Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

AN ONTOLOGY-DRIVEN FRAMEWORK FOR PROCESS-ORIENTED APPLICATIONS

Perakath Benjamin
Kumar V. Akella
Kaiser Malek
Ronald Fernandes

Knowledge Based Systems, Inc.
1408 University Drive East
College Station, TX 77840, U.S.A.

ABSTRACT

This paper describes an ontology-driven framework for process-oriented applications. The research described in this paper is motivated by the lack of information sharing mechanisms at the semantic level among process-oriented applications. Our approach addresses this problem through the determination of inter-application information flow requirements via an analysis of (i) application method ontologies and (ii) application software tool ontologies. The tool describes the overall ontology driven approach and the inter-method ontology mappings that drive the inter-tool information flow requirements. An example information integration scenario is outlined in order to illustrate the practical application of our approach. Lastly, we summarize the research and outline the benefits.

1 MOTIVATION / PROBLEM DESCRIPTION

The complexity of managing process-oriented applications in large organizations requires that work be distributed to different functional areas and managed by smaller and more versatile cross-functional teams within the larger organization. Recent years have seen the development of sophisticated software tools that support decision, design, analysis, and other activities among these cross functional teams, and these tools have greatly enhanced the effectiveness of these activities. However, the distribution of work in such a fashion generates a new problem: the necessity of *sharing* information among the different application contexts within the different functional areas managed by these teams, and among the different tools supporting the activities performed by those teams. Hence, the usefulness of such tools in the sort of distributed environment required by a complex system is a function of the degree to which those tools (and the agents that use them) can share information across their different contexts. Typically, however, the data produced by software of this kind is maintained in closed architecture databases. In the overwhelming majority of cases, each software tool has its own private data repository.

A complex representation (e.g., a simulation model or a finite capacity scheduling model) carries the information it does by virtue of some established, systematic connection between the components of the representation and the real world. It is this connection that determines the semantic content of the data being represented. Typically, however, the semantic rules of a representation system for a given application and the semantic intentions of the application designers are not advertised or in any way accessible to other agents in the organization. This makes it difficult, even impossible, for such agents to determine the semantic content of a database. We refer to this as the problem of *semantic inaccessibility*.

This problem manifests itself superficially in the forms of unresolved ambiguity (as when the same term is used in different contexts with different meanings) and unidentified redundancy (as when different terms are used in different contexts with the same meanings). But these are just symptoms; the real problem is how to *determine* the presence of ambiguity and redundancy in the first place. That is, more generally, how is it possible to access the semantics of process-oriented data across different contexts? How is it possible to fix their semantics objectively in a way that permits accurate interpretation by agents outside the immediate context of this data? Without this ability, the kind of coordination between multiple applications and sub-systems necessary for *effective enterprise process management* is not possible.

Previous approaches to mitigating the problem of inter-operable simulations address syntactic interoperability. Research attention has been focused on the larger problem of modeling and simulation *composability* (Davis and Anderson 2003; Petty and Weisel 2003). A metadata approach to modeling and simulation information exchange for military simulation

was described in (Morse, et al. 2003). The Extensible Modeling and Simulation Framework (XMSF) adopts a standard language-based approach to facilitate simulation interoperability (Brutzman et. al 2002). Missing are useful methods and tools for addressing simulation based application interoperability at the semantic level.

In this paper, we describe an approach that addresses the above problem through the determination of inter-application information flow requirements via an analysis of (i) application method ontologies and (ii) application software tool ontologies. Our research focuses on the problem of information sharing between *process-oriented applications*: applications that depend significantly on the use and manipulation of enterprise *process* or *behavioral* information. Process-oriented application method types that are within the scope of our research include finite capacity scheduling, discrete event simulation, activity costing, and project management.

2 ONTOLOGY-DRIVEN FRAMEWORK SOLUTION APPROACH

Key concepts that underlie our architecture are as follows.

