Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations	1. Thesis and Dissertation Collection, all items

2006-06

Applying semantic web concepts to support Net-Centric Warfare using the Tactical Assessment Markup Language (TAML)

Childers, Candace M.

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/2770

This publication is a work of the U.S. Government as definedin Title 17, United States Code, Section 101. As such, it is in thepublic domain, and under the provisions of Title 17, United StatesCode, Section 105, is not copyrighted in the U.S.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLYING SEMANTIC WEB CONCEPTS TO SUPPORT NET-CENTRIC WARFARE USING THE TACTICAL ASSESSMENT MARKUP LANGUAGE (TAML)

by

Candace M. Childers

June 2006

Thesis Advisor:	Don Brutzman
Co-Advisor:	Curt Blais
Second Reader:	Paul Young

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Applying Semantic Web Concepts to support Net-Centric Warfare Using the Tactical Assessment Markup Language (TAML)			**5. FUNDING NUMBERS**
6. AUTHOR Childers, Candace M.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT

The ability to analyze data quickly and transform it into important information is vital for information superiority. However, the amount of available data is increasing and the time to make decisions is decreasing. There is too much data for humans to sift through and filter for decision making, so computer automation is necessary. The current approach to automating data processing is to hard-code programs to parse particular data formats, but this approach is not flexible enough to handle the constantly changing data world. The Extensible Markup Language (XML) offers a partial solution by providing a syntactic standard for data exchange. The Tactical Assessment Markup Language (TAML) is an XML vocabulary for exchanging undersea warfare tactical data that provides a standard syntax for message exchange. However, the meaning or semantics of the data is unknown to the machine processing the data.

The Semantic Web is a set of technologies designed to add semantic information to data for machine processing. The technologies consist of several components including a common syntax for data exchange, common semantic representation, and a common ontology language. The Resource Description Framework (RDF) is used to explicitly state the relationships between resources or entities. The Web Ontology Language (OWL) is used to build models that explicitly define the concepts and properties in a domain. Since concept definitions are written in standard languages, a variety of reasoning engines might be used to process any ontology and its corresponding data instances. Reasoning engines can also apply algorithms to the data to infer useful information and present it to decision makers. Thus there is far less need for specialty hard-coded programs or proprietary data-representation schemes to hold semantic information, since the information needed to process data is captured in an OWL ontology, itself stored in XML format for exchange between systems.

Building ontologies for specific domains such as undersea warfare allows programs to understand, process, and infer new information from coherent data. Applying Semantic Web technologies to XML languages such as TAML brings the armed forces closer to a knowledge-aware Global Information Grid (GIG).

14. SUBJECT TERMS: Extensible Markup Language (XML), Global Information Grid (GIG), Semantic Web (SW), Tactical Assessment Markup Language (TAML), Resource Description Resource (RDF), Web Ontology Language (OWL), Semantic Web Rule Language (SWRL)	**15. NUMBER OF PAGES** 288
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

APPLYING SEMANTIC WEB CONCEPTS TO SUPPORT NET-CENTRIC WARFARE USING THE TACTICAL ASSESSMENT MARKUP LANGUAGE (TAML)

Candace M. Childers
Ensign, United States Navy
B.S., United States Naval Academy, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2006

Author: Candace M. Childers

Approved by: Don Brutzman
Thesis Advisor

Curt Blais
Co-Advisor

Paul Young
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to analyze data quickly and transform it into important information is vital for information superiority. However, the amount of available data is increasing and the time to make decisions is decreasing. There is too much data for humans to sift through and filter for decision making, so computer automation is necessary. The current approach to automating data processing is to hard-code programs to parse particular data formats, but this approach is not flexible enough to handle the constantly changing data world. The Extensible Markup Language (XML) offers a partial solution by providing a syntactic standard for data exchange. The Tactical Assessment Markup Language (TAML) is an XML vocabulary for exchanging undersea warfare tactical data that provides a standard syntax for message exchange. However, the meaning or semantics of the data is unknown to the machine processing the data.

The Semantic Web is a set of technologies designed to add semantic information to data for machine processing. The technologies consist of several components including a common syntax for data exchange, common semantic representation, and a common ontology language. The Resource Description Framework (RDF) is used to explicitly state the relationships between resources or entities. The Web Ontology Language (OWL) is used to build models that explicitly define the concepts and properties in a domain. Since concept definitions are written in standard languages, a variety of reasoning engines might be used to process any ontology and its corresponding data instances. Reasoning engines can also apply algorithms to the data to infer useful information and present it to decision makers. Thus there is far less need for specialty hard-coded programs or proprietary data-representation schemes to hold semantic information, since the information needed to process data is captured in an OWL ontology, itself stored in XML format for exchange between systems.

Building ontologies for specific domains such as undersea warfare allows programs to understand, process, and infer new information from coherent data. Applying Semantic Web technologies to XML languages such as TAML brings the armed forces closer to a knowledge-aware Global Information Grid (GIG).

.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. PROBLEM OVERVIEW....1
B. PROBLEM DESCRIPTION....2
C. MOTIVATION....4
D. OBJECTIVES....5
E. THESIS ORGANIZATION....5

II. BACKGROUND AND RELATED WORK....9
A. INTRODUCTION....9
B. ARTIFICIAL INTELLIGENCE (AI) PARDIGMS....9
1. Expert Systems....10
2. Case-Based Reasoning (CBR)....12
3. Knowledge Representation (KR)....13
C. SEMANTIC WEB (SW) TOOLS....15
1. Protégé....15
2. SemanticWorks....25
3. RacerPro Version 1.9....26
4. Pellet Reasoner....28
D. SEMANTIC WEB (SW) PROJECTS....29
1. Valued Information at the Right Time (VIRT)....29
2. Suggested Upper Merged Ontology (SUMO)....31
3. Cyc....33
E. SUMMARY....35

III. DETAILED PROBLEM STATEMENT....37
A. INTRODUCTION....37
B. NATURE OF THE PROBLEM....37
C. CANDIDATE SOLUTIONS....39
D. APPROACH CHOSEN....41
E. SUMMARY....42

IV. TACTICAL ASSESSMENT MARKUP LANGUAGE (TAML)....43
A. INTRODUCTION....43
B. PURPOSE....43
C. DESCRIPTION....46
D. SCHEMA DESIGN....47
E. SUMMARY....49

