Proceedings of the National Aerospace & Electronics Conference (NAECON 2000)
Dayton, Ohio, USA 10-12 October, 2000

AGENT-BASED ARCHITECTURE FOR MODELING AND SIMULATION INTEGRATION

J. TODD McDONALD[v], MICHAEL L. TALBERT[w]

[v]Air Force Operational Test and Evaluation Center, HQ AFOTEC, Kirtland AFB, New Mexico 87117, USA, todd.mcdonald@afotec.af.mil
[w]Air Force Institute of Technology, AFIT/ENG, WPAFB, Ohio 45433, USA, michael.talbert@afit.af.mil

Abstract. The Department of Defense (DOD) has an extensive family of models used to digitally simulate the mission level interactions of weapon systems. Interoperability and reuse of the underlying data files used to create simulation scenarios are of particular interest to the modeling community. We develop an architecture to support simulation interoperability by combining three key technologies: object-oriented data modeling, an underlying persistence mechanism, and an agent-oriented analysis and design methodology. We use object-oriented modeling techniques to encapsulate and organize the syntactic information contained in scenario database files while we examine the semantic information of these objects for data integration purposes. The agent architecture provides a communication capability to support collaborative development and information brokering. We demonstrate our architecture by means of prototypical applications that implement the foundational information agent layer.

Key Words. Agents, Simulation Reuse, Model Integration, AOIS, SUPPRESSOR, MSFD, JIMM

1. INTRODUCTION

The Air Force[1] Research Laboratory (AFRL) is directing an effort to provide a collaborative computing environment to support simulation scenario reuse and integration. The requirements of this collaborative environment, known as CERTCORT (**C**oncurrent **E**ngineering for **R**eal **T**ime databases **COR**relation **T**ool), and its heterogeneous data integration problem are represented pictorially in Figure 1. We describe an agent-based architecture that incorporates object-oriented data modeling techniques (OMT), semantic information modeling, and persistent database technology to accomplish legacy scenario data integration and reuse while providing traceability to authoritative data sources in an automated fashion. We use concepts and techniques from the emerging field of agent-oriented information systems (AOIS) [1,2,3] to provide a framework for application development and an agent-centric lifecycle methodology known as MaSE (**M**ulti-**a**gent **S**ystems **E**ngineering) [4].

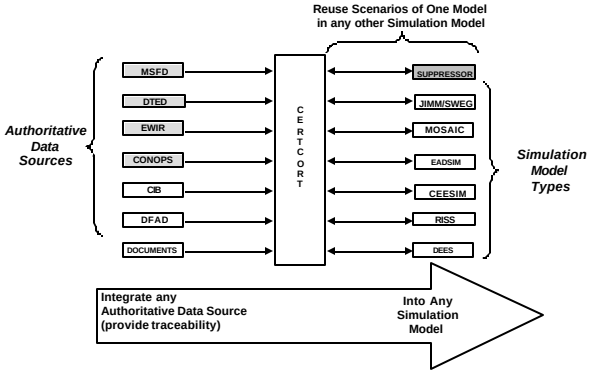

Figure 1: Heterogeneous Database Problem Domain

2. PROBLEM DOMAIN

Player-oriented military simulation models include among others the Extended Air Defense Simulation Model (EADSIM), the Suppressor Composite Mission Simulation System (SUPPRESSOR), the Joint Interim Mission Model (JIMM), and the Simulated Warfare

[1] The views expressed in this article are those of the authors and do not reflect the views of the United States Air Force, the Department of Defense, or the U.S. Government.

Environment Generator (SWEG). Two primary goals exist for integration and legacy scenario reuse within this realm. The first goal rests on the assumption that all models are based on some underlying real-world interaction. In terms of the aforementioned player-level simulations, authoritative data represents the real-world performance characteristics of both weapon systems and the human interaction required to use them.

The traceability of authoritative sources to their corresponding scenario representation is currently lost without manual correlation. Authoritative sources also come in widely different formats (syntactic representation) and with varying degrees of information (semantic content). A major goal of the CERTCORT effort is to provide automated correlation for simulation analysts as new scenarios are developed. Figure 2 pictorially represents a Multi-Spectral Force Deployment (MSFD) data file (an intelligence source detailing unit subordination relationship) and how it provides traceability to simulation specific instructions found in both a SUPPRESSOR and EADSIM scenario.

