Proceedings of the International Conference on Artificial Intelligence (IC-AI'2000).
CSREA Press, Vol III, pp. 1359-1365. June 26 - 29, 2000, Las Vegas, Nevada.

Heterogeneous Database Integration Using Agent-Oriented Information Systems[*]

J. Todd McDonald
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB OH 45433

Michael L. Talbert
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB OH 45433

Scott A. DeLoach
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB OH 45433

Abstract: *The Department of Defense (DOD) has an extensive family of models used to simulate the mission level interaction of weapon systems. Interoperability and reuse of the underlying data files used to create simulation scenarios pose great challenges in this regard. Unlike traditional data integration methods common to federated database research, the emerging field of agent-oriented information systems (AOIS) views data as the central focus of an application while also providing an overall architectural framework for application development. We develop an AOIS solution relevant to this problem domain by combining object-oriented data modeling (OMT), a persistent programming language using a commercial object-oriented database (ObjectStore®), and an agent-oriented analysis and design methodology (MaSE). Requirements from a contractor-led effort at the Air Force Research Laboratory (AFRL) known as CERTCORT are the basis for analysis and design of our system. We implement prototypical information-layer applications to conceptually demonstrate the reusability and integration of scenarios across simulation models.*

Keywords: AOIS, Agents, Modeling and Simulations, Heterogeneous Database Integration

1. Introduction

The Air Force Research Laboratory is directing an effort to provide a collaborative computing environment to support simulation scenario reuse and integration. 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 Environment Generator (SWEG). The requirements of this collaborative environment, known as CERTCORT (Concurrent Engineering for Real Time databases CORrelation Tool), and its heterogeneous integration problem are represented pictorially in Figure 1.

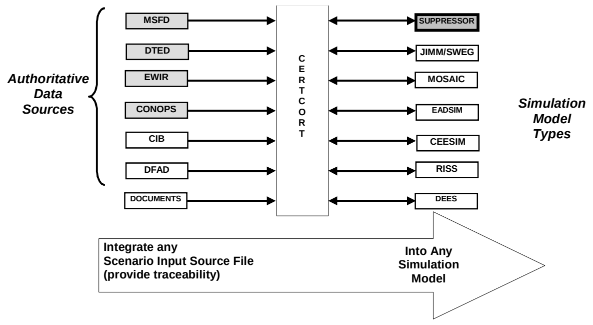

Authoritative Data Sources
MSFD
DTED
EWIR
CONOPS
CIB
DFAD
DOCUMENTS
CERTCORT
SUPPRESSOR
JIMM/SWEG
MOSAIC
EADSIM
CEESIM
RISS
DEES
Simulation Model Types
Integrate any Scenario Input Source File (provide traceability)
Into Any Simulation Model

Figure 1: Heterogeneous Database Problem Domain

1.1. Problem Domain

There are two primary goals for integration within this realm. The first concerns the mapping of real world data as conveyed through authoritative data sources (left side of Figure 1) into the language and syntax structures that are specific to a given type of model (right side of Figure 1). Figure 2 pictorially represents a Multi-Spectral Force Database file (subordination relationships among units) and its correlation into both a SUPPRESSOR and EADSIM scenario instance.

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

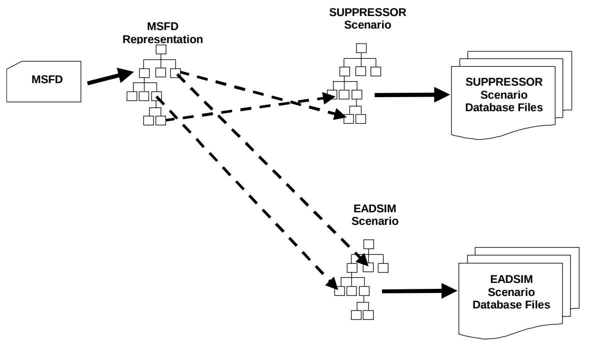

MSFD Representation
SUPPRESSOR Scenario
MSFD
SUPPRESSOR Scenario Database Files
EADSIM Scenario
EADSIM Scenario Database Files

Figure 2: Authoritative-Source-to-Model Traceability

The second form of integration deals with integration across simulation models themselves, where entities described in one model-specific grammar (e.g., SUPPRESSOR) are desired for reuse in another model-specific grammar (such as JIMM). Translation of data items from one simulation grammar to another is required in this form of integration. The grammars used to describe operational scenarios vary in their ability to capture concepts such as terrain, communication, electronic warfare, and lethal engagements.

