

# Semantic Interoperability Integrating and Augmenting Legacy Applications with OWL Ontologies

Douglas Holmes  
Java Professionals, Inc.  
11835 Maple Crest  
Moorpark, CA 93021  
805-523-2650  
[dholmes@javaprofessionals.com](mailto:dholmes@javaprofessionals.com)

Richard Stocking  
Lockheed Martin Aeronautics  
Advanced Development Programs  
1011 Lockheed Way  
Palmdale, CA 93599  
661-949-1039  
[rich.stocking@lmco.com](mailto:rich.stocking@lmco.com)

<sup>1</sup>*Abstract*—This paper describes an approach to software interoperability based on knowledge representation technologies developed in support of the emerging Semantic Web. In particular, we are interested in the interoperation of an arbitrary set (two or more) of “legacy” applications that operate in a common domain, but that were not originally designed to support each other or to share information. To do this, we develop a means to create a general meta-model of the common domain of discourse that contains, organizes and correlates the essential information that each of the applications needs to function as well as any additional information particular to the composed system. This model becomes the basis of communications between the cooperating systems. Each legacy application is able to obtain input data from the model and export output information to augment or update the model in a form that allows cooperating systems to operate on the updated information.

We represent the meta-model in RDF/XML which, we believe, provides several important advantages. RDF is a knowledge representation language that allows information to be described as a set of factual sentences. When these sentences are constrained by and consistent with an OWL ontology, this enables the use of an existing, robust collection of automatic reasoners and other logical tools to classify, evaluate, query, apply rules and otherwise operate on the model itself, independently of the cooperating applications.

We describe a “layered” set of OWL ontologies that serve as domain models for a number of legacy operations analysis simulations, aeronautical engineering design tools and software prototypes that are used to support the integrated design and development of aeronautical systems. We describe an approach to the development of the ontologies that references, and imports concepts from well known foundational ontologies, including DOLCE, GML and PSL.

We then describe an example that illustrates a methodology for creating the meta-model as an “operational scenario” that indicates the benefits to interoperability of this approach. Finally we discuss the use of SPARQL queries and SWRL rules that effectively expands the functionality of the system and greatly improves the analysis of the output of the system of cooperating simulations and tools<sup>2</sup>

## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>1</b>
<b>2. INTERNET INTEROPERABILITY: THE SEMANTIC WEB.....</b>	<b>1</b>
<b>3. THE ILIUM FRAMEWORK .....</b>	<b>1</b>
<b>4. SEMANTIC INTEGRATION .....</b>	<b>1</b>
<b>5. THE ONTOLOGIES.....</b>	<b>1</b>
<b>6. AN EXAMPLE APPLICATION .....</b>	<b>1</b>
<b>7. CONCLUSIONS .....</b>	<b>1</b>
<b>8. SUMMARY OF STYLE SPECIFICATIONS ....</b>	<b>1</b>
<b>REFERENCES .....</b>	<b>1</b>
<b>BIOGRAPHY.....</b>	<b>1</b>

## 1. INTRODUCTION

Interoperability, in the context of computational systems, refers to the capability to automatically exchange and process information in manner that preserves the meaning, or semantics, of the exchanged information and results in useful, reliable output. It is a concern that drives research and development programs across a wide range of software applications from military command and control and decision support systems to on line eGovernment and other emerging “social” systems. Enterprise Application Integration (EAI) has been a major theme in Information Technology for at least a decade. Interoperation, in industrial engineering and manufacturing software, is

<sup>1</sup> 1-4244-1488-1/08/\$25.00 ©2008 IEEE.

<sup>2</sup> IEEEAC paper# 1235, Version 2, Updated 2002:12-03

equally important. A recent NIST sponsored report [3] on the Interoperability of Software systems describes an industrial perspective on interoperability as "... complete interoperability, which is the seamless high-fidelity exchange of data between different systems, without any loss or corruption."

The value of interoperable systems is widely recognized and the costs of maintaining systems that are not interoperable are probably greater than many expect. For example, the previously mentioned NIST report notes:

"The costs of interoperability are staggering. An internal study from a major automobile manufacturer recently revealed that a lack of interoperability is costing them between \$200 million and \$400 million per vehicle program. According to a recent study of product data exchange in the automotive sector, the inability to efficiently exchange product data through the automotive supply chain alone conservatively costs the industry \$1 billion per year.<sup>6</sup> Similar estimates are available in the aerospace sector. In both industries, officials clearly believe the real costs of interoperability are much higher."

In particular, there is great interest in enabling the interoperation of some arbitrary collection of "legacy" applications that operate in a common domain, but that were not originally designed to support each other, or even to share information. Such an interoperation may be motivated by a number of objectives, including, for example, the need to integrate existing applications across an enterprise to improve business processes; a desire to create a more effective, "seamless" workflow, or perhaps a desire to compose a super system that provides important new or extended functionality, from existing components. In many of these situations, it is desirable to reuse existing software, preserve trusted functionality and user interfaces, while augmenting and extending the total capability of the collective system.

## 2. Internet Interoperability: The Semantic Web

At a fundamental level, the interoperation of software systems depends on information transport technologies that enable systems to reliably communicate, or transfer messages (signals) in a form that can be recognized and decoded as information, and on common semantics that ensure the machine interpretation and use of the received information is consistent with the meaning of the transmitted information.

---

<sup>3</sup> NAFCAM (2001) Exploiting E-Manufacturing: Interoperability of Software Systems Used by U.S. Manufacturers available at: <http://www.mel.nist.gov/div826/msid/sima/FinalReportSummary.pdf>

Since 1999, the World Wide Web Consortium (W3C) has led an initiative, known as the Semantic Web [4] intended to promote this sort of robust and *ad hoc* interoperation among arbitrary software systems connected to the Internet, in a very wide range of application domains. The transport aspect of this notion of interoperation depends on a number of widely known internet technologies, such as TCP/IP, HTTP, XML and others that are applicable to interacting systems on local networks as well as on the World Wide Web [5].

Semantic interoperability, on the other hand, depends on concepts, formalisms and technologies developed over the past several decades in the related disciplines of logic programming and artificial intelligence. RDF is a World Wide Web Consortium (W3C) standard knowledge representation language [6] that allows information to be described as a set of subject-predicate-object statements or logical propositions. These statements, commonly referred to as "triples", organize information resources (something associated with a unique URI) or literals (strings, numerals, etc.) into subjects, objects and predicates (binary relationships between subjects and objects). RDF is serialized as XML, and thus is easily transmitted and processed as any other XML document. However, RDF documents incorporate a foundation for machine readable semantics that greatly enhances interoperability between distributed software systems [7]. Additional and substantial machine interpretable semantics are added to RDF assertions when these sentences are constrained by and consistent with an OWL ontology.

OWL is a multi-layered knowledge representation language specifically intended to support and promote system interoperability on the Semantic Web. [8] OWL adds an hierarchical classification structure as well as a number of

---

<sup>4</sup> Berners-Lee, Tim, Hendler, Jim, Lassila, Ora, The Semantic Web, Scientific American available at: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>

<sup>5</sup> Berners-Lee, Tim, Blog on Design Issues available at: <http://www.w3.org/DesignIssues/>

<sup>6</sup> RDF Primer available at: <http://www.w3.org/TR/rdf-schema/#ref-rdf-primer>

<sup>7</sup> Lacey, Lee, OWL: Representing Information Using the Web Ontology Language, Trafford Publishing, 2005

<sup>8</sup> OWL Web Ontology Language Guide, available at: <http://www.w3.org/TR/owl-guide/>

<sup>9</sup> Jena Home Page available at: <http://www.hpl.hp.com/semweb/>

<sup>10</sup> Protégé Home Page available at: <http://protege.stanford.edu/>

<sup>11</sup> Baader, Calvanese, McGuinness, Nardi and Patel-Schneider, Description Logic Handbook

ways to further define binary relationships (properties). The result is in a formal language that can be used to define a logical terminology for use in describing a particular domain. This enables the use of an existing, robust collection of automatic reasoners and other logical tools to classify, evaluate, query, apply rules and otherwise operate on the descriptions that conform to the ontology. In theory, reference to this logical foundation, also supports the automatic comparison of concepts defined in different ontologies, allowing an unprecedented degree of interoperability.

### 3. THE ILIUM FRAMEWORK

For several years we have been conducting systems requirements investigations that have been dependent on the interoperation of a number of legacy engineering and analysis systems as well as some prototype software systems under development. In such investigations, conventional practice is to develop a conceptual design - that typically features some advanced technology - and to use a variety of simulations and analysis tools to examine some particular aspect of the design. Each tool provides some insight to a select set of questions in a particular context; no tool provides a comprehensive perspective in a range of situations. The interoperation is usually accomplished “manually” by a team of human operators who first analyze the results of one tool and then infer appropriate input to configure operations in a subsequent tool. Such investigations are necessarily limited in range and scope and subject to errors in inference and translation.

