Lessons Learned From Developing SAWA: A Situation Awareness Assistant

Christopher J. Matheus
Versatile Information Systems
Framingham, MA USA
cmatheus@vistology.com

Mieczyslaw M. Kokar
Northeastern University
Boston, MA USA
kokar@coe.neu.edu

Kenneth Baclawski
Northeastern University
Boston, MA USA
ken@baclawski.com

Jerzy J. Letkowski
Western New England College
Springfield, MA USA
jletkows@wnec.edu

Catherine Call
Versatile Information Systems,
Framingham, MA USA
cdcall@speakeasy.net

Michael Hinman
Air Force Research Laboratory
Rome, NY USA
michael.hinman@rl.af.mil

John Salerno
Air Force Research Laboratory
Rome, NY USA
john.salerno@rl.af.mil

Douglas Boulware
Air Force Research Laboratory
Rome, NY USA
douglas.boulware@rl.af.mil

Abstract – *SAWA is a Situation Awareness Assistant being developed by Versatile Information Systems, Inc. During the process of its development several lessons were learned about advantages and limitations of certain approaches, techniques and technologies as they are applied to situation awareness. This paper begins with an overview of SAWA and then focuses on some of the more significant lessons learned. These include the pros and cons of leveraging Semantic Web technologies, the handling of time-varying attributes and the processing of uncertainty.*

Keywords: situation awareness, level-two fusion, ontologies, formal reasoning, OWL, SWRL.

1 Introduction

The essence of situation awareness lies in the monitoring of various entities, physical and abstract, as well as various relations among the entities. Since the properties of relations, unlike the properties of objects, are not directly measurable, one needs to have some background knowledge (such as ontologies and rules) to specify how to derive the existence and meaning of particular relations. For instance, in the domain of supply logistics, relations like "suppliable" or "projected undersupply within 2 days" need to be systematically specified. Typical relations in a military context would include relations such as "unit aggregation" and "composition of the force". The number of potentially relevant relation types is practically unlimited. This presents a great challenge to developers of general-purpose situation awareness systems since it essentially means that such systems must have the potential to track any possible relation. In other words, the relation determination algorithms must be generic, rather than handcrafted for each special kind of relation. Furthermore, in order to derive a specific relation one often needs to access a number of data sources and then combine (fuse) their inputs. One way to address these challenges is to use generic reasoning tools, such as those based on the principles employed by automated theorem provers. However, to take advantage of this approach all information must be available in a formally defined knowledge base.

At Versatile Information Systems, Inc., we are developing a collection of flexible ontology-based information fusion tools needed for identifying and tracking user-defined relations. These tools collectively make up our Situation Awareness Assistant (SAWA). The purpose of SAWA is to permit the offline development of problem specific domain knowledge and then apply it at runtime to the fusion and analysis of level-one data. Domain knowledge is captured in SAWA using formal ontologies, portions of which are used to represent the incoming stream of level-one event data. The user controls the system situation monitoring requirements by specifying "standing relations", i.e., high-level relations or queries that the system is to monitor. SAWA provides a flexible query and monitoring language that can be used to request information about the current situation, predicted situations, and request notifications of current or potential future emergency conditions. In this paper we describe the structure and capabilities of SAWA and show its use on examples from the supply logistics domain. In particular, we show how to develop an appropriate ontology and associated rules, how SAWA collects and processes incoming events and how it communicates with the user. In concluding we summarize some of the more significant lessons learned from creating SAWA

2 General Approach

We view situation awareness as a fusion problem involving the identification and monitoring of higher-

order relations among level-one objects. As mentioned in the introduction, practical solutions to this problem require user-defined constraints, which we usually identified with a corpus of knowledge specific to a domain of interest, otherwise known as domain knowledge. The use of domain knowledge requires a form of representation and a means for processing or reasoning about the knowledge representations. Rather than developing ad hoc representations we advocate the leveraging of existing standards. We also believe strongly in the value of formal representations that can be used in conjunction with generic yet formal reasoning systems. Our approach to domain knowledge representation, which we will describe shortly, is thus premised on use of standards-based formal representations.

Even with appropriate domain knowledge the number of possible relations definable within the domain knowledge constraints can remain intractable. To further constrain a situation it is necessary to know something about the user's specific goals. By knowing more specifically what the user is looking for, automated systems can focus attention on just those events and candidate relations that are relevant. Our process for relevance reasoning has been reported elsewhere [1] and will not be explained in detail in this paper. We will summarize, however, by saying that relevance reasoning takes a standing relation (i.e. a goal) from the user, identifies the portion of the domain knowledge that is relevant to the standing relation, finds the attributes in the domain knowledge that must be grounded in input events and uses these attributes to identify what types of objects and which of their attributes need to be monitored in the event stream. With this mechanism, the large number of objects and attributes in a situation can be pared down to a more manageable stream of data in which only a comparatively small number of relevant relations must be monitored.