Figure 3 illustrates how typical integration of scenarios from one type of model to another deals strictly with "translating" or "mapping" the syntax structures of one into their equivalent meaning in another. This syntactic level of integration, however, does not always form the ideal basis of integration because of the diversity and complexity of the languages that are specific to a simulation model.

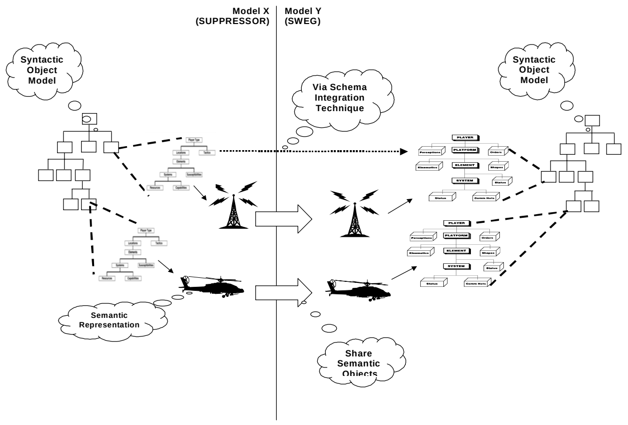


Figure 3: Model-to-Model Integration

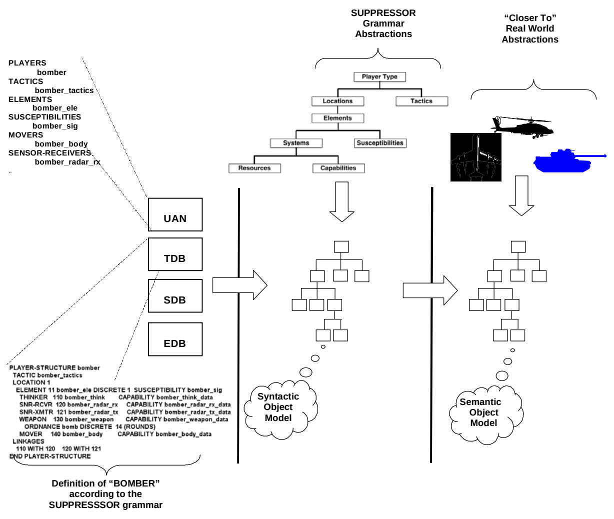
Current solutions in the realm of scenario reuse focus on schema integration techniques from the perspective of a traditional heterogeneous federated database system. We approach the integration problem from more than the syntactic level alone and describe an approach to translation based upon common semantic objects found from information discovery techniques. Figure 3 also shows how “semantic” objects that are closer to real-world abstractions are a means of reuse.

The process of mapping authoritative sources into scenario files (Figure 2) and reusing existing scenarios in different models (Figure 3) is tedious and done with minimal software support. No facility for collaborative assistance from other domain specialists exists as well. Our research explores the benefits of using agent-oriented information systems (AOIS) and implementing agent technology to achieve sharing and reuse in this heterogeneous data environment. We develop architecture to support integration of both authoritative data to a family of simulation models (Figure 2) and a model-to-model integration (Figure 3) that supports an automated approach to scenario construction.

1.2. Object-Oriented Foundations

Scenario database files are currently flat-file structures that follow pre-defined grammars of a given model. Authoritative sources like the MSFD and EWIR (Figure 1) also exist in flat-file or relational database form. We use traditional Object Modeling Technique (OMT) analysis and design methods to define grammar rules and file structures in an equivalent object-based class hierarchy. Our structural models faithfully capture the content of scenario files and authoritative data sources. Because of the legacy nature of the simulation engines themselves, scenario data files will continue to be the method of initializing and executing scenarios for a given model. We thus use object representations of both scenario files and authoritative sources as the basis of information interchange and storage.

