SAWA: An Assistant for Higher-Level Fusion and Situation Awareness

Christopher J. Matheus[*1], Mieczyslaw M. Kokar[2,1], Kenneth Baclawski[2,1], Jerzy A. Letkowski[3,1],, Catherine Call[1], Michael Hinman[4], John Salerno[4], Douglas Boulware[4]
[1]Versatile Information Systems, Inc., Framingham, MA, USA 01701
[2]Northeastern University, 360 Huntington Avenue, Boston, MA, USA 02115
[3]Western New England College, 1215 Wilbraham Road, Springfield, MA, USA 01119
[4]Air Force Research Laboratory, 525 Brooks Road, Rome, NY, USA 13441

ABSTRACT

Situation awareness involves the identification and monitoring of relationships among level-one objects. This problem in general is intractable (i.e., there is a potentially infinite number of relations that could be tracked) and thus requires additional constraints and guidance defined by the user if there is to be any hope of creating practical situation awareness systems. This paper describes a Situation Awareness Assistant (SAWA) that facilitates the development of user-defined domain knowledge in the form of formal ontologies and rule sets and then permits the application of the domain knowledge to the monitoring of relevant relations as they occur in evolving situations. SAWA includes tools for developing ontologies in OWL and rules in SWRL and provides runtime components for collecting event data, storing and querying the data, monitoring relevant relations and viewing the results through a graphical user interface. An application of SAWA to a scenario from the domain of supply logistics is also presented.

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~~defined[MMK1]~~ relations. These tools collectively make up our **S**ituation **Aw**areness **A**ssistant (**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

[*] cmatheus@vistology.com; phone 1 508 788-5088; fax 1 508 788-9944; http://www.vistology.com

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
SAWA: An Assistant for Higher-Level Fusion and Situation Awareness	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Versatile Information Systems Inc,5 Mountainview Drive,Framingham,MA,01701**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**10**	

associated rules, how SAWA collects and processes incoming events and how it communicates with the user. We also discuss the advantages of the approach as compared with other solutions that we are aware of.

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 we believe 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. In particular we have chosen to use the Web Ontology Language OWL[2] 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[3].

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 (X_1, $X_2 \ldots X_n$) implies the truth of some other information (i.e., $X_1 \wedge X_2 \wedge \ldots X_n \rightarrow Y$). 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 a rule language, and for this purpose we have selected the Semantic Web Rule Language, SWRL[4].

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.

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[5] and will not describe it in detail here. A simplified version of the ontology is shown in Figure 1 with the key concepts being use of objects that have attributes with specific values being 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.