2.1 Inter-Application Ontology Mappings to Determine Information Integration Requirements

The use of ontology mappings as the primary mechanism for discovering inter-application information flow requirements. Two types of ontology mappings were investigated:

- Type I: Mappings between the process-oriented application methods (scheduling, simulation, and activity costing); and
- Type II: Mappings between the application software tools that are used to implement the application methods.

This paper will describe the representative Type I mappings that we designed.

2.2 Neutral Process Language for Inter-Application Translators

The use of a vendor-neutral process language as the basis for building translators between process-oriented applications. This approach has similar motivations as the Process Specification Language (PSL) standard under development by the National Institutes of Standards Technology (NIST) [http://www.mel.nist.gov/psl/]. In our research, we use the IDEF3 Process Modeling Language as the vendor-neutral process language [www.idef.com].

The conceptual architecture of our ontology-driven framework from process-oriented applications is shown in Figure 1.

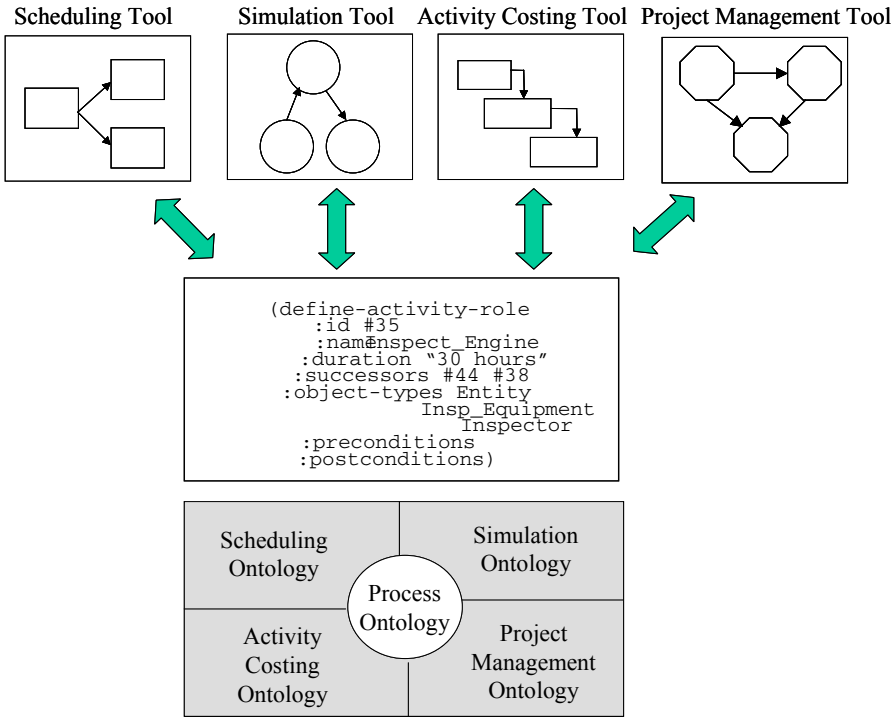

Figure 1: Ontology Driven Framework for Process-Oriented Applications

Translators enable the information flow between the different process-oriented application such as scheduling, simulation, activity costing, and project management to a "neutral" process language, IDEF3. The advantage of using a neutral language to facilitate information flow between multiple languages is that it provides translation efficiency. The efficiency gain in a generalized application integration situation involving 'n' applications is illustrated in Figure 1.

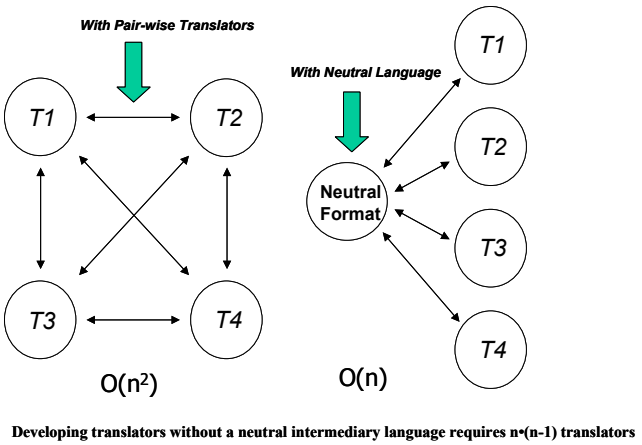

Figure 2: Using a Neutral Language to Increase Application Integration Efficiency