To remedy these kinds of problems, we have developed and employed a Semantic Web Framework, called the Ilium Framework, as means to assemble a collection of simulations and tools. Among this collection, the Framework provides services to automate systems interoperation as well as to situate all systems in the same information context. We believe this situated collection offers analysts and engineers a broad perspective on the operational environment that can include, for example, links between high level military goals and effects produced by individual engagements, modern “network-centric” command and control structures, formal definitions of situations and situation awareness as well as the effects of cooperating systems in the context of our investigation. In such efforts, ensuring semantic consistency across an increasingly wide range of systems has been a major concern. This paper describes our approach to software semantic interoperability. That approach is based on knowledge representation technologies developed in support of the emerging Semantic Web.

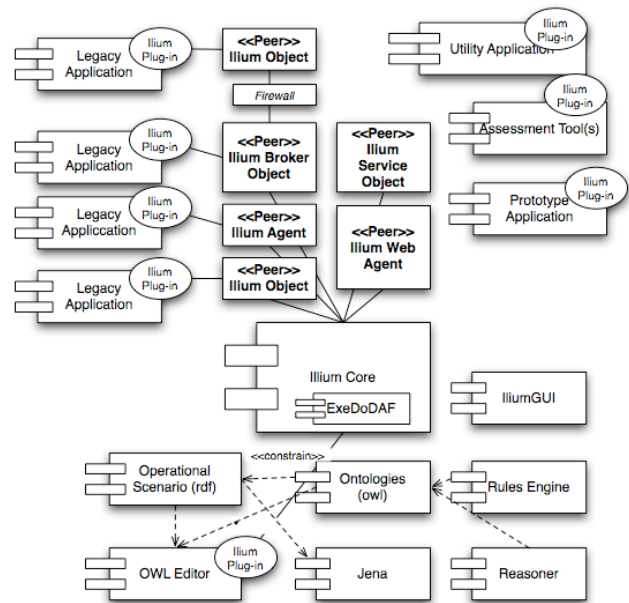


Figure 1. Ilium Framework Architecture

The Ilium Framework has been developed as an integration environment that leverages, and to a certain extent, creates a local Semantic Web. The framework supports an API that allows legacy applications, that may be written in a wide variety of programming languages and depend on a variety of operating systems, to be configured, or “wrapped” as Java plug-ins. The framework provides communication, control, user interface, and other services to synchronize and otherwise orchestrate the activities of the collection. Figure 1 illustrates a generic configuration of the Ilium Framework. In this view, several types of Ilium Objects manage interactions among legacy applications. Basic Ilium Objects act, mirror and manage interactions between equivalent objects or concepts in the legacy application and other systems plugged into the Framework. Broker Objects provide fine grained control, when appropriate requesting specific services and monitoring the state of their surrogates. Agents provide an intelligent interface between specific legacy objects or processes and the rest of the Framework.

Finally, the framework incorporates Jena <sup>[9]</sup>, a popular open source set of Semantic Web tools, and supports Protégé <sup>[10]</sup> as well as an associated set of DL reasoners and rule engines. This latter set of tools and services supports various knowledge-based applications and is used to ensure semantic consistency across the plugged-in systems.

<sup>9</sup> Oberle, Daniel, Semantic Management of Middleware, Springer, 2006

<sup>10</sup> OWL Web Ontology Language Guide, available at: <http://www.w3.org/TR/owl-guide/>

## 4. SEMANTIC INTEGRATION

To do this, it is important to have a common reference for the meaning of terms and symbols used by all the applications in the Framework. Arguably, every software system is, at some level, a model of the domain it serves. Conventionally, this model is expressed explicitly, in a domain model, or implicitly, in the requirements documentation, prior to the system design. In the Ilium Framework, we provide a means for users to specify a formal, logical model - which we call a “scenario” - of the operating system in the domain of discourse. That formal model provides the integrating principle for the component systems that are plugged into the framework. However, to accommodate various component legacy systems, the model is created, in contrast to the conventional approach, after the systems have been implemented.

The creation of a consistent operational model is facilitated by the existence a collection of Ilium domain ontologies. An ontology, in this sense, is a kind of machine terminology or lexicon that prescribes the meaning of terms used by the system. Every significant concept in the domain of discourse and possible relationships among them are described in the Framework’s collection of ontologies. The ontologies establish the fundamental “meaning” for these terms and symbols in the Framework that effectively express a domain theory for the assembled system and provide a mathematical basis for machines to compute and operate on those meanings. This approach is not unique.

Notably, Oberle [7] proposes a similar approach to interoperability, but focuses on middleware and other software artifacts. We attempt a more direct approach, focusing on the semantics of the domain itself.

In another sense, domain ontologies are akin to database schema and to class libraries in object-oriented programming languages. In the Ilium Framework, we exploit those similarities to forge links between plugged-in component systems. The broad perspective on the domain resident in the ontologies provides a flexible, extensible means to situate component systems, that typically express a narrower perspective, within the composite system. Thus, versions of concepts, used in component systems (e.g. classes in object oriented systems and equivalent notions in procedural systems), can then be identified and represented as subsumed concepts in the ontology.

Figure 2 illustrates a portion of a simple ontology for such a simplified “Shoe Store” domain. Thus, we see, in this domain there are some physical things, such as Products and Persons and non-physical things such as Transactions. There are also different, more specialized, kinds of these things (Shoes, Shoestrings, Orders, Sales, Customers and SalesPersons) and some intrinsic attributes that describe, and distinguish, the concepts. And there are some properties that relate the various concepts. For example, a Transaction is an activity in which some Persons and a Product participate; a ShoeSales activity has a shoe that is soldIn the Transaction which is soldTo a Customer and soldBy a SalesPerson.

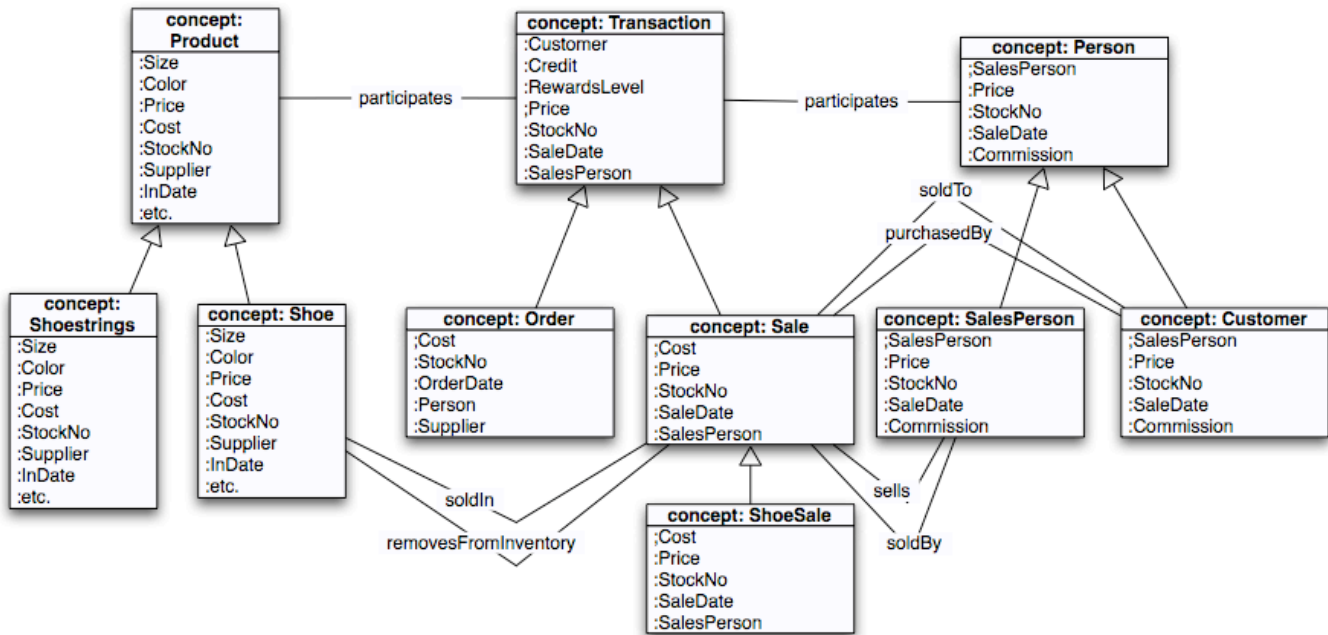


Figure 2. A Simple Shoe Store Ontology

In addition to establishing a common semantic foundation for component systems, the Ilium ontologies define the conceptual relationship between the more limited

perspective of a component system and a broad view of the domain. Concepts defined in the high-level ontologies subsume all the concepts contained in the various plugged-

in systems. Thus, they both offer an index to attribute data for those concepts and indicate relationships to equivalent concepts in other systems. For example, a “Shoe” concept in a “Shoe Store” Ilium Ontology subsumes a “ShoeProduct” class used in an Inventory Management system and a “ShoeInterest” concept in a Customer Relations Management System. Then, the system is assured that a particular pair of brown loafers is the proper subject of a decision to re-order more of shoes of that type.