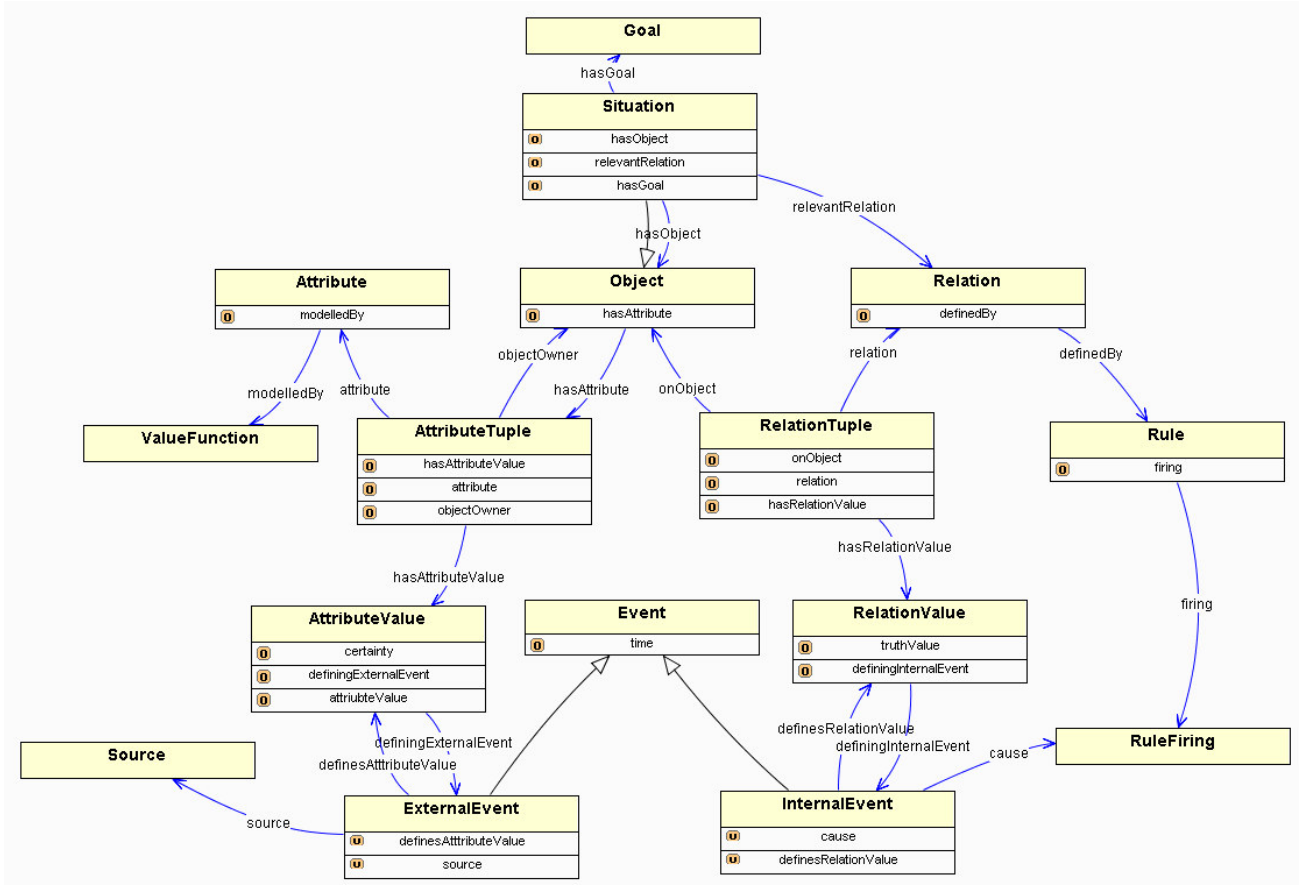

Figure 1: SAW Core Ontology. This ontology serves as the basis for all domain specific ontologies and rule sets. According to the ontology a Situation consists of Objects and Relations and a Goal (standing relation). Objects have AttributeTuples that are associated with specific Attributes and a collection of AttributeValues defined according to ExternalEvents. Relations are realized through RelationTuples that connect pairs of Objects with RelationValues defining by the firing of Rules.

3. SAWA HIGH-LEVEL ARCHITECTURE

The SAWA High-Level Architecture has two aspects as shown in Figure 2: 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).

