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

USING ONTOLOGIES FOR SIMULATION INTEGRATION

Perakath Benjamin
Kumar Akella
Ajay Verma

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

ABSTRACT

This paper describes the motivations, methods, and solution concepts for the use of ontologies for simulation model integration. Ontological analysis has been shown to be an effective initial step in the construction of intelligent systems. However, the modeling and simulation community has not taken advantage of the benefits of ontology management technology. Moreover, the popularity of semantic technologies and the semantic web has provided several beneficial opportunities for the modeling and simulation communities of interest. The paper outlines the technical challenges in simulation integration and describes an ontology-based method that addresses these challenges. An example military combat simulation application scenario is used to illustrate the practical benefits of the simulation model integration approach.

1 MOTIVATIONS

A key motivation for distributed, federated simulation modeling is to allow for the decomposition of the target system into smaller, more manageable components and to distribute the model development effort among different organizations and functional groups. Once the component simulation models have been developed, there is a need for mechanisms to assemble a simulation model of the entire target system in a manner that ensures accomplishment of system performance goals. The U.S. federal government's ongoing investments in distributed simulation and initiatives such as the High Level Architecture (HLA) have been motivated significantly by this idea. The idea of distributed simulation is also critical to the success of Simulation Based Acquisition (SBA) for federal acquisition programs.

Designing and building a federated simulation involves several inter-related activities, including the following activities.

1. Members of the simulation modeling team must *agree* upon a core "community of interest" ontology.
2. The distributed and negotiation-based nature of the simulation model design process requires the sharing of knowledge that is described at different levels of abstraction using a variety of terminology sets / ontologies and housed in a heterogeneous collection of simulation tools / databases.
3. Simulation application designers lack of a common set of template reference simultaion models (i.e., a "simulation model design handbook") that has generally accepted validity and well defined criteria for application.
4. The need to reuse pre-existing ("legacy") simulation models, data, and tools.
5. The need to fuse and harmonize information at multiple levels of abstraction.
6. The need to faciliate semantic information exchange between different simulation models, tools, and databases.

The research described in this paper addresses the technology requirements implied by the above activities. We now outline the challenge of component based simulation and composability—key ideas that motivate our work.

1.1 The Challenges of Component-Based Distributed Simulation and Composability

From a software engineering perspective, a "component" is an independently deployable set of software services. The component-based approach to software development is an area of intense research and development in the information technology community. This approach seeks to provide the mechanisms that will allow developers to assemble software systems from collections of re-usable parts or building blocks. This approach has been very successful in the hardware world as evidenced by the "plug and play" hardware peripherals for computers. The path to success has been considerably slower in the software arena owing to the inherent complexity of software. The convergence to a relatively robust set of "standard" languages and protocols in the past few years augurs well for increasing plug compatibility in the software domain. The challenge of component-based software has not, however, been adequately addressed in the simulation community. In particular, lacking are methods and tools that address the hard technical challenges of semantic *interoperability* and semantic *composability*.

1.1.1 Key Challenge 1: Semantic Inaccessibility

The semantic rules of the component simulation tools and the semantic intentions of the component designers are not advertised or in any way accessible to other components in the federation. This makes it difficult, even impossible, for a given simulation tool to determine the semantic content of the other tools and databases in the federation, termed the *problem of semantic inaccessibility*. This problem manifests itself superficially in the forms of unresolved ambiguity and unidentified redundancy. But, these are just symptoms; the real problem is how to *determine* the presence of ambiguity, redundancy, and their type in the first place. That is, more generally, how is it possible to access the semantics of simulation data across different contexts? How is it possible to fix their semantics objectively in a way that permits the accurate interpretation by agents outside the immediate context of this data? Without this ability—semantic information flow and interoperability—an integrated simulation is impossible.

1.1.2 Key Challenge 2: Logical Disconnectedness

Even given a solution to the problem of semantic interpretability, however, a further problem impedes full cooperation among disparate systems/sub-systems. Suppose, for instance, we have determined that a certain representation R1 in a design model M1 is semantically equivalent to a representation R2 in a given analysis model M2, and that both R1 and R2 stand for the same entity. Thus, the models M1 and M2 both carry information about P. Suppose now that the information about P in M2 is updated. This requires a change in the information carried about P in M1. The fact that it is known that R1 and R2 are semantically equivalent in and of itself has no bearing whatever on whether the implications of the change in M2 will be propagated to M1. The problem in question is that the constraints between the particular pieces of information generated by various component tools within the simulation federation are rarely maintained. This is referred to as the *problem of logical disconnectedness*.

2 ONTOLOGY-BASED SIMULATION MODEL INTEGRATION METHOD

The principal hypothesis that underlies our method is as follows: *Semantic Information Exchange and Integration* for distributed simulation based applications will be effectively enabled using an *ontology-directed approach* that automates the mediation, sharing, and interchange of *semantic information* from multiple types of domain and simulation specification models (Benjamin et. al 2006a).

2.1 Ontologies and Ontological Analysis in Simulation Integration

A key assumption is that the simulation modeling environment has a set of pre-existing (and possibly partial) ontology models. Different types of ontology models have been found to be useful for simulation model integration as shown in Figure 1.