V. SEMANTIC WEB (SW) CONCEPTS 51
A. INTRODUCTION 51
B. WHAT IS THE SEMANTIC WEB (SW)? 51
1. The Vision 51
2. Semantic Web (SW) Layers 52
C. ENABLING TECHNOLOGIES 62
1. Providing a Common Syntax Using XML 63
2. Resource Description Framework (RDF) 64
3. RDF Schema (RDFS) 70
4. Web Ontology Language (OWL) 71
5. Semantic Web Rule Language (SWRL) 75
D. SUMMARY 78
VI. SERIALIZING TAML DOCUMENTS AS RDF/XML 81
A. INTRODUCTION 81
B. WHY THE RESOURCE DESCRIPTION FRAMEWORK (RDF)? 81
1. Advantages 81
2. Disadvantages 84
C. DESIGN 85
1. Resources 85
2. Predicates 87
3. Objects 89
4. Containers 90
5. Reification 93
6. Development Tools 93
D. QUERY LANGUAGES 97
1. XQuery 97
2. SPARQL Protocol and RDF Query Language (SPARQL) 99
E. USING SPARQL TO QUERY TAML RDF/XML DOCUMENTS 100
1. Jena 100
2. Examples 103
F. SUMMARY 109
VII. TAML OWL ONTOLOGY 111
A. INTRODUCTION 111
B. USE CASES 111
1. Semantic Validation 111
2. Classifying Instances 113
3. Querying Over Multiple Domains 114
C. ONTOLOGY LANGUAGE CHOICE FOR TAML APPLICATIONS 116
D. OWL ONTOLOGY DESIGN FOR TAML 117
1. Ontology Editor 118
2. Domain and Scope 119
3. Ontology Reuse 120
4. Class Hierarchy 121
5. Properties 123

6. Class Definitions 125
E. ONTOLOGY TESTING 134
1. Built-in Ontology Tests 134
2. Reasoning Tests with RacerPro Reasoner 136
3. Testing for Ontology Consistency 137
4. Testing for Semantic Consistency 137
5. Testing the Classification Use Case 139
F. SUMMARY 141

VIII. USING SWRL TO ADD RULES TO THE TAML CONTACT CLASSIFICATION ONTOLOGY 143
A. INTRODUCTION 143
B. SEMANTIC WEB RULE LANGUAGE (SWRL) 143
1. SWRL Built-Ins 144
2. SWRL Rule Format 147
3. Protégé-OWL SWRLTab Plug-in 148
C. RULE DESIGN FOR TAML 151
1. Datatype Range Restriction Rules 151
2. Checking Semantic Consistency with SWRL 152
3. Prioritizing Hostile Contacts 154
4. Same As/Different From Rules 156
5. Adding Complex Rules 157
D. SWRL LIMITATIONS 159
E. USING RULE ENGINES TO EXECUTE SWRL RULES 160
F. SUMMARY 162

IX. EXPERIMENTAL RESULTS 163
A. INTRODUCTION 163
B. SEMANTIC WEB (SW) LANGUAGE ASSESSMENTS 163
1. Tactical Assessment Markup Language (TAML) Schema 165
2. TAML Serialized as RDF 166
3. OWL-Only TAML Contact Classification Ontology 167
4. OWL/SWRL TAML Contact Classification Ontology 169
C. ASSESSMENT OF QUERY LANGUAGES 170
D. SUMMARY 171

X. CONCLUSIONS AND RECOMMENDATIONS 173
A. CONCLUSIONS 173
B. RECOMMENDATIONS FOR FUTURE WORK 174
1. Execute and Test SWRL Rules 174
2. TAML Contact Classification Application 175
3. Probabilistic-OWL (PR-OWL) 175
4. Domain-Specific Reasoners 176
5. Valued Information at the Right Time (VIRT) 177
6. Autonomous Vehicle Command Language (AVCL) 177
7. Further TAML Ontology Development and Testing 177

APPENDIX A. RDF/XML SERIALIZATION OF TAML DOCUMENTS.....179
A. DESCRIPTION.....179
B. TRAFALGAREXAMPLE.RDF.....179
APPENDIX B. TAML CONTACT CLASSIFICATION ONTOLOGY SOURCE 199
A. DESCRIPTION.....199
B. TAMLCONTACTCLASSIFICATION.OWL SOURCE.....201
APPENDIX C. TAML OWL / SWRL CONTACT CLASSIFICATION ONTOLOGY SOURCE.....217
A. DESCRIPTION.....217
B. TAMLCONTACTCLASSIFICATIONSWRL.OWL SOURCE.....219
LIST OF REFERENCES.....255
INITIAL DISTRIBUTION LIST.....263

LIST OF FIGURES

Figure 1. An illustration of the rule-based system architecture [From Ref. University of Tokyo Center of Collaborative Research, 2006]. 11
Figure 2. An illustration of the formalized four-step process for Case-Based Reasoning (CBR) [From Ref. Aamodt, 1994].....12
Figure 3. Knowledge Representation (KR) consists of defining knowledge about entities and the laws that govern them at a symbolic level [From Ref. Schmidt, 2005]. 14
Figure 4. The Protégé-2000 architecture [From Ref. Gennari, 2002]. 17
Figure 5. Screenshot of Protégé's OWL GUI interface. 19
Figure 6. A graphical view of *VegetarianPizza* using the OWLViz plug-in [From Ref. OWLViz Guide, 2004].....21
Figure 7. A graphical view of the *Contact* class from the Jambalaya plug-in 23
Figure 8. A screenshot of the SWRLTab plug-in [From Ref. http://protege.stanford.edu/plug-ins/owl/swrl].....24
Figure 9. Screenshot of the SemanticWorks graph editing tool [From Ref. http://www.altova.com/products_semanticworks.html]. 25
Figure 10. Screenshot of RacerPro running inside the Protégé ontology editor ….....27
Figure 11. Visualization of the component-based architecture for VIRT [From Ref. Hayes-Roth, 2005]. 30
Figure 12. High-level view of the SUMO ontology [From Ref. http://ontolog.cim3.net]32
Figure 13. A diagram of the Cyc components [From Ref. http://www.cyc.com/cyc/technology/whatiscyc] 34
Figure 14. Vision of the GIG for combining various data sources to create interoperability [From Ref. http://akss.dau.mil/dag/Guidebook/IG_c7.3.4.1.asp].....38
Figure 15. An illustration of the mappings needed to connect eight different systems for information exchange [From Ref. Brutzman and Grimley, 2006].....44
Figure 16. An illustration of the mappings needed to connect eight different systems when TAML is added as an external model between systems [From Ref. Brutzman and Grimley, 2006].....45
Figure 17. Illustration of the levels of interoperability required for machine understanding of data across systems [From Ref. Obrst, 2006]. 53
Figure 18. Definition of an ontology as a specification of a conceptualization and a description of a domain [After Ref. Gruber 1993]. 54
Figure 19. Modified Semantic Layer Cake showing the components that make up the SW and highlighting the technologies that enable those layers [After Ref. Berners-Lee 2001]. 62
Figure 20. RDF Graph showing the graph representation of the above N-triples.....66
Figure 21. The SWRL source for a rule defining the property, *has40YearOldBrother*....77
Figure 22. Fragment of Trafalgar.xml illustrating the use of *id* attributes. 85
Figure 23. Fragment of TrafalgarExample.rdf demonstrating how unique names are created for operations.....86