Inventory Management, Financial Transaction, Store Operations and Payroll Applications that all have some notion of a pair of shoes. Nevertheless, each sees the same pair of shoes differently, names them differently, and, in fact is interested in different attribute data about the shoes. All of these perspectives are unified in the upper ontologies and appropriate relations between them are specified. Moreover, new relationships - and indeed , new related concepts relevant to the extended collection of systems can be described and correctly applied.

Figure 3 illustrates this idea for the Simple Shoe Store domain described in Figure 2. In this case there are notional

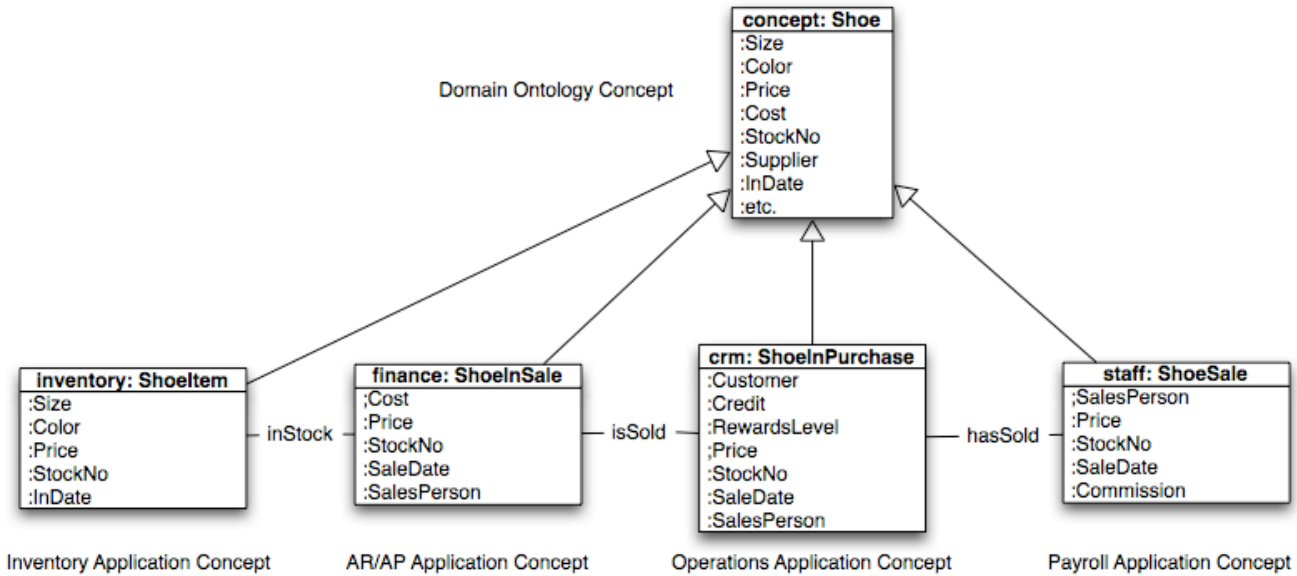


Figure 3. Foundation ontology concept subsumes application concepts

Given such an ontology, it is possible to construct a particular model of operations expected to be supported in the domain from instances of the concepts defined in the ontology. The model may describe an actual or expected event, activity or process (e.g. a transaction, the manufacture of an artifact, a system engineering process). The resulting model, or “scenario”, describes - and constrains - a coherent, consistent semantic interoperation of the composite system. So, in the case of the Shoe Store example, there is confidence that, in a conforming system, all references to some “Shoe” by any of the component systems are both about the same individual and are appropriate to the type of individual (that is, a “shoe” is something that is worn by a person and not, for example, a document).

The operational model, having been constructed in a way that satisfies the domain theory, is also something that preserves truth. That is, all valid statements in the model must be logically true under the axioms in the theory. Thus, to the degree that we believe the theory to be an accurate description of the actual physical domain, the

model is trusted to provide a reliable information about the “real” world of which it is a model, as well as insight into events and objects in it. [11, 12]. Logical consistency in the model provides confidence in its ability to correctly describe corresponding states in the “real” domain the model describes.

<sup>11</sup> Protégé Ontology Editor documentation available at <http://protege.stanford.edu>

<sup>12</sup> Top Braid Composer Datasheet available at <http://www.topquadrant.com/topbraidcomposer.html>

<sup>12</sup> Welty, Chris and Nicola Guarino. 2001. Support for Ontological Analysis of Taxonomic Relationships. *J. Data and Knowledge Engineering*, 39(1):51-74. October, 2001.

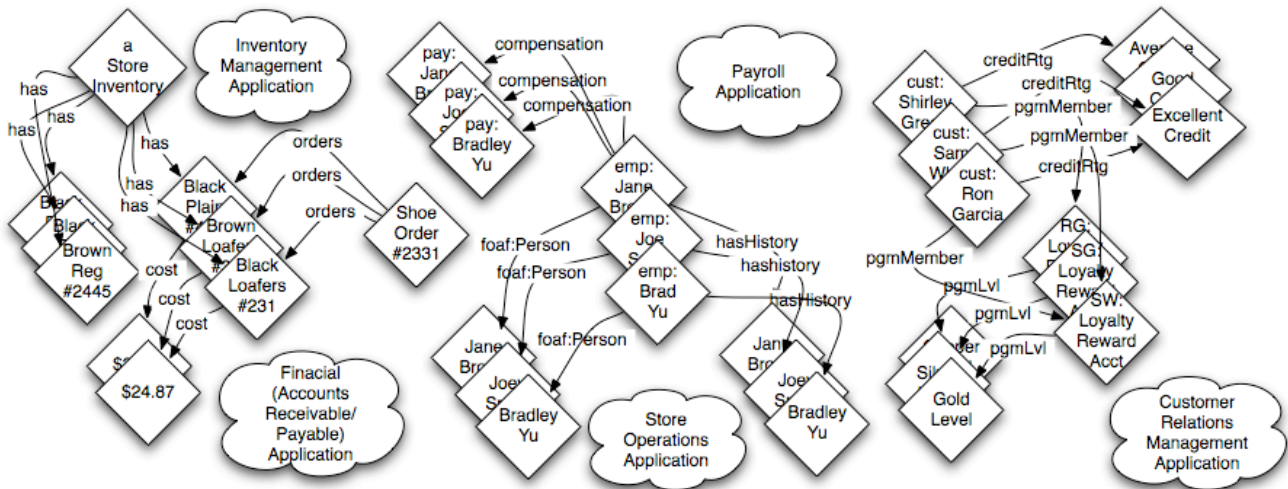


Figure 4: Excerpt of an Operational Scenario

Within the Framework, the scenario (model) organizes and correlates all the information that each of the applications needs to function, as well as any additional information particular to the composed system. Figure 4 illustrates a portion of a notional operational scenario for the Shoe Store example.

This model also serves as a system knowledge base for knowledge based components and becomes the basis of communications between the cooperating, plugged-in component systems. The Ilium Framework uses the model to create Java objects that coordinate and control the system, as well as specialized agents that support and extend the functionality of the composite system. Each plugged-in application is able to obtain input data from the model and export output information to augment or update the model in a form that allows cooperating systems to operate on the updated information and to report results to users in a flexible report format.

## 5. THE ONTOLOGIES

Ontologies define and make explicit the semantics, or the “meaning”, assigned to computed data, in a particular domain. Ontologies represented in OWL, are intended to be “machine readable” so that the data computed according to defined concepts in one location can be properly acted on (e.g. interpreted, processed, reasoned about) by different machines and different governing programs in another location. [13] Consequently, OWL ontologies are not easily developed without Editors and other tools. Fortunately, a number of good tools are available. Ilium ontologies (domain models) themselves are developed using the

<sup>13</sup> Natalya F. Noy and Deborah L. McGuinness, *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford University, Stanford, CA, 94305

Protégé Editor/OWL Plug-in<sup>[14]</sup> and more recently, the Top Braid Composer Semantic Web Ontology Editor [15]. In most important aspects, they both provide access to large, active user communities and wide range of tools and methodologies. Thus, the basic Ilium ontology structure benefits from knowledge engineering experience and methods (e.g. OntoClean [16]), as well as various production tools available with Protégé and DL Reasoners to ensure logical consistency and completeness in the model that will provide the foundation for meaningful interoperations of the composed system.

<sup>14</sup> Alan Rector, *Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL, K-CAP'03*, October 23–25, 2003, Sanibel Island, Florida, USA. pp 121-8 2003

<sup>15</sup> Cyc Homepage available at <http://www.cyc.com/>

<sup>15</sup> Open Cyc home page available at <http://www.opencyc.org/>

<sup>15</sup> SUMO Description and Home Page available at <http://www.ontologyportal.org/>

<sup>16</sup> SENSUS Description and Home Page available at <http://www.isi.edu/natural-language/projects/ONTOLOGIES.html>

<sup>16</sup> DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering, ontology and documents available at <http://www.loa-cnr.it/DOLCE.html>

In general, Ilium ontology development has been influenced and guided by knowledge engineering methods articulated in the following rules:

- There is no one correct way to model a domain— there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.
- Ontology development is necessarily an iterative process.
- Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain. [17]

Ontologies organize the concepts (e.g. physical objects, principles, ideas, etc.) in a domain and describe the relationships between them. However, an attempt to define all the possible concepts in a single multi-purpose monolithic model is problematic and, in fact nearly impossible to accomplish. To avoid this problem the Ontologies are partitioned into modular perspectives and layered to support increasing detail as the notion of domain becomes more specific. This approach is consistent with experience in disciplines that have developed and used large scale ontologies over many years [18] The modular ontology approach is also emerging Semantic Web “best practice”, and is supported by the OWL knowledge representation language.