Determining the information flow requirements for these inter-tool translators occurs through an analysis of the application (method and tool) ontologies shown at the bottom of Figure 1. The IDEF3 "neutral" process language depicted at the center of the figure facilitates the *efficient* transfer of information between multiple (process-oriented) tools. The absence of a neutral (i.e., vendor-independent) process language would require the design and development of pair-wise translators between the tools, leading to additional translator development and maintenance effort. Once the neutral language has been designed, the translator design involves the development and analysis of the ontologies of (i) the different methods and (ii) the different tools. A key step is to determine mappings between the different method and tool ontologies (a formal specification of their concepts). The steps involved in implementing our ontology-driven approach are described in the following list.

1. Determine Inter-Method Mappings: This activity involves (i) developing an ontology of the different application methods that are within the scope of the application integration effort and (ii) analyzing the ontologies to identify the mappings between the ontologies. The method ontologies that are the focus of the mappings described in this paper are (i) Process Modeling, (ii) Finite Capacity Scheduling, and (iii) Simulation.
2. Determine Inter-Tool Mappings: This activity involves (i) developing an ontology of the specific software tools used to implement the application methods entailed by the integration effort and (ii) analyzing the tool ontologies to identify the mappings between the tool ontologies.
3. Determine Inter-Tool Information Flow Requirements: In this activity, the results of the inter-method and inter-tool ontology mappings are used to derive the meaningful information flows among the applications that need to be integrated.
4. Design Tool Translators: In this activity, the software translator between the different application tools is designed. Each translator may be (i) one-way or (ii) two-way according to the specific needs of the target enterprise application scenario.
5. Build and Test Translators: This activity involves (i) building the translators designed in step 4 and (ii) testing the application with test models and test data.
6. Use the Translators in the Context of Process-Oriented Integrated Enterprise Application Scenarios: This final activity refers to the operational use of the translators in the context of actual application executions.

The next section describes a set of inter-method mappings and a strategy for determining the inter-tool mappings.

3 PROCESS METHOD AND TOOL MAPPINGS

This section summarizes the mappings between three (process-oriented) methods: (i) Process Modeling, (ii) Simulation Modeling, and (iii) Finite Capacity Scheduling. We also outline a strategy for determining mappings between the corresponding tools that provide automated support for these methods.

3.1 Method Mappings

Table 1 summarizes the concept (ontology) mappings between process modeling, simulation modeling, and finite capacity scheduling (for commonly re-occurring concept types).

Table 1: Inter-Method Mappings: Process Modeling, Simulation Modeling, and Finite Capacity Scheduling