1. Domain Ontologies. These are representations of the knowledge in a well-circumscribed domain of interest. A domain might be organized based on natural phenomena (e.g., Biology, Physics, Geology) or based on man-made systems (e.g., Transportation, Telemetry, Semiconductor Equipment, etc.).
2. Community Of Interest (COI) Ontologies. These are knowledge models that are organized around communities of practice that share a common goal. Examples of Communities of Interest include Military Command and Control, Modeling and Simulation, Military Combat, Clinical Medicine, etc.
3. Simulation Tool Ontologies. These ontologies represent knowledge that is encoded within simulation modeling tools. These ontologies capture the (implicit or explicit) ontological commitments made by the designers of these simulation tools. In our research, we have studied the ontologies of several commercial simulation tools including ARENA, Witness, and FLAMES, and a few simulation tools used primarily within the military M&S community such as EAAGLES, AMBER, and JSAF.

Our experience indicates that it is possible to design the COI Ontologies using multiple domain ontologies. This design process involves the integration and harmonization of multiple, possibly overlapping domain ontologies as depicted in Figure 1. We have also identified the need to perform comparative ontology analysis that maps the Community of Interest Ontologies to the different Simulation Tool ontologies (for the set of simulation tools that are used in the context of a federated simulation exercise).

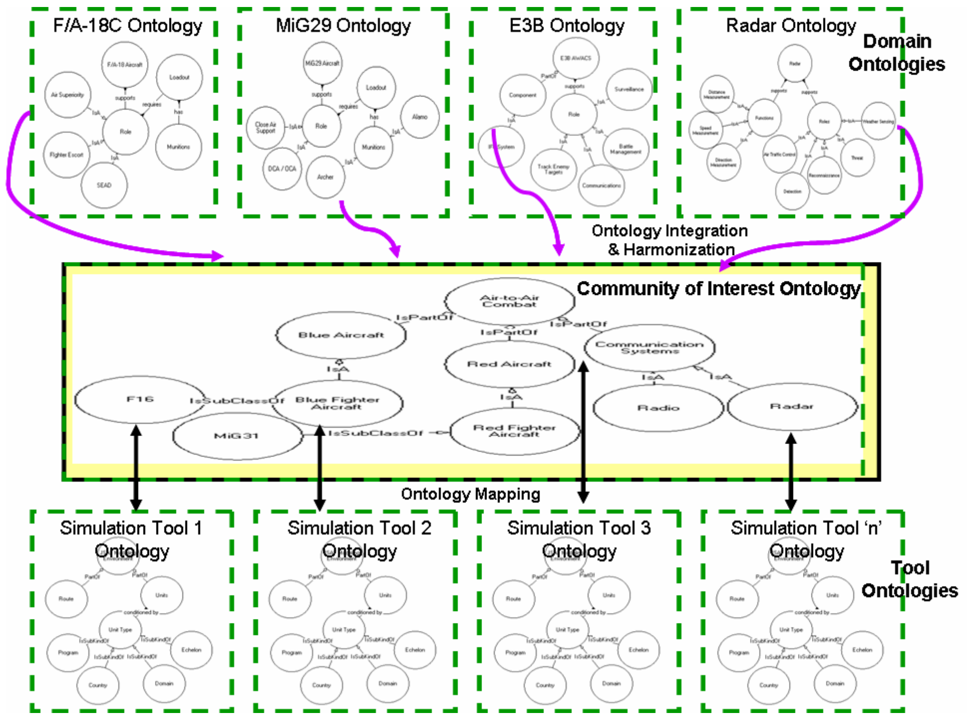


Figure 1. Ontology-based Simulation Integration involves Analysis Using Multiple Ontologies

A set of baseline ontology models will establish a mechanism for performing mapping / comparative analyses between the target simulation tool ontologies and reference Community of Interest (COI) ontology. The value derived through ontology model comparisons to determine potential integration mismatches is illustrated in Figure 2.

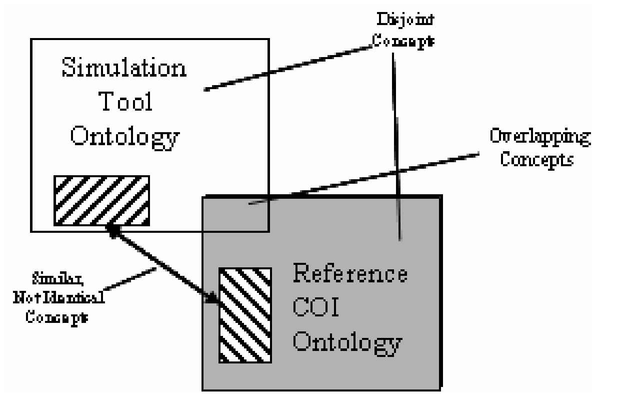


Figure 2. Using Ontology Comparison for Mismatch Assessment

Broadly, knowledge modeling mismatches are categorized as Language Level and Knowledge Level mismatches (see Klein 2001). Language Level mismatches occur because of differences in syntax and expressivity (of the languages used to represent multiple ontologies / applications). *Knowledge Level* mismatches are grouped into two broad categories: (1) Conceptualization Mismatches and (2) Explication Mismatches.

Conceptualization mismatches occur because of the differences in the *granularity* and *scope* of different ontologies (for example, a general ontology of space transportation resources versus an ontology of resources used for range system reconfiguration).