The Top, most general layer describes key (system and software) concepts of the Ilium Framework, and the basic or “foundational” concepts that are used to organize and provide structure more specific layers. We believe that reference to “foundational ontologies” that expose ontological commitment and underpin analysis at more detailed layers in the structure is a key to broad interoperability among distributed ontologies and systems

---

<sup>17</sup> Nicola Guarino, Claudio Masolo, Stefano Borgo, Aldo Gangemi and Alessandro Oltramari, Ontology Infrastructure for the Semantic Web: Wonder World Deliverable D18, Laboratory for Applied Ontology, Trento, Italy, 2001 available on line at <http://www.loa-cnr.it/DOLCE.html>

<sup>18</sup> DUL.owl ontology available at [www.loa-cnr.it/ontologies/DUL.owl](http://www.loa-cnr.it/ontologies/DUL.owl)

<sup>18</sup> Amy Knutilla, Steven Polyak, Craig Schlenoff, Austin Tate, Shu Chiun Cheah, Steven Ray, and Richard Anderson Process Specification Language: An Analysis of Existing Representations, NIST report available at <http://www.mel.nist.gov/msidlibrary/doc/psl-1.pdf>

<sup>18</sup> Process Specification Language Ontology available at <http://www.mel.nist.gov/psl/>

that depend on them. We have reviewed (and continue to be informed by) a number of respected Upper Ontologies, including Cyc [19, 20], SUMO [21] and SENSUS [22]. We eventually determined that the DOLCE ontology [23, 24] specifically developed in the European Union Wonder Web to support distributed ontologies on the Internet, is best aligned with our interoperability goals. We have subsequently found DOLCE to be sufficiently expressive to accommodate all our varied needs and to be a very useful guide to domain analysis.

---

<sup>19</sup> Ontology for Geography Markup Language (GML3.0) owl ontology available at [oki.cae.drexel.edu/~wbs/ontology/2004/09/ogc-gml](http://oki.cae.drexel.edu/~wbs/ontology/2004/09/ogc-gml)

<sup>20</sup> Ontology for Geography Markup Language (GML3.0) of Open GIS Consortium (OGC) Home Page available at <http://loki.cae.drexel.edu/~wbs/ontology/ogc-gml.htm>

<sup>20</sup> Peter Maguire, Using THUNDER for Campaign Studies, DSTO-TN-0303, DSTO, Melbourne, August 2000

<sup>21</sup> User Manual, SEAS Version 3.7, U.S. Air Force, SMC/XR, February 2007

<sup>22</sup> Bijan Parsia and Evren Sirin, Pellet and OWL DL Reasoner, MINDSWAP Research Group, University of Maryland, College Park available at <http://iswc2004.semanticweb.org/posters/PID-ZWCSLQK-1090286232.pdf>

<sup>22</sup> Pellet Home Page available at <http://pellet.owldl.com/>

<sup>22</sup> FaCT++ Home Page available at <http://owl.man.ac.uk/factplusplus/>

<sup>22</sup> OWLIM Home Page available at <http://www.ontotext.com/owlim/>

<sup>22</sup> A SPARQL Tutorial available at <http://jena.sourceforge.net/ARQ/Tutorial/index.html>

In particular, we import a simplified “light weight” OWL version of DOLCE, known as DUL (DOCLE Ultra Lite) [25] as the starting point of all the Ilium Ontologies ( as imported by the basic Ilium Framework ontology). DUL divides the world into both 4 physical dimensional categories of Event, State, Action and 3 dimensional categories, including Information Realization, Object, Quality and Region. A more detailed description of DOLCE and DUL is beyond the scope of this paper; numerous papers and other documentation is available at the LOA DOLCE web site <http://www.loa-cnr.it/DOLCE.html>. Figure 5 is a view of the basic Ilium Ontology and imported DUL owl ontology depicted in the Top Braid Composer Editor. Classes and Properties with the prefix “dul:” are imported from

DUL.owl. The Class Structure (tree) is visible in the left panel. A more detailed analysis of the DUL “Physical Object” class is visible along with several Classes that have been imported from the Ilium Geo ontology and subsumed by the DUL PhysicalObject class. Thus, a “geo:TopographicFeaure” is defined as a kind of “PhysicalObject”. Likewise, Object Properties are seen in the right panel. Properties, like Classes, may have subordinate Properties that specialize the parent Property. In the depicted example, “Dul:isRoleOf” is a specialization of the “dul:classifieds” relation. The center panel displays details (axioms, documentation, etc.) related to the selected Class.

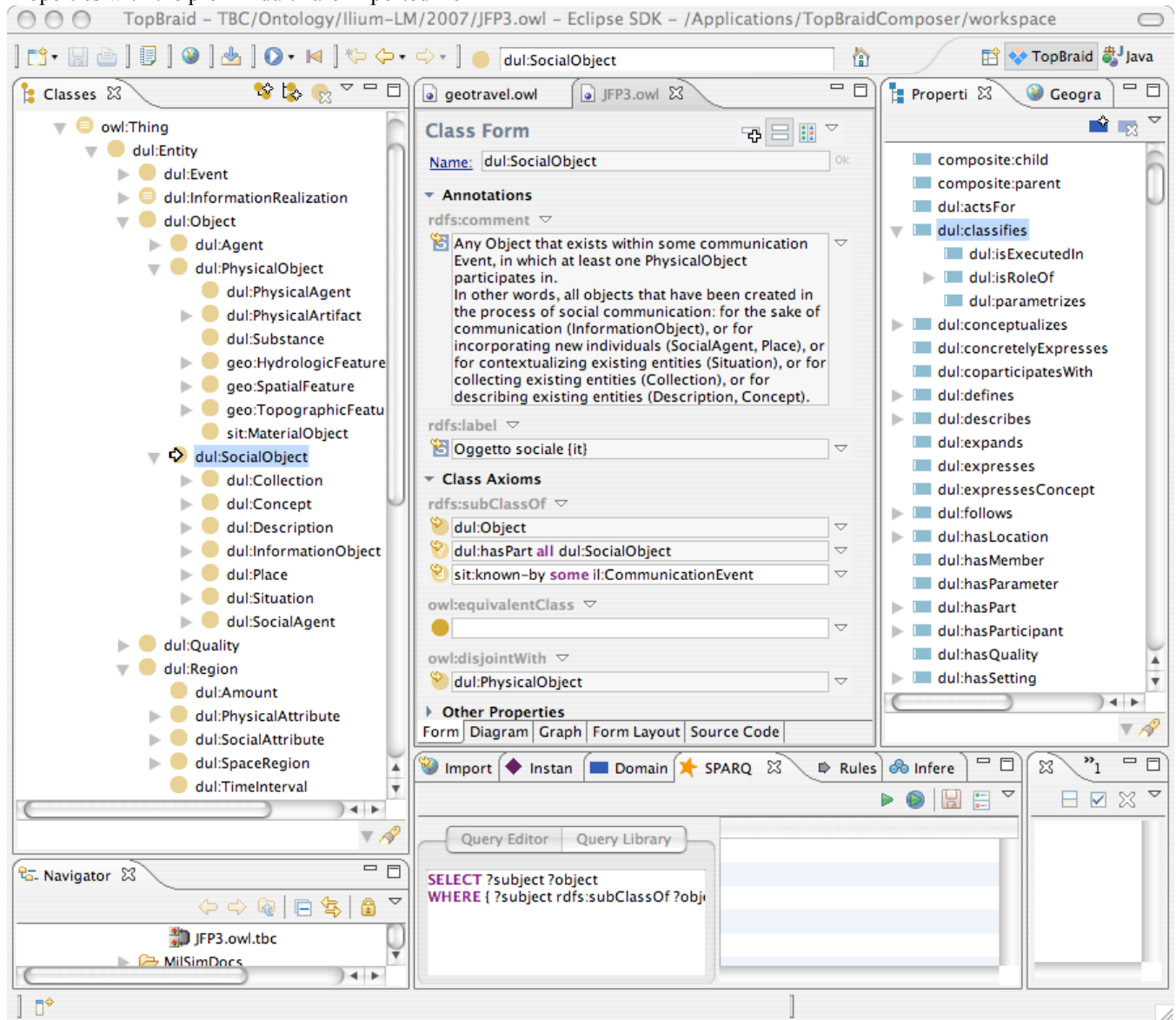


Figure 5. DOLCE Ultra Lite Foundational Ontology

The Middle layer includes a number of specialized perspectives on a generic domain. These ontologies focus on material assets (IliumAsset), organizations (IliumOrg),

geographic and cartographic features (IliumGeo) and situations (IliumSit). At this level, analytical standards associated with these perspectives are incorporated, and

where possible, imported, into the ontologies to leverage those standards and improve long-term compatibility with other ontologies that reference those standards. Thus, the Ilium Asset ontology incorporates process and service concepts from the NIST Process Specification Language [26,

27] and the Ilium Geo ontology incorporates concepts from (and may import) the Open Geographic Modeling Language ontology [28, 29]. Figure 6 depicts a view of the IliumAsset Ontology in the Editor.