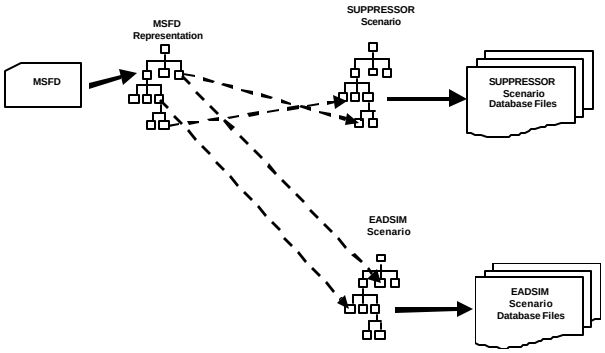

Figure 2: Source-to-Model Traceability

The second form of integration deals with reuse across simulation models themselves. This is where entities described in a model-specific grammar (such as SUPPRESSOR) are desired for reuse in another model-specific grammar (such as JIMM). It also describes direct translation of an entire scenario into a different model, while retaining the same weapons, tactics, and operations of the original scenario. This level of integration typically requires the translation of various data items from one simulation specific grammar to another. Such effort requires in-depth knowledge of both simulations and a manual error-prone method of subjective translation by an analyst.

Player-level simulations tend to have a "player" or "platform" definition that generically describes all possible systems from aircraft, naval, land, and space vehicles. As such, simulations vary in their ability to model terrain, communication, zones, and electromagnetic effects; likewise, scenario description languages (referred to as "grammars") vary in their resolution ability to capture such concepts.

Because grammars follow a pre-defined format, object-oriented modeling is well suited for analyzing and describing the contents of a given scenario. As such, objects encapsulate the text-based nature of scenario data files in a hierarchical manner. We use the term "syntactic" object model to mean a faithful representation of the grammar structure of a simulation. Likewise, the term "semantic" object model refers to associations, classes, and inheritance relationships derived from a syntactic model that provide higher levels of abstraction. Human analysts can work and conceive of scenarios in semantic terms much more easily than in the scenario-specific syntax of a grammar. It is this expert-knowledge of a simulation grammar that makes translation from model to model difficult, time consuming, and subjective. Figure 3 illustrates the concept of model-to-model translation in terms of abstractions. Integration can occur from a purely "syntactic" understanding of a scenario or from a more information-based method that relies on the reuse of "semantic" objects closer to real-world abstractions.

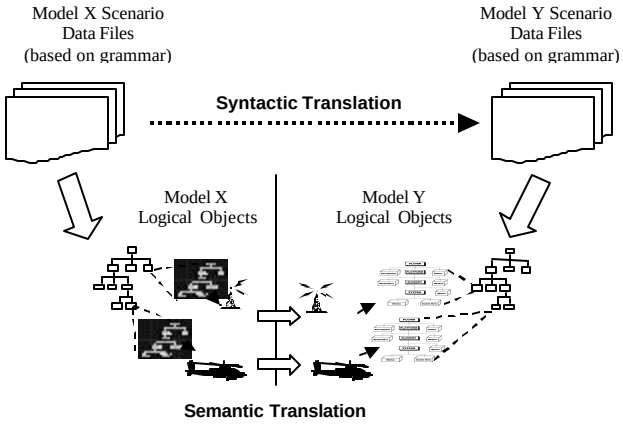

Figure 3: Model-to-Model Integration

We develop an architecture that supports integration of both authoritative source data to a family of simulation models (Figure 2) and a model-to-model integration (Figure 3) that allows an automated approach to scenario construction. Traditional data-centric approaches to model integration try to find schematic structures of a textual nature that are common to models within a given domain. Reuse is seen in terms of mapping schematic structures into a global schema

that provides the necessary translation from one model to another. We describe an approach to translation based upon common semantic objects that are found from information discovery techniques performed on underlying scenario data files. Objects play a key role in our understanding of ontology and our particular heterogeneous data integration strategy. Their role is briefly discussed next.

3. OBJECT-ORIENTED FOUNDATION

The information systems view of our architecture is not seen in terms of "how" or "where" scenario data is stored. Instead, a collection of "information agents" is seen to actively encapsulate scenario information that finds its source in flat-file, relational, or object-based formats. Agents perform the task of translating scenario and authoritative data sources into the common data model of our architecture: *objects*. In this sense, both the ontological definition and meaning of data within our system is seen in terms of objects used to encapsulate files and simulation grammars. The derivation of an object model is obtained by applying traditional object-oriented data modeling techniques (OMT) to the grammar definition and format definitions of simulation models and source authoritative data. Figure 4 illustrates the encapsulation process for a set of SUPPRESSOR scenario data files. The object representation of a grammar is referred to as the "syntactic" model while other derivable object models are seen as "semantic" views of this syntax model.