Explication mismatches are of three types: (i) Terminology, (ii) Modeling Style, and (iii) Encoding. *Terminology* mismatches arise because of the ambiguity of natural language and the naming conventions used by different organizations. The most common terminology mismatches are those caused by synonyms (using different words to refer to the same concept) and homonyms (the difference in meaning of the same word used in different contexts).

Modeling Style mismatches occur because of modeling paradigm differences (for example, using temporal logic based on time points versus time intervals) and modeling conventions (for example, to allow for the distinction between types and instances, and allowing for the representation of relational properties such as transitivity and symmetry).

Encoding mismatches occur due to formatting conventions (date formats) and differences in scale (units of measure differences, etc.).

2.2 Example Application Scenario

Before describing our methodology, we outline an example military combat simulation application scenario. This scenario is used to illustrate the method in the subsections that follow.

Figure 3 show an example tactical air combat scenario. In this case, two F/A-18C Hornets are pitted against two MiG-23 Floggers. The latitude and longitudes of the aircraft, MiG23 Orbit, trigger locations (CM1 and CM2), F/A-18 Orbit, and route for the BASE are depicted. Also represented in the figure are the aircraft call signs, initial altitude, initial heading, initial speed, and competency factor; radius of orbits (MiG-23 and F/A-18); distance between two trigger locations (CM1 and CM2); and distance between aircraft and nearest trigger locations. The F/A-18C fighters employ Electronic Suppression Measures (ESM) and radar to continuously track and monitor the position of MiG23 aircrafts. While the MiG23 aircrafts don't have ESM capability, their radars are active to locate F/A-18C fighters.

We now describe our method and illustrate its application using the above example scenario. Our Ontology-Based Simulation Model Integration Method involves four inter-related activities: (1) Assess Simulation Goals and Context, (2) Establish / Configure Baseline Models, (3) Perform Simulation Integration Assessment, (4) Determine Integration Information Flow Requirements, and (4) Generate Model Integration Advise.

2.3 Simulation Integration Method

The activities of the simulation integration method are described in greater detail in the following paragraphs.

2.3.1 Assess Simulation Goals and Scope

Preliminary and important steps in the development of the simulation models are to define the simulation modeling goals and the purpose, scope, and level of detail. Access to an explicit representation of simulation objectives, scope, levels of detail, experiment plans, and performance criteria would assist (by providing "context") with the downstream design and integration of the component simulation models. In particular, the results of this assessment may be used later to perform in-context interpretations of the component simulation models, the simulation data, and the different ontology models.

In our example simulation based training model scenario, the simulation goals and scope are expressed as follows:

