Use Cases for Ontologies in Information Fusion

Mieczyslaw M. Kokar
Electrical & Computer Engineering
Northeastern University
360 Huntington Avenue
Boston, MA 02115
kokar@coe.neu.edu

Christopher J. Matheus
Versatile Information Systems
5 Mountainview Drive
Suite B2
Framingham, MA 01701, USA
cmatheus@vistology.com

Kenneth Baclawski
Computer and Information Science
Northeastern University
360 Huntington Avenue
Boston, MA 02115, USA
ken@baclawski.com

Jerzy A. Letkowski
College of Business Administration
Western New England College
Springfield, MA 01119, USA
jletkows@wnec.edu

Michael Hinman
Air Force Research Laboratory
Rome Research Site
Rome, NY 13441, USA
hinmanm@rl.af.mil

John Salerno
Air Force Research Laboratory
Rome Research Site
Rome, NY 13441, USA
John.Salerno@rl.af.mil

Abstract – Ontologies are becoming increasingly popular due to recent efforts to extend the capabilities of the World Wide Web through the addition of formal semantics. While ontologies have traditionally been used as precise languages to facilitate efficient exchange of information among people, the "Semantic Web" is extending this role to software agents. For this to be possible, ontologies must be formalized in languages processable by computers, such as OWL, the W3C's Web Ontology Language. The purpose of OWL ontologies is to permit software agents to understand web content and to interact intelligently with Web services (which may themselves be software agents). The use of such ontologies, however, need not be constrained to the Web. Recently, ontologies have found their way into higher-level information fusion where they are providing a means for describing and reasoning about sensor data, objects, relations and general domain theories. To the best of our knowledge, there is as of yet no documented effort to capture the main uses of ontologies in information fusion. In this paper we start filling this void by presenting a number of "use cases," i.e., scenarios of the use of ontologies in the context of higher-level information fusion. In this paper we develop use cases in which ontologies are used both for the fusion process itself and for the development of fusion systems. The use cases cover scenarios in which the agent roles are played by people, software or both.

Keywords: Information fusion, ontology, use cases, consistency checking, querying

1 Introduction

An ontology is a formulation of the entities that are relevant to a domain as well as the relationships between these entities. Ontologies originated in philosophy, but they are increasingly being used in many scientific and engineering domains. The traditional use of ontologies is to serve as precise languages to facilitate efficient exchange of information among people. Recently, ontologies have begun to be used for communication among software agents. Indeed, it has long been recognized that independently developed software agents can communicate only if they have a shared understanding of the meaning of the data being exchanged. Ontologies, especially formal ontologies, are an effective means by which two software agents can acquire such a shared understanding.

Fusion systems combine information acquired from multiple sources. When the sources are very similar (e.g., two sensors of exactly the same type in different locations) and the fusion task is highly limited (e.g., tracking objects), then there is no strong argument for introducing ontologies. However, if the sources are dissimilar or the fusion tasks are more diverse, then ontologies can be more effective than ad hoc techniques.

Although many ontology languages have been proposed, the one that is emerging as the standard is the Web Ontology Language (OWL) of the World Wide Web Consortium (W3C). OWL is the basis for the proposed "Semantic Web," which is a layer above the current Web, that supports semantic understanding of Web content. The original purpose of OWL ontologies was to permit software agents to understand Web content and to interact intelligently with Web services (which may themselves be software agents). The use of ontologies, however, is not limited to the Web, and ontologies have recently been introduced into higher-level information fusion where they provide a mechanism for describing and reasoning about sensor data, objects, relations and general domain theories (cf. [1, 2, 3, 4, 5, 6, 7, 8, 9]).

The use of ontologies in general has been addressed by Jasper and Uschold [10] and [11]. Usage scenarios and goals for ontologies in general have been developed in the context of the Ontology Definition Metamodel standardization effort [12]. The World Wide Web Consortium has also developed scenarios for the use of ontologies as part of its Semantic Web effort [13].