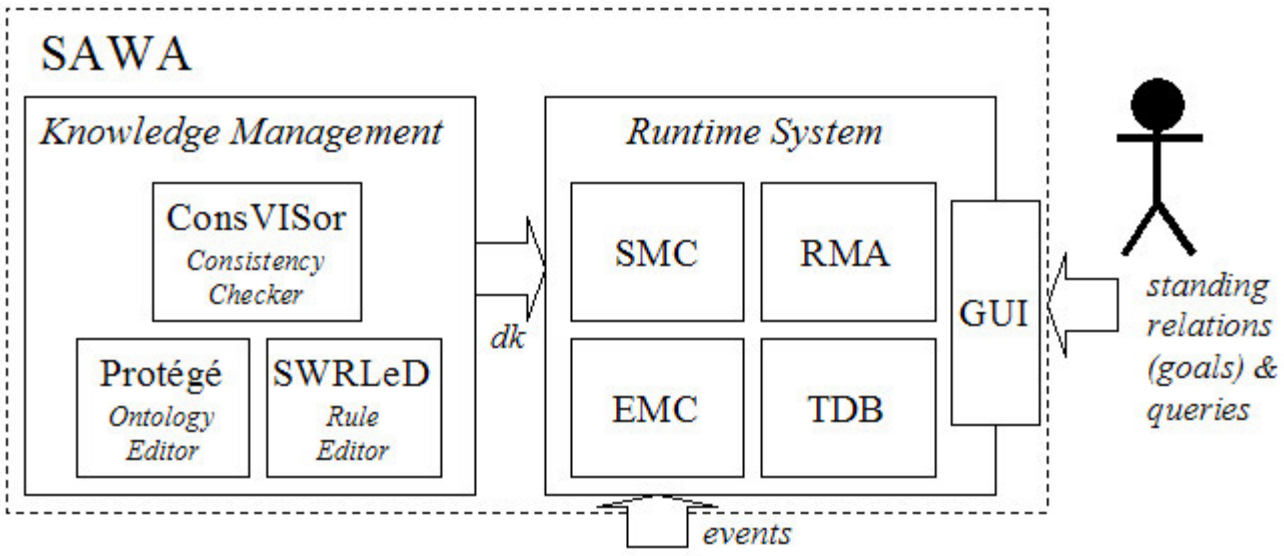

Figure 2: SAWA High-Level Architecture. On the left side of the diagram is the Knowledge Management suite of tools that is used to develop the domain knowledge (dk) that serves as input to the Runtime System, shown on the right hand side. The user interacts with the system through the GUI by issuing 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).

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.

2.4. Ontology Editor

The OWL language is based in RDF, 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[6] but the most widely used is Protégé[7]. 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.

2.5. 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[8], 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. Figure 3 shows a snapshot of the Web Service version of 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, an informative message is provided as shown in Figure 3. 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[9] and as such can produce symptom reports in OWL that can be automatically processed by other OWL-cognizant programs.