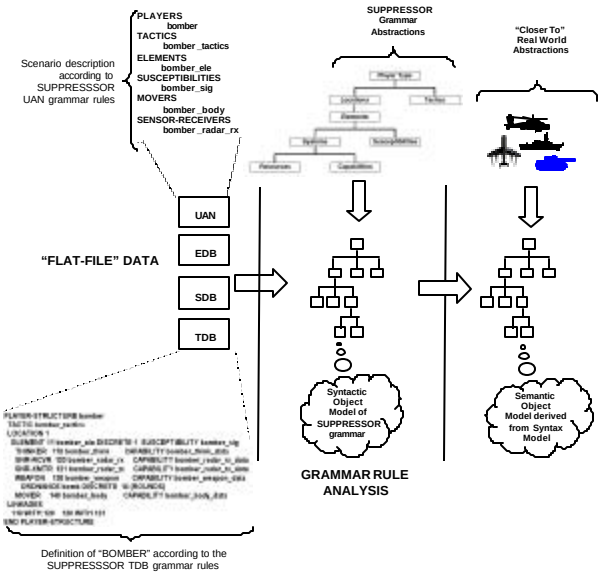

Figure 4: Object Encapsulation of Scenario Files

The object-oriented syntax models for both scenario database files and authoritative data sources can serve various purposes in our architecture. Once in an objectified form, methods can be derived for information visualization purposes, text translation (XML, HTML), persistent object creation, or appropriate conversion to other simulation object structures. Figure 5 illustrates this concept.

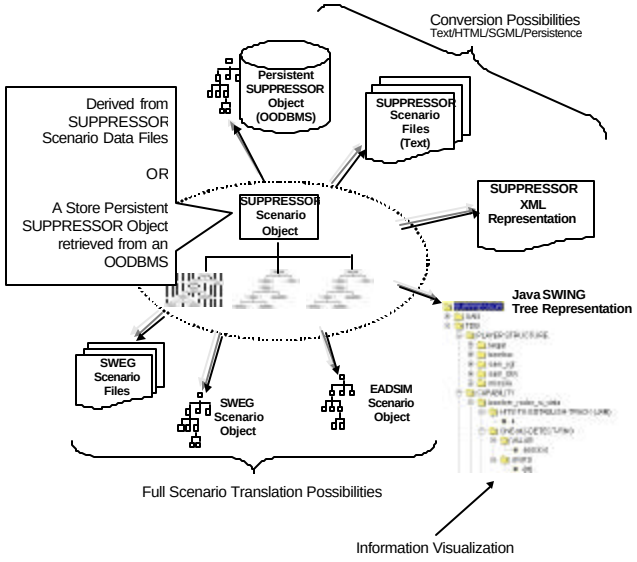

Figure 5: Translation Possibilities for Scenario Object

Semantic object models are derived based on analyst requirements and domain specific needs for information reuse and integration. For example, command chain relationships among players of a scenario are common to many simulation models. The SUPPRESSOR model has particular grammar data items that convey rudimentary subordination relationships. However, the syntactic representation of this data does not capture the parent/subordination relationship necessary for representing this information in a real world or semantically appropriate way. Post-processing of the scenario instance data must occur in order to create additional associations, inheritance chains, and classes that express the parent/subordinate relationship (as seen in Figure 6). In this sense, methods built into the object model can be used to post-process these object bundles to create semantically appropriate information.

Likewise, the MSFD data file (a traceable authoritative source) also contains subordination information of forces modeled in a scenario. The MSFD is record oriented and its syntactic object class representation is a simple aggregation of record classes with multiple attributes. The actual command chain information is

not derivable from this syntax model apart from instance data and some form of complicated post-processing. Once this post-processing is accomplished (as depicted in Figure 6), the subordination information in terms of parent and subordinate units can now be expressed in a more semantically appropriate manner.

