management and Control: Tools in Action. Retrieved March 7, 2005 from http://www.dmem.strath.ac.uk/~pball/simulation/simulate.html.

Banks, J. (1996). Software for Simulation. *Proceedings of the 1996 Winter Simulation Conference*.

Banks, J. (2001). Panel Session: The Future of Simulation. *Proceedings of the 2001 Winter Simulation Conference*.

Banks, J., & Carson, J. S. II (1985). Process-interaction Simulation Languages. *Simulation 44:5*, 225-235.

Banks, J. & Carson J. S., (1986). Introduction to Discrete-event Simulation. *Proceedings of the 18th Winter Simulation Conference*.

Bapat, V. & Swets, N. (2000). The Arena Product Family: Enterprise Modeling Solutions. *Proceedings of the 2000 Winter Simulation Conference*.

Barros, F. J. (1995). Dynamic Structure Discrete Event System Specification: a New Formalism for Dynamic Structure Modeling And Simulation. *Proceedings of the 27th conference on Winter simulation*.

Belanger, T. C. (1994). The Indispensable Task Network. *AIPE Facilities.*

Benjamin, P., Akella, K.V., Malek, K., & Fernandes, R. (2005). An Ontology-Driven Framework For Process-Oriented Applications. *Proceedings of the 2005 Winter Simulation Conference 2005*.

Berners-Lee, T. (1998). *What the Semantic Web can represent*. Retrieved September 14, 2006 from http://www.w3.org/DesignIssues/RDFnot.html.

Berners-Lee, T. (1999). *Weaving the Web*.

Berners-Lee, T. (2000). Semantic Web. Presentation at XML 2000. Retrieved September 14, 2006 from http://www.w3.org/2000/Talks/1206-xml2k-tbl/.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The Semantic Web. *Scientific American 284* (5): 34–43.

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

Bobeanu, C., Kerckhoffs, J. H., & Van Landeghem, H. (2004). Modeling of Discrete Event Systems: A Holistic and Incremental Approach using Petri Nets. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Volume 14 Issue 4.

Bock, C. (1999). Three Kinds of Behavior Model. *Journal of Object-Oriented Programming*. Volume 12, Number 4.

Bock, C. (2003). UML 2 Activity and Action Models. *Journal of Object Technology*, Volume 2, Number 4.

Bock, C., & Gruninger, M., (2005). PSL: A Semantic Domain for Flow Models. *Journal of Software and Systems Modeling, 4:2*.

Brutzman, D., Zyda, M., Pullen, J.M., & Morse, K.L. (2002). *Extensible Modeling and Simulation Framework (XMSF) Challenges for Web-based Modeling and Simulation*. Retrieved September 5, 2005 from http://www.movesinstitute.org/xmsf.

Butler, M.H. (2006). *Is the Semantic Web Hype?*. Retrieved June 12, 2006 from http://www.hpl.hp.com/personal/marbut/isTheSemanticWebHype.pdf

Carey, S. A., Kleiner, M.S., Hieb, M.R., & Brown, R., (2002a). Standardizing Battle Management Language – Facilitating Coalition Interoperability. *MSIAC M&S Journal*, Vol. 4 #2.

Carey, S. A., Kleiner, M.S., Hieb, M.R., & Brown, R. (2002b). Development of a C2 Standard of Task Representation for C4ISR Systems, Simulations, and Robotics: Battle Management

Language. Paper presented at the 2002 Command and Control Research and Technology Symposium. Track 3: Modeling and Simulation.

Carroll, J. J., & DeRoo, J. (2004). OWL Web Ontology Language Test Cases. Retrieved September 14, 2006 from: http://www.w3.org/TR/owl-test/.

Cassandras, C.G., & Lafortune, S. (1999). *Introduction to Discrete Event Systems*. Kluwer.