Objects offer an ideal form of encapsulation for the underlying data content of scenarios and allow a natural form of persistence when object-oriented databases (OODBMS) are introduced into the architecture. Figure 4 illustrates the concept of encapsulating scenario database files by an object structure derived from an OMT analysis and design of the grammar for a simulation such as SUPPRESSOR. This "syntax" model becomes the ideal unit of storage because it can also be used to readily reproduce the underlying flat-file structure (which is required as input to execute an actual simulation). Semantic object models can also be derived from this structure that introduce real world abstractions and further provide the basis for reuse and integration.



```
PLAYERS
        bomber
TACTICS
        bomber_tactics
ELEMENTS
        bomber_ele
SUSCEPTIBILITIES
        bomber_sig
MOVERS
        bomber_body
SENSOR-RECEIVERS
        bomber_radar_rx
..
```

```
PLAYER-STRUCTURE bomber
  TACTIC bomber_tactics
  LOCATION 1
   ELEMENT 11 bomber_ele DISCRETE 1  SUSCEPTIBILITY bomber_sig
    THINKER   110 bomber_think       CAPABILITY bomber_think_data
    SNR-RCVR  120 bomber_radar_rx     CAPABILITY bomber_radar_rx_data
    SNR-XMTR  121 bomber_radar_tx     CAPABILITY bomber_radar_tx_data
    WEAPON    130 bomber_weapon       CAPABILITY bomber_weapon_data
      ORDNANCE bomb DISCRETE  14 (ROUNDS)
    MOVER     140 bomber_body        CAPABILITY bomber_body_data
  LINKAGES
   110 WITH 120    120 WITH 121
END PLAYER-STRUCTURE
```

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 object structures. Figure 5 illustrates this concept.

The strength of an OODBMS in this problem domain is that persistence can be achieved for scenario representations without having to change or translate them into another data format. Scenarios stored in an object-oriented database already exist in the common data model that is necessary for reuse and integration. Scenario and authoritative data representation are accomplished by parsing the scenario files dynamically into an appropriate object instance of that simulation's class structure or by retrieving previously created scenarios and authoritative sources that are persistent objects in an OODBMS. Persistent stores are also available for transaction processing, query capability, and information retrieval applications that are separate from our agent-based architecture.

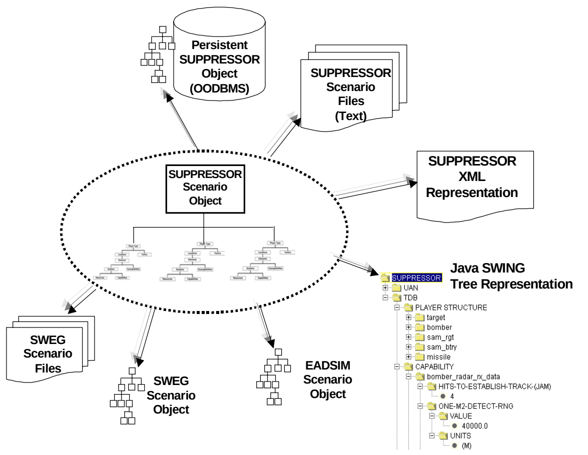


Figure 5: Translation Possibilities for Scenario Object

A promising approach to solve the integration problem of heterogeneous data sources, and the thrust of our research, is to provide access to a possibly large number of information agents that dynamically or persistently represent scenarios of different models. Scenarios can exist in this environment in both native file structure and OODBMS formats while an information agent is used to provide the mapping of this representation to an information brokering system. For relational data sources, an active information agent can perform necessary data translation steps into the common data model (in our case, object). Figure 6 illustrates the concept of representing OODBMS stores and native file formats with information agents. This research demonstrates both of these capabilities, and the definition and use of agents in our architecture are discussed next.

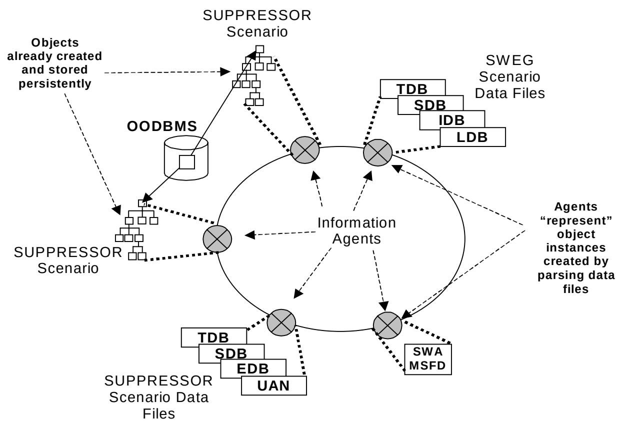

SUPPRESSOR Scenario
Objects already created and stored persistently
SWEG Scenario Data Files
TDB
SDB
IDB
LDB
OODBMS
Agents "represent" object instances created by parsing data files
Information Agents
SUPPRESSOR Scenario
TDB
SDB
EDB
UAN
SWA
MSFD
SUPPRESSOR Scenario Data Files

Figure 6: Information Agent Data Representation

2. AOIS Design

To narrow the discussion and scope of what we define an "agent" to be, we realize that agents are conceptualized or implemented by many in the AI field using concepts reserved solely for humans. From this perspective, agents can be characterized in terms of knowledge, belief, intention, and obligation. Other advocates require stricter properties such as mobility, veracity, mobility, and rationality. We view 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.