Figure 24. Fragment of Trafalgar.xml showing child elements as properties of their parent element 86
Figure 25. Fragment of TrafalgarExample.rdf showing how unique names are created for elements without unique *id* attributes. 87
Figure 26. Fragment of Trafalgar.xml showing child elements which descibe element *Contact* 88
Figure 27. Fragment of TrafalgarExample.rdf showing child elements as properties of their parent class *Contact* 88
Figure 28. Comparison of XML serialization (above) and RDF serialization (below) of literal objects 89
Figure 29. Illustration of a resource as an object in TAMLExample.rdf 90
Figure 30. Example showing several elements of the same name, *TimePosition* under one parent element, *Track* 91
Figure 31. Example of the use of rdf:*Seq* in TAMLExample.rdf 92
Figure 32. Example of the use of rdf:*Bag* in TAMLExample.rdf. 92
Figure 33. Screenshot of TAMLExample.rdf graph from Altova SemanticWorks. 95
Figure 34. Screenshot of the W3C online parser used to validate RDF/XML. Available online at http://www.w3.org/RDF/Validator. 96
Figure 35. XML describing the make-up of a book, adapted from [Hunter 2004]. 98
Figure 36. An illustration of SimpleBooks.xml (top), a query to retrieve all book elements (middle), and the result XML (bottom) [After Ref. Hunter, 2004] 99
Figure 37. A screenshot of the results of an example query displayed as a table within the Twinkle GUI [From Ref. http://www.ldodds.com/projects/twinkle] 102
Figure 38. A screenshot of the results of an example query displayed as text within the Twinkle GUI [From Ref. http://www.ldodds.com/projects/twinkle] 102
Figure 39. A screenshot of the results of an example query displayed as rs/xml within the Twinkle GUI [From Ref. http://www.ldodds.com/projects/twinkle] 103
Figure 40. SPARQL query requesting all resources with a *Longitude* and *Latitude* property. 104
Figure 41. Results of SPARQL SELECT/WHERE query when run against TrafalgarExample.rdf. Results are sorted by *Longitude* and displayed in table format 105
Figure 42. SPARQL query which returns all subjects who have *Speed* equal to ten and *Course* equal to 45.0 106
Figure 43. Results of query defined in Figure 42 when run against TAMLExample.rdf. Results are displayed in table format 106
Figure 44. SPARQL query demonstrating the use of the UNION keyword 107
Figure 45. Results of query defined in Figure 44 when run against TAMLExample.rdf 107
Figure 46. A sample ASK query 108
Figure 47. Result of the ASK query defined in Figure 46 when run against TrafalgarExample.rdf 108
Figure 48. A query asking for a description of the resource with taml:*Name* equal to "Africa" 109

Figure 49. Result of the DESCRIBE query defined in Figure 48 when run against TrafalgarExample.rdf 109
Figure 50. Use case diagram for applying concept semantics and validating semantic consistency of TAML documents 112
Figure 51. Use case diagram for classifying TAML contact instances. 114
Figure 52. Use case diagram for querying over multiple domains. 115
Figure 53. A diagram of the Protégé-OWL architecture [From Ref. Knublauch 2004]. 119
Figure 54. View of *Contact* hierarchy from the Protégé Ontoviz plug-in. 121
Figure 55. This screenshot illustrates the definition for *Contact* asserted in Protégé with an emphais on Necessary conditions 127
Figure 56. This screenshot illustrates the necessary and sufficient rules for *FriendlyContact* asserted in Protégé. 129
Figure 57. Invoking the built-in ontology test function in Protégé-OWL....135
Figure 58. The reasoner returns an error when two properties contradict each other, but does not provide a useful reason for the error....139
Figure 59. Protege highlights required fields in red if they are missing a required value. 139
Figure 60. A screenshot of a SWRL rule inside the Protégé SWRL editor. The rule is designed to reclassify instances of *SuspectContact* as instances of *HostileContact* when contact range is less than 10000. 149
Figure 61. The OWL source for the rule defined in Figure 60 150
Figure 62. SPARQL query which returns the id of all Contacts where the *hasSemanticError* property is equal to "true". 154
Figure 63. SPARQL query which returns the *id* of all *Contact*s where the *isImmediateThreat* property is equal to "true" 156
Figure 64. Multiple logical translations are needed in order to be able to execute SWRL rules using a rule engine integrated into Protégé [From Ref. Plas 2006] 161
Figure 65. TAML code from Trafalgar.xml which illustrates the ambiguity in TAML documents. 165
Figure 66. Comparison of XML serialization (above) and RDF serialization (below) of literal objects. 166
Figure 67. A view of the properties defined for *Contact* in the Protégé GUI 199
Figure 68. A view of the *FreindlyContact* class definition in the Protégé GUI....200
Figure 69. A screenshot from the Protégé SWRLTab showing the SWRL rules defined for the TAML Contact Classification Ontology 218

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Table of the logical symbols used to represent OWL elements in the OWL plug-in [From Ref. Knublauch, 2004] 20
Table 2. List and description of all the classes in TAMLContactClassification.owl. 122
Table 3. List and description of the datatype properties in TAMLContactClassification.owl. 124
Table 4. List and description of the object properties describing *Contact* in TAMLContactClassification.owl 125
Table 5. List and description of axioms defining the class *Contact*. 126
Table 6. List and description of axioms defining the class *FriendlyContact*. 130
Table 7. List and description of axioms defining the class *HostileContact*. 131
Table 8. List and description of axioms defining the class *SuspectContact*. 132
Table 9. List and description of axioms defining the class *NeutralContact*. 132
Table 10. List and description of axioms defining the class *UnknownContact*. 133
Table 11. List and description of test cases used to test proper inference of Classification in TAMLContactClassification.owl 140
Table 12. List and description of the SWRL comparison built-in operators used to define rules within the TAML Contact Classification Ontology [After Ref. http://www.daml.org/2004/04/swrl/builtins.html] 145
Table 13. Description of the SWRL multiply built-in operator used within the TAML Contact Classification Ontology [After Ref. http://www.daml.org/2004/04/swrl/builtins.html]. 145
Table 14. Description of other built-ins not used in the TAML Contact Classification Ontology [After Ref. http://www.daml.org/2004/04/swrl/builtins.html] 146
Table 15. List and description of the SWRL rules designed to indicate when an instances includes a range violation on a datatype property. 152
Table 16. List and description of the SWRL rules designed to indicate when an instance includes a semantic inconsistency 153
Table 17. List and description of the SWRL rules designed to prioritize hostile contacts by their threat level. 155
Table 18. List and description of the SWRL rules sameAs and differentFrom, designed to indicate whether two instances refer to the same or a different real world *Contact*. 157
Table 19. List and description of the SWRL rules added to the TAML Contact Classification Ontology to demonstrate how complex rules can be formed 158
Table 20. An overview of the capabilities and limitations of SW languages. 164