In this article, we present some of the main uses

of ontologies in information fusion. The uses are formalized using "use cases" which capture the common features of classes of usage scenarios. In Section 2 we introduce the concept of an ontology and describe the main constructs of OWL. The concept of a use case is discussed in Section 3 which also introduces the notation most commonly used for drawing use case diagrams. We then present the two main uses of ontologies in information. In Section 4 we give the use case for ontology based information fusion processing. This use case is concerned with the use of ontologies while a fusion system is processing data. A specific scenario is presented that shows how ontologies facilitate the ability to evaluate very general queries as a fusion system is running. Ontologies can also be used during the development of a fusion system. This use of ontologies is discussed in Section 5. One specific use of ontologies for software development is to verify the correctness of the software, and a scenario illustrating this is presented in this section. There are many other uses of ontologies in information fusion that are not covered by these two use cases. Some of these are mentioned in Section 6 which concludes the paper.

2 Ontologies

An ontology is an explicit, formal, machine-readable semantic model that defines the classes (or concepts) and their possible inter-relations specific to some specified domain [14]. To construct an ontology one must have an ontology specification language, of which there are several to choose from. Many forces are driving the increasing interest in ontologies, but one that is rapidly gaining momentum is the emergence of web-enabled agents [14]. These agents can reason about and dynamically integrate the appropriate knowledge and services at run-time based on formal ontologies. Ontologies are also the basis for the Semantic Web [15], where they are being used to create machine-readable, semantic-descriptions of Web content that can be shared, combined and reasoned about automatically by theorem provers and intelligent agents.

As part of its Semantic Web effort, the W3C has been engaging in the development of a new XML-based language called the Web Ontology Language (OWL) [16]. OWL is an emerging standard for ontologies and knowledge representations, based on the Resource Description Framework (RDF) [17] and the DARPA Agent Markup Language (DAML), which is the immediate predecessor of OWL. OWL is a declarative, formally defined language that fully supports specialization/generalization hierarchies as well as arbitrary many-to-many relationships. Both model theoretic and axiomatic semantics have been fully defined for the elements in OWL/DAML providing strong theoretical as well as practical benefits in terms of being able to precisely define what can and cannot be achieved with these languages. The field is relatively young, yet several tools have been developed and many more are on the horizon for creating OWL ontologies and processing OWL documents. In our work, we create ontologies as UML diagrams and then programmatically convert them into DAML/OWL representations [18].

Ontologies capture potential objects and potential relations; that is to say, they do not describe what is in the world but rather what can be in the world. Ontologies, however, can be used to annotate or mark-up descriptions of instances of the world in what are called instance annotations.

It should be noted that the job performed by ontology languages cannot be accomplished with purely syntactic languages such as XML Schema. An XML Schema specification can define the structure of objects (i.e., their composition) but it cannot capture the semantic meaning implicit in the relations that might exist between objects. To do this requires knowledge about how the classes of data objects relate to one another. This type of knowledge, called meta-data (i.e., data about data relationships), is what ontologies capture. With appropriate meta-knowledge to explain how the data are to be interpreted, intelligent systems can reason about the data and make inferences that can be proven to be correct. It is this power that makes ontologies critical to our formal approach to situation awareness and fusion.

2.1 Examples of Ontologies

In this paper we use two small pieces of the Situation Awareness Ontology that we have developed in our research. The first one is shown in Figure 1. This is a UML [19] representation of an ontology. It does not include all the details which can be represented in an ontology representation language, like OWL [16]. While the OWL representation is appropriate for computer consumption, the UML representation, on the other hand, is more appropriate for humans. The aspects captured in this UML representation include classes, subclasses and associations. Classes are represented as boxes. Each class defines a "type", i.e., it describes instances that have some characteristics in common. In particular, it means that instances are linked to instances of other classes according to the properties, which in this diagram are shown as arrows. A hollow arrowhead represents a special property called subclass.

So in Figure 1 we see nine classes and eleven properties, one of them being the subclass property (Situation is a subclass of Object). Situation is the central class in this ontology. According to this ontology, a situation must have a Goal. This is specified by the fact that the Situation class is associated with the Goal class and that this property has a multiplicity constraint of 1..*, i.e., there must be at least one goal instance for any situation instance. A situation also includes some objects and relations among them. The fact that Situation is a subclass of Object indicates that every instance of Situation is also an instance of Object. In other words, situations themselves are objects. This

is a very important aspect of representing situations, since it means that situations can have attributes and properties like any other object. Following through the diagram we can see that relations consist of relation tuples. Moreover, we can see that objects have attributes, which are attribute tuples. The tuples consist of attribute type and attribute value. The value, in turn, is determined by the value function, which is expressed with respect to a unit.