As seen in the semantic model for MSFD command hierarchy, there are six different levels of subordination involving national, corps, divisional, company, battalion, regimental, and group placement. Even though the SUPPRESSOR model only conveys two levels of subordination (i.e., parent and subordinate), there is an appropriate mapping from these two particular semantic object models that can be derived. Other semantic models can be derived from a simulation's syntax model based on the information visualization need of an analyst or the information content discovery need of an information retrieval (IR) system. The result of applying OO modeling techniques to simulation grammars and scenario input files forms foundational objects by which encapsulation and further data integration can be accomplished. The goal, of course, is to exploit to the fullest extent possible the energy spent in deriving the object model for a given simulation model type or data input.

In previous work [5], the SUPPRESSOR syntactic object model was fully elaborated along with the MSFD syntactic object model. For demonstration purposes, the command chain hierarchy was chosen as a suitable semantic model in which integration and reuse could be accomplished. We further propose a

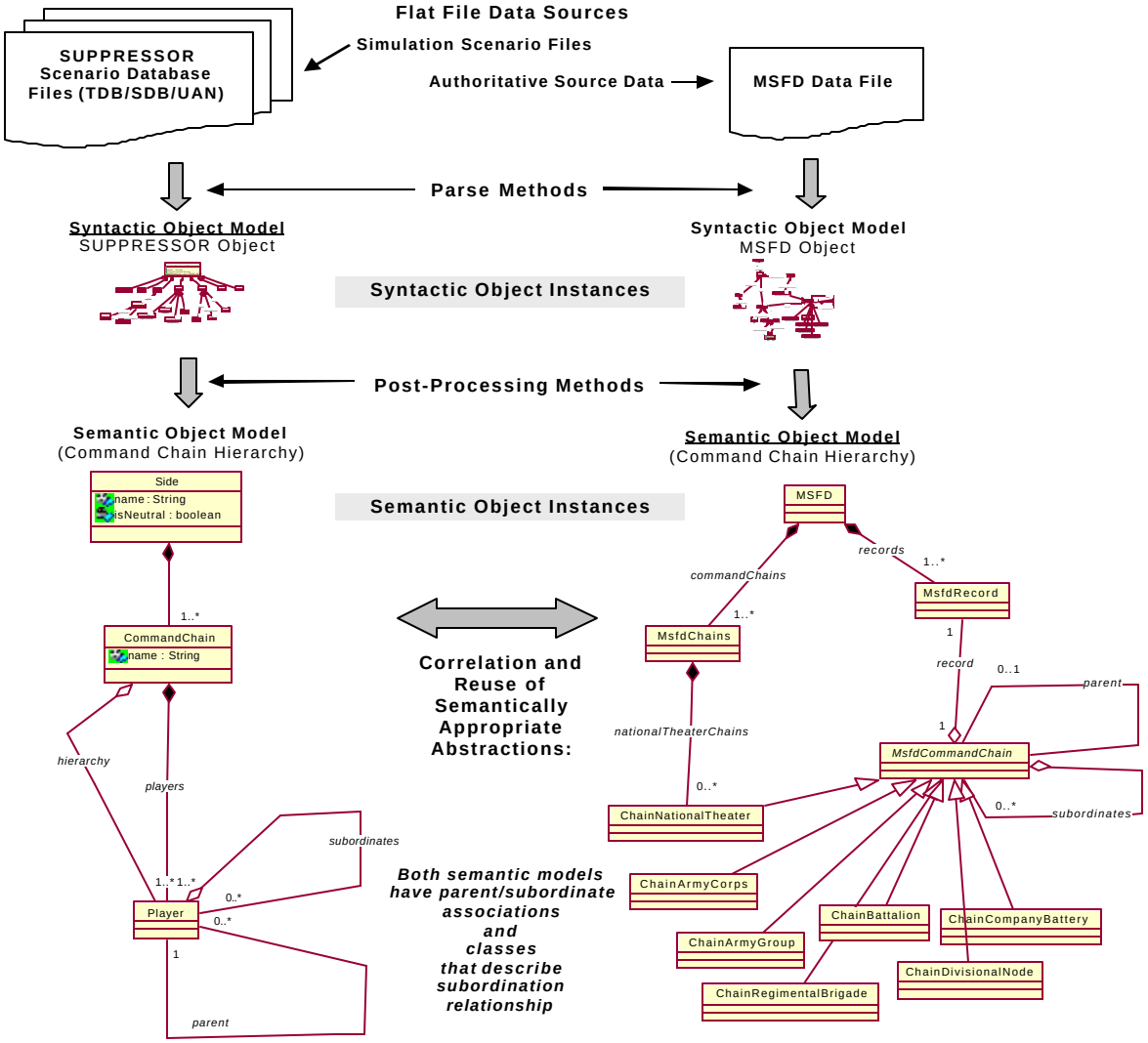

Figure 6: Authoritative-Source-to-Simulation-Scenario Semantic Integration

model-to-model integration approach based on a common semantic object model. This concept, though not fully developed in past research, is based on the need to provide legacy integration of pre-existing scenarios that are written in various simulation models.