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Program Interface
ARP	Another RDF Parser
A-TAS	Anti-Submarine Warfare Tactical Assessment System
BSC	Business Standards Council
C4I	Command, Control, Communications, Computers and Intelligence
CBR	Case-Based Reasoning
CVTSC	Carrier Tactical Support Center
DAWG	Data Access Working Group
DIG	Description Logic Implementation Group
DISA	Defense Information System Agency
DL	Description Logic
DoD	Department of Defense
DON	Department of the Navy
DTD	Document Type Definition
FOL	First-Order Logic
GIG	Global Information Grid
GUI	Graphical User Interface
HTML	HyperText Markup Language
JC3IEDM	Joint Consultation Command & Control Information Exchange Data Model
Jess	Java Expert System Shell
KIF	Knowledge Interchange Format
KR	Knowledge Representation
LGPL	Lesser GNU Public License
MILO	Mid-Level Ontology
NDR	Naming and Design Rules
MEBN	Multi-Entity Bayesian Networks
NLP	Natural Language Processor

NPS	Naval Postgraduate School
OWL	Web Ontology Language
OWL-DL	Web Ontology Language Description Logic
OWL-PR	Probabilistic Web Ontology Language
OWL-QL	Web Ontology Language Query Language
OWL-S	Web Ontology Language for Services
PSM	Problem-solving Method
RacerPro	Renamed ABox and Concept Expression Reasoner Professional
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	Resource Description Framework Query Language
ROE	Rule of Engagement
SHriMP	Simple Hierarchical Multi-Perspective
SKSI	Semantic Knowledge Source Integration
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SUMO	Suggested Upper Merged Ontology
SW	Semantic Web
SWRL	Semantic Web Rule Language
TAML	Tactical Assessment Markup Language
URI	Uniform Resource Identifier
USW	Undersea Warfare
USW-DSS	Undersea Warfare Decision Support System
VIRT	Valued Information at the Right Time
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language
XQuery	XML Query Language
XSLT	Extensible Style Language for Transformations

ACKNOWLEDGMENTS

First I would like to thank my thesis advisor, Dr. Don Brutzman for introducing me to an exciting problem and teaching me all about XML and TAML. His enthusiasm for the Semantic Web kept me focused and excited through the long nights of researching, building ontologies and writing.

I would like to thank my co-advisor, Curt Blais for the countless brainstorming sessions where we were able to bounce modeling decisions off each other. Thank you for getting me through the tough parts, especially the installing of Twinkle.

I would like to thank my second reader, Captain Paul Young for sparking my interest in the Semantic Web and providing outside insight to the problem.

I would like to thank the USW-XML Working Group for their interest in my work and their feedback throughout this process.

I would like to thank Altova for providing NPS with a university-partner license permitting free use of all their XML software tools.

I would like to thank Franz Inc. for providing an academic license permitting the free use of the RacerPro Semantic Web reasoner.

I would like to thank my family for their moral support throughout all my years of school. Their encouragement and visible pride in my accomplishments gave me the drive and desire to excel.

Lastly and most importantly, I would like to thank my husband Patrick for his constant support. Thank you for playing golf on the weekends, so I could work without distraction. Most importantly thank you for proofing my chapters and encouraging me to do my best. I love you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

> We must have information superiority: the capability to collect, process and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same.
>
> -- Army Vision 2010

A. PROBLEM OVERVIEW

This work explores the enhancements to information superiority offered by the adoption of Semantic Web (SW) technologies. The addition of SW languages such as the Extensible Markup Language (XML), Resource Description Framework (RDF), Web Ontology Language (OWL), and the Semantic Web Rule Language (SWRL) allow systems to exchange, process, analyze, and reason against information. The application of SW technologies to the tactical domain are described and demonstrated in this work.

Semantic Web (SW) languages and technologies are being developed to extend upon the successful model provided by the World Wide Web (WWW). The various components of the SW are described, and the application of these components to the Tactical Assessment Markup Language (TAML) for Undersea Warfare (USW) is explored.

Information overload is an increasing problem in the current battlespace. The amount of information available to warfighters is increasing while the amount of time available to process the information is decreasing. The focus of this work is the application of standard languages to military systems in order to increase the capabilities of machine automation. The goal is to describe and explore languages and technologies that have the potential to bring the armed forces closer to net-centric warfare.

The SW addresses the problem of adding machine-interpretable semantics (or meaning) to domain specific data and vocabularies. XML provides a foundation for the Semantic Web by providing a syntactic standard for data representation. Each component of the SW adds a new level of expressiveness to data. Together the components of the SW are intended to create a rich model of a domain which can be used by machines to

process and analyze data for valuable and relevant information. The SW increases the capabilities of machines to quickly provide meaningful information to warfighters.

This work takes a pragmatic approach to implementing SW technologies in a tactical domain, and offers an analysis of the advantages and limitations of this approach to solving the problem of information overload.

B. PROBLEM DESCRIPTION

Computer technology is rapidly changing. New hardware and software are constantly being created in the fast-paced computer world. The Department of Defense (DoD) is continually buying new computer equipment and software systems in order to extend its capabilities. As technology has grown so has the idea of connecting computers to improve their capabilities through shared functionality. Communications technology has driven this new recognition of power. Computers have the capability to send and receive data through networks. This awareness is driving the need to find a way to create interoperability among systems with different hardware and software components that were not originally created with interoperability in mind. Interoperability is the ability for two systems to communicate, share information, and perform tasks concurrently (Young, 2002).

The Global Information Grid (GIG) refers to the net-centric warfare environment of the future (Winters and Tolk, 2005). The DoD is transitioning from a platform age to a net-centric age where information is the central focus. Information superiority depends on the efficient collection, analysis and presentation of information, but currently raw data is presented in a variety of inconsistent forms throughout the DoD. There is a need for syntactic and semantic interoperability in order to merge data together to form useful information for warfighters. Syntactic interoperability refers to the ability to transport data between systems for communication while semantic interoperability refers to the effective and consistent interpretation of the meaning of data between systems (Hillman, 2003). The amounts of raw data are staggering, so computer automation is key to collecting, analyzing and converting data to useful information.

The Tactical Assessment Mark-up Language (TAML) is an Extensible Markup Language (XML) vocabulary recently defined by the USW-XML Working Group. The

goal of TAML is to enhance interoperability and understanding of USW messaging through utilization of common XML tag sets. The use of XML in USW messaging enforces standardization, so that TAML documents might be understood anywhere in the Undersea Warfare (USW) community. Since XML primarily provides syntactic interoperability, semantic issues such as well-defined meanings for elements are not yet formally addressed in TAML. This thesis explores that problem space.