Process Concepts	**Simulation Concepts**	**Scheduling Concepts**	**Discussion**
Flow Object, Participant Object	Entity	Item, Part	The inputs and outputs for activities are declared explicitly in process and simulation models; activity inputs and outputs are often implicit in a scheduling model.
Process	Process, Activity	Activity, Task	Differences between the process, simulation, and schedule models are observed in the types of attributes that are relevant for each method and in the level of abstraction that is typically adequate to address the modeling goals.
Waiting Space	Queue, Buffer	N/A	Because simulation is often used for the analysis of queue behaviors, waiting spaces are often modeled in greater detail in simulation than in process modeling. It is not common to explicitly model queues in schedule models.
Agent, Resource Resources are often classified in different ways: (i) Dedicated vs. Shared; (ii) Consumable vs. Non-Consumable; (iii) Human, Equipment, Facilities, etc.	Resource	Resource	Differences in how resources are modeled in process, simulation, and schedule models are observed in the types of attributes that are relevant for each method and the level of abstraction that is typically adequate.
Intra-Activity Constraints (Resource, Timing, etc.)	Intra-Activity Constraints are often expressed as Resource - Activity Dependencies and Activity Time Specifications (Often Stochastic)	Intra-Activity Constraints are often expressed as Resource - Activity Dependencies, Activity Time Specifications (Usually Deterministic), and Calendar Constraints	Differences occur in both the constraint types and in the level of constraint specification detail.
Inter-Activity Logical Constraints (a) Input – Output Dependencies (b) Convergence (Fan-In) and Divergence (Fan-Out) Dependencies	Inter-activity logical Constraints (a) Push and Pull Interactivity Flow Dependencies (b) Convergence often manifests as Assembly constraints and Divergence often manifests as Disassembly Constraints	Inter-Activity Logical Constraints (a) Task Input and Task Output Specifications (b) Convergence often manifests as Logical "AND" convergence constraints and Divergence often manifests as Task Parallelism (divergent "AND") Constraints	Differences occur based on whether the representation of a flow object ("entity") is implicit or explicit. We have observed that flow objects are often implicit in schedule models and almost always implicit in simulation models. The modeling of the assembly and dis-assembly of physical systems is often common in simulation models and less common in schedule models.
Inter-Activity Temporal Constraints (Precedence Constraints)	Inter-Activity Temporal Constraints (Precedence Constraints)	Inter-Activity Temporal Constraints (Precedence Constraints)	Differences occur in both the constraint types and in the level of constraint specification detail.

3.2 Inter-Tool Mapping Development Strategy

Once the inter-method ontology mappings have been developed, an important next step is to determine mappings between the tools that support these methods. We will summarize our strategy for developing these mappings based on an ongoing research and development project called TEAMS (Toolkit for Enabling Adaptive Modeling and Simulation) (Benjamin, Graul, and Erraguntla 2002). TEAMS facilitates space transportation system operations process analysis using multiple analysis methods including simulation, scheduling, and cost analysis. The use of a standard and expressively rich process modeling language, IDEF3, provides the basis for the rapid generation of analysis models. The PROSIM® commercial tool provides automated support for IDEF3-based process modeling. Automated support for generating different types of analysis and execution support models has been implemented: (i) discrete event simulation (Arena and Witness) models, (ii) scheduling models (WorkSim and MSProject) models, and (iii) cost (SMARTABC® and SMARTCOST®) models. Additional analysis tool interfaces are under development to facilitate rapid and cost effective space transportation system operations analysis. The TEAMS process-oriented, re-configurable, plug-and-play analysis framework solution concept is illustrated in Figure 3 (Benjamin, Graul, and Erraguntla 2002).

Suppose that tools T1 and T2 need to exchange information and that their ontologies are TO1 and TO2, respectively. Further, we'll refer to the IDEF3 Neutral Process Representation Language as NPRL. The strategy for determining the information flow requirements between T1 and T2 is summarized in the following steps.

1. Determine TO1 <-> NPRL mappings
2. Determine TO2 <-> NPRL mappings
3. Use (1) and (2) to determine TO1 to TO2 mappings

In addition to the translator efficiency gains described earlier in Section 2, the advantage of using NPRL in this process is that it effectively assists with the *conceptual disambiguation*. The generic concept descriptions (NPRL) provide a reference point that is unaffected by terminological or implementation-specific ontology differences and similarities. For example, the NTRL might use the term "Unit of Behavior (UOB)" to generically denote terminological variants of this concept such as "Activity," "Task," and "Operation."

The disambiguated inter-tool ontology mappings that result from step 3 (above) provide the foundation for developing the inter-tool (TO1 <-> TO2) information exchange requirements.

In the TEAMS framework, we've established the technical viability and practical benefits of the above strategy by developing translators between the ARENA simulation tool and the WorkSim scheduling tool using IDEF3 as the intermediary NPRL (Benjamin, Graul, and Erraguntla 2002).