2.1 Ontology Representation in OWL

In our current efforts we have been exploring the use of recent developments for the Semantic Web [2]. In particular we have chosen to use the Web Ontology Language OWL [3] for defining ontologies that serve as the basis for data and knowledge representation within our situation awareness systems. The advantages of using OWL includes the fact that it is defined by a formal set of semantics and that there are a growing number of automated systems to formally process OWL documents, including editors, consistency checkers and reasoning engines [4].

OWL was designed to capture the classes, properties and restrictions pertinent to a specific domain. As such, OWL can capture basic class hierarchies, properties among classes and data and simple constraints on those properties and classes. OWL, however, cannot capture all types of knowledge relevant to a given domain. In particular, it does not provide a way to represent arbitrarily complex implications, in which knowledge of the existence of a collection of facts (X1, X2,…Xn) implies the truth of some other information (i.e., X1 v X2v…Xn $\Rightarrow$Y). For example, there is no way in OWL to define the relationship of "uncle(X,Y)", which requires knowing that X is male, X has a sibling Z, and Z has a child Y. The joining of collections of interrelated facts into implication rules as illustrated in this example is very common when defining relationships important to domains involving situation awareness. We therefore need the ability to define portions of our domain knowledge using rules, and for this purpose we have selected the Semantic Web Rule Language, SWRL [5].

2.2 Rule Representation in SWRL

SWRL is built on top of OWL and, like OWL, has a formally defined semantics, making it a natural choice for use in our situation awareness applications. SWRL does, however, have some shortcomings that make it less than ideal. Because it was officially introduced as a draft recommendation in just the spring of 2004, it is relatively new and is still evolving; this means there are few tools and applications for use with SWRL and it remains a moving target which may undergo radical changes that will introduce inconsistencies for early adopters. Furthermore, SWRL predicates are limited to binary arity. While it is possible to represent concepts dependent on higher-arity relations using SWRL, the process of doing so significantly complicates the resulting rules, making them difficult to read and maintain. Still, the advantages of SWRL justify the exploration of its use for situation awareness, which can be seen as one of the objectives of our current work. Our results do not as of yet provide sufficient evidence on which to fully judge SWRL's future potential in this area, although we remain optimistic.

2.3 SAW Core Ontology

We are interested in building systems for situation awareness that are generic in nature. That is to say that the systems should be applicable to a wide variety of problem domains simply through the redefinition of the domain knowledge that they use. For this approach to work, some core concepts need to be established that will be used as the basis for the development of specific domain knowledge ontologies and rule sets. For this reason we have developed a SAW Core Ontology that serves as the representational foundation of all domain knowledge that is built on top of it. We have reported on this core ontology in earlier papers [6] and will not describe it in detail here. The key concepts capture by the ontology are the use of objects that have attributes with specific values defined by external events that occur over time; in addition, relations combine pairs of objects with truth values defined over time by the firing of rules that define the relations.

3 SAWA High-Level Architecture

The SAWA High-Level Architecture has two aspects as shown in Fig. 1: a set of offline tools for Knowledge Management and a Runtime System of components for applying the domain knowledge to the monitoring of evolving situations. The knowledge management tools include an ontology editor, an ontology consistency checker and a rule editor. The runtime system consists of

a Situation Management Component (SMC), an Event Management Component (EMC), a Relation Monitor Agent (RMA), a Triples DataBase (TDB) and a Graphical User Interface (GUI). The user interacts with the system through the GUI by issuing standing relations (goals) and queries. Events from the outside world come into the runtime system and are processed for redistribution to other components by the Event Management Component (EMC).

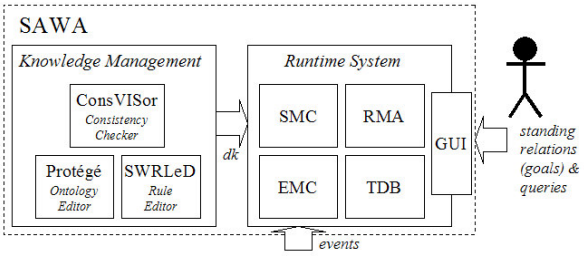

Fig. 1. SAWA High-Level Architecture.