In 1998, Tim Berners-Lee introduced the concept of the Semantic Web (SW) (Berners-Lee, 1998). The goal of the SW is to represent data in a way that enables machines to interpret and process information. XML provides the foundation for the SW by providing a standardized way to represent data in different domains using tagsets, but XML alone does not represent data in a way that machines can interpret it. Thus several languages have been developed for use on top of XML, including: Resource Description Framework (RDF) (http://www.w3.org/RDF), Web Ontology Language (OWL) (http://www.w3.org/TR/owl-features), and Semantic Web Rule Language (SWRL) (http://www.w3.org/Submission/SWRL). These languages add metadata to data in order to enhance machine understanding of a domain. In this context, "understanding" loosely refers to the ability to process data and infer information using logic. RDF is used to add metadata about a specific resource. The metadata adds semantic meaning by defining relationships between resources, better enabling a machine to understand and process information. OWL is used to create an ontology or formal specification of a specific vocabulary defining a domain. An ontology helps to represent the data in ways that enable a computer to infer information about the relationships between objects. With the additional information provided by RDF and OWL, computers are thus able to analyze and logically process the data stored in XML documents.

Current software practices require that programs be explicitly configured by human programmers to process the data in TAML documents. Computer programs can then answer the questions about data that they are directly programmed to answer. However, with the addition of RDF information to TAML and the addition of an ontology supporting the expanded vocabulary, computers may be able to interpret the data and answer key semantic questions that only humans were capable of defining and answering before. This capability allows a reasoning engine (such as the Renamed ABox

and Concept Expression Reasoner Professional-RacerPro) to infer answers to questions without an explicit program defining the logical rules that tell it how.

C. MOTIVATION

The amount of tactical data available within the modern battlespace can be overwhelming. Data gathering techniques are abundant, and it is impossible for humans to quickly and efficiently analyze all of the data being produced about a particular situation. However, the quickly evolving nature of combat benefits from the ability to dissect information quickly. Humans need to be presented with the most pertinent data, so they can convert it to useful information or knowledge. Current machines process data at rapid speeds, but they do not have the ability to reason about the content in order to derive important information and present it to decision makers. Computers do not understand the data they are processing; therefore, they must be hard-wired to process the data, which limits their capabilities.

The SW offers a possible solution to automating the processing and analysis of data. SW technologies are a way to give meaning to data which machines can then process. The SW is based on languages that provide standardization of data definitions, knowledge representation, and logical processing. This standardization allows computers to handle any vocabulary, but leaves enough flexibility to make it possible to represent any domain. The key to automation with SW technologies is the ability for any machine with a SW-capable parser to read and understand the rules of a domain-specific ontology. Whenever a machine receives information from a new source, it can correctly process the information if it has the associated ontology. Separating the rules of logic from the hard-coded software means that semantic processing becomes consistent, accessible, verifiable and scalable. Proceeding even further, ontologies can be mapped to each other or to a common upper ontology to enable interoperability between differently expressed knowledge domains. The SW may provide the machine automation needed to process and analyze the overwhelming amounts of tactical data available to warfighters.

D. OBJECTIVES

This thesis seeks to assess whether SW technologies are a viable solution to the problem of providing real-time analysis and interoperability of data in tactical situations. SW technologies are already being used in the commercial world to provide interoperable solutions between domains that need to share information.

To achieve the objective of assessing tactical viability, this thesis examines the benefits of adding semantic meaning to TAML documents by representing the data using RDF/XML. RDF/XML is a semantic representation of data that is understandable by machines with an RDF parser. Representing data as RDF/XML allows machines to apply a more powerful reasoning and query capability to data. This thesis demonstrates such capabilities by serializing TAML data as RDF/XML files.

This thesis also examines the reasoning and querying power added to TAML by creating an OWL ontology to explicitly define the TAML vocabulary. Semantic use cases for the TAML language are introduced in the thesis and a corresponding ontology is built to answer the common questions posed for the semantic use cases. These results are then assessed to determine whether SW technologies can demonstrably enhance tactical information processing.

This thesis is mainly concerned with determining how SW technologies can add useful semantic information to tactical information in order to automate information processing and decision making. Producing a working system as part of this assessment provides both an immediate real capability and demonstrates SW methodologies for future work.

E. THESIS ORGANIZATION

Chapter I provides an introduction that includes an overview of the problem to be explored, the motivation for exploring the problem, the technical approach taken to explore solutions to the problem, and the organization of the thesis document.

Chapter II provides a concise summary of current SW tools and projects. The tools reviewed are the Java Expert System Shell (Jess) Rule Engine, the Protégé ontology editor and its plug-ins, the SemanticWorks ontology editor, the RacerPro reasoner and the Pellet reasoner. Current project topics include a discussion of delivering Valued

Information at the Right Time (VIRT), and an overview of the Cyc and SUMO upper-ontology projects.

Chapter III provides a detailed problem description. The chapter focuses on the nature of the problem, the candidate solutions and the approach chosen to address the problem.

Chapter IV introduces the Tactical Assessment Markup Language (TAML) by providing its purpose and goals, a description of the language and an overview of the schema design.

Chapter V introduces the Semantic Web (SW) and its technologies. The chapter reviews the SW layers and the languages used to describe data at each layer. The languages reviewed include the Extensible Markup Language (XML), the Resource Description Framework (RDF), the Web Ontology Language (OWL), and the Semantic Web Rule Language (SWRL).

Chapter VI examines the advantages and disadvantages of RDF/XML. It also describes the steps taken to serialize TAML documents as RDF/XML, as well as a comparison of the XQuery language for XML and the SPARQL RDF Query Language (SPARQL).

Chapter VII describes the design and development of a TAML ontology focused on modeling contacts. The goal of the ontology is to classify contacts as friendly, hostile, neutral, suspicious, or unknown. The chapter also explores the use of a reasoner in testing the ontology and providing automated classification of contacts.

Chapter VIII describes the process of adding SWRL rules to the TAML Contact Classification Ontology in order to increase the semantic model of the Undersea Warfare (USW) domain. The chapter explores the advantages of adding SWRL rules and the current limitations of the language.

Chapter IX discusses the results of the thesis by assessing the ability of the example ontologies to fulfill the requirements of the use cases.

Chapter X discusses the conclusions of this thesis based on the work accomplished. The chapter includes a summary of the successes and failures experienced as well as recommendations for future work.

Appendix A provides a description of serializing Tactical Assessment Markup Language (TAML) documents as RDF/XML files. The appendix also contains the source of the example file, TrafalgarExample.rdf.

Appendix B describes the TAML Contact Classification Ontology and contains the source file, TAMLContactClassification.owl.

Appendix C describes the addition of SWRL rules to the TAML Contact Classification Ontology. The appendix also contains the source file for the rule-enhanced ontology.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

> Over time, the knowledge engineering field will have an impact on all areas of human activity....knowledge engineering will catalyze a global effort to collect, codify, exchange, and exploit applicable forms of human knowledge.
>
> -- Hayes-Roth, Waterman, and Lenat, *Building Expert Systems*,1983

A. INTRODUCTION

This chapter provides needed background by summarizing the related work that is referenced in this thesis. The chapter begins by discussing the Artificial Intelligence (AI) paradigms that form the history of AI technologies such as the Semantic Web. The important AI paradigms discussed are Expert Systems, Case-Based Reasoning (CBR) and Knowledge Representation (KR). The SW is a set of languages for providing KR on the Web. This chapter also discusses several tools available for building SW applications. The tools described are Protégé, SemanticWorks, RacerPro and Pellet. Protégé provides several additional plug-ins for developing ontologies using OWL which are also described. Finally the chapter provides an overview of several current SW projects and applications including Valued Information at the Right Time (VIRT), the Suggested Upper Merged Ontology (SUMO) and Cyc.