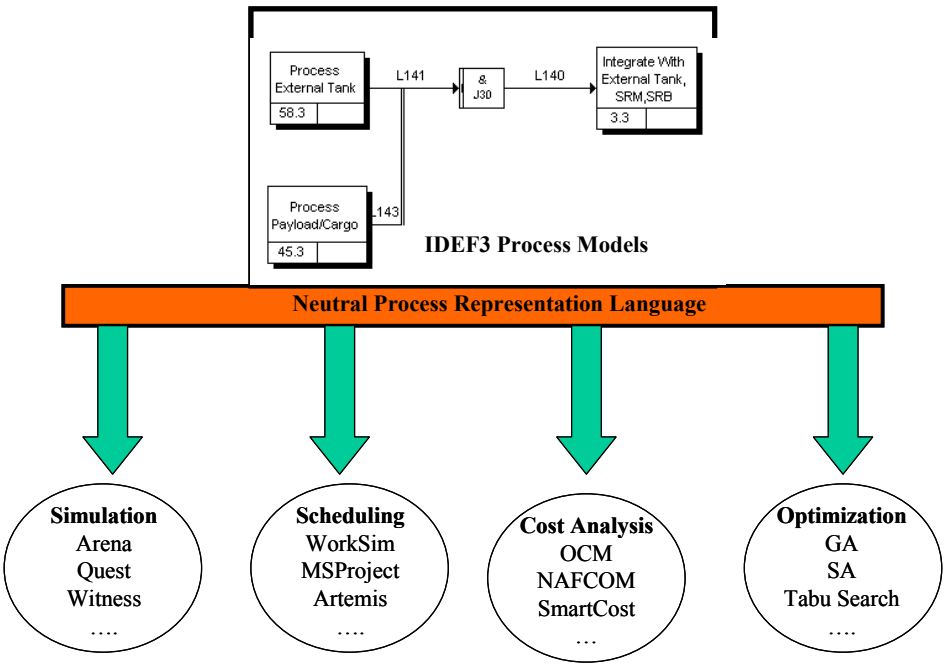

Figure 3: The TEAMS Process-Centric Operations Analysis Framework Solution Concept

4 EXAMPLE ONTOLOGY-ENABLED PROCESS INFORMATION INTERCHANGE APPLICATION

This section describes an example application scenario to illustrate the working of the ontology driven application integration framework approach. The example is based on a simulation-based application that was developed in (Benjamin, Graul, and Erraguntla 2002).

4.1 Background

The goal of the application was to answer the following types of questions: (i) What is the estimated throughput of the current process (estimated number of items processed per year, estimated average processing time, and estimated processing time variability)?; (ii) What is a feasible schedule for executing this process given a predetermined (a) start date and (b) end date? Simulation modeling is used to address questions of type (i) and finite capacity scheduling is used to address questions of type (ii).

In our example application, we studied a space transportation system ground operations process. The scope of the model encompasses the existing facilities, Ground-Support Equipment (GSE), and range infrastructure along with the flight hardware elements. The following activities were performed in order to answer the application questions.

1. Develop IDEF3 Process Model
2. Design and Generate Simulation Model
3. Design and Generate Schedule Model
4. Perform Simulation Experiments
5. Analyze Simulation and Schedule Model Outputs

4.2 Example Ontology Mapping Descriptions

The purpose of the example description is to illustrate the working of the ontology-driven approach for process application integration. We will focus on explaining the inter-model and inter-tool ontology mappings. The inter-tool mappings are described relative to three tools: the PROSIM® process modeling tool, the Arena simulation modeling tool, and the WorkSim scheduling tool.

4.2.1 Flow Object Mappings

Figure 4 represents a process flow network in PROSIM® with two flow objects / participants objects: ELV ET Barge and RLV ET Barge. Flow objects are characterized by simulation-specific information such as inter-arrival time, batch size, and arrival point. Active flow objects are introduced into the system and assigned to a process upon arrival.

Figure 5 represents the "corresponding" process flow network in Arena with two flow objects ("entities"): ELV ET Barge and RLV ET Barge. In addition to the process information specified in PROSIM®, Arena allows for the representation of additional information such as the maximum number of entities generated, the time when the first entity is generated, and time units. Active entities are always generated at the system entry point, and these entities are removed from the system at the system exit point.