Figure 1 also indicates that the value of an attribute or a relation is defined by events. Events are further represented in Figure 2. According to the Event sub-ontology events are part of an event stream. Each event in an event stream is associated with one or more objects. Objects then have attributes, as we already stated above.

3 Use Case Formalism

Most, if not all, software being developed these days is object-oriented. The software development phases, according to software engineering rules, include specification, design, implementation, testing and maintenance. According to the software engineering, software is first modeled, then implemented and tested. The software engineering community came to an agreement to use a common modeling language - the Unified Modeling Language (UML) [19]. As the first step in the modeling of a newly developed software system, most developers model the aspects of the interaction of the system with the external environment using use case diagrams. In this paper we use this modeling formalism of the UML to show the role of ontologies in information fusion. But first we need to introduce this formalism.

Use case modeling uses three main representational icons: the oval to represent a use case, the straw man to represent an actor and lines that connect actors with use cases, use cases with other use cases, or actors with other actors. Lines can end with arrowheads. Two types of arrowheads are used, possibly with some adornments. A use case captures some part of the functionality of the system. For instance, "Collect Sensory Data" could be such a functionality. Actors can be either humans interacting with the system or other systems that are not part of the system being modeled. More precisely, actors don't represent specific individuals but rather roles played by either humans or other systems. A line between an actor and a use case represents the fact that the actor is involved in the use case. When the line ends with an arrowhead, it means that the element on the opposite side of the arrowhead initiates the use case. If the arrow is a hollow arrow, this means that the element at the opposite side of the hollow arrowhead is a special case (a subtype) of the element. For instance, one use case can be a special kind of another use case. Moreover, lines can have adornments enclosed in guillemots that represent stereotypes. In use case diagrams the only stereotypes used are ⟨⟨include⟩⟩ and ⟨⟨extend⟩⟩. The ⟨⟨include⟩⟩ stereotype means that the use case at the arrow tail always includes the function of the use at the arrowhead. The ⟨⟨extend⟩⟩ stereotype represents the fact that the use case at the arrowhead is a functionality that is extended sometimes by the use case at the arrow tail, and thus the functionality shown at the arrow tail is an exception rather than the rule.

4 Use Case: Ontology Based Fusion

Possibly the most common use of ontologies in information fusion is for the fusion processing itself. The use case diagram is shown in Figure 3. The use case for fusion in general is shown at the top of this figure using the oval labeled with "Fusion". This use case represents the various forms of fusion scenario in which multiple sources of information are processed in response to requests by one or more users. The sources of information can be sensors, Level2+ processes, databases and documents. These are all collectively referred to as data authors.

The annotations at the ends of the edges represent the number of instances of the actor that can participate in the use case. The symbol 1..* near the User actor means that there must be at least one user participating in the fusion use case. The first number is the lower limit on the number of participants and the second number is the upper limit on the number of participants. An asterisk means that there is no upper limit.

The Fusion use case is the general case of a fusion process which may or may not use ontologies. Those scenarios that use ontologies are represented by the Ontology Base Fusion use case. This use case has two additional actors. There must be an ontology (or set of ontologies). This is represented by the Fusion Ontology actor. There can also be other ontology based systems (which may or may not be fusion systems). When another ontology based system is participating, it acts like a Data Author, except that ontologies may be used to achieve interoperability. By contrast, for most Data Authors, it is entirely the responsibility of the fusion system to understand the formats and protocols that are produced by the Data Authors. This is a significant advantage of ontology based systems.

Another important feature of ontology based fusion is represented by the remaining three use cases in the diagram. These three use cases are concerned with querying the fusion system as it is running. The basic use case is called Querying, and it includes two others: the sending of the query and the process of inferring the answer. Inference is a logical reasoning process which requires a theorem prover or rule based system. Querying can be done by either a user or another ontology based system.