Cavarra, A., Riccobene, E., & Scandurra, P. (2004) *A Framework to Simulate UML Models: Moving from a Semiformal to a Formal Environment*. Paper presented at the ACM Symposium on Applied Computing.

Clementson, A. T. (1986). Simulation with Activities using C.A.P.S/E.C.S.L (the British Approach to Discrete-event Simulation). *Proceedings of the 1986 Winter Simulation Conference*.

Cota, B. A., Fritz, D. G., & Sargent, R. G. (1994). Control Flow Graphs as a Representation Language. *Proceedings of the 1994 conference on Winter Simulation Conference*.

Cota, B. A., & Sargent, R.E. (1992). A modification of the process interaction world view. *ACM Transactions on Modeling and Computer Simulation (TOMACS), 2* (2): 109-129.

Crain, R. C. (1997). Simulation using GPSS/H. *Proceedings of the 1997 Winter Simulation Conference*.

Daconta, M. C., Obrst, L. J., & Smith, K. T. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing.

DaCosta, B. (2002). XML Support for OneSAF Objective System Behaviors. *Proceedings of the 11th CGF Conference*.

DaCosta B., Lucas, T., Outar, R., & Helton, D. (2003). OneSAF Repository Framework: Defining, Storing, and Interchanging XML Data. *Proceedings of the Spring 2003 Simulation Interoperability Workshop*.

Davis, P. K., & Anderson, R. H. (2003). *Improving the Composability of Department of Defense Models and Simulations*. RAND National Defense Research Institute, 2003.

Davis, D. A., & Pegden, C. D. (1988). Introduction to SIMAN. *Proceedings of the 1988 Winter Simulation Conference*.

Dean, M., Schreiber, G., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., & Stein, L.A. (2004). *OWL Web Ontology Language Reference*. World Wide Web Consortium (W3C) Recommendation. Retrieved September 14, 2006 from http://www.w3.org/TR/owl-ref/.

Dubiel, B., & Tsimhoni, O. (2005). Integrating Agent Based Modeling into a Discrete Event Simulation. *Proceedings of the 2005 Winter Simulation Conference*.

Fensel, D. (1998). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer.

Filippov, A. (2003). AnyLogic Technical Overview, Retrieved November 12, 2005 from: http://www.anylogic.jp/download/any5presentation.pdf.

Fineberg, M. L. (1995). *A comprehensive taxonomy of human behaviors for synthetic forces* (IDA Paper P-3155). Alexandria, VA: Institute for Defense Analysis.

Fishman, G. S. (1978). *Principles of Discrete Event Simulation*. New York: John Wiley & Sons.

Fishwick, P. A. (2004). Toward an Integrative MultiModeling Interface: A Human-Computer Interface Approach to Interrelating Model Structures. *SCS Transactions on Modeling and Simulation*.

Fishwick, P. A., & Miller, J. A. (2004). Ontologies for Modeling and Simulation: Issues and Approaches. *Proceedings of the 2004 Winter Simulation Conference*.

Franta, W. R. & Maly, K. (1977). An Efficient Data Structure for the Simulation Event Set. *Communications of the ACM* 20(8): 596-602 (1977)

Gerber, W. J., & Lacy, L. W. (2004a). Standard Ontological Behavior Representation to Support Composability (extended abstract). *Proceedings of the 13th Behavior Representation in Modeling and Simulation Conference*.

Gerber, W. J., & Lacy, L. W. (2004b). Behavior Composability Support Through Standardized Ontology Representations. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*.

Glynn, P.W. (1989). A GSMP formalism for discrete event systems. *Proceedings IEEE 77*, 1 Jan. 14-33, Vol. 77, Issue 1.

Gomez-Perez, A., Fernandez-Lopez, M., & Corcho, O. (2004). *Ontological Engineering*. London: Springer.

Gravitz, P., Sheehan, J., & McLean, T. (1999) Common Activities in Data Interchange Format (DIF) Development. *Proceedings of the Spring 1999 Simulation Interoperability Workshop*.