B. ARTIFICIAL INTELLIGENCE (AI) PARDIGMS

AI is the science and engineering of creating intelligent computer programs and machines. Intelligence is hard to define but in the context of computing it refers to the computational aspect of achieving goals (McCarthy, 2004). Several approaches to AI using computers have been explored with varying success since Alan Turing introduced the "Turing Test" in 1950. A few general approaches are discussed in the following sections.

1. Expert Systems

An expert system is a class of computer programs made up of rules that characterize relationships, provide an analysis of a problem(s), and recommend a course of action to solve a particular class of problems. The idea for expert systems originated in the 1970s and commercial systems were fielded in the 1980s. The problems solved by expert systems are the same problems that are normally solved by human experts such as medical problems. The systems are designed by asking human experts to define "rules of thumb" which are then encoded as rules in the system (Wikipedia, 2006c).

Expert systems are specialized programs developed to solve problems in a particular domain. These systems solve problems where there are multiple possible answers and the correct answer depends on real-world uncertainties (Wikipedia, 2006c). Some of the methods used to solve these problems use imperfect methods such as fuzzy logic. However, expert systems provide the advantage of separating the problem-solving program from the problem domain. This approach increases the adaptability of the programs when the assumptions of the domain change.

a. Rule-Based Expert Systems

Some expert systems are implemented through rule-based programming which is declarative. A declarative program typically expresses rules that tell a computer what to do without necessarily specifying how. Declarative programs are executed by a run-time system that interprets and invokes rules based on input information (Friedman-Hill, 2003). A set of rules written by a developer are typically used by a rule engine to fill in missing information and solve a problem. Rules themselves are made up of premises and actions or conclusions. Whenever a premise is evaluated as true, the rule engine performs the associated action or makes the associated conclusion.

The architecture of a rule-based system includes an inference engine, a rule base and a working memory as shown in Figure 1. The inference engine is the central part of the rule-based system (Friedman-Hill, 2003). It controls the process of applying rules to obtain outputs. The inference engine contains a pattern matcher, an

agenda and an execution engine. The pattern matcher compares the rules to the facts in the working memory to determine the agenda which is the ordered list of rules to execute. The execution order of the rules is determined by the engine. The execution engine is the component that executes the rules and returns the outputs to the system. The rule base is the component that stores all of the rules known to the system. This is the part that is developed by a rule author working with a domain expert for the specific topic of interest. The working memory or fact base holds all of the information or facts that the system is working on, including the premises and conclusions of each rule. The working memory is indexed to improve efficiency (Friedman-Hill, 2003).

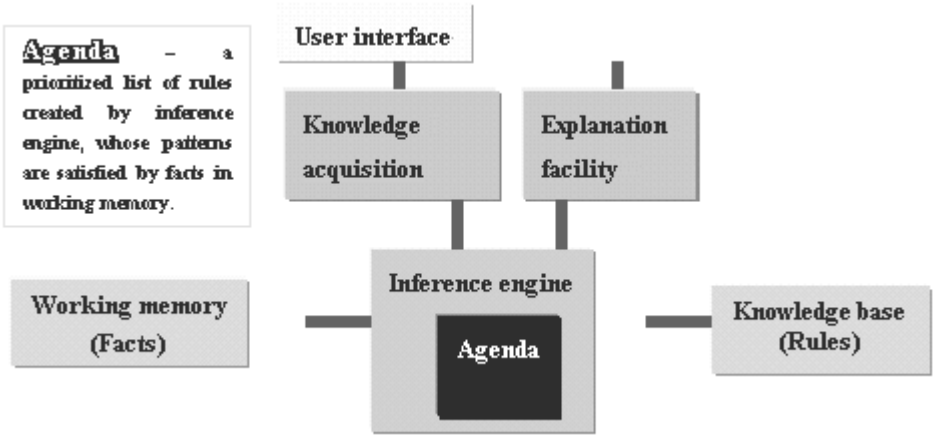


Figure 1. An illustration of the rule-based system architecture [From Ref. University of Tokyo Center of Collaborative Research 2006].

b. Java Expert System Shell (Jess) Rule-Based System

The Java Expert System Shell (Jess) is an expert rule-based system developed by Sandia National Laboratories in the 1990s. Jess is derived from the C Language Integrated Production System (CLIPS) (Wikipedia, 2006a) which is an expert system tool that was developed in 1984. CLIPS is a complete environment for constructing object-based expert systems like Jess (Riley, 2004). Jess is written in the Java programming language and works with any Java 2 virtual machine. The Jess engine implements the Rete algorithm for pattern matching. The Rete algorithm trades memory space for speed, so Jess typically uses a lot of memory. However, the engine is capable of firing more than 80,000 rules per second (Friedman-Hill, 2003). Jess is a powerful and efficient environment for processing rules for a variety of applications.

2. Case-Based Reasoning (CBR)

CBR is a problem solving technique employed by humans that has been formalized for machines. CBR is an approach to solving new problems based on the available solution of similar problems (Morris, 1994). For example, doctors sometimes use the diagnoses of one patient to diagnose a patient with similar symptoms. The goal of AI is to provide machines with reasoning capabilities similar to human reasoning. CBR is an attempt to do that by formalizing the human approach which uses past experiences to solve current problems. The idea for formalizing human CBR for use in machines originated with Roger Schank at Yale University in the early 1980s.

CBR has been formalized for machines as a four-step process as illustrated by Figure 2. When a machine is given a problem to solve, it retrieves a similar problem or case from memory, reuses the solution from the previous case, tests the solution with the current problem and revises it if necessary, storing the problem and solution in memory for future reference.

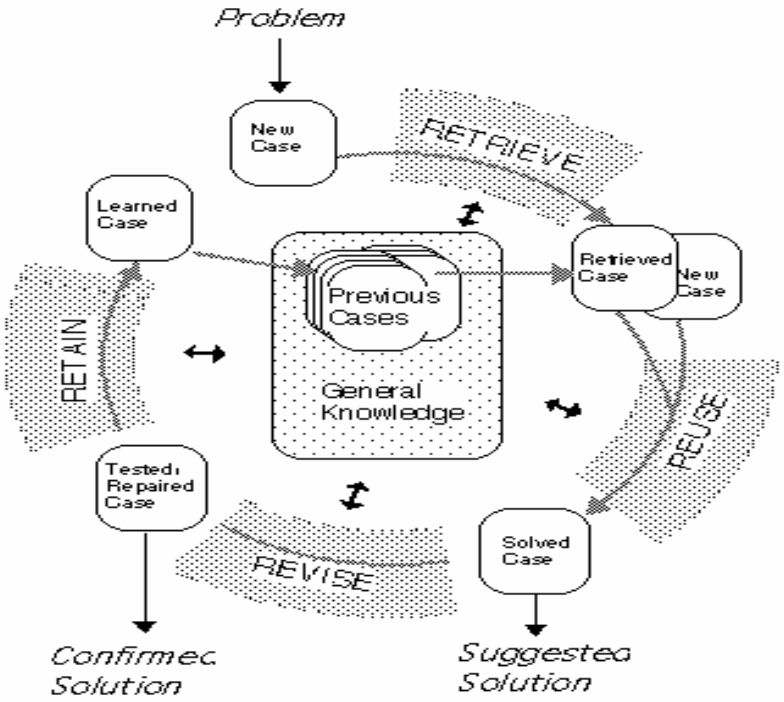


Figure 2. An illustration of the formalized four-step process for Case-Based Reasoning (CBR) [From Ref. Aamodt 1994].