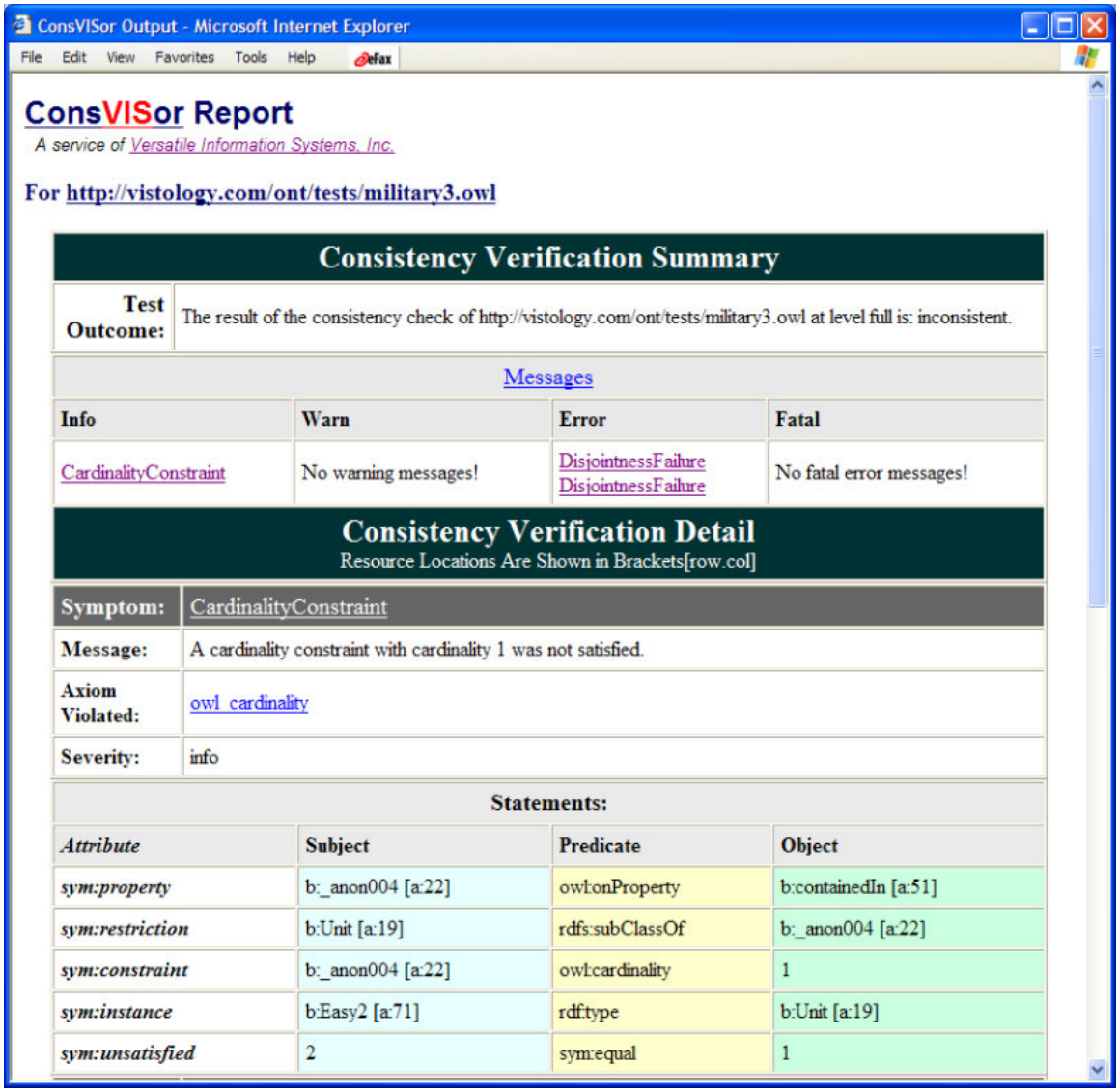

Figure 3: ConsVISor. This screenshot shows sample HTML-based output from the ConsVISor Web Service available at http://www.vistology.com/consvisor. Whereas this HTML output is intended to ease processing by human users, ConsVISor's output may also be received as an OWL document for automated processing by other programs that understand OWL.

2.6. 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 Figure 4. 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, atomic 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 upon which 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 alpha testing 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.