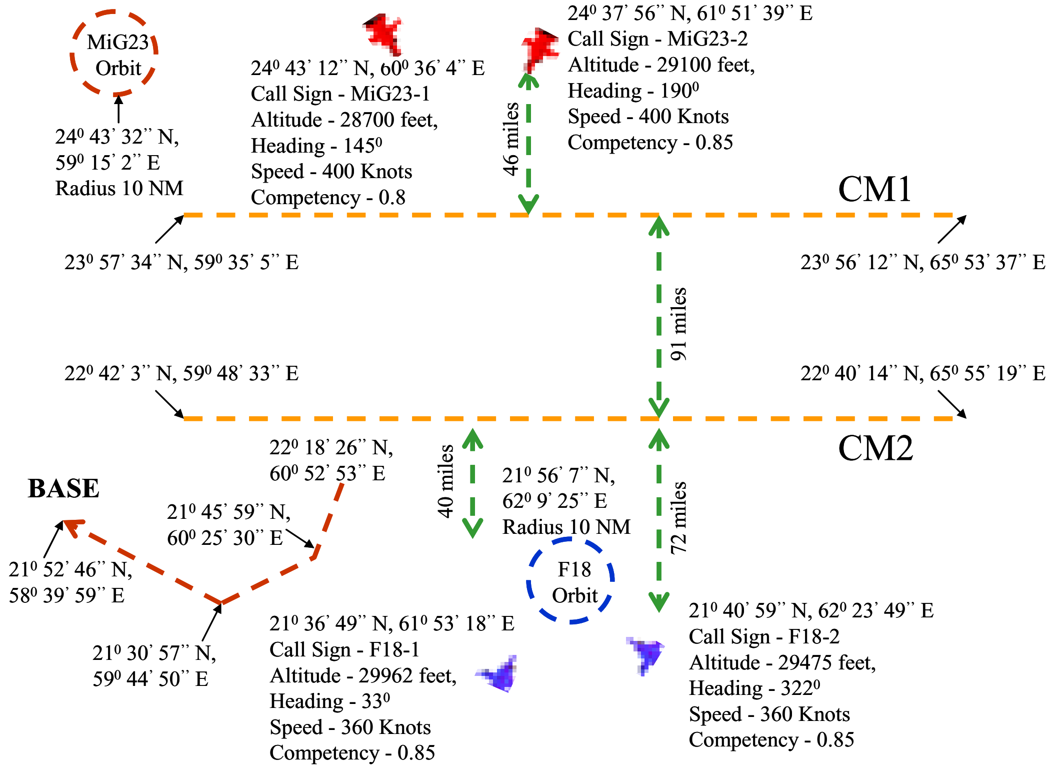


Figure 3. Example Tactical Air Combat Scenario

1. Enhance pilot skills in the areas of Target Acquisition, Communications, Situational Awareness, Weapons Employment, and Electronic Warfare for 2V2 DCA missions. Instructor observations include assessment of pilot performance in critical areas such as airspace management, sensor employment, and post engagement status communication. The performance factors (for pilot performance evaluation) include aircraft closure rate, weather, enemy electronic emissions, multiple targets, onboard weapons, and radar efficiency.
2. The reference ontology models (in this case, the COI reference ontology is the "military combat simulation ontology") is used to unambiguously interpret the terminology and concepts that are embodied within the description of the simulation goals (e.g., the meaning of the terms "Target Acquisition, Communications, Situational Awareness, Weapons Employment, and Electronic Warfare").

2.3.2 Establish and Allocate Model Requirements

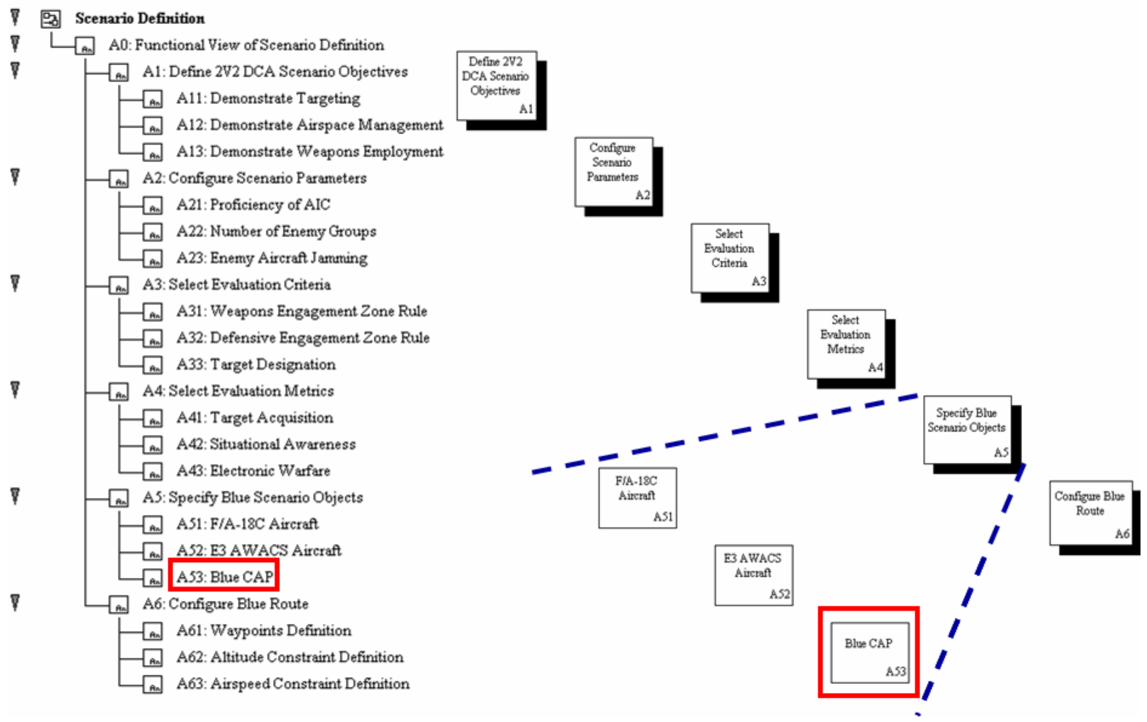
The next simulation activities are (1) determine the simulation modeling requirements and (2) allocate the requirements to the different component models within the simulation federation. In our example, the overall combat mission functions may be decomposed into (a) the Friendly Aircraft combat functions, (b) Enemy Aircraft combat functions, and (c) the Combat Environment functions. An IDEF0-based representation of the functional requirements (for the example scenario) is shown in Figure 4. Perform Simulation Integration Assessment

Simulation integration assessment involves four inter-related activities: (1) Perform Process Mismatch Analysis, (2) Perform Object Mismatch Analysis, (3) Perform Abstraction Analysis, and (4) Perform Data Mismatch Analysis. These activities are described in the following subsections.

2.3.2.1 Perform Process Mismatch Analysis

This activity identifies simulation integration and composability issues that arise because of inter-process mismatches. The analysis is performed across multiple processes in simulations targeted for integration. Multiple types of process comparisons have been developed, including the comparisons described in the following list.

1. Input – Output Inconsistencies: if the output of activity A is input to B, then there should be a precedence between A and B. If there is a precedence relationship between B and A, then the two information pieces (object dependencies and precedence) contradict each other.