Gruber, T. R. (1993). A Translation approach to portable ontology specifications. *Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge Based Systems 5* (2), 199-220.

Guo, L., Chen-Burger, Y., & Robertson, D. (2004). Mapping a Business Process Model to a Semantic Web Service Model. *Proceedings of the IEEE International Conference on Web Services (ICWS'04).*

Guru, A., Savory, P., & Williams, R., (2000). A Web-based Interface for Storing and Executing Simulation Models. *Proceedings of the 2000 Winter Simulation Conference*.

Hall, R.J., & Zisman, A. (2004a). OMML: A Behavioral Model Interchange Format. *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04).*

Hall, R. J., & Zisman, A. (2004b). Model Interchange and Integration for Web Services. TAV-WEB Proceedings/ACM SIGSOFT SEN

Hanrahan, R. P. (1995). *The IDEF Process Modeling Methodology*. Retrieved July 23, 2005 from http://www.stsc.hill.af.mil/crosstalk/1995/06/IDEF.asp.

Harrell, C. R., Ghosh, B., & Bowden, R. (2000). *Simulation Using ProModel.*

Harrell, C. R., & Price, R. N. (2000). Software/modelware tutorials I: Simulation modeling and optimization using ProModel. *Proceedings of the 32nd conference on Winter simulation.*

Harrell, C. R., & Price, R. N. (2003). Simulation Modeling Using Promodel Technology, *Proceedings of the 2003 Winter Simulation Conference.*

Harrison, G.A., Maynard, D.S., & Pollak, E. (2004). Automated Database And Schema-Based Data Interchange For Modeling And Simulation. *Proceedings of the 2004 Winter Simulation Conference.*

Harward, G.B. (2005). *Suitability of the NIST Shop Data Model as a Neutral File Format for Simulation*. master's thesis, BYU.

Hayes, P., Horrocks, I., & Patel-Schneider, P. F. (2004). OWL Web Ontology Language Semantics and Abstract Syntax

Henriksen, J. O., & Crain, R. C. (2000). GPSS/H: a 23-Year Retrospective View. *Proceedings of the 2000 Winter Simulation Conference.*

Hieb, M., Pullen, J., Sudnikovich, W., & Tolk. A. (2004) Extensible Battle Management Language (XBML): A Methodology for Web Enabling Command and Control for

Network Centric Warfare. *Proceedings of the 2004 Command and Control Research and Technology Symposium The Power of Information Age Concepts and Technologies*.

Hobbs, R. L. (2003). Using XML to Support Military Decision-Making. Paper presented at the 2003 XML Conference and Exposition.

Ingalls, R. G. (1986). Automatic Model Generation. *Proceedings of the 1986 Winter Simulation Conference*.

Joint Warfighting Center (1997). *Handbook for the Design and Use of Operational Templates.*

Karayanakis, N. M. (1995). *Advanced System Modelling and Simulation with Block Diagram Languages*. Boca Raton : CRC Press.

KBSI (2005). *IDEF3 Process Description Capture Method*. Retrieved September 14, 2006 from http://www.idef.com/idef3.html

Kelton, W. D., Sadowski, R. P., Sturrock, D. T. (2003). *Simulation with Arena*, 3rd ed., McGraw-Hill.

Kreutzer, W. (1986). *Systems Simulation: Programming Styles and Languages*. Wokingham, England: Addison-Wesley.

Lacy, L. W. (2001). Semantic Web Applications for Modeling and Simulation. Retrieved September 13, 2006 from http://www.daml.org/2001/07/dmso-applications/semantic-web-071101.ppt

Lacy, L. W. (2004) *DARPA DAML Final Report*. DRC Report #DRC E-8970U.

Lacy, L. W. (2005) *OWL: Representing Information Using the Web Ontology Language*. Victoria, Canada: Trafford Publishing.