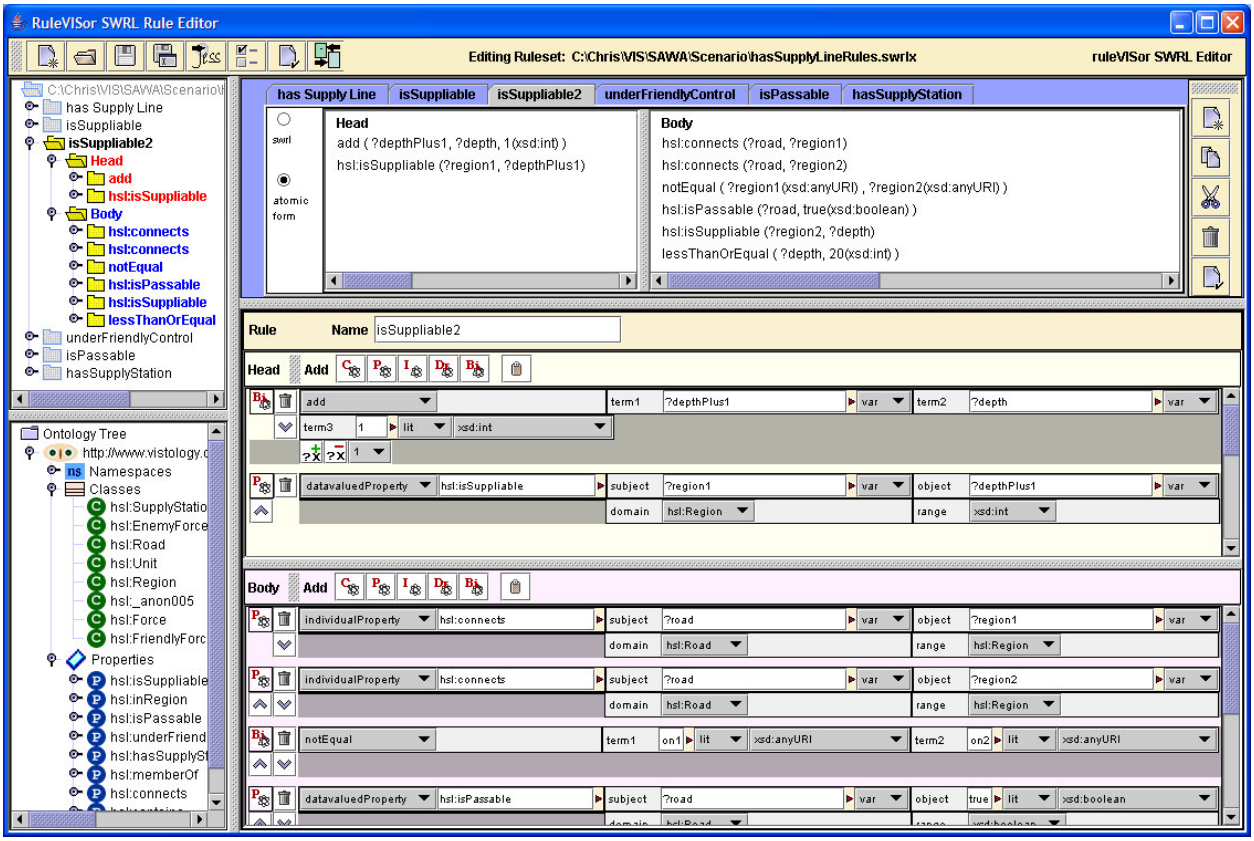

Figure 4: RuleVISor. This screenshot of the RuleVISor SWRL editor shows its use on a set of rules used by the Supply Logistics scenario described in Section 6.

5. SAWA RUNTIME SYSTEM

The SAWA Runtime System, also called the SAWA Engine, is depicted in Figure 5 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.

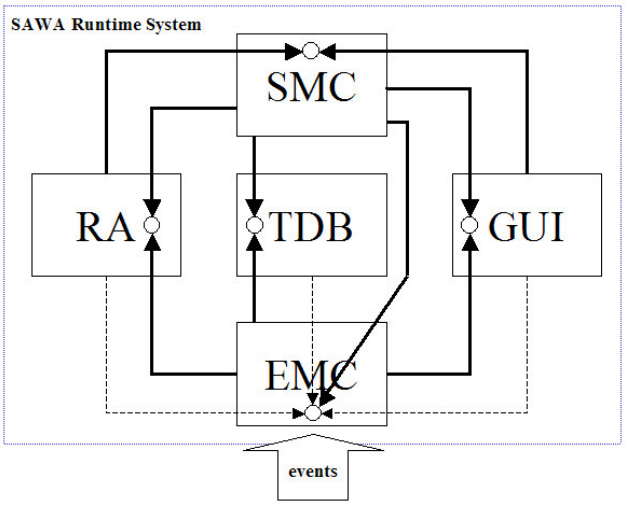

Figure 5: SAWA Runtime System. The communication links between the various SAWA components are shown with solid lines representing permanent connections and dotted lines indicating transient connections.

2.7. 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.

2.8. 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 Figure 6. 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).

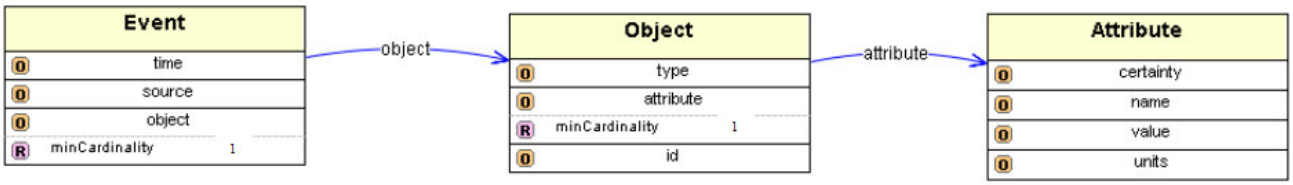

Figure 6: Event Ontology. Simple ontology used to represent incoming events for processing by the EMC. Each Event describes one or more Objects each having one or more Attributes for which a value and certainty measure are defined.

2.9. 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.

2.10. 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.[1] More complex knowledge structure can be represented using collections of interrelated triples (see XXX[MMK2]). 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 OQL: OWL Query language[10]. 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" 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.

2.11. 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 Figure 7.

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 Figure 7. 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).

[1] For simplicity we are here ignoring the use of namespaces on the subject and object.