Scenario Definition
A0: Functional View of Scenario Definition
A1: Define 2V2 DCA Scenario Objectives
A11: Demonstrate Targeting
A12: Demonstrate Airspace Management
A13: Demonstrate Weapons Employment
A2: Configure Scenario Parameters
A21: Proficiency of AIC
A22: Number of Enemy Groups
A23: Enemy Aircraft Jamming
A3: Select Evaluation Criteria
A31: Weapons Engagement Zone Rule
A32: Defensive Engagement Zone Rule
A33: Target Designation
A4: Select Evaluation Metrics
A41: Target Acquisition
A42: Situational Awareness
A43: Electronic Warfare
A5: Specify Blue Scenario Objects
A51: F/A-18C Aircraft
A52: E3 AWACS Aircraft
A53: Blue CAP
A6: Configure Blue Route
A61: Waypoints Definition
A62: Altitude Constraint Definition
A63: Airspeed Constraint Definition
Define 2V2 DCA Scenario Objectives A1
Configure Scenario Parameters A2
Select Evaluation Criteria A3
Select Evaluation Metrics A4
Specify Blue Scenario Objects A5
Configure Blue Route A6
F/A-18C Aircraft A51
E3 AWACS Aircraft A52
Blue CAP A53
Figure 4. Example Functional Requirements Analysis – an Important Step in Federated Simulation Model Design

2. Missing inputs: any input to an activity should either be produced by some other activity in the process or is an input to the process itself (external input). Here the inputs necessary for a given activity are calculated to be the inputs required for each task in the activity minus the objects produced internally within the activity.
3. Unused outputs: any object produced by an activity in a process should either be consumed by some other activity or should be the goal or desired output of the process itself (output of the process).
4. Rate mismatch: the rate of flow of “flow objects” out of a given activity must be compatible with the rate at which a downstream activity is expecting flow object arrivals.
5. Infeasible resource allocations: the sum of allocation of resources to the component model activities must be less than or equal to the overall model resource availability.

Figure 5 shows a detailing of our example scenario that illustrates the information exchange between two combat simulation tools. The illustration also identifies the types of data that must be exchanged in order to simulate the scenario.

Suppose that we use Simulation “Tool 1” to simulate the dynamics of multiple aircraft and Simulation “Tool 2” to mimic the performance of a ground based radar in the emerging scenario. Assume that several aircraft are airborne with friendly Blue aircraft and enemy Red aircraft. Suppose also that a friendly ground station is tracking various aircraft through radars that ultimately influences the strategy employed by Blue force in combating the Red force. In this example, Tool 1 simulates the Red and Blue forces, while Tool 2 simulates the Ground Based Radar system. Initially the Red force is outside the range of the ground based radars and hence the Blue force is operating without the knowledge of the whereabouts of the Red force. In time, as the Red force gets closer it becomes visible to ground based radar. The radar tracks both Blue and Red aircraft. Therefore, it is initially tracking all the Blue force aircraft as they move within range. As soon as the Red aircraft become visible, they are also tracked and their position and heading information are dynamically conveyed to Blue force. The dynamic information updates enable the Blue force to react in an appropriate manner as it engages the Red force in combat.

The process mismatch analysis information for the above scenario is illustrated in Figure 6. This example illustrates the types of mismatches that might occur as the Blue aircraft, the Ground Based Radar, and the Red aircraft communicate with each other during the simulation execution. In particular, the example shows mismatches between the following attributes: (1) Coordinate_System, (2) Time_Unit, (3) Distance_Unit, (4) Time_Step_Value, and (5) Reference_Frame.