Theoretically, the modeled aspects of any given weapon system or player interaction can be described as a correlating function between a data item (or set of data items) found in a simulation grammar with a particular real-world weapon characteristic that is being modeled. This function can provide the necessary basis and mapping for data items in a simulation grammar into a generic semantic object model that captures the information content. Figure 7 expresses this mapping function in a way that shows correlation of the data item elements of a "bomber" player as defined in the SUPPRESSOR grammar to their equivalent representation in a generic semantic object model. This syntax-to-semantic conversation is

between grammar data items and the real-world attributes or characteristics of a weapon system they are defining. The derivation of *f(x)* and the construction of the semantic object model are left for future research.

In the same manner, a correlating function *g(f(x))* can also be found from a given populated instance of a generic semantic object model that represents weapon systems and characteristics back to some different model, which in particular belongs to the CERTCORT domain. Figure 7 demonstrates how a simulation entity like a bomber can be defined in a SUPPRESSOR model and integrated and reused in a JIMM scenario using the concept of a common semantic object model. Our approach differs from traditional heterogeneous database approaches in that the underlying data files are seen more in the context of an information retrieval system than they are as schematic representations of some underlying database. Semantic modeling is used

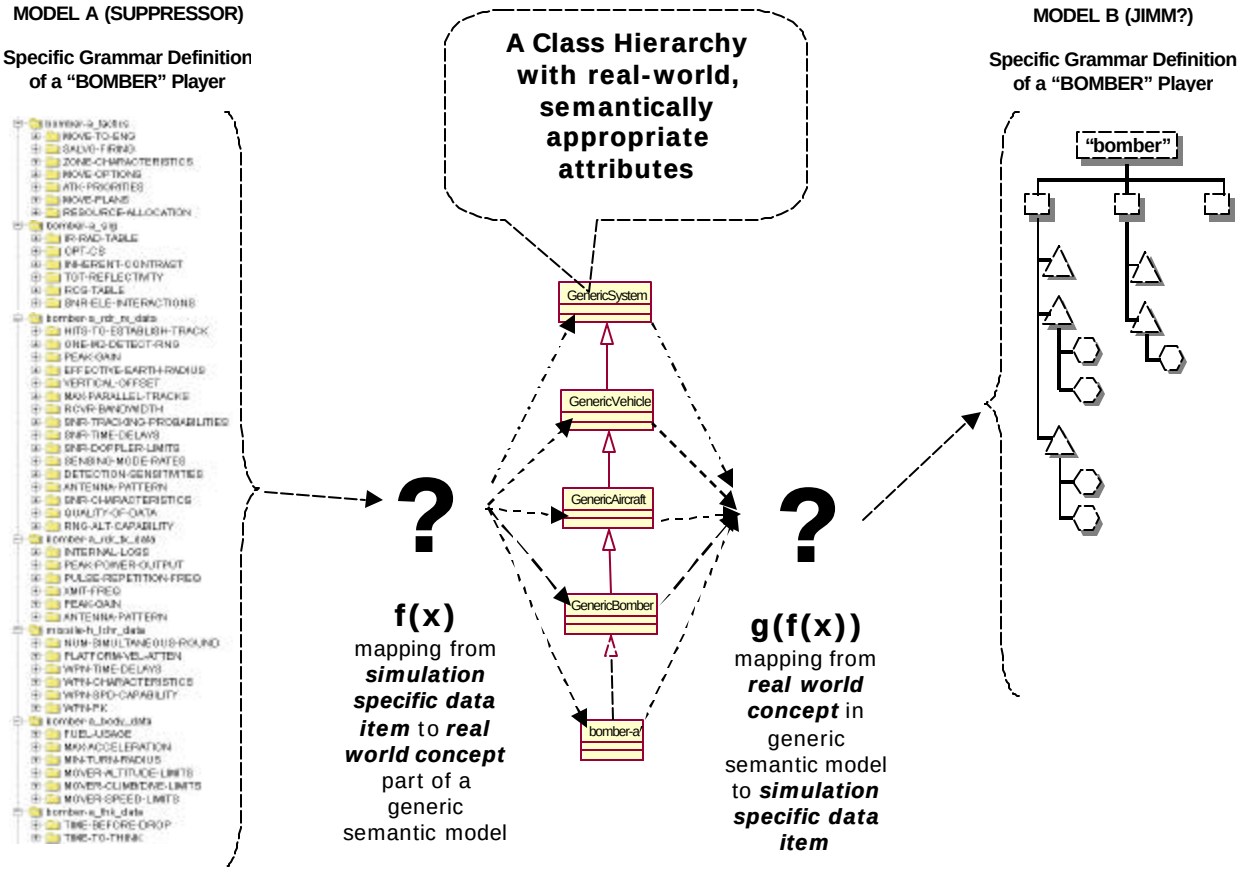

Figure 7: Model-to-Model Semantic Integration Concept

accomplished by some function, *f(x)*, which is currently not defined. This function is based on the valid assumption that a many-to-many relationship exists

to expose this underlying information content found in these representative documents while the syntax model

preserves the closest representation back to an underlying flat-file or relational data source.

Though we use models and authoritative data sources part of the CERTCORT domain as representative examples for applying our data modeling approach, these techniques can be applied in general to the larger body of simulations and models that are within DOD. In this sense, the achievements of our research show benefit for other simulation and modeling communities to include operational and educational war-gaming and one-on-one and campaign level models. These models and the construction of scenarios executed in them have similar problems of interoperability and reuse as well. Our research uses objects as natural encapsulations of complex data. These **objects** have