4 SAWA Knowledge Management

Knowledge Management in SAWA is handled by a loosely coupled suite of tools for developing and maintaining OWL ontologies and SWRL rule sets.

4.1 Ontology Editor

The OWL language is based in RDF [7], which has an XML-based representation. As such, any text or XML editor could be used to develop OWL ontologies. The manual coding of OWL is, however, tedious and prone to error, making specialized editors highly desirable. There are a number of editors available for OWL [8] but the most widely used is Protégé [9]. Protégé is a general-purpose ontology management system developed long before OWL but for which OWL plug-ins have been developed. Using Protégé with the basic OWL plug-in permits the use of Protégé's frame-based editor to construct OWL classes, properties and restrictions among them as well as to develop annotations for OWL ontologies. This approach is adequate but not as convenient as a graphical editor that allows the visual display and manipulation of the relations between objects and properties. Fortunately there is a plug-in for Protégé called ezOWL that provides a graphical editor on top of the basic OWL-plugin. All of the ontologies depicted in this paper are screenshots taken from ezOWL. ezOWL has its limitations (for example it does not cleanly display more than two properties between two classes) and does not always produce correct OWL code, but it is currently the best available visual editor for OWL and does a satisfactory job, provided the resulting code is checked for consistency.

4.2 Consistency Checker

Developing an accurate and consistent ontology is not easy, particularly as the complexity of the domain increases. For all but the most trivial problems it is imperative that newly constructed ontologies be automatically validated for logical consistency; this is also invaluable when combining multiple ontologies that may individually be consistent but are collectively incompatible. It has been the authors' experience that seldom is the first design of an ontology complete and consistent, and the use of consistency checking tools has saved tremendous amounts of development time. SAWA includes ConsVISor [10], an OWL/RDF consistency checker, in its suite of knowledge management tools. ConsVISor is both a standalone Java application and a free Web Service provided by Versatile Information Systems, Inc. at http://www.vistology.com/consvisor.

ConsVISor's purpose is to analyze OWL and RDF documents looking for symptoms of semantic inconsistencies. Not only does it detect outright semantic violations, it also identifies situations where logical implications have not been fully specified in a document. For example, if an ontology places a minimum cardinality constraint on a property for a specific class and an instance of that class is created without having the minimum number of property values. Emphasis is placed on providing highly informative feedback about detected symptoms so as to aid the correction of underlying errors by the human user. ConsVISor's output however is based on an OWL-based Symptom Ontology [11] and as such can produce symptom reports in OWL that can be automatically processed by other OWL-cognizant programs.

4.3 Rule Editor

SWRL rules in their XML representation are syntactically and (frequently) semantically difficult to read and write. It was therefore decided that SAWA needed an easy to use editor to assist in the construction and maintenance of SWRL rules. With SWRL being so new, there were no SWRL editors available and so we decided to implement one, which we are calling RuleVISor. A screenshot of RuleVISor being used on a rule set for the Supply Logistics scenario described in Section 6 is shown in Fig. 2. The rules are displayed along the top left hand side of the editor in a directory style layout for easy selection and high-level scanning. The rule that is currently being edited appears in two forms in the right-hand section of the editor. At the top of this section is the display of the contents of the rule head and body in either an easy to read atomic form, which is shown in the screenshot, or as raw SWRL code (not shown). Below this display is the section where editing of the rule takes place, including the optional naming of each rule. This section is split into a portion at the top for editing the head followed by a portion for editing the body. Within either of these the user has the option of adding or deleting binary atoms, unary atoms, instances, data value ranges and built-in functions simply by clicking on the appropriate icons. Each clause in a rule head or body appears in a three-row region that provides the name of the atom, the terms it operates over and possibly other constraints such as term type restrictions. The values of the terms can either be typed in by the user or dragged from other areas of the editor. The primary source for dragged items is the Ontology Tree that appears in the lower left hand corner.

The Ontology Tree displays the contents of the ontologies in whose context a rule set is to be built. Of most interest here are the Classes and Properties of the ontology, which are used to populate the term slots of atoms used in the rule heads and bodies. Class and Property names may be dragged to any text entry box in the editor but they will only be accepted by the box if the value being dragged matches the type that the box expects. This form of primitive type checking represents the beginning of a much more sophisticated policy for consistency checking based on ConsVISor that is planned for a future version of RuleVISor.

RuleVISor is currently in beta testing (contact the authors if you are interested in testing RuleVISor) but has been used extensively for rule development purposes by the authors and has proven to be a great time saver over the manual editing of SWRL rules. Perhaps the biggest advantage afforded by RuleVISor is the ability to deal with rule definition at a conceptual level that abstracts out the syntactic complexities of the XML-based representation of SWRL. RuleVISor also assists with the generation of Jess rules as it has a built in translator that understands some of the more subtle complexities of converting from SWRL to Jess.