One could argue that the query capability could be implemented in a fusion system without the use of an ontology. While it is true, such a query capability would be limited to the procedures that were hard-coded into the fusion process. For instance, one might list a number of queries and then write procedures for

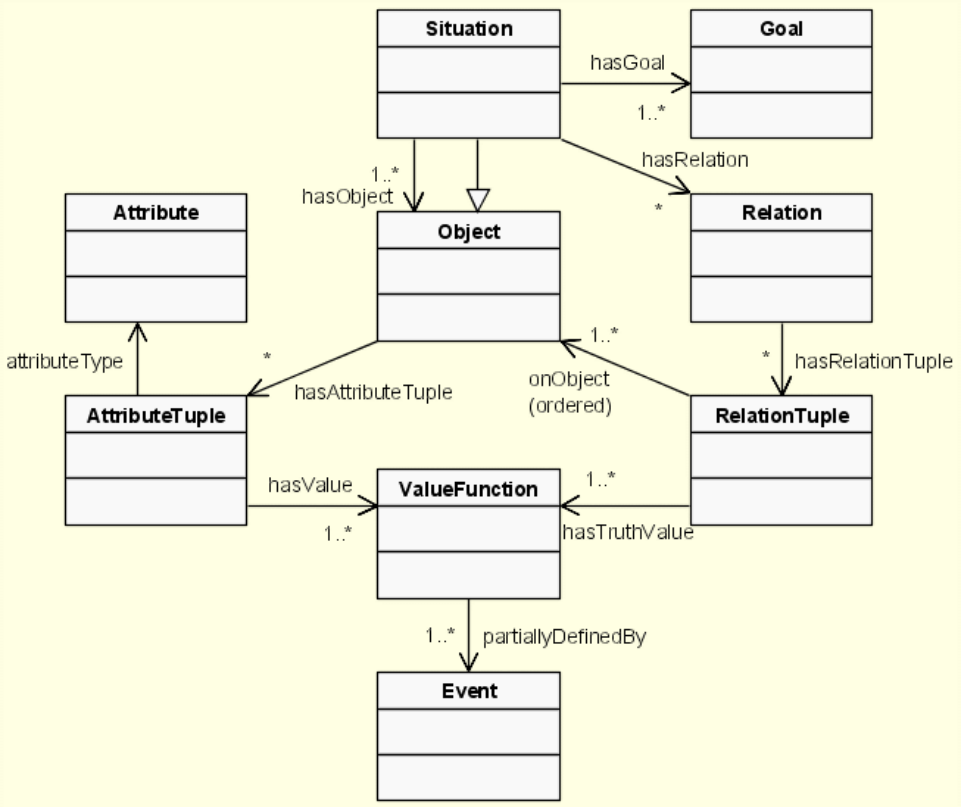

Fig. 1: SAW Core Lite Ontology

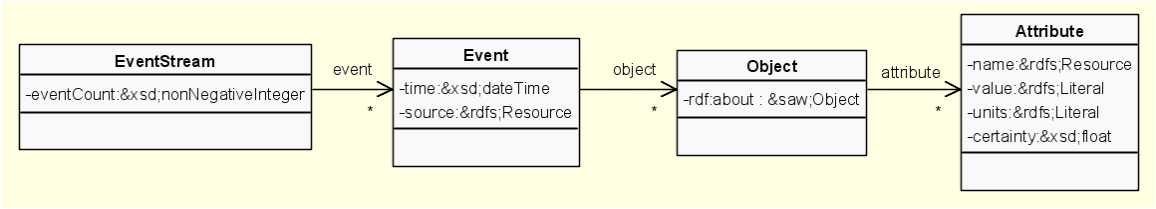

Fig. 2: SAW Event Ontology

answering each of the queries. The ontology based approach is different. An ontology based system is assumed to have a generic reasoner that can answer any query that can be formulated in the language defined by the ontology. In general, the number of possible queries that can be formulated in the language defined by an ontology is infinite. Obviously, this kind of a capability cannot be implemented using a procedural approach.