A CBR system consists of a database of past cases and their solutions, rules for measuring the similarity between cases, and rules for adapting past solutions to new cases. Case or problem attributes such as problem type are used to compare cases. CBR systems learn by solving new problems and placing the matched problem and solution into the database for future use. Unlike the expert systems described above, CBR systems focus on solving problems in unusual or exceptional situations. CBR systems model exceptional situations and use the models to solve similar situations. Thus CBR is most useful in domains where there is a high ratio of exceptional cases (Morris, 1994). CBR is not useful when the database of cases does not contain a case sufficiently similar to the current case being solved. CBR systems are relatively easy to develop, since domain experts usually remember exceptional cases, but users of CBR systems must be aware of their limits (Morris, 1994).

3. Knowledge Representation (KR)

An important principle of AI is that intelligent behavior by machines can be achieved through the processing of symbols which represent increments of knowledge (Alesso and Smith, 2005). Knowledge Representation (KR) languages are used to represent knowledge with well-defined syntax and semantics so machines are able to manipulate the knowledge and deduce new knowledge. In the KR approach, domain specific knowledge is separated from problem-solving and reasoning procedures. Larger databases for storing information and faster computers are making KR a more feasible approach to AI. KR systems and languages are used to symbolize knowledge about the world, such as definitions of entities and the laws that govern their behavior, as shown in Figure 3.

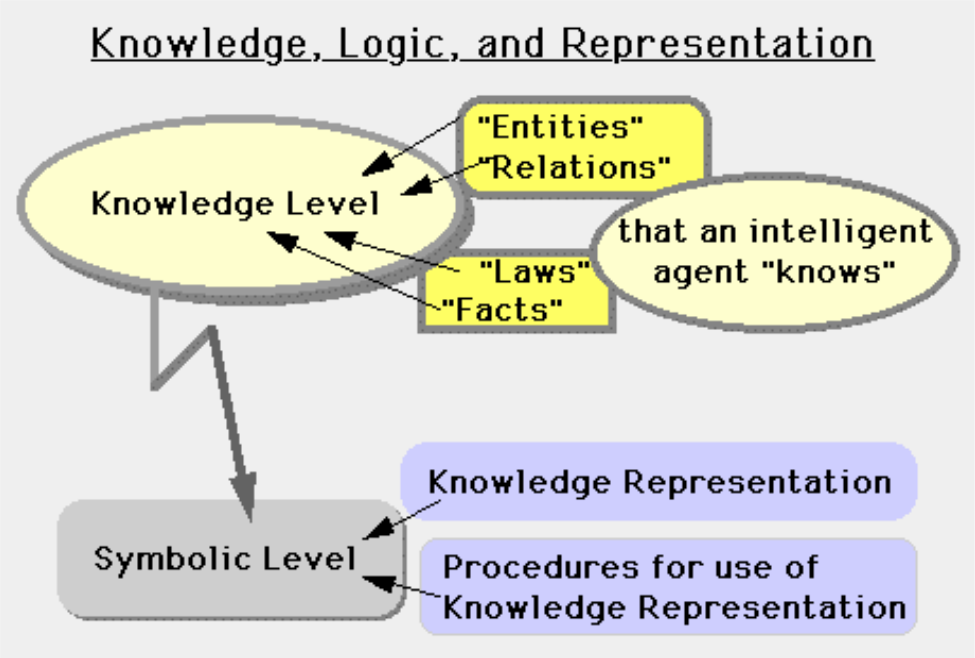


Figure 3. Knowledge Representation (KR) consists of defining knowledge about entities and the laws that govern them at a symbolic level [From Ref. Schmidt 2005].

Three well-established techniques exist for KR and inference: frames and semantic networks, logic-based approaches, and rule-based systems. Frames and semantic network systems capture information about objects and concepts into a hierarchy. The principle of inheritance can then be used to infer the properties of a subclass from the properties of a higher class. Inheritance and the identification of objects with similar properties are two types of reasoning done with frame systems. This technique is limited and is usually used to represent simple systems.

Logic-based knowledge representation techniques use logical formulas to represent complex relationships among objects and concepts using well-defined syntax and semantics. When knowledge is represented as logical formulas, logical theorems can be applied to derive new information. However, it is sometimes difficult to precisely define logical relationships and the reasoning is sometimes inefficient.

Rule-based approaches allow for the representation of knowledge using sets of IF-THEN conditions. This approach is less formal and more flexible than the logic-based

approach. The key component of each of these techniques is the separation of the knowledge-representation from the reasoning procedure (Alesso and Smith, 2005).

Description Logic (DL) languages are a family of logic-based KR languages first developed in the 1980s. They are used to formally define the terminology of a domain (Wikipedia, 2006b). The semantics of the definitions written in DL languages can be translated to FOL constructs. Therefore, reasoners can be used to infer information from the data formally defined by DL languages.

In the past, KR systems have been centralized, essentially requiring adoption of a common terminology for the world (Berners-Lee, 1999). Adoption of centralized systems is often limited. In addition, increasing the scope of these systems becomes harder with increased size. The SW offers a DL language, OWL and a rule language, SWRL which are more flexible than past KR languages because domain experts are not limited in the terminology available to them for describing their domain.

C. SEMANTIC WEB (SW) TOOLS

A variety of tools exist to aid the process of building, editing, and accessing SW applications. These tools remove some of the burden of developing SW ontologies and applications from the developer. The tools range from application program interfaces (APIs) to full graphical user interfaces (GUIs).

1. Protégé

Protégé is an open-source frame-based ontology editor developed by Stanford Medical Informatics (Gennari and Musen, 2002). Protégé is licensed under the Mozilla Public License Version 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html). The public license allows developers to use Protégé within commercial products but restricts modifications on the source code. Source code modifications must be clearly stated in file headers and licensed under the Mozilla Public License. However, plug-ins developed for Protégé do not have to be licensed under the Mozilla Public License.

The development of Protégé has been driven by the need for ontologies in the biomedical field, but the methodologies used and the tool itself are domain independent.