The agent concept is seen by some also as a natural and appropriate way to deal with information complexity. A new paradigm has emerged that looks first at the information systems level of a problem and addresses how it relates not only to objects, but to the idea of autonomous agents as well. The term agent-oriented information systems (AOIS) describes the adaptation of agent-oriented principles to the entire information life cycle design process. Information agents as such can be viewed as entities that represent their information source as knowledge and beliefs and then offers capabilities and commitments about those beliefs to other interested parties. In this sense, information agents serve the role of an information "provider" in the context of an AOIS.

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. Our architecture allows IR capabilities to be added in the future but initially deals solely with the replacement of traditional data storage services with a collection of information agents linked by an information brokering system. Figure 7 illustrates the traditional notion of one type of middle-agent system known as a matchmaker. We use this as a basis for information registration and exchange in our system.

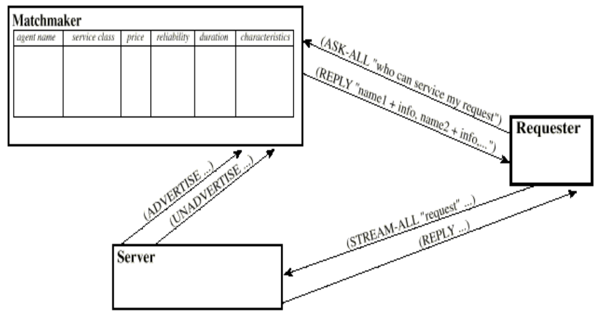


Figure 7: Matchmaker Brokering Scheme

In order to introduce agent-oriented principles into our problem domain, we require two key building blocks. First, an agent-oriented systems analysis and design technique is 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 it is definitively agent-centric and not object-centric. Second, a multi-agent development environment is chosen to implement and build the communication requirements of agents specified by our agent-oriented methodology. The methodology used to transform our domain requirements into agent architecture is discussed next.

2.1. *Multi-agent Systems Engineering (MaSE)*

Systems engineering approach to software development follows an orderly and logical design process that successfully captures system requirements and transforms them into real world software and hardware components. The process for developing a multi-agent system is no different. The key difference is that agent concepts and constructs are used to synthesize the problem domain of a system in addition to normal object-oriented interactions. We use MaSE as a generalized methodology similar to Object Oriented Analysis and Design (OOAD) to capture the agent-oriented aspects of our domain. The major phases of this methodology cascade as follows: 1) domain level design, 2) agent-level design, 3) component design, and 4) system design.

An agent-oriented analysis of a problem using MaSE specifically seeks to 1) map requirements to implementation, 2) develop a methodology for determining what agents are needed in a system, and 3) develop a methodology for designing conversations to support the collaborative goals of a group of agents. Though MaSE does not assist in determining whether a problem domain is best represented by agents, it is an appropriate starting place when agent orientation has been chosen as the desired abstraction.

Regardless of the implications of whether machines can “think” or act “rationally”, agents can be seen as another way of abstracting a problem into more definable pieces. We find agents particularly helpful in this problem domain as an abstraction tool because of the complexity of involved with integrating such a large number of data sources and finding a common architecture for reuse among such a large number of diverse simulation models.

The MaSE methodology itself flows from requirements analysis and derives system level goals. These goals are decomposed into subgoals that can be conjuncted or disjuncted that reflect a higher level of control, or goals that are subsumed by other goals. These decomposed goals can then be converted into roles that are more familiar to UML notation. Roles can be combined under one agent and agents that interact with humans and system resources may also be defined.

For purposes of our research, and to focus the scope of our implementation, we chose one particular simulation model (SUPPRESSOR from Figure 1) and one particular authoritative data source (MSFD from Figure 1) as a basis for our requirements. Figure 8 shows the results of applying MaSE to our problem domain and the agent types that are directly traceable to our decomposed requirements. Though they are a small subset of the entire CERTCORT data domain, these requirements represent both the collaborative nature and automated generation facilities desired in the final system.