Consider the situation awareness ontology shown in Figure 1. It is easy to imagine some of the queries that might be asked. For example, one might be interested in determining the objects which are located in a particular region, or one might want to know the physical units of the attributes possessed by the objects in a particular aggregate (grouping of objects). Still other examples would involve various forms of inference. One might be interested in the location of an aggregate of objects. While objects have locations, aggregations of them do not directly have locations. This must be inferred from the relationship between the objects and the aggregate (sometimes called the "part-of" relationship). Any of these queries (and many others) could be expressed procedurally. However, one would have to design and develop a new procedure for each one. Ontology based systems have a significant advantage over traditional systems in this regard.

The use of the word "query" is somewhat misleading because it suggests that one is only concerned with retrieving information. The intention is that one can not only retrieve information but also modify information. This use of the word is consistent with how it is used in databases. The ability to modify a running system is a significant new feature of ontology based systems. Of course, all systems include some parameters that can be specified while the system is running. Ontology based systems can be modified in any manner that is specified by the ontology. While this can be done procedurally, each such modification must be developed separately. Once again, the ontology based systems have the advantage.

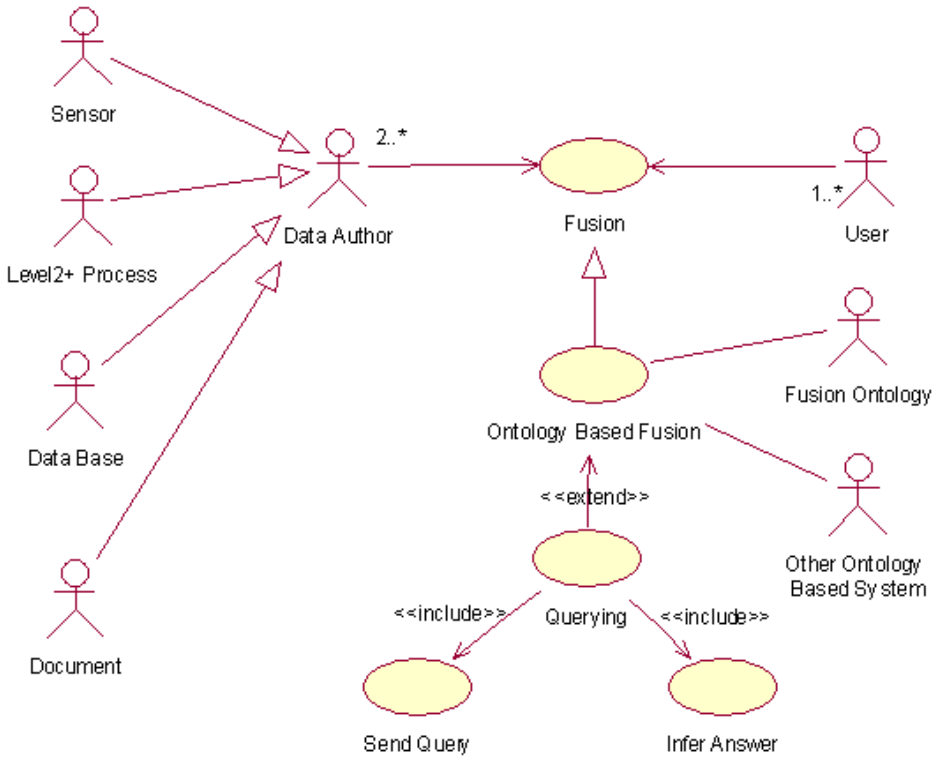

Fig. 3: Fusion Use Case

5 Use Case: Development of Ontology Based Fusion Systems

Information fusion systems are complex systems that require a substantial development effort. Ontologies can help assure that the fusion system correctly satisfies its specifications. The use case diagram for fusion system development is shown in Figure 4. The three main use cases in this diagram are System Development, Fusion System Development and Ontology Based Fusion System Development. The first use case captures some of the characteristics of general case software system development. The actors in this use case are Application Developer and Domain Expert. The Application Developer represents the engineers who are developing the hardware and software of the fusion system. The Domain Expert represents the individuals who provide the domain-specific expertise for the system.

The diagram shows that the Fusion System Development use case is a special case of System Development. Developing a fusion system requires fusion specific features to be developed, such as data association and fusion rules. These features along with their incorporation into the overall system are developed by a Fusion Expert.