Lacy, L. W. (2006). Interchanging Discrete Event Simulation Models using PIMODES and SRML. *Proceedings of the Fall 2006 Simulation Interoperability Workshop.*

Lacy, L. W., & Dugone, T. (2000a). Using XML To Share Offline Simulation Data. *Proceedings of the 2000 Summer Computer Simulation Conference 2000.*

Lacy, L., & Dugone, T. (2000b). Computer Generated Forces Behavior Representation and Reuse Using the eXtensible Markup Language (XML). *Proceedings of the Fall 2000 Simulation Interoperability Workshop*.

Lacy, L., Dugone, T., & Youngren, R. W. (2001). Standard Data Exchange Methods for Equipment Characteristics And Performance Data. *Proceedings of the 2001 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC).*

Lacy, L. W., & Gerber, W. J. (2004). Potential Modeling and Simulation Applications of the Web Ontology Language – OWL. *Proceedings of the 2004 Winter Simulation Conference*.

Lacy, L., & Henninger, A. (2003). Developing Primitive Behavior Ontologies using the Ontology Web Language. *Proceedings of the 2003 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC).*

Lacy, L., & O'Brien, L. (1997). Conceptual Modeling for WARSIM 2000. *Proceedings of the 1997 Interservice/Industry Training Systems and Education Conference*.

Lacy, L., Randolph, W., Harris, B., Youngblood, S., Sheehan, J., Might, R., & Metz, M. (2001). Developing a Consensus Perspective on Conceptual Models for Simulation Systems. *Proceedings of the Spring 2001 Simulation Interoperability Workshop.*

Lacy, L., Stone, G.; & Dugone, T. D. (1999a). Sharing HLA Scenario Data. *Proceedings of the Fall 1999 Simulation Interoperability Workshop*.

Lacy, L., Stone, G.; & Dugone, T. D. (1999b). XML Data Interchange Format Standards for HLA-Related Data Interoperability. *Proceedings of the 1999 Southeastern Simulation Conference*.

Lacy, L., Stone, G.; & Dugone, T. D. (2001). Representing Computer Generated Forces Behaviors Using eXtensible Markup Language (XML) Techniques. *Proceedings of the Tenth Conference on Computer Generated Forces.*

Lacy, L., & Tuttle, C. (1998). Interchanging Simulation Data using XML. *Proceedings of the 1998 Fall Simulation Interoperability Workshop*.

Laughery, R. (1998). Computer Simulation As A Tool For Studying Human-Centered Systems. *Proceedings of the 1998 Winter Simulation Conference.*

Law, A. M., & Kelton, W. D. (2000). *Simulation Modeling and Analysis*. McGraw-Hill.

Liles, D. H., & Presley, A. R. (1996). Enterprise Modeling Within An Enterprise Engineering Framework. *Proceedings of the 1996 Winter Simulation Conference*.

Lubell, J. (2001). XML Representation of Process Descriptions In *Professional XML Meta Data.* Wrox Press.

Markovitch, N. A., Profozich, D. M. (1996). Simulation Modelling Support via Network Based Concepts. *Proceedings of the 1996 Winter Simulation Conference*.

Mayer, R. J., Menzel, C. P., & Mayer, P. (1991). *IDEF3 Technical Report.*

McGuinness, D. L. (2002). Ontologies Come of Age. In D. Fensel, J. Hendler, H. Lieberman, & W. Wahlster (Eds.), *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press. Retrieved from: http://www.ksl.stanford.edu/people/dlm/papers/ontologies-come-of-age-mit-press-(with-citation).htm

McGuinness, D. L., Fikes, R., Hendler, J., & Stein. L. A. (2002). DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems*, Vol. 17, No. 5, pages 72-80.

McGuinness, D. L., van Harmelen, F. (2004). *OWL Web Ontology Language Overview. World Wide Web Consortium (W3C) Recommendation*. Retrieved September 13, 2006 from http://www.w3.org/TR/owl-features/.