inherent strength for separating data from the processing requirements of that data. Information visualization and information discovery are achieved by adding appropriate post-processing methods on data read from different sources (relational, object, or flat-file). Our submitted work [6] elaborates fully the concept of information bundling in terms of syntactic and semantic object modeling along with how object-oriented database technology is incorporated into our architecture.

Our approach to a distributed collaborative architecture based on agent-oriented information systems (AOIS) holds promise for the general field of DOD simulation and modeling as well. In addition to looking at data integration in terms of derived semantic views, we also take a lifecycle approach to this problem domain by incorporating an agent-oriented system design methodology known as MaSE [4,7]. By applying agent-oriented analysis to this problem domain, we map requirements for scenario construction and integration into agent based layers that directly translate from design into implementation. The application of MaSE to a subset of the CERTCORT requirements is detailed fully in our original research [5] and summarized in [7]. The applicability of agent architecture to our problem domain is discussed next.

4. AGENT ARCHITECTURE

Agents are a relatively new paradigm introduced over the last decade. We view the term "agents" both in terms of a programming paradigm that offers higher level abstractions above objects and as autonomous entities that have active properties. Multi-agent systems, in particular, require explicit definition of communication (known as conversations) and the specification of message elements between agents that achieve common goals. As such, agents can be defined as objects with goals and a common communication language [4].

Our research uses the **agent** concept as a natural abstraction that can capture active requirements of a system. In this sense, the communication ability provided by the agent-architecture and the encapsulation of information as objects represented by information agents within our system replaces the traditional concept of data access in an application framework. The use of persistence mechanisms for objects is a natural extension to our architecture that remains supported yet orthogonal to our application development paradigm [6, 7]. We implement functionality by classes of agents known as *layers*. Three layers are initially conceived for our architecture: the information layer, collaboration layer, and assistance layer. Each layer contains one or more types of agents whose goals and function have been directly distilled from CERTCORT requirements via the MaSE methodology. Figure 8 conceptually shows these basic layers.