Ontology based fusion system development is a special case of general fusion system development which adds a number of optional features. This use case has three additional actors: Fusion Ontology, Ontology Reasoning Tool, System Database. Unlike the actors discussed so far, none of these new actors would ever be a person. The Fusion Ontology actor is the ontology (or set of ontologies) that form the basis for information fusion in the domains of interest. It is represented as an actor because it is normally developed separately, and therefore is external to the system being developed. In a sense, it plays a similar role to a database, except that it is used to store meta-information about the system being developed. In particular, it keeps information about the types (classes) of inputs to the fusion system, the types of outputs from the system, and possibly the types of information items that are passed among the components of the system (applicable when the system consists of a number of components). Additionally, this ontology has information about the types of objects that the system will represent and the types of relationships (properties) that will be implemented and maintained by the system. As development proceeds, instances of these types and properties are introduced and can be stored in a repository represented in the diagram by the System Database actor.

The Ontology Reasoning Tool actor is responsible for formal reasoning tasks such as inference, querying, consistency checking and code generation. The first of these, inference, is always present in any use of ontologies. The others are optional features, and so they are linked to the Ontology Based Fusion System Development use case by means of the ⟨⟨extend⟩⟩ connector.

Queries to the System Database are more powerful than ordinary database queries because they make use of inferencing carried out by the Ontology Reasoning Tool. The query can involve any of the types and properties in the ontology. One can, for example, ask

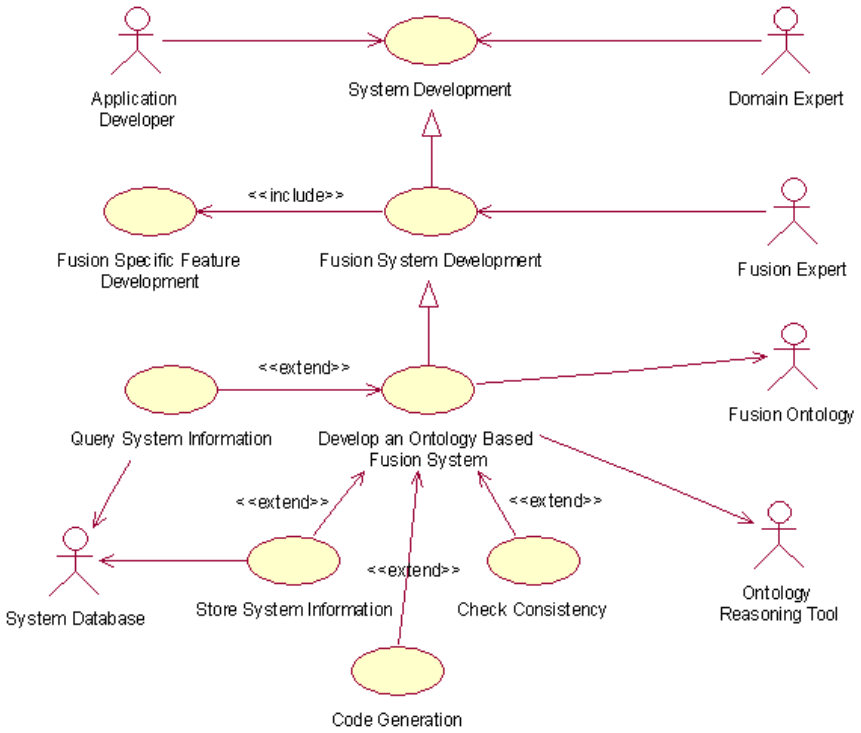

Fig. 4: Development Use Case

whether the inputs to the system are consistent with the specifications. One can also determine whether software (source code) is consistent with the ontology. It is even possible to generate some or all of the code for the system from the information and meta-information provided by the Fusion Ontology and System Database actors.

Here is an example of software verification using ontologies. A fusion system based on the ontology in Figure 2 had a class Event that looked like this:

```
public class Event {
    private int eventCount = 0;
    ...
    }
```

The event count was incremented by 1 each time that an event was constructed. The Event class compiled correctly and even passed various tests. However, it is incorrect because the eventCount should be an attribute of the EventStream class, not the Event class. The intention was that the eventCount was a mechanism for assigning event identifiers for the events in a stream of events. The initial implementation was assigning the same identifier to every event in an event stream.