5 SAWA Runtime System

The SAWA Runtime System, also called the SAWA Engine, is depicted in Fig. 3 along with the communication channels between its sub-components. SAWA is implemented in Java, includes Jess as the basis for its reasoning functions and uses our proprietary RDF/OWL/XSD parser. The SAWA Engine consists of the following sub-components: the Situation Management Component (SMC) which is the system's central controller, the Event Management Component (EMC) which processes all incoming events, the Relation Monitoring Agent (RMA) which monitors relevant events for the status of relations occurring in the evolving situation, the Triples DataBase (TDB) which maintains a historical record of all situation events and permits the processing of queries, and the Graphical User Interface (GUI) which handles all user interaction with the system. The function of each of these components is described further in the subsections that follow.

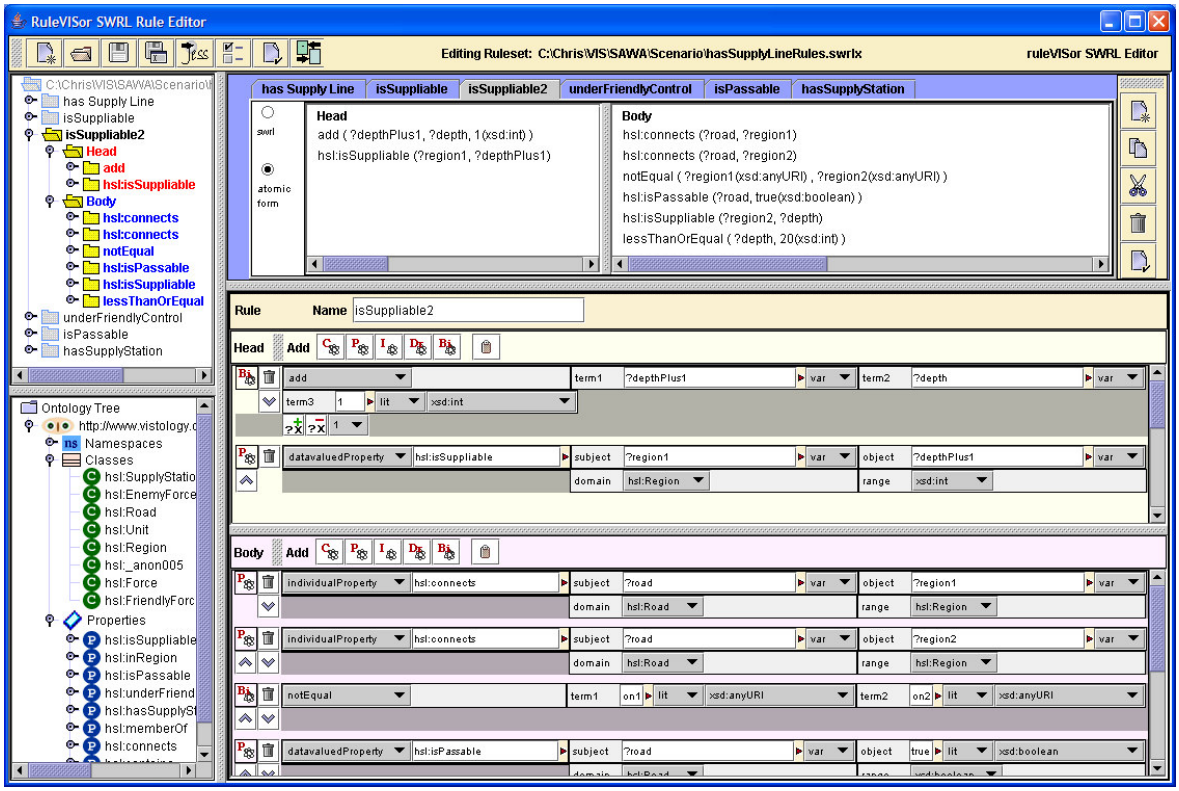

Fig. 2: RuleVISor.

.