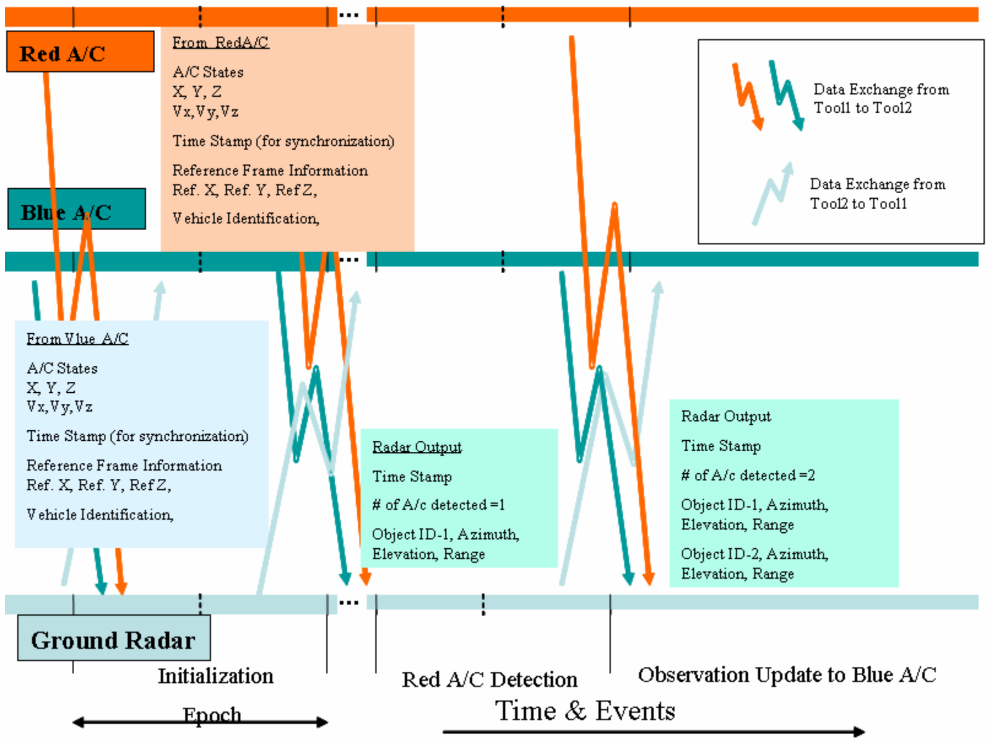

Red A/C
From RedA/C
A/C States
X, Y, Z
Vx,Vy,Vz
Time Stamp (for synchronization)
Reference Frame Information
Ref. X, Ref. Y, Ref Z,
Vehicle Identification,
Data Exchange from Tool1 to Tool2
Data Exchange from Tool2 to Tool1
Blue A/C
From Vlue A/C
A/C States
X, Y, Z
Vx,Vy,Vz
Time Stamp (for synchronization)
Reference Frame Information
Ref. X, Ref. Y, Ref Z,
Vehicle Identification,
Radar Output
Time Stamp
# of A/c detected =1
Object ID-1, Azimuth, Elevation, Range
Radar Output
Time Stamp
# of A/c detected =2
Object ID-1, Azimuth, Elevation, Range
Object ID-2, Azimuth, Elevation, Range
Ground Radar
Initialization
Red A/C Detection
Observation Update to Blue A/C
Epoch
Time & Events

Figure 5. Dynamic Information Exchange in a Federated Air-To-Air Combat Simulation Scenario

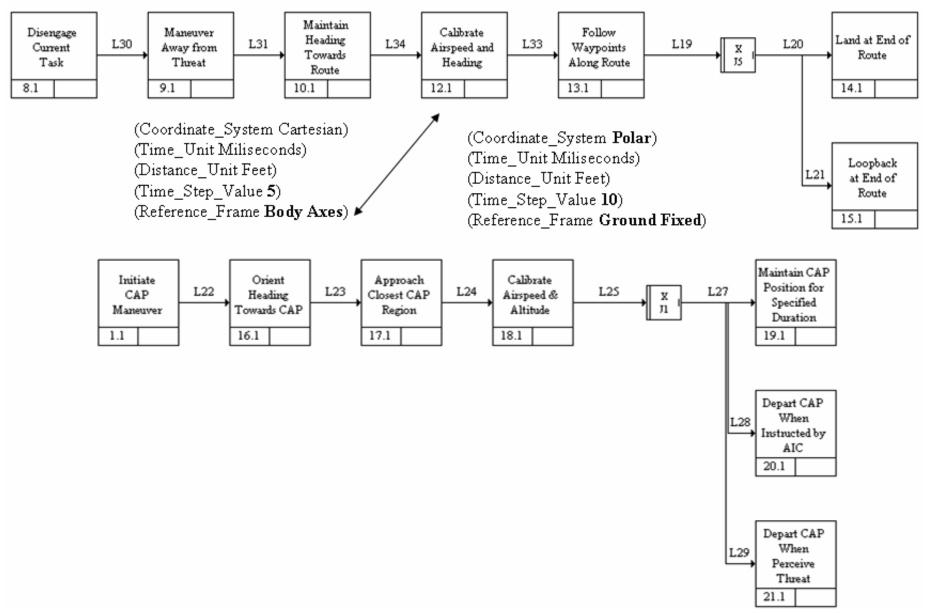

Disengage Current Task
8.1
L30
Maneuver Away from Threat
9.1
L31
Maintain Heading Towards Route
10.1
L34
Calibrate Airspeed and Heading
12.1
L33
Follow Waypoints Along Route
13.1
L19
X J5
L20
Land at End of Route
14.1
L21
Loopback at End of Route
15.1
(Coordinate_System Cartesian)
(Time_Unit Miliseconds)
(Distance_Unit Feet)
(Time_Step_Value 5)
(Reference_Frame Body Axes)
(Coordinate_System Polar)
(Time_Unit Miliseconds)
(Distance_Unit Feet)
(Time_Step_Value 10)
(Reference_Frame Ground Fixed)
Initiate CAP Maneuver
1.1
L22
Orient Heading Towards CAP
16.1
L23
Approach Closest CAP Region
17.1
L24
Calibrate Airspeed & Altitude
18.1
L25
X J1
L27
Maintain CAP Position for Specified Duration
19.1
L28
Depart CAP When Instructed by AIC
20.1
L29
Depart CAP When Perceive Threat
21.1

Figure 6. Process Mismatch Analysis Example

Ontological analysis helped facilitate the process mismatch analysis techniques described earlier in this section. In particular, the ontology mappings between multiple ontolologies identified the semantic and terminological differences of the information represented in the simulation process models.

2.3.2.2 Perform Object Mismatch Analysis

This activity identifies simulation integration issues that arise because of simulation object mismatches. This is accomplished by performing *comparative analyses* of the objects that reside within the different simulation models. We have developed different types of comparative analysis techniques including (1) Syntactic Analysis, (2) Topological Analysis, (3) Lexical Semantic Analysis, (4) Compositional Semantic Analysis, and (5) Information Fusion.

The above example illustrates the mismatches in the terminology and semantics of multiple Radar ontologies. Natural Language Processing (NLP)-based semantic analysis techniques are used to interpret and compare the text within the "Description" fields shown in the figure.