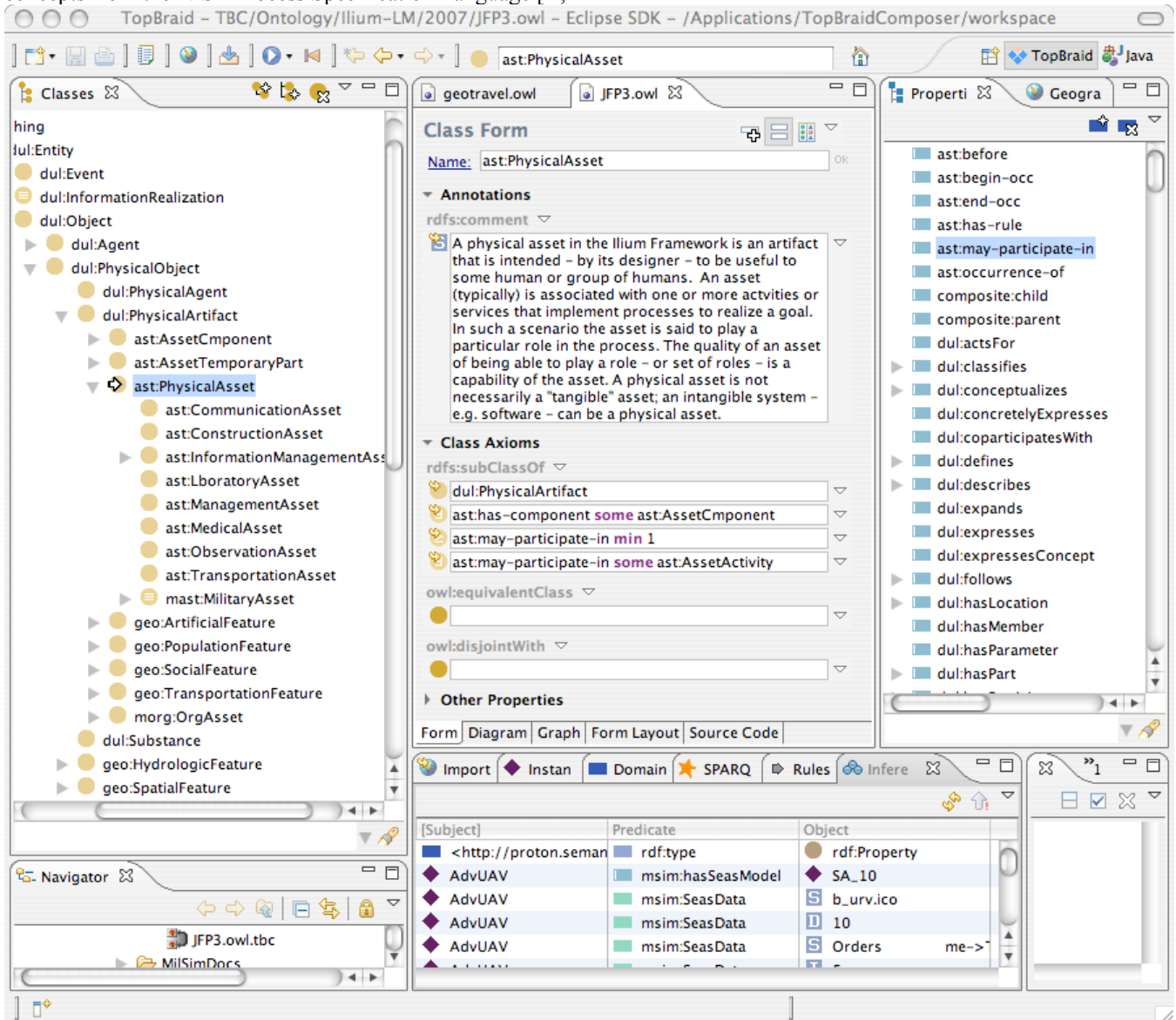


Figure 6. Ilium Asset Ontology

Below the Middle layer are found domain specific ontologies that mirror and extend the ontological perspectives of the Middle layer. We currently have developed a layer of Military OWL ontologies to serve as domain models for a number of legacy operations analysis simulations, aeronautical engineering design tools and software prototypes that are used to support the integrated design and development of aeronautical systems. For example, the MilAsset ontology models military platforms, command and control systems and similar physical assets; a specialized extension of that describes concepts peculiar to Unmanned Aircraft.

Figure 7 depicts a view of the MillAsset ontology in the Editor. Concepts and Properties with the prefix “mast:” are elements of the MilAsset ontology. In this view, specialized classes formilitary ManeuverableAircraft are depicted as sub-classes of FixedWingAircraft, and a large number current maneuverable aircraft types are specified. Aircraft types may have specialized models that describe, for example, sensor configurations or performance characteristics of interest, that are particular to one or more plug-ins. In this view of the editor, an number of model relationships are visible in the “Properties” window at the right of the frame. In the SPARQL panel at the bottom, a

SPARQL query (discussed in more detail in Section 6) has retrieved the “SEAS Plane” model information for a particular F-15 instance.