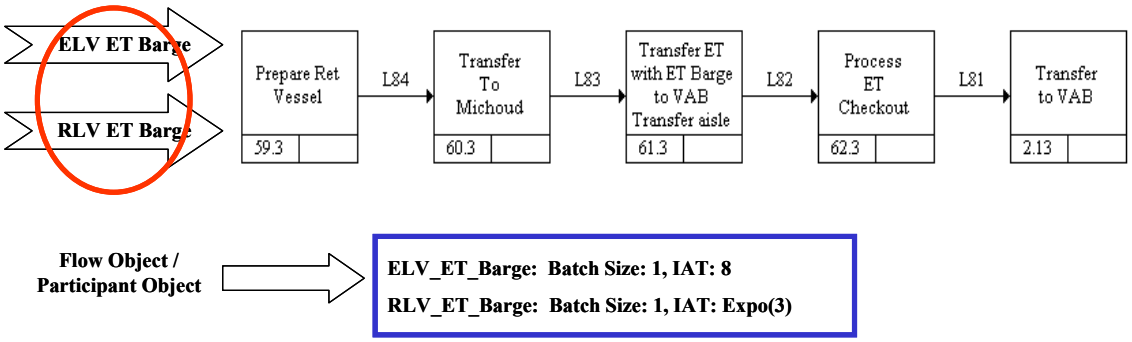

Figure 4: Flow Object Representation in PROSIM®

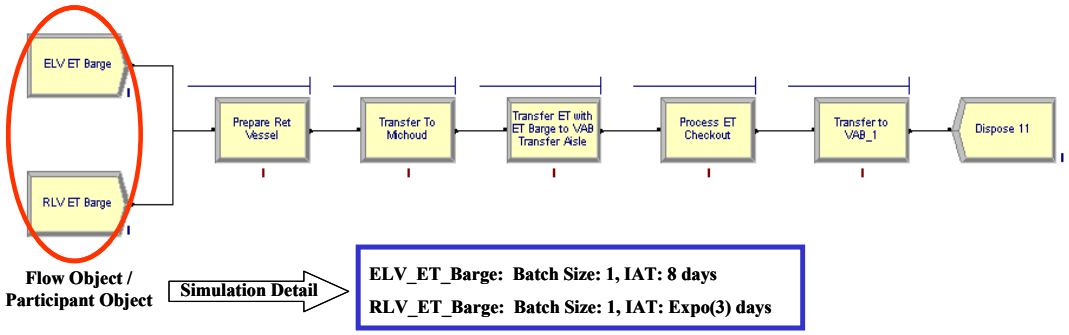

Figure 5: Flow Object Representation in Arena

Flow objects are represented as items (or parts) in WorkSim. Some of the information related to simulation (and represented in PROSIM® and Arena) is irrelevant to scheduling and thus not represented in WorkSim. Other information about the flow object (item) is relevant for scheduling / WorkSim such as the scheduled start date (for forward scheduling) or desired end date (for backward scheduling). This type of information is often not represented in process and simulation models.

4.3 Intra-Activity Constraint Mappings

Figure 6 shows how resource rules and waiting spaces are assigned to activities in PROSIM®. Resource rules are used to represent the restrictions on the use of resource objects with flow objects at an activity. Waiting spaces, the physical areas where flow objects wait for resources, are assigned to the activity. Multiple activities may be assigned to a single waiting space. Different types of dispatching rules may be assigned to waiting spaces such as first-in-first-out, last-in-first-out, etc.

Unlike PROSIM®, both resource rules and waiting spaces are assigned to processes in Arena. Queues / buffers are inherently tied to processes, but the user has the option of assigning multiple processes to a single queue. The processing times are categorized as value added, transfer, wait, and non-value added. Unlike PROSIM®, Arena allows for the grouping of similar resources skill sets. This feature is not available in PROSIM®.

The concept of waiting spaces is not commonly used in scheduling (and not currently represented in WorkSim). However, WorkSim provides for the representation of different types of resource rules. Unlike Arena, processing times in WorkSim are not tagged as value added, transfer, wait, etc.