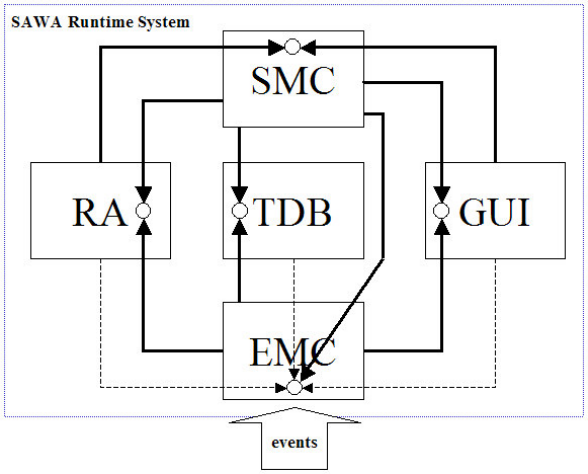

Fig. 3: SAWA Runtime System.

5.1 Situation Management Component

The Situation Management Component (SMC) is the central controller for SAWA. It interacts with the GUI to provide options to the user and to accept the user's commands to start, stop and query situations. In addition, it serves as the communication channel between the GUI and the TDB and RMA. The SMC initializes the monitoring of situations by instructing the EMC to start listening to specific event streams and informs the RMA, TDB and GUI how to connect to the EMC to receive their appropriate streams of processed events. The SMC is also responsible for performing relevance reasoning and for passing the appropriate set of relevant rules to the RMA and the set of relevant objects and attributes to the EMC.

5.2 Event Management Component

The Event Management Component (EMC) receives streams of raw event data and converts them into appropriate streams of events for the GUI, RMA and TDB. Each of these components receives a specific type of event stream: the RMA only receives relevant events encoded as Jess-formatted triples; the TDB receives all events in the form of OWL triples; the GUI receives relevant events in the form of object-attribute instances. The raw input streams are expected to be annotated using an event ontology with references to objects defined in the core ontology and the appropriate domain ontology. The event ontology currently being used in SAWA is shown in Fig. 4. This event ontology is known only to the EMC which converts all event information into appropriate structures for the other components; the isolation of the other components from the event ontology was done so as to permit the use of other event ontologies dependent upon the source of the event streams (which at this time is a simulator of fused level-one object data).

5.3 Relation Monitoring Agent

The Relation Monitoring Agent (RMA) performs the task of monitoring the stream of relevant events and detecting the truth-value of relevant relations that might exist between objects occurring in the evolving situation. The RMA performs this task using the relevant rules defined by the domain knowledge in conjunction with the standing relation. These relevant rules are processed in the forward-chaining Rete network of a Jess inference engine. As events come in, they are processed through the Rete network and as a result may end up firing one or more rules. The firing of a rule results in the instantiation of a relation that is then reported to the GUI via the SMC. At the moment all rule firings result in relations that have an associated certainty rating of 1.0 (i.e., 100%). We are working on a new implementation of the reasoning engine that will incorporate uncertainty reasoning and will thus afford the detection of relations having incomplete certainties.

5.4 Triples Database

In RDF and OWL all information is represented in the form of triples. Each triple represents a predicate that relates a subject to an object. For example, to state that S2 is a SupplyStation requires a triple of the form: S2 rdf:type SupplyStation. More complex knowledge structure can be represented using collections of interrelated triples (see [12]). The triples representing the domain knowledge, user input and the incoming events all need to be maintained in a way that they can be readily processed. In SAWA this is accomplished through the Triples DataBase (TDB).

The TDB's primary purpose is to maintain an accurate history of all events so that they can be queried by the user at any time. It is currently developed on top of Jess and makes use of Jess' built-in query capabilities to implement an engine for OWL-QL: OWL Query language [13]. The TDB also supports "what-if" queries in which a set of hypothetical facts are asserted, a query is run to produce what-if results, and the hypothetical facts are retracted along with all facts deduced from them. The TDB accomplishes this what-if capability using the "logical"

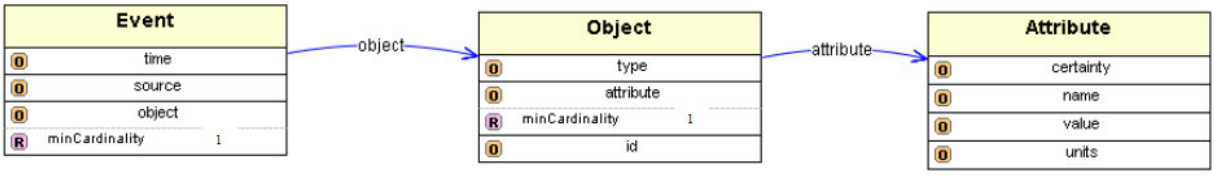

Fig. 4. Event Ontology.

retraction feature of Jess. While both the general query mechanism and the what-if query mechanism work as designed, they are quite inefficient and not particularly suited for new real-time operations. Consequently we are in the process of developing our own inferencing and query engine optimized for the processing of triples.