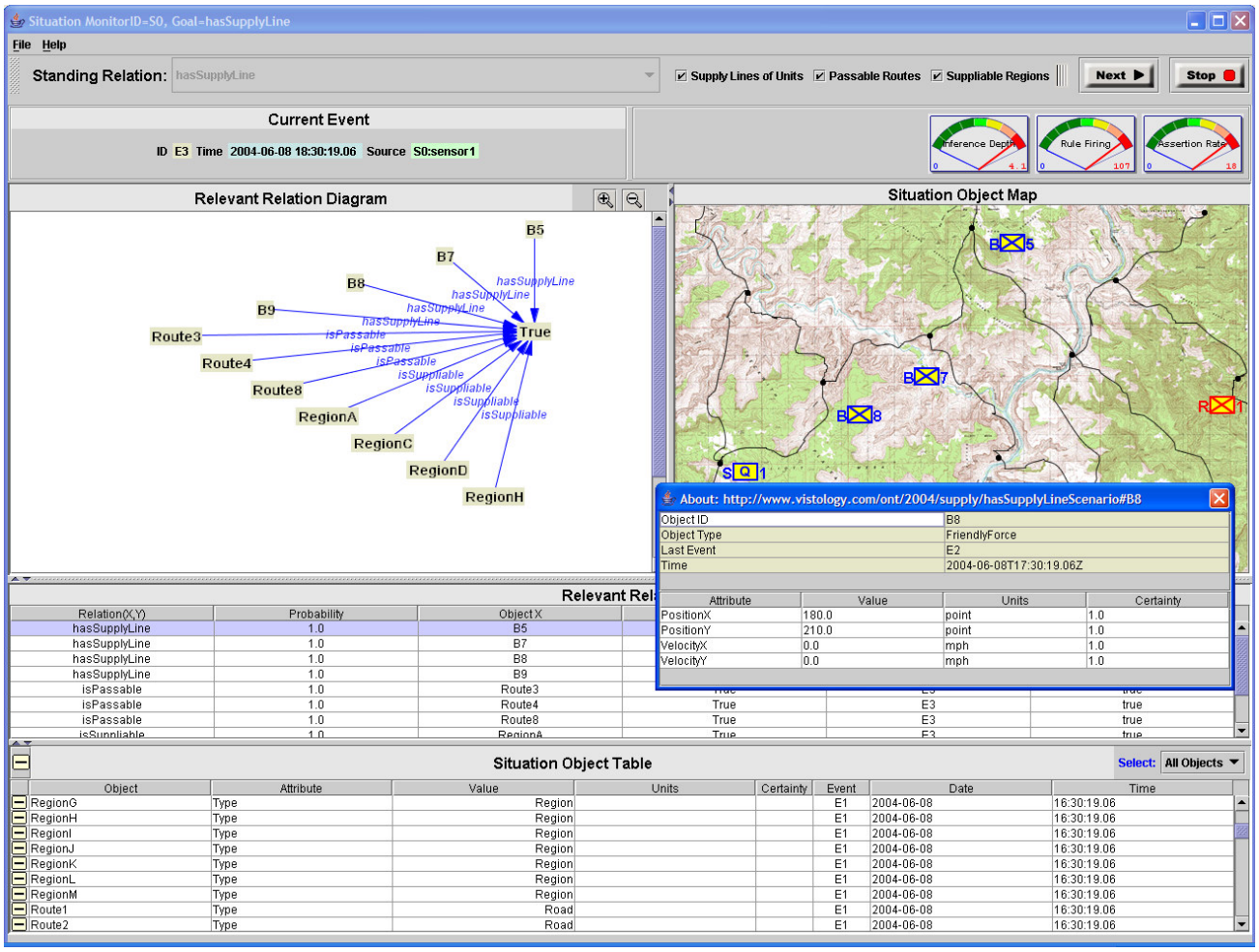

Figure 7: The SAWA GUI. The GUI provides the means for specifying the standing relation (i.e., goal), executing queries, and monitoring the evolution of events, objects, attributes and relations. Objects and attributes are displayed in the Situation Object Table and also on the Situation Object Map. Relevant relations appear in the Relevant Relations table as well as in the Relevant Relation Diagram. Clicking on objects on the map or events, objects or relations in the tables brings up a sub window of supplemental information as shown in the figure for Unit B8. The dials in the upper right hand corner are used for monitoring the performance of the inferencing engine.