4.4 Agent / Resource Mappings

PROSIM® allows for the specification of information such as system-wide capacity and per-hour resource use cost for agents / resources. Arena and WorkSim allow for the representation of additional information such as calendar constraints, capacity and efficiency exceptions, categories based on skills/capabilities, etc.

4.5 Inter-Activity Constraint Mappings

Figure 7 illustrates the representation of inter-activity convergence (Fan-In) and divergence (Fan-Out) constraints in PROSIM®. PROSIM® Fan-In and Fan-Out "Junction Boxes" are used to represent different logical inter-activity constraints such as "And (&)," "Exclusive Or (X)," and "Inclusive Or (O)." The process flow diagram in Figure 7 illustrates examples of Fan-In and Fan-Out "&" constraints.

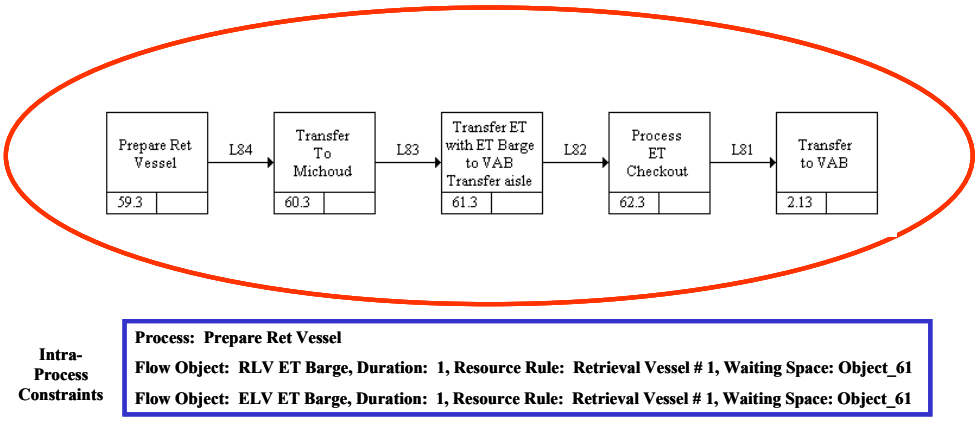

Figure 6: Process, Waiting Space, and Resource Rule Representation in PROSIM®

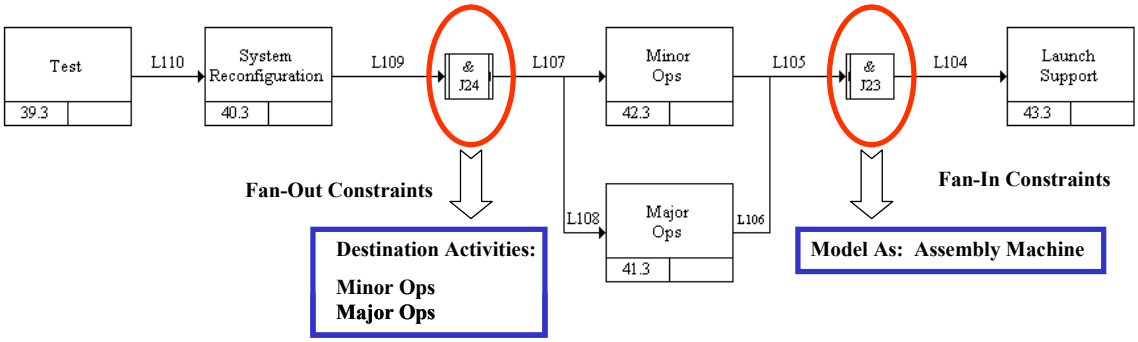

Figure 7: Representing Fan-In and Fan-Out Constraints in PROSIM®

Both convergence (Fan-In) and divergence (Fan-Out) constraints may be represented in Arena. Arena also allows for the specification of additional details such as the cost associated with cloning and assembly based on user-defined entity attributes (e.g., perform assembly operations on entities whose weight is less than 10 lbs, etc.).