5.5 Graphical User Interface

The Graphical User Interface permits the user to define standing relations, execute queries and monitor the current state of events, objects, attributes and relations. Its use on a Supply Logistics scenario (described in the next section) is illustrated in Fig. 5.

6 A Supply Logistics Scenario

SAWA is currently being applied to the domain of supply logistics. A simple scenario based on the concept of supply lines has been constructed for the purposes of demonstrating the basic system functions. The goal or "standing relation" for this scenario is to constantly monitor the relation "hasSupplyLine" for all friendly units. A supply line is defined as the existence of a continuous path of roads under friendly control connecting a unit (e.g., B5, B6, etc.) to a supply station (e.g., S1). The specific layout for this scenario can be seen in the map display in the GUI screenshot in Fig. 5. Roads connect pairs of regions (their centroids indicated by solid dots). There are six friendly blue units (i.e., B5, B6, B7, B9 and S1), including one supply station (S1), and one unfriendly red unit (R1).

The screenshot in Fig. 6 shows the simple supply logistics ontology that goes along with this scenario. Note that all of the classes in this ontology are implicitly sub classes of the Object class in the SAW Core Ontology described in Section 2.3 - this is necessary for the domain specific ontology to work with the otherwise generic mechanisms of the SAWA Engine. Note also that this ontology is a gross simplification of what would be expected for a more complete ontology necessary to support more practical supply logistics scenarios (which the authors are currently working on). This ontology was created using ezOWL, which produced the screenshot shown in Fig. 6 as well as the OWL code used in the running of the scenario.

The rule set developed for this scenario is partially shown in the screenshot of RuleVISor in Fig. 2. These rules define that a unit hasSupplyLine if the unit is in a region that isSuppliable. A region isSuppliable if it hasSupplyStation and is underFriendlyControl or if it is connected to another region by a Passable road and that other region isSuppliable. A region is underFriendlyControl if it contains a friendly unit. A region hasSupplyStation if the region contains an object and that object is a supply station (note that this rather obvious sounding rule is an implication that cannot be readily captured in OWL alone).

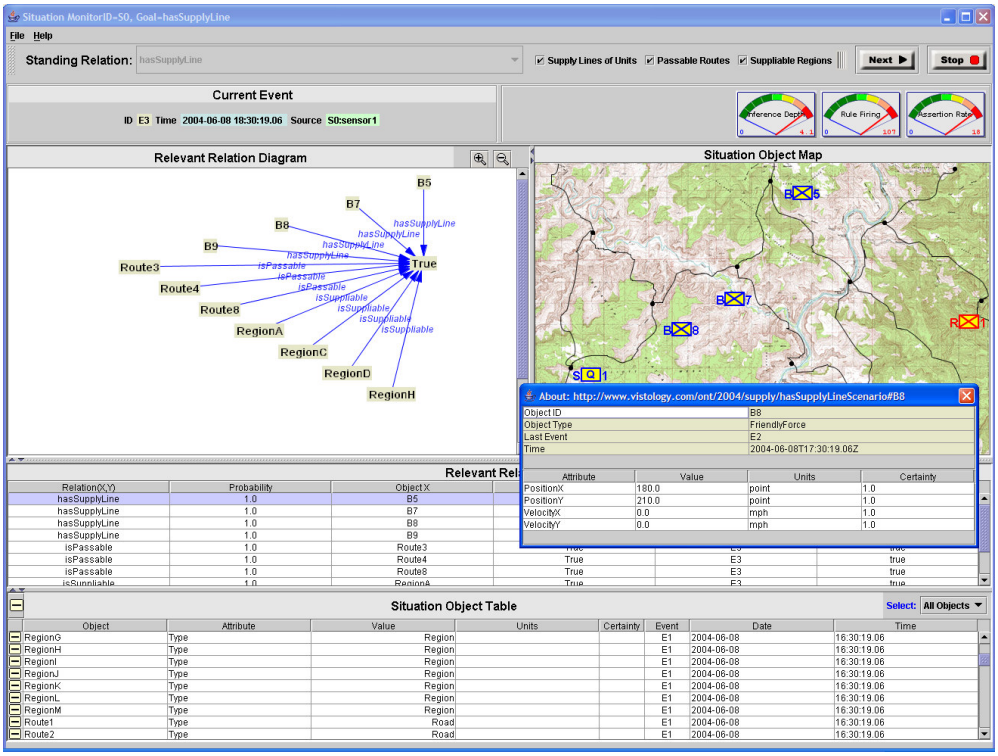

Fig. 5. The SAWA GUI.