Perform Abstraction Analysis This activity will perform abstraction level mismatch analysis to (1) identify simulation integration issues that arise because of multiple levels of abstraction and (2) to determine information flow requirements between simulations that are at different levels of abstraction.

The *level of abstraction* of a model determines the amount of information that is contained in the model (Benjamin et. al 1998). The quantity of information in a model decreases with the levels of abstraction. Thus, a "low level abstraction" model contains more information than a "high level abstraction" model. To illustrate, consider the simple example shown in Figure 7. Model M transforms Input I to Output O. A decomposition of M into M1, M2, and M3 shows a detailing of input – output transformations that is hidden at the more abstract level. Thus, I1, I2, and I3 are transformed by M1, M2, and M3 to O1, O2, and O3, respectively, at the "lower" level of modeling abstraction. This shows how the quantity of information contained at the lower level is more than at the higher levels. Therefore, it is convenient to think of abstractions as a mechanism for selectively "hiding" information.

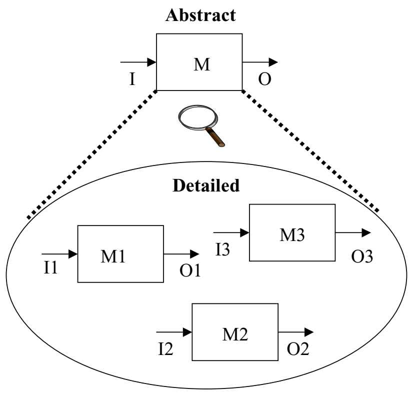


Figure 7. Abstractions in Modeling

Our abstraction analysis is used with different types of models: (1) Process Models, (2) Ontology Models, and (3) Information Models.

An essential ingredient of abstraction analysis is ontology-based reasoning. The focus Community of Interest (COI) Ontology Reference Model and the mappings between the COI ontology and the different component simulation tool ontologies are used to (1) interpret and (2) disambiguate (at the semantic level) the concept descriptions within the multi-level modes. We have designed several "abstraction analysis rules" that are used for the ontology-based abstraction analysis. These rules are described below.

Consistency with respect to mean: checks whether the abstraction parameter (e.g., time taken) is the same, on average, across different levels of abstraction. This involves identifying the beginning and the end of a given process and then navigating through each activity to calculate the total processing time.

Consistency with respect to variance of process duration: checks whether the variation in the abstraction parameter (e.g., time taken) is the same across multiple levels of abstraction. This involves calculating the variance of the parameter, performing F-tests for equality of variances, etc.

Object definition consistency (omission): checks for differences in object specifications across different levels of abstraction. This involves identifying objects in the detailed model that are not defined in the high-level model. This also involves evaluating object definitions/properties from the ontology models.

Object definition consistency (substitution): checks for differences in object specifications across different levels of abstraction. An object in the detailed model may exist in the high-level model but may be known by a different name, or may be a generalization. For example, the detailed model may distinguish different types of fighter planes but the high-level model may refer to all of them by one name. Again, this involves evaluating object definitions/ properties from the ontology models.

Variable unit mismatch: This rule checks if the variables used in the two models to be integrated are consistent with respect to units. Detailed models sometimes use a finer unit of time (e.g., hours or minutes instead of days or weeks).

2.3.2.3 Perform Data Mismatch Analysis

This activity identifies simulation integration issues that arise because of simulation data mismatches. This activity focuses attention on the *data models* used to represent information within the component simulation tools that are part of the federation. The data model representations covered by our mismatch analysis include (1) the Unified Modeling Language (UML) and (2) the Entity Relationship (ER) modeling language. The data mismatch analysis results are used to (1) diagnose potential simulation integration issues and (2) determine simulation integration (inter-tool) information flow requirements.

2.3.3 Assess and Harmonize Mismatch Analysis Results

The mismatch analysis results from the (1) process, (2) object, and (e) data perspectives are harmonized. The results of this activity are used to (1) mediate the information flows between the component simulation tools through mismatch resolution and (2) determine (execution-time) information exchange requirements between the component simulation tools. We have implemented an Ontology-Driven Translation Generation technique that uses the inter-tool ontology mappings to automate the generation of translators between the component simulation tools (see Benjamin et. al. 2006a).

3 SOLUTION APPLICATION EXAMPLE

We have designed a solution application framework that provides automated support for the simulation integration method. The solution is called the Ontology-based Simulation Integration Framework (OSIF) (Figure 8). OSIF provides a "visual programming environment" for rapidly composing, building, and maintaining federated simulations.

Central to the OSIF solution concept are (1) OSIF Model Libraries and (2) Reference Libraries. The "OSIF Model Libraries" contain template models that encapsulate structural and behavioral information that will allow for the rapid composition of simulation from re-usable component parts. The modeling procedure will involve selecting appropriate templates, editing the templates, and composing complex models by connecting multiple components together. The model libraries will include ontology templates, process (behavior) templates, information meta-models, design patterns, and example simulation models. The "Reference Libraries" refer to re-usable and extendible "domain models," including reference ontologies, reference process models, and reference information meta-models. The reference libraries will provide a mechanism for composing and integrating (at the semantic level) external simulation models and tools from other environments / vendors. The model Composition Workspace is used to browse and edit the model libraries and provide a graphical user interface for M&S application composition. The Composability Assessment Tools (CAT) are invoked to (2) diagnose mismatches between candidate models that are to be integrated and (2) generate model reconfiguration advice to revise model library components or candidate new models that need to be integrated into the OSIF. The Ontology Driven Translator Generator is used to generate translator code between external models / tools and the OSIF.