6 Conclusion

Ontologies are an increasingly important mechanism for integration of disparate software systems, and they are also beginning to be used in information fusion systems. In this paper we showed some of the main current and potential uses of ontologies in information fusion. In both cases there are significant advantages for using introducing ontologies that are either difficult to achieve or even impossible without ontologies. The advantages include support for interoperability, flexible querying, run-time modifiability, validation against specifications and consistency checking.

There are still many other possibilities for the use of ontologies in information fusion that are not covered by the two use cases that were presented. Systems that do not share a single ontology might still be able to interoperate by mapping or merging ontologies. It may be possible to construct systems that not only use ontologies but also modify them or even learn them dynamically. We are currently working toward the development of tools that can achieve these goals.

Acknowledgments

This work was partially funded by the Air Force Research Laboratory, Rome, NY under contract numbers F30602-02-C-0039 and F30601-03-C-0076.

References

[1] A-C. Boury-Brisset. Ontology-based approach for information fusion. In Proceedings of the Sixth International Conference on Information Fusion, pages 522–529, 2003.

[2] T. Horney, E. Jungert, and M. Folkesson. An ontology controlled data fusion process for a query language. In Proceedings of the Sixth International Conference on Information Fusion, pages 530–537, 2003.

[3] K. Sycara, M. Paolucci, and M. Lewis. Information discovery and fusion: Semantics on the battlefield. In Proceedings of the Sixth International Conference on Information Fusion, pages 538–544, 2003.

[4] C. J. Matheus, M. M. Kokar, and K. Baclawski. A Core Ontology for Situation Awareness. In Proceedings of the Sixth International Conference on Information Fusion, pages 545–552, 2003.

[5] E. P. Blasch and S. Plano. Ontological issues in higher levels of information fusion: User refinement of the fusion process. In Proceedings of the Sixth International Conference on Information Fusion, pages 634–641, 2003.

[6] A. I. Chao, B. C. Krikeles, A. E. Lusignan, and E. Starczewski. An extensible, ontology-based, distributed information system architecture. In Proceedings of the Sixth International Conference on Information Fusion, pages 642–649, 2003.

[7] C. J. Matheus, K. P. Baclawski, and M. M. Kokar. Derivation of ontological relations using formal methods in a situation awareness scenario. In Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2003, pages 298–309. SPIE, 2003.

[8] M. M. Kokar and J. Wang. Using ontologies for recognition: An example. In Proceedings of the Fifth International Conference on Information Fusion, pages 1324 – 1343, 2002.

[9] M. M. Kokar and J. Wang. An example of using ontologies and symbolic information in automatic target recognition. In Sensor Fusion: Architectures, Algorithms, and Applications VI, pages 40–50. SPIE, 2002.

[10] M. Ushold, R. Jasper, and P. Clark. Three approaches for knowledge sharing: A comparative study. In KAW-99, 1999.

[11] R. Jasper and M. Ushold. A framework for understanding and classifying ontology applications. In Proceedings of the IJCAI-99 Ontology Workshop, 1999.

[12] L. Hart, P. Emery, B. Colomb, K. Raymond, D. Chang, Y. Ye, E. Kendall, and M. Dutra. Usage scenarios and goals motivating development of an ontology definition metamodel, 2004. http://www.omg.org/cgi-bin/apps/doc?ontology/04-01-01.pdf.

[13] J. (Ed.) Heflin. Owl web ontology language use cases and requirements, 2003. http://www.w3.org/TR/2003/PR-webont-req-20031215/.

[14] D. McGuinness. Ontologies and online commerce. IEEE Intelligent Systems, 16(1):8–14, 2001.

[15] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, May 2001.

[16] OWL-ref. OWL Web Ontology Language Reference, 2003. http://www.w3.org/TR/owl-ref.

[17] RDF. Resource description framework (RDF) model and syntax specification, Feburary 1999. http://www.w3.org/TR/REC-rdf-syntax.

[18] P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar, and J. Smith. UML for ontology development. The Knowledge Engineering Review, 17, 1:61–64, 2002.

[19] OMG. Unified Modeling Language Specification, Version 1.3. Technical report, Object Management Group, 1999.