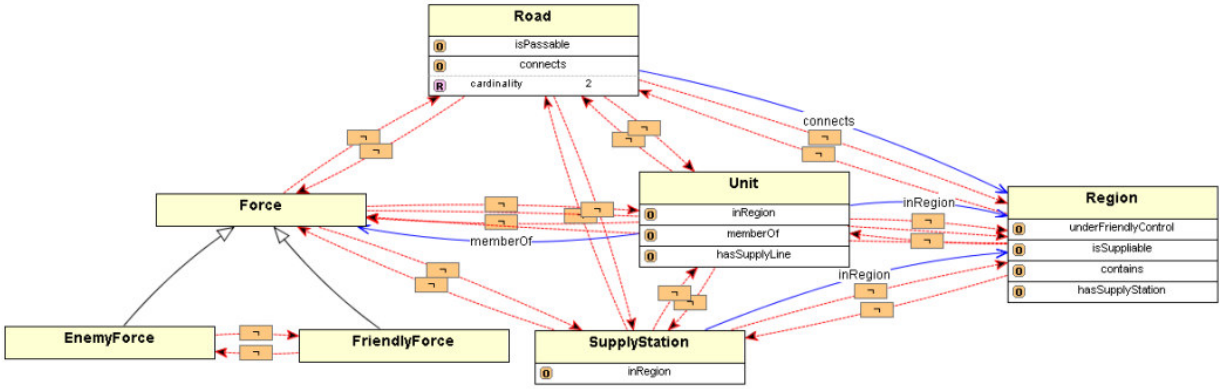

Fig. 6. Simple Supply Logistics Ontology.

To simulate the running of the scenario several snapshots were developed as OWL annotations to define the state of the world at sequential time slices. In each time slice one of the units was moved in such a manner as to create changes in the set of relations that would hold true. These snapshots were then presented to a running SAWA application in which the user specified the standing relation to be hasSupplyLine applied to all friendly units. The system correctly detected the standing relations that held true at each time slice and reported these back to the GUI for display to the user; the GUI screenshot in Fig. 5 shows the display after a couple of time steps.

7 Conclusions: Lessons Learned

In the process of developing SAWA we encountered a number of challenges. Paramount among these were 1) the practical application of emerging Semantic Web technologies, 2) the need to represent and reason about time and 3) the difficulties of dealing with uncertainty as it propagates through the reasoning processes.

7.1 Semantic Web Technologies

We have found ontologies to be invaluable for representing and reasoning about knowledge pertinent to a situation's domain. OWL, along with the formal reasoning techniques that have been developed around it, represents the state of the art in ontologies. That being said, OWL is not without its limitations when viewed from a system implementer's perspective. Foremost among these is the restriction to binary properties, which comes from its foundation on RDF triples. While it is possible to represent higher-arity relationships through the use of reified statements [14] this process is cumbersome and necessitates greater care in ontology design and implementation. Another issue with the language is the lack of the ability to perform joins across multiple entities. This problem means you are unable to constrain a property's object values based on the properties those objects possess. As illustrated earlier, there is no way to fully represent the property of uncle(X,Y) in which you would want to constrain X to being the brother of some other person who is the parent of Y. To obtain this sort of expressive power requires the use of a more powerful language containing the ability to perform implications.

SWRL provides this capability for representing implication through the use of rules and is explicitly designed as an extension to OWL. SWRL is a more recent development and therefore is more prone to updates and changes; furthermore, there are scant few systems currently supporting the language. This is a real problem because the language is rather verbose in its XML syntax, making it difficult to read and write by hand. SWRL also retains the restriction to binary properties, which can make relatively simple rules much more complicated. For example, we considered a rule that could be used to tell whether a given unit had on hand a specific type of supply at some point in time. The head of such a rule might look as follows: onHand(Unit, Supply, Time). Because this is a ternary relationship it was necessary to create a reified statement with multiple properties each appearing as a separate atom in the rule head; as a result the intended meaning of the rule was difficult to identify without significant analysis. Although developing rules of this sort is possible, and is amenable to automation, it currently makes the creation and maintenance of SWRL rules very challenging, relegating them to consideration in advanced research efforts. We envision future remedies for many of the challenges that currently exist when applying OWL and SWRL to practical situation awareness applications, and we are ourselves working on some of these. We remain confident that in time Semantic Web technologies will become integral parts of future situation awareness systems.