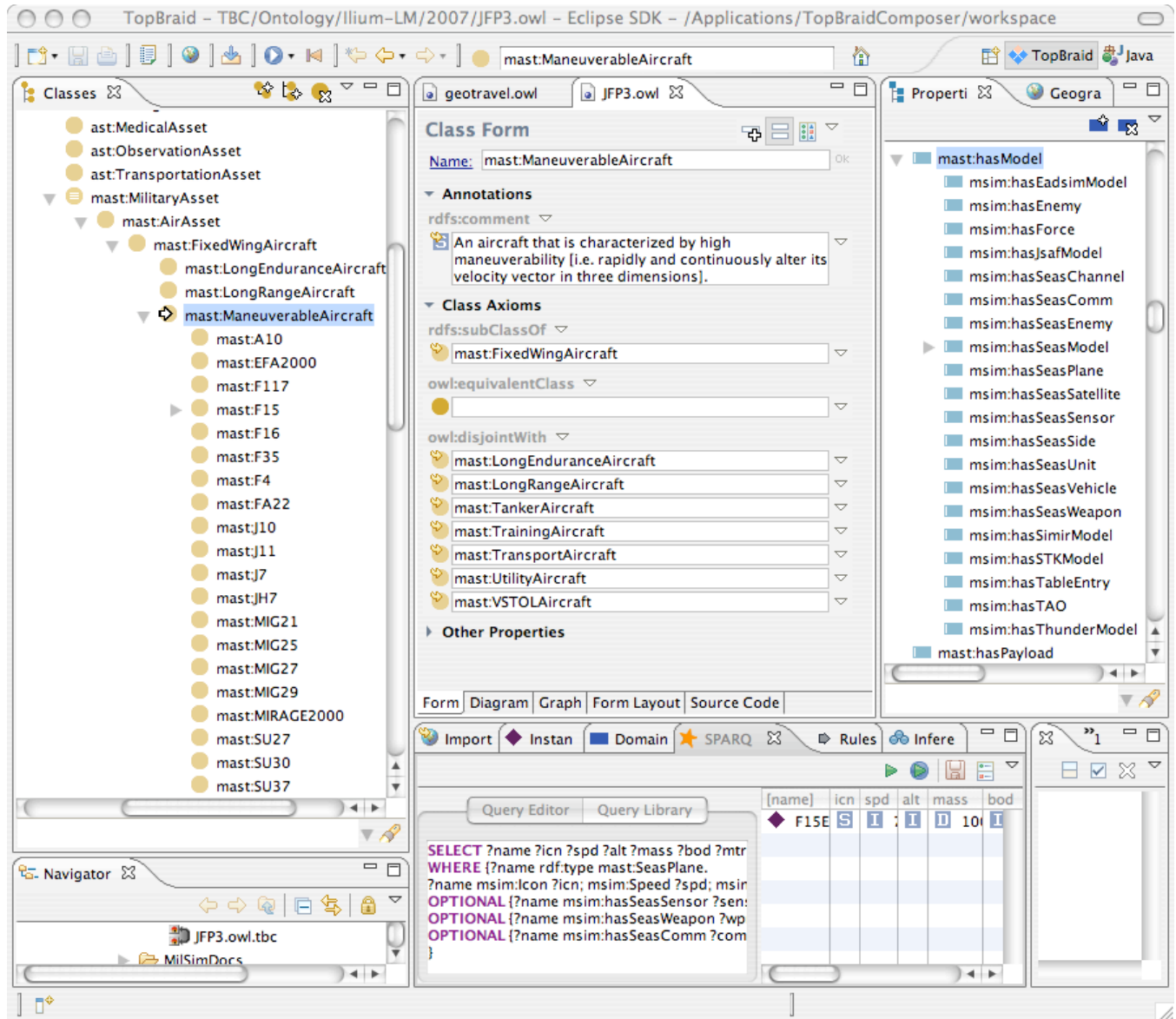


Figure 7. Military Asset Ontology

A number of other military ontologies also exist. The MilOrg ontology describes military and related political organizations as well as particular Roles they play in military affairs, Tasks they are required to perform and specialized military Actions that perform them. MilGeo defines a military perspective on geography, with emphasis on terrain, population and cultural features of interest to military operations. MilGeo contains classes that describe military facilities, including Forts, Airbases, NavalPorts and Forward Operating Locations. The MilSit ontology describes Conflicts, Campaigns, Missions, Plans, and similar concepts. It also defines exactly what constitutes a situation, and classifies various types of situations. Specialized ontologies relating to military information and communications exist and may be imported in some cases to provide increased detail for those concerns. Finally, to support military Modeling and Simulation applications, the MilSim ontology provides a platform for importing relevant ontologies for a particular study or exercise, and a location to create concepts peculiar to individual simulations.

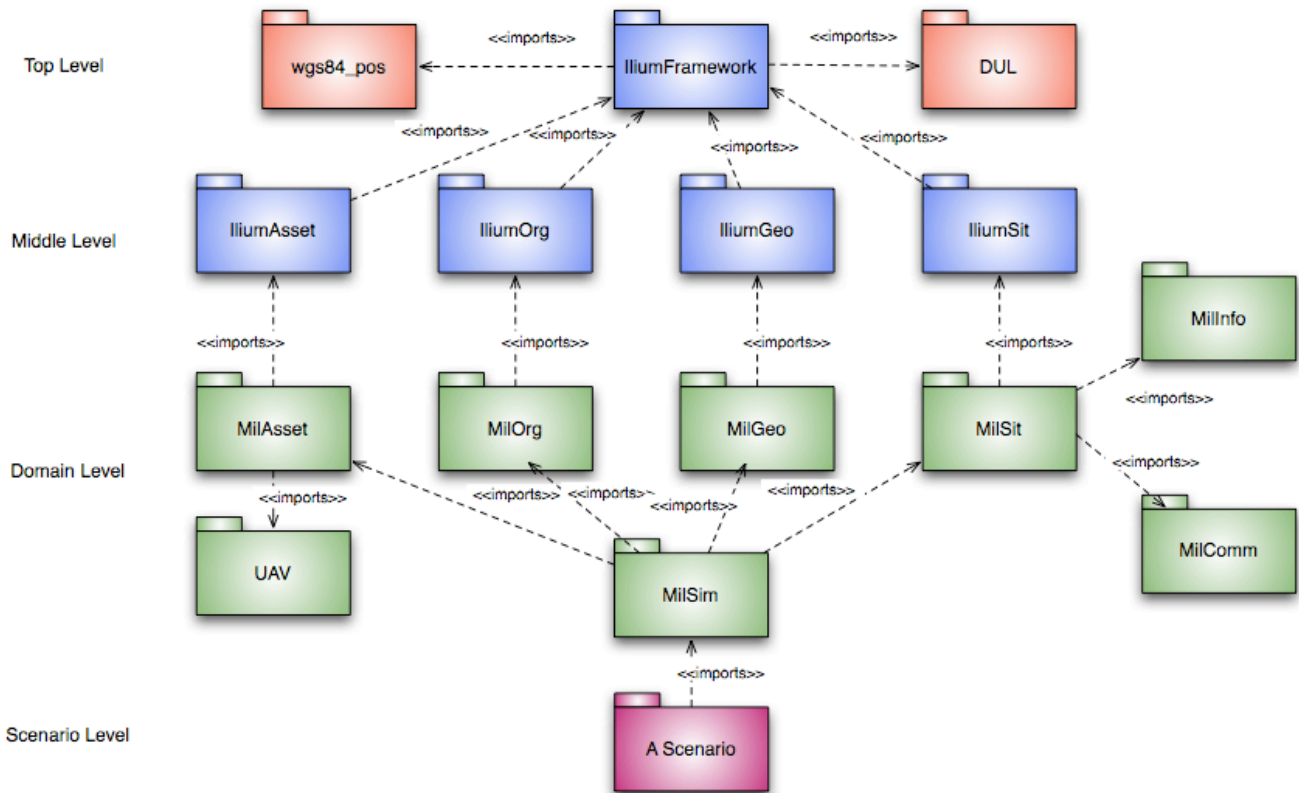


Figure 8. Ilium Ontology Suite (typical)

It is possible, and in fact, common, to employ only a few of these ontologies in particular studies. For example, many studies are only concerned with physical platforms and therefore only need the `MilAsset` ontology (which imports `IliumAsset`, `IliumFramework` and `DUL`). Figure 8 illustrates the existing Ilium Ontology Suite and a typical import pattern for a complex Modeling and Simulation application scenario. Currently the illustrated suite (less `MilInfo`, `MilComm`, and `UAV`) defines 1240 OWL Classes, 274 OWL Object Properties, and 188 OWL Datatype Properties.

## 6. AN EXAMPLE APPLICATION

We are currently using the described approach to conduct systems requirements and other engineering analyses for military aerospace systems. In these investigations, it is useful to consider detailed aspects of the system design (or,

software prototype behavior) in the context of large scale, network-centric military operations. In the past year, in particular, we have assembled a collection of well known and widely accepted military simulations to prepare for an extensive investigation of requirements for autonomy in unmanned military platforms.

It is believed that useful insights in the study will be sensitive to important details of system behavior, supporting sensor, platform and communications technologies as well as the combined effects of these technologies on an advanced, highly networked military force. Thus the study environment must provide a means to reliably model and evaluate significant technical detail and to measure the effects, as well as propagation of those effects, throughout a broad operational context.

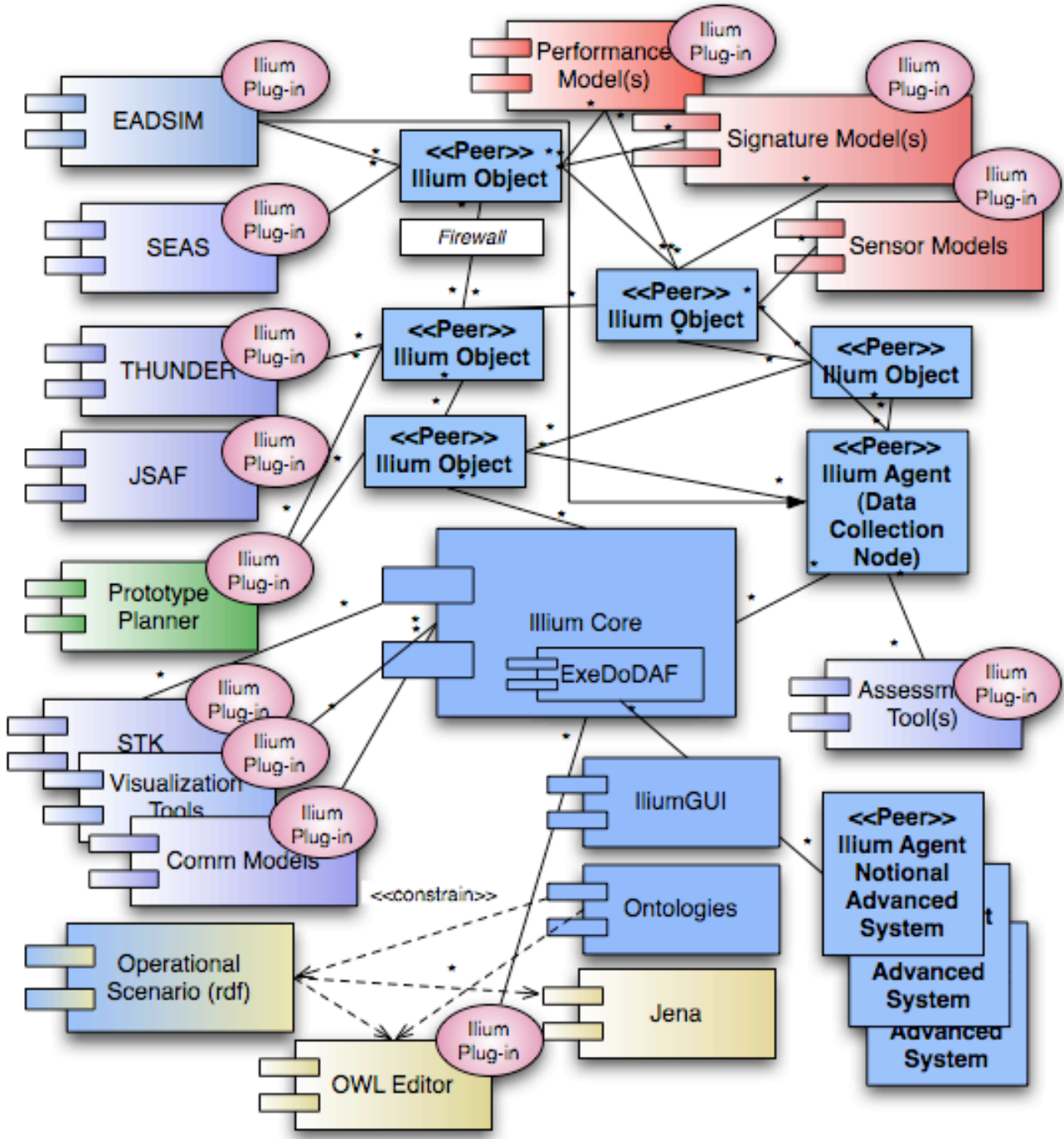


Figure 9. A Military Modeling and Simulation Configuration of the Ilium Framework