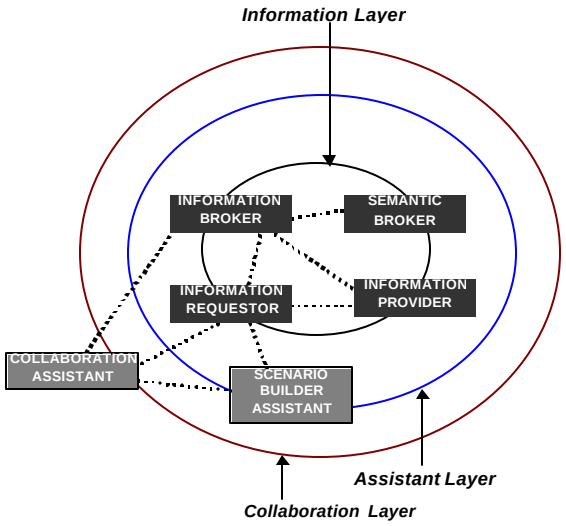

Figure 8: CERTCORT Agent Layers

The information layer can be seen as four different types of agents that collectively handle the data storage, processing, and sharing requirements of our system. Distributed or localized *information* agents thus *encapsulate* and *represent* the information content of underlying data sources within this layer, which in our case can be authoritative sources or simulation scenario data stores. The information is represented to other agents in the system using serialized objects embedded in a standard agent communication protocol. Our agent communication language (ACL) is based upon Knowledge Query Manipulation Language (KQML) structures familiar to the realm of agent

technology. Figure 9 illustrates how information agents in our system represent the content of scenario data files (SUPPRESSOR/SWEG) and authoritative data files (MSFD). The figure also illustrates how object instances can be retrieved from an OODBMS where persistent instances have been previously created and stored.

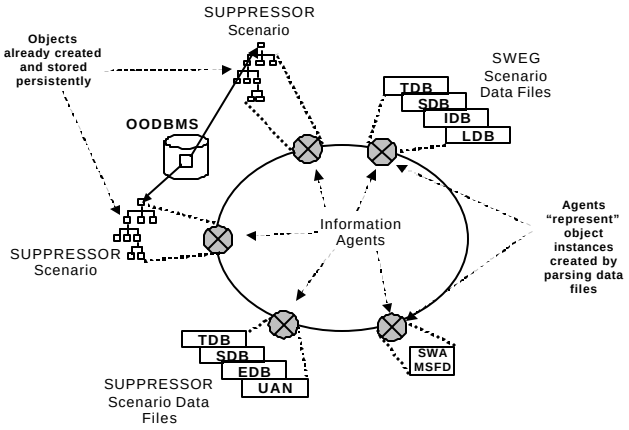

Figure 9: Information Agent Data Representation

CERTCORT requirements involve the automation and correlation of source data into respective scenarios and the automated construction of new scenarios using intelligent assistance. Current scenario construction can only be done manually by one analyst at a time, with a desired goal to allow collaborative development that automatically validates and fuses scenario data to avoid conflicts or errors. The *collaborative assistance* layer and the *intelligent assistance* layer (Figure 8) represent agents that incorporate these requirements into our architecture. Though left for future research, agents are an attractive programming paradigm to represent these particular problems in goal based plans that can be distributed across computer resources.

The *information* layer itself consists of three particular agent classes that we elaborate for implementation and demonstration purposes. Figure 10 shows the relationships of the information *provider*, information *requestor*, and information *broker* agent classes. This MaSE diagram also shows the various types of conversations that support goal directed behavior among these agent types. Information provider agents are used to actively encapsulate data sources, which can be flat-file, relational, or object. Our architecture reflects the reasoning ability and "active" nature these providers need to have in order to respond to requests for information. Cooperative information agents are based on the traditional notion of information retrieval (IR) systems where agents search with other agents for information and respond to queries in a plan-based manner [2]. Our paradigm is based upon the traditional notion of one type of middle-agent architecture known as a matchmaker [8]. This configuration allows IR capabilities to be added in the future but initially replaces the traditional data storage services with a collection of information agents linked by an information brokering system, which acts as a basis for information registration and exchange.

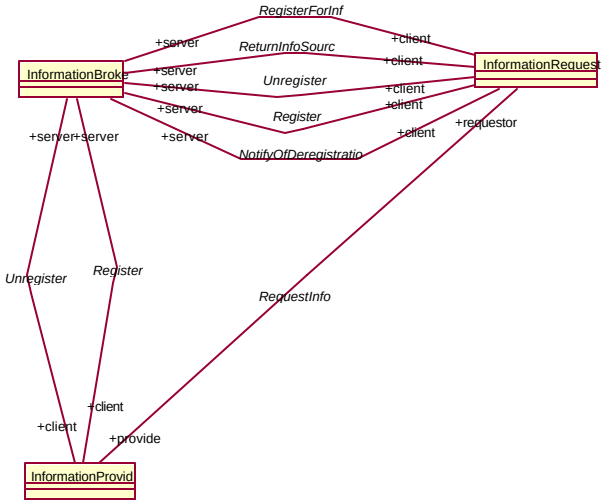

Figure 10: Information Layer Agent Specification

In order to introduce agent-oriented principles into the modeling and simulation problem domain, two building blocks are needed. An agent-oriented systems analysis and design technique (such as MaSE) should be used to break the problem area down from requirements to design on into its implementation as an agent hierarchy. This technique may be similar to normal object-oriented design methodologies, but should be definitively agent-centric and not object-centric. Second, a multi-agent development environment is needed to implement and build the communication requirements of agents specified by the agent-oriented methodology. We use a customized agent framework known as *agentMom* to implement our agent classes in the Java programming language.