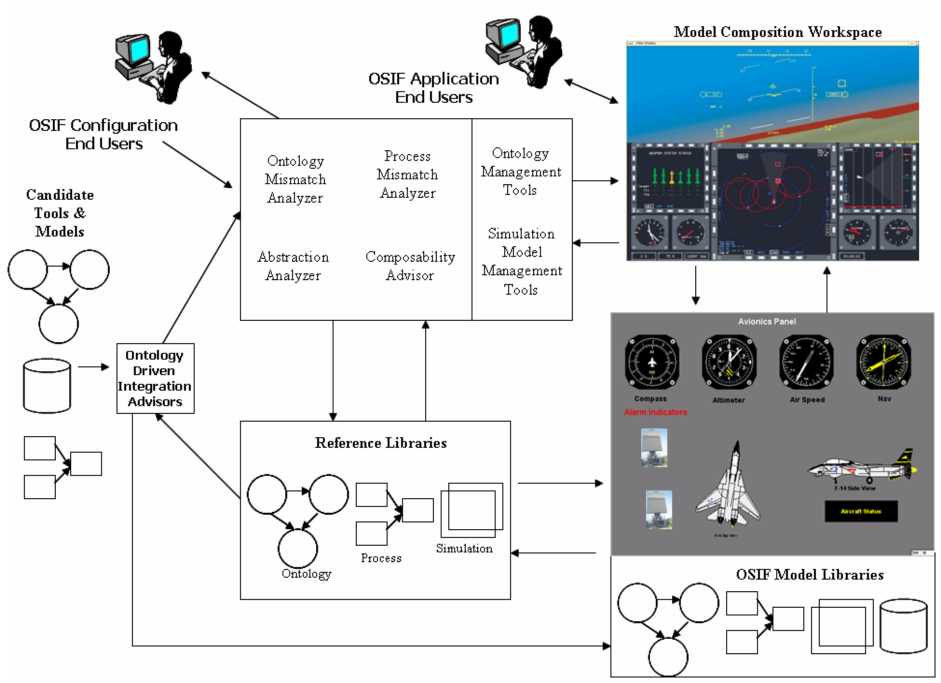

OSIF Configuration End Users
OSIF Application End Users
Model Composition Workspace
Candidate Tools & Models
Ontology Mismatch Analyzer
Process Mismatch Analyzer
Ontology Management Tools
Abstraction Analyzer
Composability Advisor
Simulation Model Management Tools
Ontology Driven Integration Advisors
Avionics Panel
Compass
Altimeter
Air Speed
Nav
Alarm Indicators
Aircraft Status
Reference Libraries
Ontology
Process
Simulation
OSIF Model Libraries

Figure 8. OSIF Conceptual Architecture

4 POTENTIAL BENEFITS

The principal short-term benefit of the solution described in this paper is the *significant reduction in time and effort needed for developing and deploying distributed simulation applications*. Other important benefits are (1) significant increases in the quality of information sharing and communications for distributed modeling and simulation applications, and (2) significant reductions in the time and effort for *semantic* knowledge sharing, communication, and *semantic integration* for modeling and simulation applications. We are currently working on designing and configuring multiple demonstration applications of this research that will facilitate the development and execution of distributed military simulations to support *Simulation Based Acquisition*. Longer term, we anticipate that this solution will provide a sustainable mechanism for building simulations that interoperate and share information at the semantic level.

REFERENCES

Benjamin, P.C., Erraguntla, M, Delen, D., and Mayer, R.,1998. Simulation Modeling at Multiple Levels of Abstraction, In *Proceedings of the 1998 Winter Simulation Conference*, Washington, DC. 1998.

Benjamin, P. and M. Graul (2006a). A framework for adaptive modeling and ontology-driven simulation. *Proceedings of SPIE, Enabling Technologies for Simulation Science X, Vol.6227*

Klein, M., Combining and Relating Ontologies: An Analysis of Problems and Solutions. Workshop on Ontologies and Information Sharing, pp. 53-62, In Proceedings of IJCAI – 2001, Seattle, WA. August.

Petty M. D., and E. W. Weisel, "A Composability Lexicon", *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, Orlando FL, March 30-April 4 2003, pp. 181-187, 2003.

AUTHOR BIOGRAPHIES

PERAKATH BENJAMIN is Vice President (R&D) at Knowledge Based Systems, Inc. (KBSI), and manages and directs the R&D activities at KBSI. Dr. Benjamin has been responsible for developing advanced technology that is being applied throughout industry and government.

KUMAR V. AKELLA is a research scientist at Knowledge Based Systems, Inc. (KBSI). Dr. Akella's areas of expertise include simulation modeling, ontology-based semantic search, computer based training, design of experiments, data and text mining.

AJAY VERMA is a research scientist at Knowledge Based Systems, Inc. (KBSI) since 2000. Dr. Verma has conducted extensive research and developed applications in dynamics and controls, simulation, data mining, system identification, signal processing, image processing, optimization, neural networks.