Meeks, A. O., Aviles, G., & Lacy, L. W. (2004). Auto-Authoring Instruction from Ontological Representations of Procedures. *Proceedings of the 2004 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC).*

Mendling, J., Pérez de Laborda, C. , & Zdun, U. (2005). Towards Semantic Integration of XML-based Business Process Models. In K. D. Althoff, A. Dengel, R. Bergmann, M. Nick, & T. Roth-Berghofer (Eds.) *Proceedings of the WM2005: Professional Knowledge Management - Experiences and Visions. Semantic Model Integration Workshop (SMI 2005)* as part of the 3rd Conference Professional Knowledge Management (WM 2005), Kaiserslautern, Germany, April 2005, pages 513-517.

Menzel, C., & Grüninger, M. (2001). A Formal Foundation for Process Modeling. *Proceedings of the 2001 International Conference on Formal Ontology in Information Systems*.

Mili, F., & Ghanekar, S. (2005). *Building and Using an OWL-S Ontology of Tasks*, Paper presented at the 2005 OWL Workshop at the International Semantic Web Conference.

Miller, J. A., & Baramidze, G. (2005). Simulation and the Semantic Web. *Proceedings of the 2005 Winter Simulation Conference*.

Miller, J. A., & Fishwick, P. A. (2004). *Investigating Ontologies for Simulation Modeling*. Paper presented at the 2004 Simulation Symposium.

Miller, J. A., Silver, G. A., & Lacy, L. W. (2006). Ontology Based Representations Of Simulation Models Following The Process Interaction World View. *Proceedings of the 2006 Winter Simulation Conference* (in press).

Nainani, B. (2005). Supporting the Business Process Lifecycle using Standard-based Tools. *WebServices Journal*, Vol.5 Issue 4.

Nance , R. E. (1993)., A History of Discrete Event Simulation Programming Languages , *ACM SIGPLAN Notices*, Volume 28 Issue 3.

Nance, R. E. & Sargent, R.G. (2002). Perspectives on the evolution of Simulation. *Operations Research 50* (1): 161-172.

Narain, S. (1991). An Axiomatic Basis for General Discrete-event Modeling. *Proceedings of the 23rd Winter Simulation Conference*.

Nielsen, N. R. (1991). Application of AI Techniques to Simulation. In Fishwick, Modjeski (Eds.) *Knowledge-Based Simulation Methodology and Application.*

Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical Report KSL-01-05, Retrieved September 13, 2006 from: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

Odhabi, H. I., Paul, R. J., & Macredie, R. D. (1998). Developing a Graphical User Interface for Discrete Event Simulation. *Proceedings of the 30th Winter Simulation Conference*.

O'Reilly, J. (2002). Introduction to AWESIM. *Proceedings of the 2002 Winter Simulation Conference*.

Oscarsson, J., Moris, M. U. (2002). Documentation of Discrete Event Simulation Models for Manufacturing System Life Cycle Simulation. *Proceedings of the 2002 Winter Simulation Conference*.

Overstreet, C. M., & Nance, R. E. (1985). A Specification Language to Assist in Analysis of Discrete Event Simulation Models. *Communications of the ACM*, Volume 28 Issue 2

Pace, D. K. (2001). Conceptual Model Development for C4ISR Simulations. Paper presented at the Fifth International Command and Control Research and Technology Symposium. Retrieved September 13, 2006 from: http://www.dodccrp.org/events/2000/5th_ICCRTS/cd/papers/Track2/059.pdf

Page, E. H. (1994). Simulation Modeling Methodology: Principles and Etiology of Decision Support, PhD Dissertation.

Pegden, C. D. (1983). Introduction to SIMAN. *Proceedings of the 1983 Winter Simulation Conference*.