WorkSim allows for the representation of "And (&)"inter-activity constraints types. "Exclusive Or (X)" constraints are not currently supported by WorkSim.

5 SUMMARY

This paper described an ontology driven approach to facilitating semantic information sharing among process-oriented applications. A key idea is the use of a Neutral Process Representation Language (NPRL) to increase the translation effectiveness among applications that must share information. The role of domain and tool ontologies in determining translation information requirements was described. Attention was focused on determining mappings between Process Modeling, Simulation Modeling, and Finite Capacity Scheduling method ontologies. The practical benefits of the approach were illustrated through a space transportation system ground processing operations analysis example.

ACKNOWLEDGMENTS

The research described in this paper was partially supported by NASA, contract number NAS10-02040, and the Office of Naval Research, contract number N00014-05-C-0072.

REFERENCES

Benjamin, P., Graul, M., and Erraguntla, M. 2002. Methods and tools for aerospace operations modeling and simulation: Toolkit for enabling adaptive modeling and simulation (TEAMS). *WSC '02: Proceedings of the 2002 Winter Simulation Conference,* San Diego, California, 763-771.

Brutzman, D., Zyda, M., Pullen, M. J., and Morse, K. J. 2002. Challenges for Web-based modeling and simulation, *Findings and Recommendations Report: Technical Challenges Workshop, Strategic Opportunities Symposium*, October 22, 2002. The MOVES Institute, Monterey, CA.

Davis, K. and Anderson R. H. 2003. *Improving the composability of Department of Defense models and*

simulations, RAND National Defense Research Institute, Santa Monica CA, 2003.

Morse, K. L., M. D. Petty, P. F. Reynolds, W. F. Waite, and P. M. Zimmerman. 2004. Findings and recommendations from the 2003 Composable Mission Space Environments Workshop," *Proceedings of the 2004 Spring Simulation Interoperability Workshop*, 313-323, Arlington VA.

Petty, M. D., and E. W. Weisel, 2003. A composability lexicon, *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 181-187, Orlando FL.

AUTHOR BIOGRAPHIES

PERAKATH BENJAMIN, Ph.D., a Vice President at Knowledge Based Systems, Inc. (KBSI), manages and directs the R&D activities at KBSI. He has over 18 years of professional experience in systems analysis, design, development, testing, documentation, deployment, and training. Dr. Benjamin has a Ph.D. in Industrial Engineering from Texas A&M University. Dr. Benjamin has been responsible for the development of process modeling, software development planning, and simulation generation tools that are being applied extensively throughout industry and government. At KBSI, Dr. Benjamin was the principal architect on an NSF project to develop intelligent support for simulation modeling that led to the development of the commercial simulation model design tool, PROSIM®.

KUMAR V. AKELLA, Ph.D. a researcher at Knowledge Based Systems, Inc. (KBSI), received a Ph.D. in Mechanical Engineering from Texas A&M University. Dr. Akella's areas of expertise include simulation modeling, design of experiments, numerical modeling, data mining, and text mining. His current work at KBSI includes simulation modeling and analysis, planning and scheduling, enterprise modeling, computer based training, and ontology development.

KAISER MALEK, a systems analyst at Knowledge Based Systems, Inc. (KBSI), received a Master's degree in Industrial Engineering from Oklahoma State University. Mr. Malek has been working as a systems developer at KBSI since 2001. Mr. Malek's responsibilities include analysis, design, and development of software applications, simulation and modeling of business processes, web-based technology for software development, enterprise integration, and project coordination.

RONALD FERNANDES, Ph.D. a senior research scientist at Knowledge Based Systems, Inc. (KBSI), received a Ph.D. in Computer Science from Texas A&M University (TAMU). His areas of expertise include data mining, neural networks, process modeling, knowledge management, workflow systems, the design of large-scale computer software, parallel and distributed computing, communication / network protocols, network design and analysis, and network management. Dr. Fernandes directs and manages the commercial software development group at KBSI.