7.2 Reasoning about Time

One of the most distinguishing characteristics of real-world situations is the aspect of time. It is also arguably the biggest challenge. What we would like to be able to do is have perfect knowledge of the state of all relevant objects at all times. This is clearly impossible in all but the most trivial situations. We are thus left having to work with streams of events in which partial snapshots of the world are received, likely in the form of object-attribute values, perhaps with some indication of the degree of certainty to ascribe to each value. From this collection of information received at various times we must piece

together a picture of what is actually happening at the current moment, and what might happen in the future.

The real problem is that while some things stay relatively static the more interesting ones are dynamic and continue to change after they have been sensed. SAWA was designed to work on the Level 2 problem of identifying higher order relations (e.g., aggregation, group-wise activity, attacking) but in order to do this it became necessary to model the behavior of dynamic objects. What we really needed was a Level 1 system able to handle object modeling which SAWA could query at any moment for the current state of an object. Unfortunately none was available and we were forced to address this problem ourselves. It was much too inefficient to attempt to model objects using the rule-based engine SAWA uses for higher-level reasoning. Our work around solution to this problem was to implement a "currentValue()" built in function in our rule language which permits the value for a dynamic object to be estimated at any point in time. The calculations are done using simple dynamic models defined for each object in the domain ontology and implemented in the engine's native language (Java). The calculations are always based on the most recent observation for the value (there is no merging of predicted values with observed values – observed values are always assumed to be definitive).

This approach is far from ideal and we would much rather have SAWA focusing on what it does best (i.e., reasoning about relations) rather than modeling Level 1 dynamics. One advantage this approach did afford, however, was that it makes it possible to extrapolate values into the future as easily as to estimate their current values. The consequence is that, with all dynamic projections, the certainty of the values decreases the further away one gets in time from the observation point.

7.3 Managing Uncertainty

Uncertainties in sensed values pose several challenges for situation awareness systems. Although we know they inherently exist it is not always possible to have them measured or reported with the values; even when they are reported, their precise meaning is often ambiguous. But let us assume we are able to obtain certainty measures and they have some well-defined semantics. In a reasoning system like SAWA the question arises as to how to propagate the certainty measures through the logical stages of reasoning. In our rule-based system this requires a way of calculating the certainties of the rule conclusions based on the certainties present in the values in the body. After considering various possibilities we decided on a Bayesian approach. The challenge here was how to implement the approach without overly complicating the development of the rules by the domain expert. Our goal was to permit the user to write ordinary (non-probabilistic) rules and have the system handle the propagation of certainty measures automatically behind the scenes. On the other hand the user should also be able to provide prior probabilities and conditional dependency tables if necessary and possible. In either case the propagation of certainty should be carried out with as much information as is available. In order to do this we would need to embed the Bayesian reasoning deep within the inference engine. For this reason we have implemented our own proprietary Rete-based inference engine and have begun adding support for the Bayesian propagation of certainty.

Acknowledgements

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

References

[1] C. Matheus, K. Baclawski and M. Kokar, Derivation of ontological relations using formal methods in a situation awareness scenario. In Proc of SPIE Conference on Multisensor, Multisource Information Fusion, pages 298-309, April 2003.

[2] T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, May 2001.

[3] OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/.

[4] http://www.w3.org/2004/OWL/.

[5] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, 2004. http://www.w3.org/Submission/SWRL/.

[6] C. Matheus, M. Kokar and K. Baclawski, A Core Ontology for Situation Awareness. In Proceedings of FUSION'03, Cairns, Queensland, Australia, pages 545-552, July 2003.

[7] Resource Description Framework (RDF) Concepts and Abstract Syntax.. W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-concepts/.

[8] European OntoWeb Consortium, A Survey of Ontology Tools, May 2002. http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip.

[9] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, S. W. Tu The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. 2002.

[10] K. Baclawski, M. Kokar, R. Waldinger and P. Kogut, Consistency Checking of Semantic Web Ontologies. 1st International Semantic Web Conference (ISWC)}, Lecture Notes in Computer Science, LNCS 2342, Springer, pp. 454--459, 2002.

[11] K. Baclawski, C. Matheus, M. Kokar, J. Letkowski and P. Kogut, Towards a Symptom Ontology for Semantic Web Applications. In Proceedings of Third International Semantic Web Conference, Hiroshima, Japan, pages 650-667, November, 2004.

[12] RDF Primer. W3C Working Draft. Edited by F. Manola and E. Miller, 2002. http://www.w3.org/TR/rdf-primer/.

[13] OWL-QL: OWL Query Language, 2003. http://ksl.stanford.edu/projects/owl-ql/

[14] Defining N-ary Relations on the Semantic Web: Use With Individuals. July 2004, Noy and Rector (eds).