Pegden, C. D., Shannon, R. E., & Sadowski, R.P. (1995). *Introduction to Simulation Using SIMAN*. McGraw-Hill.

Pew, R. W., & Maver, A. S. (editors) (1998). *Modeling Human and Organizational Behavior: Application to Military Simulations*. Report of the US National Research Council's panel on Modeling Human Behavior and Command Decision Making (Representation for Military Simulations). also published by National Academies Press.

Pidd, M. (1984). *Computer Simulation in Management Science*. Chichester: John Wiley & Sons.

Pidd, M. (2002). *Computer Simulation in Management Science,* Fourth Edition. Chichester: John Wiley & Sons.

Pidd, M. (2004). Simulation Worldviews – So What?. *Proceedings of the 2004 Winter Simulation Conference*.

Pilone, D. (2005). *UML 2.0 in a Nutshell*. O'Reilly

Pooch, U. W., & Wall, J. A. (1993). *Discrete Event Simulation*. Boca Raton, Florida: CRC Press.

Praehofer, H., & Pree, D. (1993). Visual modeling of DEVS-based multiformalism systems based on higraphs. *Proceedings of the 25th Winter Simulation Conference*.

Pritsker, A. A. B. (1986). *Introduction to Simulation and Slam II*

Pritsker, A. A. B., O'Reilly, J. J., & LaVal, D. K. (1999). *Simulation with Visual SLAM and AweSim*. New York: John Wiley & Sons.

ProcessModel (1999) User's Manual, ProcessModel Corp., Provo, UT.

Radiya, A., & Sargent, R. G. (1994). A Logic-Based of Discrete Event Modeling and Simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS),* Volume 4 Issue 1.

Reichenthal, S. W. (2002). SRML-Simulation Reference Markup Language W3C Note. Retrieved September 13, 2006 from: http://www.w3.org/TR/SRML.

Reichenthal, S. W. (2004). SRML Case Study: Simple Self-Describing Process Modeling and Simulation. *Proceedings of the 2004 Winter Simulation Conference*.

Reichenthal, S. W., & Gustavson, P. L. (2003). Manufacturing BOMs with SRML for Process-Oriented Federations. *Proceedings of the Fall 2003 Simulation Interoperability Workshop*.

Risner, S., Porter, K., Lacy, L., O'Brien, L., & Kollmorgen, G. (1998). Conceptual Modeling in the Joint Simulation System (JSIMS). *Proceedings of the 1998 Fall Simulation Interoperability Workshop*.

Roberts, R. S. (1991). Simulation Languages and Database Theory: Some Considerations from the Entity-Relationship Model. *Proceedings of the 1991 Winter Simulation Conference*.

Rohrer, M. W. (2000). Software/modelware tutorials I: AutoMod product suite: AutoMod tutorial. Proceedings of the 32nd Winter Simulation Conference Hierarchical modeling for discrete event simulation (panel)

Russell, N., van der Aalst, W., ter Hofstede, A., & Wohed, P. (2006). On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling, Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 53., Markus Stumptner, Sven Hartmann and Yasushi Kiyoki, Ed.

Schenck, D. A., & Wilson, P. R. (1994). *Information Modeling: The EXPRESS Way*. Oxford University Press.

Schriber, T. J. (1991). *An Introduction to Simulation Using GPSS/H*. New York.

Schriber, T. J., & Brunner, D. T. (2001). Inside Discrete-Event Simulation Software: How it Works and Why it Matters. *Proceedings of the 2001 Winter Simulation Conference*.

Schruben, L. (1983). Simulation Modeling with Event Graphs. *Communications of the ACM*, Volume 23, Number 11.

Schruben, L. (1992) Graphical model structures for discrete event simulation. *Proceedings of the 24th conference on Winter simulation*.

Seila, A. F. (2005). The Case for a Standard Model Description for Simulation. *International Journal of Simulation and Process Modeling*, Volume 1, Nos. 1/2.