We used a representative simulation (SUPPRESSOR) and a representative authoritative data source (MSFD) in order to demonstrate the information layer of our architecture. Future research will work to fully elaborate the object models of other simulations and data sources part of the CERTCORT family. Additional agent layers are envisioned and the implementation of collaborative and intelligent assistance is foreseen as natural extensions of this architecture without change in the underlying information systems design.

5. CONCLUSIONS AND DISCUSSION

Agents provide unique benefits to information integration in the modeling and simulation context above those provided by traditional heterogeneous database architectures. For instance, semantic models in our domain require post-processing of scenario instance data that exists in the form of syntactic models. This "active" property of a data source is handled well by properties information agents provide. Federated databases as well tend to be "data" centric and not "application" centric. Multi-agent systems provide a life cycle approach that can provide direct traceability of user requirements into system components and agent classes.

Our architecture also supports a distributed and cross-platform capability. Though Java facilitates some degree of cross-platform capability, our agent types can be distributed across network resources to best match developer needs and computing power. The three primary types of information layer applications can be distributed across different nodes and operating systems in a network. The *information broker*, *information provider*, and *information requestor* agent types function in different "roles" within the information layer of our agent architecture and in essence replace traditional data access in a normal IS paradigm. Communication and information exchange is handled by Java classes part of the *agentMom* framework using Knowledge Query Manipulation Language (KQML) like syntax. Data in our system that is encapsulated in object form is shared using the Java serialization and networking API.

Another key advantage of an AOIS approach to systems development is that it can keep the "focus" of system development on the data without binding to a particular data storage mechanism. Agents in this sense provide the ability to abstract away the underlying data representation of information sources within an information system. The ability to represent object, relational and flat-file data in a common data model where reuse and integration can occur is a key element for successful applications in our problem domain. The incorporation of both syntactic and semantic object modeling to our data domain is key to our data integration approach and represents our concept of a common ontology.

Finally, agent-based systems can be expanded to provide greater functionality without drastic architectural changes. Different "layers" can be constructed independent and concurrent with other layers. Intelligent interfaces and the ability to achieve coordinated plan-based goals are unique to multi-agent system paradigms in this respect. AOIS also has expression in terms of both information-gathering systems and information retrieval systems, though our implementation does not include them. Future work involves incorporation of other simulation models and authoritative sources as well as the development of additional agent layers.

7. REFERENCES

[1] Petit, M., P. Heymans and P. Schobbens. "Agents as a Key Concept for Information Systems Requirements Engineering." Position paper on AOIS at CAiSE 99, 1999.

[2] Dignum, F. "Are information agents just an extension of information systems or a new paradigm?" Workshop on AOIS at CAiSE 99, 1999.

[3] Wagner, G. "Toward Agent-Oriented Information Systems." Technical report, Institute for Information, University of Leipzig, March 1999.

[4] DeLoach, Scott A. "Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems." Proceedings of a Workshop on Agent-Oriented Information Systems (AOIS'99). 45-57. Seattle, WA. May 1, 1999

[5] McDonald, J. "Agent Based Framework for Collaborative Engineering Model Development," MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, AFIT/GCS/ENG/00M-16, March 2000.

[6] M. Talbert, McDonald, J., "Legacy Scenario Information Reusability for Simulation Interoperability", submitted to Ninth International Conference on Information and Knowledge Management (CIKM), Washington, D.C., Nov 2000.

[7] McDonald, J., M. Talbert, and S. Deloach, "Heterogeneous Database Integration Using Agent-Oriented Information Systems", to Appear in Proceedings of the International Conference on Artificial Intelligence '2000, Las Vegas, NV, Jun 2000.

[8] Decker, K., M. Williamson and K. Sycara. "Matchmaking and Brokering." Technical report, The Robotics Institute, Carnegie Mellon University (USA), Pittsburgh, May 16, 1996.