To do that we have assembled a collection of trusted military simulations that span the range of such applications from fine-grained simulations that focus on detailed interactions between two entities to coarse-grained simulations of military campaigns. Figure 9 illustrates the Ilium Framework configured to accommodate those simulations, as well as representative prototype systems. In this case, the legacy applications are simulations, analysis systems and design tools. Prototypes are notional UAV

autopilots, route planners, decision support systems and similar applications. The Framework augments these applications with control objects and agents that coordinate computations in the composed system and specialize agents that typically represent the characteristics of a new system or technology that cannot be easily or adequately simulated in any of the component legacy systems. In our current study, the following systems have Ilium Framework plug-ins and are used as components in the study environment.

THUNDER is a campaign simulation that models national military forces and military operations that extend over months. [30] Characteristics of individual systems are evaluated statistically and their effects on the overall campaign are not explicitly known. Political, social and cultural objects and concepts are not included in the model. In this configuration, THUNDER is used to generate force level tasking (mission orders) and to evaluate and adjust for the results of missions with respect to campaign plans.

SEAS is a force level simulation that simulates battles between major forces in combat operations that typically last for hours and as much as a day. SEAS features a flexible, rules based decision logic that can influence behavior at both the commander and individual combatant level. SEAS executes the missions requested by THUNDER and provides a manageable dynamic context for examining the behavior of prototypes in a range of typical operational situations. [31] A dynamic plug-in supports interaction with the Framework and other plugged-in components.

EADSIM is a trusted model of air defense systems that is, in particular, sensitive to many important design attributes of individual systems. In certain, modified forms, it can reference advanced sensor and engagement decision models. In our study configuration, EADSIM simulates enemy air defenses in selected (e.g. significant to our investigation) portions of the virtual battle. A dynamic plug-in supports system control as well as interaction with the Framework and other plugged-in components.

A prototype aircraft mission planning system plug-in supports virtual real time mission plan creation and updates. In particular, the system provides automatic route planning for selected platforms that have been assigned missions by THUNDER and that are subsequently executed (in simulation) in SEAS and EADSIM.

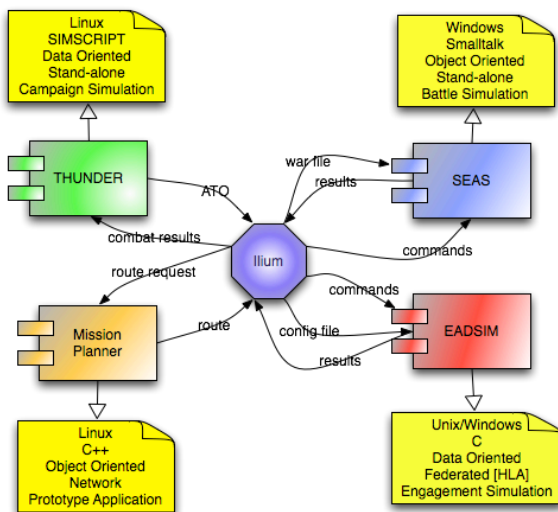


Figure 10: Legacy Components in the Ilium Framework

The Ilium Framework itself provides software agents that are used to model notional or experimental UAV characteristics and behaviors. Depending on the objectives of a particular study, the Framework, may also provide agents that address advanced Command and Control concepts to coordinate the interaction of the various component systems.

We maintain a semantic consistency among the plugged-in component applications by developing a single operational scenario as initial input for a study, and deriving the necessary application configuration data from that source. We create an RDF model of the scenario, based on the Ilium suite of ontologies, that includes:

- political context (issues, objectives, sensitivities, etc.)
- military context (centers of gravity, campaign objectives, etc.)
- geophysical environment
- military units (order of battle, unit equipment lists, etc.)
- command and control
- assets (platforms, weapons, ISR systems, etc.)

Figure 11 below is an excerpt from an operational scenario set in the Southwest U.S. depicting a description of an Air Force Wing assigned to a notional Joint Task Force. The Wing is based at Bishop Air Base and has six Fighter Squadrons assigned to it. Additional detail about each of those squadrons, as well as the base, is found in the model (in this case, an `rdf:resource`) associated with it.

```
<morg:AirForceWing rdf:ID="USAF_366_Air_Exp_Wg">
  <dc:creator>Doug Holmes</dc:creator>
  <mgeo:basedAt rdf:resource="#BishopAB"/>
  <geo:positionedAt rdf:resource="#BishopAB_pos"/>
  <morg:assignedOrg rdf:resource="#USAF_390_Ftr_Sq"/>
  <morg:assignedOrg rdf:resource="#USAF_494_Ftr_Sq"/>
  <dc:date>10/20/07</dc:date>
  <morg:assignedOrg rdf:resource="#USAF_496_Ftr_Sq"/>
  <rdfs:comment>366 AEW [F-22; F-15; KC-135; C-130]</rdfs:comment>
  <morg:assignedOrg rdf:resource="#USAF_389_Ftr_Sq"/>
  <morg:assignedOrg rdf:resource="#USAF_391_Ftr_Sq"/>
  <morg:assignedOrg rdf:resource="#USAF_495_Ftr_Sq"/>
  <rdfs:label>366 Air Expeditionary Wing</rdfs:label>
</morg:AirForceWing>
```

Figure 11. An RDF model of a notional USAF Wing

Figure 12 is another excerpt from the same scenario illustrating a model of a particular Fighter Squadron and one of the F-15E aircraft it operates.

```

<morg:AirForceSquadron rdf:ID="USAF_8_Ftr_Sq">
  <rdfs:label>8th Fighter Squadron</rdfs:label>
  <geo:positionedAt rdf:resource="#GeorgeAB_pos"/>
  <msim:hasSeasModel rdf:resource="#BAFGA"/>
  <mast:hasModel rdf:resource="#BAFGA"/>
  <rdfs:comment>An F-15E Squadron</rdfs:comment>
  <dc:date>9/26/07</dc:date>
  <dc:creator>Doug Holmes</dc:creator>
  <mgeo:basedAt rdf:resource="#GeorgeAB"/>
</morg:AirForceSquadron>

<F-15E rdf:ID="F-15E_05">
  <dul:isReferenceOfRealization
rdf:resource="#AirObject_2007"/>
  <mast:equipment-of rdf:resource="#USAF_7_Ftr_Sq"/>
  <dc:creator>Doug Hollmes</dc:creator>
  <dc:date>9/26/07</dc:date>
  <msim:hasSeasPlane rdf:resource="#F15E"/>
  <mgeo:basedAt rdf:resource="#GeorgeAB"/>
  <geo:positionedAt rdf:resource="#GeorgeAB-pos"/>
  <rdfs:label>F-15E 005</rdfs:label>
</F-15E>

```

Figure 12. An RDF model of a Fighter Squadron and one of its aircraft

Note that, these models also have associated SEAS models, that contain information peculiar to the SEAS simulation, about these entities. This information is used to configure SEAS to properly represent these particular entities. We are also able to insert objects and information that may be of interest in our study, that is not represented in any of the component simulations or other applications. Where those objects are related to an object that is simulated, that relationship permits inferences about the effects on them as a result of actions that are computed in a simulation. For example, it is possible to indicate that GeorgeAB “defends” the capital city and lend special significance to the actions of aircraft that are based there, even though none of the simulations in the composite system have any notion of “capital city”. Finally, the operational scenario provides an explicit record of the assumptions that underly the study and can also include and explicit representation of system and study goals. This practice improves analysis and may enable future knowledge based analytical tools.

Throughout the process of constructing the operational scenario, and when it is complete, we use one or more Description Logic Reasoners to ensure the logical consistency of the the model. A number of these automatic theorem provers are freely available, including Pellet [32, 33], FaCT++ [34], and OWLIM [35] are used in the Framework to compute logical entailments and to “complete” RDF models, as well as to ensure the consistency of models. In the later capacity, frequent checks will identify errors in the construction; some (e.g. Pellet) also indicate the source of the error and aid in repairing it. As a result, we are confident that the operational scenario is a sound model. The primary use of the Reasoners allows us to significantly extend the explicit RDF model that is created. For example,

a squadron is explicitly specified as “assignedTo” a wing, and that relationship is the inverse of “assignedOrg” then the system can infer that the wing has the squadron as an assigned organization, even though that fact has never been asserted. Similarly, if the same relationship is defined as a transitive relationship, it is possible to infer that a flight that is assigned to the squadron is also assigned to the wing.