Sheehan, J. (2001). Data Provisioning Using Authoritative Data Sources, Paper presented at the NDIA SBA Conference. Retrieved September 5, 2005 from: http://www.dtic.mil/ndia/2001sbac/sheehan.pdf.

Sheehan, J., Prosser, T., Conley, H., Stone, G., Yentz, K, & Morrow, J. (1998). Conceptual Models of the Mission Space (CMMS): Basic Concepts, Advanced Techniques, and Pragmatic Examples. *Proceedings of the Spring 1998 Simulation Interoperability Workshop*.

Smith, M. K., Welty, C., & McGuinness, D. L. (2004). *OWL Web Ontology Language Guide. World Wide Web Consortium (W3C) Recommendation*. Retrieved September 13, 2006 from http://www.w3.org/TR/owl-guide/.

Son, Y. J., Jones, A. T., & Wysk, R. A. (2000). Automatic Generation of Simulation Models from Neutral Libraries: an Example. *Proceedings of the 2000 Winter Simulation Conference*.

Son, Y. J., Jones, A. T., & Wysk, R. A. (2003). A Component Based Simulation Modeling from Neutral Component Libraries. *Computers & Industrial Engineering 45* (2003) 141–165.

Sowa, J. F., (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks/Cole.

Storrle, H., & Hausmann , J. H. (2005). Towards a Formal Semantics of UML 2.0 Activities. *Software Engineering* 2005: 117-128

Sulistio, A., Yeo, C. S., & Buyya, R. (2004). A Taxonomy of Computer-based Simulations and its Mapping to Parallel and Distributed Systems Simulation Tools. Retrieved September 13, 2006 from: http://www.gridbus.org/papers/simulationtaxonomy.pdf

Swain, J. J. (2003). Simulation Reloaded, *OR/MS Today*, Retrieved September 13, 2006 from: http://www.lionhrtpub.com/orms/orms-8-03/frsurvey.html.

Swegles, S. (1997). Business Process Modeling with SIMPROCESS. *Proceedings of the 1997 Winter Simulation Conference.*

Sycara, K., Martin, D., McGuinness, D. L., McIlraith, S. & Paolucci, M. (2004). OWL-S Technology for Representing Constraints and Capabilities of Web Servcies. Paper presented at the W3C Workshop on Constraints and Capabilities for Web Services.

Syrjakow, M., Syrjakow, E., & Szczerbicka, H. (2002). Towards a Component-Oriented Design of Modeling and Simulation Tools. *Proceedings of Conference on AI, Simulation & Planning In High Autonomy Systems*.

Thatte S. (2001) XLANG: Web Services For Business Process Design, Microsoft Corporation.

Trick, M. A. (2005). Types of Simulation. Retrieved September 13, 2006 from: http://mat.gsia.cmu.edu/simul/node7.html

Vitolins, V., & Kalnins, A. (2005). Semantics of UML 2.0 Activity Diagram for Business Modeling by Means of Virtual Machine. *Proceedings Ninth IEEE International EDOC Enterprise Computing Conference*, pp. 181.-192.

WfMC (1999) Workflow Management Coalition, Interface 1: Process Definition Interchange Process Model, Document Number WfMC TC-1016-P, Retrieved September 13, 2006 from: http://www.wfmc.org/standards/docs/TC-1016-P_v11_IF1_Process_definition_Interchange.pdf

Whitman, L., Huff, B., & Presley, A. (1997). Structured models and dynamic systems analysis: the integration of the IDEF0/IDEF3 modeling methods and discrete event simulation. *Proceedings of the 29th conference on Winter simulation*.

Woolfson, M. M, & Pert, G. J. (1999). *An Introduction to Computer Simulation*. Oxford University Press.

Zeigler, B. P. (1976). *Theory of Modelling and Simulation*. New York: Wiley & Sons.

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of Modeling and Simulation* (2nd). San Diego: Academic Press.