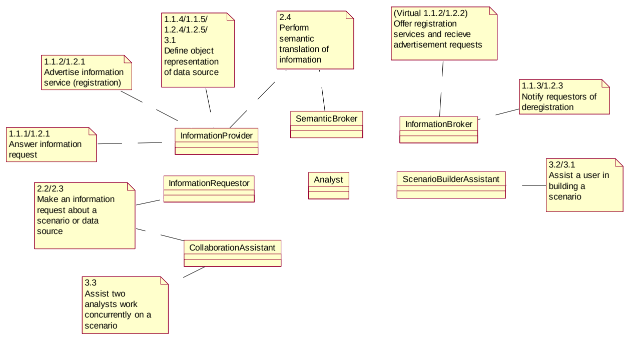

1.1.4/1.1.5/ 1.2.4/1.2.5/ 3.1 Define object representation of data source
2.4 Perform semantic translation of information
(Virtual 1.1.2/1.2.2) Offer registration services and recieve advertisement requests
1.1.2/1.2.1 Advertise information service (registration)
1.1.3/1.2.3 Notify requestors of deregistration
SemanticBroker
InformationBroker
1.1.1/1.2.1 Answer information request
InformationProvider
InformationRequestor
Analyst
ScenarioBuilderAssistant
3.2/3.1 Assist a user in building a scenario
2.2/2.3 Make an information request about a scenario or data source
CollaborationAssistant
3.3 Assist two analysts work concurrently on a scenario

Figure 8: Agent Derivation from Roles

Once appropriate agent types have been defined under MaSE, traditional use cases can be defined according to normal UML notation for each goal that an agent is responsible for. Collaborative goals require the interaction of multiple agents and thus more complex interactions. A message passing sequence between two or more agents requires at least the definition of one conversation type. Sequence and collaboration diagrams can be used to define collaborative scenarios.

Conversations are designed using the scenarios as the minimum messages that must be passed. State based sequence diagrams (not shown) are derived directly from scenarios and define all required states, including failure, for each agent type involved in a conversation.

A final product of MaSE includes an agent hierarchy that fulfills system goals and also defines all necessary conversations between agent types. Figure 9 illustrates the three primary agent types that were defined in our domain as part of the "information layer" and the conversation types between them.

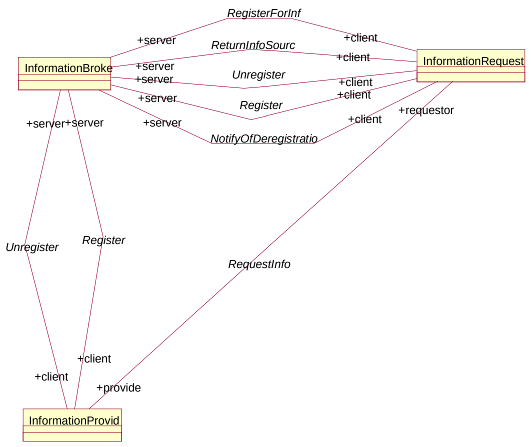


Figure 9: Agent Types and Conversation Hierarchy

Agents are a powerful abstraction tool. By extension, we use agent types to form "layers" representing increased system functionality and requirements implementation. Our research focused on agents that are suited at "registering" information providers (entire scenarios or authoritative data sources) and also "registering" applications that request data and information from those providers. Based on our matchmaker scheme, an information "broker" is the middle-agent responsible for matching requestors with providers. Figure 10 illustrates both this "information" layer along with other agent layers that will support collaboration among more than one analyst developing scenarios and that will introduce intelligent user interfaces to provide expert knowledge in creating scenarios

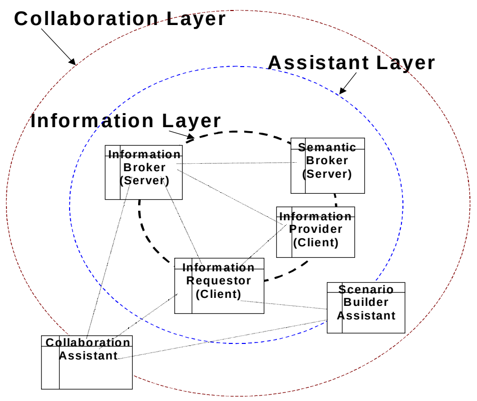

Collaboration Layer
Assistant Layer
Information Layer
Information Broker (Server)
Semantic Broker (Server)
Information Provider (Client)
Information Requestor (Client)
Scenario Builder Assistant
Collaboration Assistant

Figure 10: CERTCORT Agent Layers

2.2. Multi-Agent Development Framework

To implement our agent design, a multi-agent development framework named *agentMom* was chosen. This environment defines basic classes for agents, conversations, and messages. Our research demonstration was written in Java (JDK 1.1.6) and utilized serialized objects as the primary means of information exchange between agent types. Figure 11 illustrates the communication architecture between class types defined by the agentMom framework.