The screenshot in Figure 8 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 Figure 8 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 Figure 4. 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).

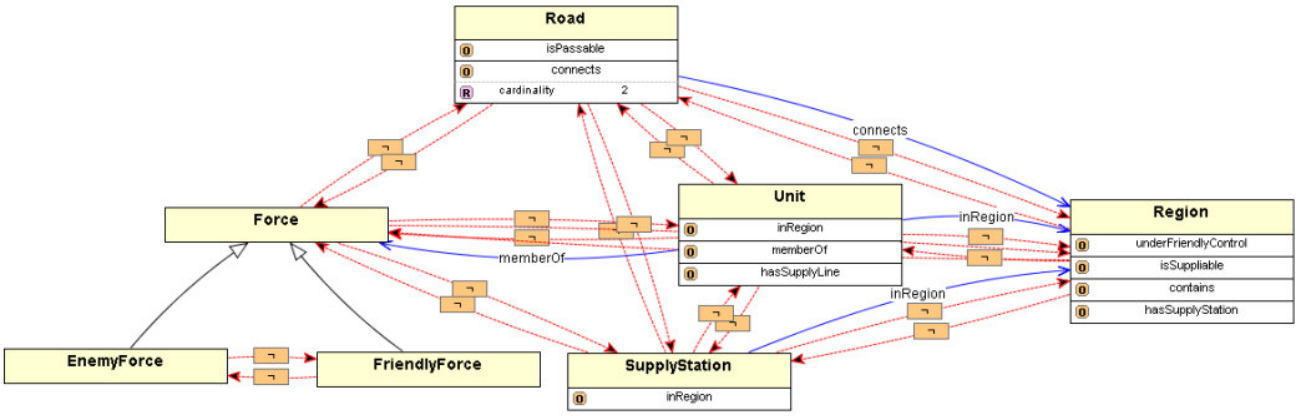

Figure 8: Simple Supply Logistics Ontology. This simple ontology captures just enough information needed for reasoning about supply lines, which serves as the standing relation in our supply logistics scenario. Each of the classes represented in the ontology is actually a subclass of the Object class defined in the SAW Core Ontology shown in Figure 1.

To simulate the running of the scenario several snapshots where 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 around in such a manner as to create changes in the set of relations that would hold true. These snapshots where then presented to a running SAWA application in which the user specified the standing relation to be *hasSupplyLine* as 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 which displayed them for the user; the GUI screenshot in Figure 7 shows the display after a couple of time steps.

7. CONCLUSION

This paper described the Situation Awareness Assistant, SAWA. SAWA is designed to monitor the evolution of higher-order relations within a situation using formal and generic reasoning techniques for level-two fusion. The system is grounded in the use of the formal languages of OWL and SWRL, which permit the representation of ontologies and rules. For a specific application of SAWA, a domain theory consisting of a domain specific ontology and a corresponding set of rules are first constructed or reused from a previous application. A standing relation, or goal, is then defined by the user, which is used to determine the relevant portion of the domain knowledge for the current objectives as well as to identify the relevant object and object-attributes that the system needs to monitor in the event stream. As relevant events are detected they are passed on to the relation-monitoring agent, which analyzes them for the possible occurrence of higher-order relations. As higher-order relations are detected they are passed onto the GUI, which displays them in both tabular and graphical forms for the user along with other data pertaining to the events, objects and their attributes. The GUI also provides the capability for querying the system's triple database using basic OQL queries or with "what-if" queries that can produce hypothetical situations against which a query is run. A scenario from the domain of supply logistics was briefly described.

REFERENCES

[1] Relevance Reasoning paper, 2003.
[2] OWL, 2003.
[3] OWL Tools, 2003.
[4] SWRL, 2004.
[5] A Core Ontology for Situation Awareness, FUSION03, 2003.
[6] Ontology Tools Eval Paper, 2003.
[7] Protégé.
[8] ConsVISor, 2003. http://www.vistology.com/consvisor.
[9] A Symptom Ontology for Semantic Web Documents, ISWC04, 2004.
[10] OQL: OWL Query Language, 2003.