Once we have an operational scenario that has been classified by a Reasoner, we then use the completed model to configure the composite system - Ilium and the plugged-in systems - to execute the simulation. We use SPARQL, a W3C standard query language designed to access RDFL to extract data from the scenario to create the input files needed to configure the various applications.[36]

```

SELECT ?name ?pos ?long ?lat ?alt ?vis
WHERE {?name rdf:type mast:SeasLocation.
?name geo:positionedAt ?pos.
?pos pos:long ?long.
?pos pos:lat ?lat.
?pos pos:alt ?alt.
?name msim:Visible ?vis
}

```

Figure 13. A typical SPARQL query

In a similar fashion, SPARQL is used to extract data from the scenario to create the necessary Ilium java surrogate, control and agents. SPARQL can also be used to review the scenario and answer questions that have arisen since the inception of the study. At this point, we are able execute the scenario with confidence that it will produce reliable results.

## 7. CONCLUSIONS

We have developed a methodology and supporting tools for creating an “operational scenario” that supports the semantic interoperability of an *ad hoc* collection of legacy applications and extends their capabilities. The method depends on and ensures the logical integrity of the composite system. Therefore, when we assemble as a collection of legacy component systems that were not originally intended to interoperate with other systems, we can be confident that the composite system will produce consistent results. Logical consistency implies that, to the degree that we trust the interpretation underlying the model, the results of operations on the model are trustworthy. If, for example, the model is based on a Newtonian interpretation of physics, the model ought to provide reliable answers to questions about automotive and even aeronautical engineering, but probably not to all questions in astronomy or cosmology. This methodology extends the utility of trusted simulations allowing integration of fine-grained simulations that are sensitive to design requirements with high level, coarse-grained simulations that are sensitive to acquisition issues and policy. The potential benefits of this sort of interoperation may range from obvious production efficiencies to clearer insights into system

requirements. Finally, an important side-effect of the approach is the OWL/RDF knowledge base formed by the combination of the operational scenario and the results of the operations of the legacy systems. That knowledge base and the use of SPARQL queries and SWRL rules effectively expands the functionality of the system and greatly improves the analysis of the output of the system of cooperating simulations and tools.

We have prepared a foundation for the application of Semantic Web and other knowledge based tools in the analysis and design of unmanned systems. We anticipate the development and application of these tools and the addition of autonomy directed Multiple Agent Systems (MAS) that use RDF/XML inter-agent communications in the coming year.

## REFERENCES

- <sup>1</sup> NAFCAM (2001) Exploiting E-Manufacturing: Inoperability of Software Systems Used by U.S. Manufacturers available at: <http://www.mel.nist.gov/div826/msid/sima/FinalReportSummary.pdf>
- <sup>2</sup> Berners-Lee, Tim, Hendler, Jim, Lasilla, Ora, The Semantic Web, Scientific American available at: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>
- <sup>3</sup> Berners-Lee, Tim, Blog on Design Issues available at: <http://www.w3.org/DesignIssues/>
- <sup>4</sup> RDF Primer available at: <http://www.w3.org/TR/rdf-schema/#ref-rdf-primer>
- <sup>5</sup> Lacey, Lee, OWL: Representing Information Using the Web Ontology Language, Trafford Publishing, 2005
- <sup>6</sup> OWL Web Ontology Language Guide, available at: <http://www.w3.org/TR/owl-guide/>
- <sup>7</sup> Jena Home Page available at: <http://www.hpl.hp.com/semweb/>
- <sup>8</sup> Protégé Home Page available at: <http://protege.stanford.edu/>
- <sup>9</sup> Baader, Calvanese, McGuinness, Nardi and Patel-Schnieder, Description Logic Handbook
- <sup>10</sup> Oberle, Daniel, Semantic Management of Middleware, Springer, 2006
- <sup>11</sup> OWL Web Ontology Language Guide, available at: <http://www.w3.org/TR/owl-guide/>
- <sup>12</sup> Protégé Ontology Editor documentation available at <http://protege.stanford.edu>
- <sup>13</sup> Top Braid Composer Datasheet available at <http://www.topquadrant.com/topbraidcomposer.html>
- <sup>14</sup> Welty, Chris and Nicola Guarino. 2001. Support for Ontological Analysis of Taxonomic Relationships. *J. Data and Knowledge Engineering*. **39**(1):51-74. October, 2001.
- <sup>15</sup> Natalya F. Noy and Deborah L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, Stanford University, Stanford, CA, 94305
- <sup>16</sup> Alan Rector, Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL, *K-CAP'03*, October 23–25, 2003, Sanibel Island, Florida, USA. pp 121-8 2003
- <sup>17</sup> Cyc Homepage available at <http://www.cyc.com/>
- <sup>18</sup> Open Cyc home page available at <http://www.openecyc.org/>
- <sup>19</sup> SUMO Description and Home Page available at <http://www.ontologyportal.org/>
- <sup>19</sup> SENSUS Description and Home Page available at <http://www.isi.edu/natural-language/projects/ONTOLOGIES.html>
- <sup>20</sup> DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering, ontology and documents available at <http://www.loa-cnr.it/DOLCE.html>
- <sup>21</sup> Nicola Guarino, Claudio Masolo, Stefano Borgo, Aldo Gangemi and Alessandro Oltramari, Ontology Infrastructure for the Semantic Web: Wonder World Deliverable D18, Laboratory for Applied Ontology, Trento, Italy, 2001 available on line at <http://www.loa-cnr.it/DOLCE.html>
- <sup>22</sup> DUL.owl ontology available at [www.loa-cnr.it/ontologies/DUL.owl](http://www.loa-cnr.it/ontologies/DUL.owl)
- <sup>23</sup> Amy Knutilla, Steven Polyak, Craig Schlenoff, Austin Tate, Shu Chiun Cheah, Steven Ray, and Richard Anderson Process Specification Language: An Analysis of Existing Representations, NIST report available at <http://www.mel.nist.gov/msidlibrary/doc/psl-1.pdf>
- <sup>24</sup> Process Specification Language Ontology available at <http://www.mel.nist.gov/psl/>
- <sup>25</sup> Ontology for Geography Markup Language (GML3.0) owl ontology available at [oki.cae.drexel.edu/~wbs/ontology/2004/09/ogc-gml](http://oki.cae.drexel.edu/~wbs/ontology/2004/09/ogc-gml)
- <sup>26</sup> Ontology for Geography Markup Language (GML3.0) of Open GIS Consortium (OGC) Home Page available at <http://loki.cae.drexel.edu/~wbs/ontology/ogc-gml.htm>
- <sup>27</sup> Peter Maguire, Using THUNDER for Campaign Studies, DSTO-TN-0303, DSTO, Melbourne, August 2000

<sup>28</sup> User Manual, SEAS Version 3.7, U.S. Air Force, SMC/XR, February 2007

<sup>29</sup> Bijan Parsia and Evren Sirin, Pellet and OWL DL Reasoner, MINDSWAP Research Group, University of Maryland, College Park available at <http://iswc2004.semanticweb.org/posters/PID-ZWSCSLOK-1090286232.pdf>

<sup>30</sup> Pellet Home Page available at <http://pellet.owldl.com/>

<sup>31</sup> FaCT++ Home Page available at <http://owl.man.ac.uk/factplusplus/>

<sup>32</sup> OWLIM Home Page available at <http://www.ontotext.com/owlim/>

<sup>33</sup> A SPARQL Tutorial available at <http://jena.sourceforge.net/ARQ/Tutorial/index.html>

## BIOGRAPHY

**Douglas Holmes** is co-founder and Senior Partner of Java Professionals, Inc. In the past twenty-two years, he has managed and participated in numerous artificial intelligence and knowledge-based programs for DARPA and other research agencies, as well as commercial applications in the petroleum and other sectors. He is currently developing ontologies and applying Semantic Web technology to support research and development of military unmanned systems. He also has over twenty years experience as an Air Force Fighter Pilot and Fighter Weapons School Instructor. Mr. Holmes has a B.S. in Mathematics and Basic Sciences from the U.S. Air Force Academy and a M.S. in Management Information Systems from Golden Gate University.

**Richard Stocking** is the lead Program Investigator/PM for Net Centric Operations – Warfare Analysis efforts for Lockheed Martin Aeronautics Advanced Development Programs (The Skunk Works™). He is currently leading efforts researching autonomous UAV operations. Current efforts include the integration of Multiple Agent Systems and other autonomy systems within the Ilium Framework. He has over thirty years experience and over 11,000 flight hours with multiple C4ISR systems in the US Army and US Navy. Mr. Stocking has a M.S. in Systems Technology from the Naval Postgraduate School.