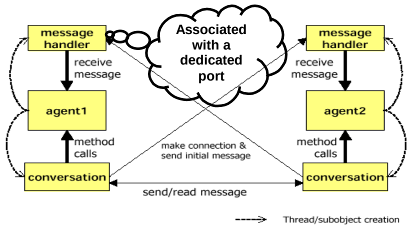


Figure 11: agentMom Communication Architecture

Figure 12 illustrates the distributed, cross-platform nature of the framework along with representative applications functioning in different “roles” within the information layer of our agent architecture.

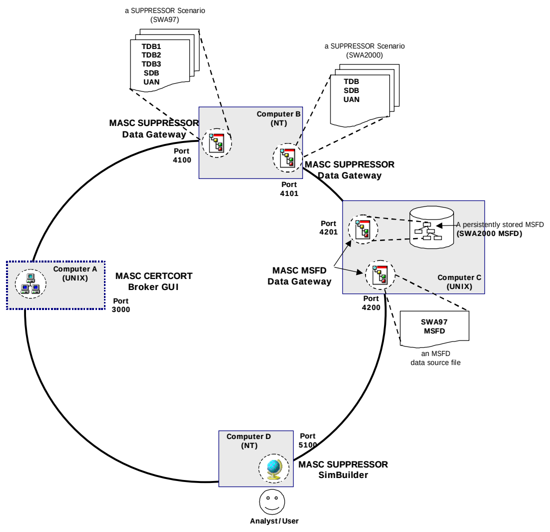


Figure 12: Application Demonstration

We implement an architecture that allows persistent objects to be both created and retrieved from an ObjectStore® object-oriented database management system (OODBMS). Persistence in regards to this framework is seen as orthogonal to the agent-oriented information system, yet support is provided by the framework itself for OODBMS access.

3. Discussion

Agents provide unique benefits to information integration in this context above those provided by traditional heterogeneous database architectures. Semantic models in our domain require post-processing of instance data; this is best supported in the context of an "active" data source that information agents can provide. Federated databases tend to be "data" centric and not "application" centric, however multi-agent systems provide a life cycle approach that can provide direct traceability of user requirements into system components and agent classes.

AOIS technology keeps the "focus" of system development on the data without binding to a particular data storage mechanism. Agents also provide the ability to abstract away the underlying data representation of information sources within information systems. Agent based systems can be expanded to provide greater functionality without drastic architectural changes. Intelligent interfaces and the ability to achieve coordinated plan-based goals are not possible from a database-centered approach to systems development. Scenario model integration and construction has certain information retrieval aspects that are naturally suited to underlying information agent architecture. AOIS has implementation in terms of both information-gathering systems and the encapsulation of traditional data sources normally part of a database management system.

4. References

1. Biller H. and E. Neuhold. *"Semantics of Data Models: Database Semantics."* in Readings In Artificial Intelligence & Databases. Ed. Mylopoulos, J. and Brodie, M. San Mateo CA: Morgan Kaufman Publishers, 1989 [received 3 Sep 1976].
2. Hammer, J. and D. McLeod. *"An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems."* International Journal of Intelligent and Cooperative Information Systems 2(1):51-83, March 1993.
3. Decker, K., M. Williamson and K. Sycara. "*Matchmaking and Brokering*." Technical report, The Robotics Institute, Carnegie Mellon University (USA), Pittsburgh, May 16, 1996.
4. Wooldridge, M. and N. Jennings. "*Intelligent Agents: Theory and Practice.*", Knowledge Engineering Review, 10(2): 115-152, 1995.
5. Wagner, G. "*Toward Agent-Oriented Information Systems.*" Technical report, Institute for Information, University of Leipzig, March 1999.
6. Dignum, F. "*Are information agents just an extension of information systems or a new paradigm?*" Workshop on Agent Oriented Information Systems at CAiSE 99, 1999.
7. Petit, M., P. Heymans and P. Schobbens. "*Agents as a Key Concept for Information Systems Requirements Engineering.*", Position paper, Agent Oriented Information Systems Workshop at CAiSE 99, 1999.
8. International Bi-Conference Workshop on Agent-Oriented Information Systems, 1999. http://www.aois.org.
9. 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.
10. DeLoach, Scott A. "Using agentMom." Unpublished document. Air Force Institute of Technology (AU), Wright-Patterson AFB, OH. October 1999.