The goal of Protégé is to minimize the need for a knowledge engineer during ontology development, so that domain experts themselves become the primary ontology developers. The Protégé tool makes ontology development more intuitive through GUIs, so that domain experts with first-hand knowledge of the world being modeled are able to build the ontology directly, without having to be knowledge engineers themselves. The likelihood of errors is thereby reduced and ontology building is streamlined by allowing domain experts to build, test and utilize knowledge bases (Sandahl, 1994).

Protégé has been developed using a build-and-test methodology and has evolved through four major builds (Gennari and Musen, 2002). Protégé I was developed by Mark Musen in 1987 as a tool for building knowledge-acquisition applications for medical planning. Protégé I assumed that knowledge is acquired in stages and each stage makes it easier to acquire knowledge in the next stage. The knowledge bases built by Protégé I were not general purpose, so they are not reusable. Protégé II focused on making the tool more general by creating the idea of reusable problem solving methods (PSMs). Protégé II allowed developers to create PSMs separate from the knowledge base. A frame-based ontology described the particular domain of interest. A more general PSM (such as a planning algorithm) was then applied to the ontology to acquire further knowledge. PSMs themselves can thus be considered reusable components of the knowledge base. Protégé II was implemented to run on the NeXTStep operating system which limited its user community. The third generation build was aimed at increasing the user base of Protégé by implementing the tool on the Windows operating system. The user community and feedback grew once Protégé/Win was released for install on Windows machines.

Protégé-2000 is the current build of the tool which implements a plug-in architecture in order to increase the extensibility of the tool as shown in Figure 4. The plug-in architecture makes Protégé easy to extend. Developers have the ability to customize Protégé by building tab, slot, backend, project and import/export plug-ins (Gennari and Musen, 2002). Numerous plug-ins have been written to provide a variety of functionality.

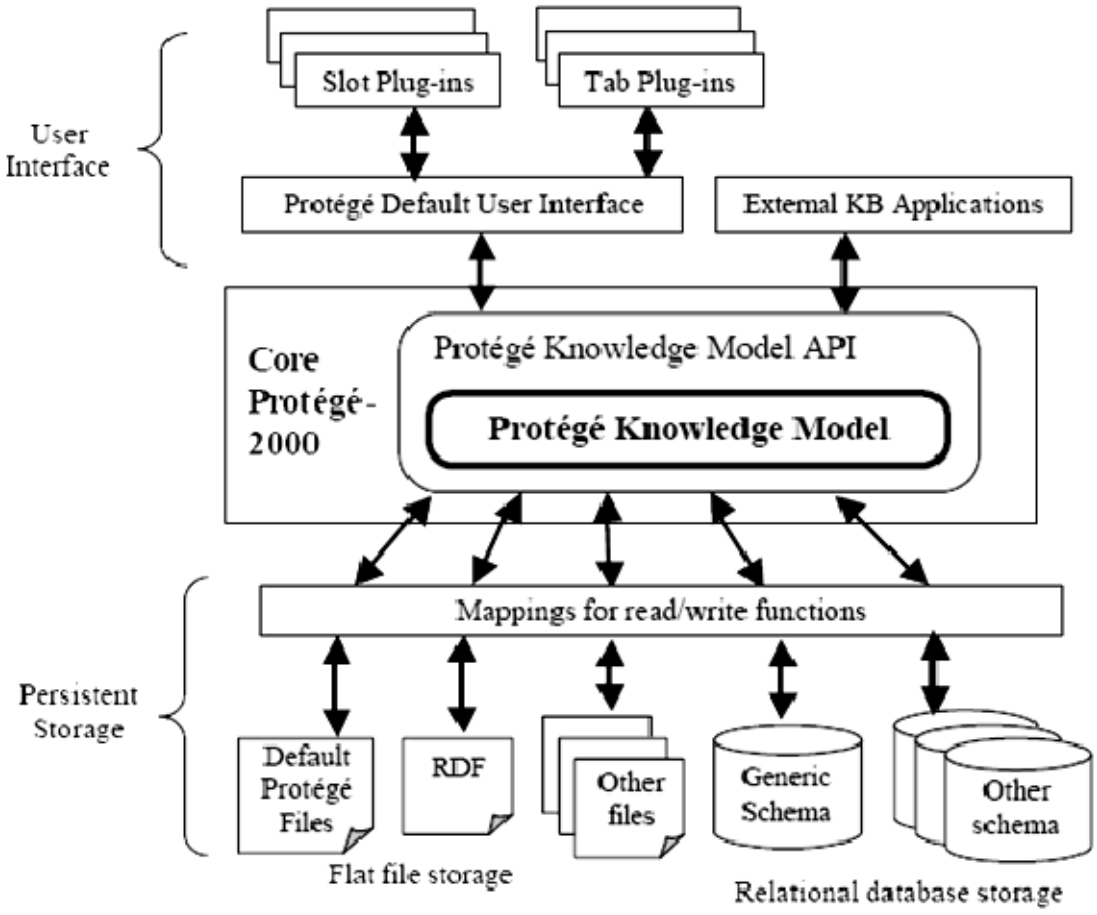


Figure 4. The Protege-2000 architecture [From Ref. Gennari 2002].

Tab plug-ins customize the functions of Protégé by allowing developers to create wholly new Protégé views. For example, the SWRL tab is a plug-in which allows developers to add SWRL rules to their ontology. Similarly, a slot plug-in is used to change the form view of Protégé. For example, developers are able to customize the form that is displayed for entering instances. Backend plug-ins are used to modify the ontology storage format. For example, a database backend plug-in allows developers to save their ontology to a database thereby allowing them to save large numbers of classes, slots, instances and properties. Import/export plug-ins create translations to and from different KR and data formats for display in Protégé. For example, an XML export plug-in translates frame ontology information into an XML schema for storage. Project plug-ins are used to create custom applications which access and modify the ontology stored in

Protégé by using the Protégé API. The plug-in architecture of Protégé thus allows software developers to extend and customize the functionality of the tool to fit their particular domain and application needs.

Protégé is a powerful tool that provides a framework for creating knowledge-based applications. The tool is flexible and extensible, so its uses are quite broad. Protégé provides a storage mechanism for ontologies and an API for accessing them. The developer is free to use the tool and the API to suit his particular needs.

a. OWL Plug-in

The OWL plug-in is a SW extension of Protégé which is used to edit OWL ontologies, access Description Logic (DL) reasoners, and to acquire instances for semantic markup (Knublauch, Fergerson, Noy and Musen, 2004). The plug-in is extensible in order to support the development of a wide variety of SW applications. It provides intelligent assistance in developing and editing ontologies. The OWL language is based on RDF which is a difficult language to write and edit manually. The OWL plug-in provides a user-friendly graphical display which is used to create OWL ontologies in order to facilitate more rapid and accurate development. The OWL plug-in user display is shown in Figure 5. The plug-in may be easily customized to a particular domain, so a developer is able to create any application. The OWL plug-in benefits from Protégé's large user community, flexible architecture and reusable components.

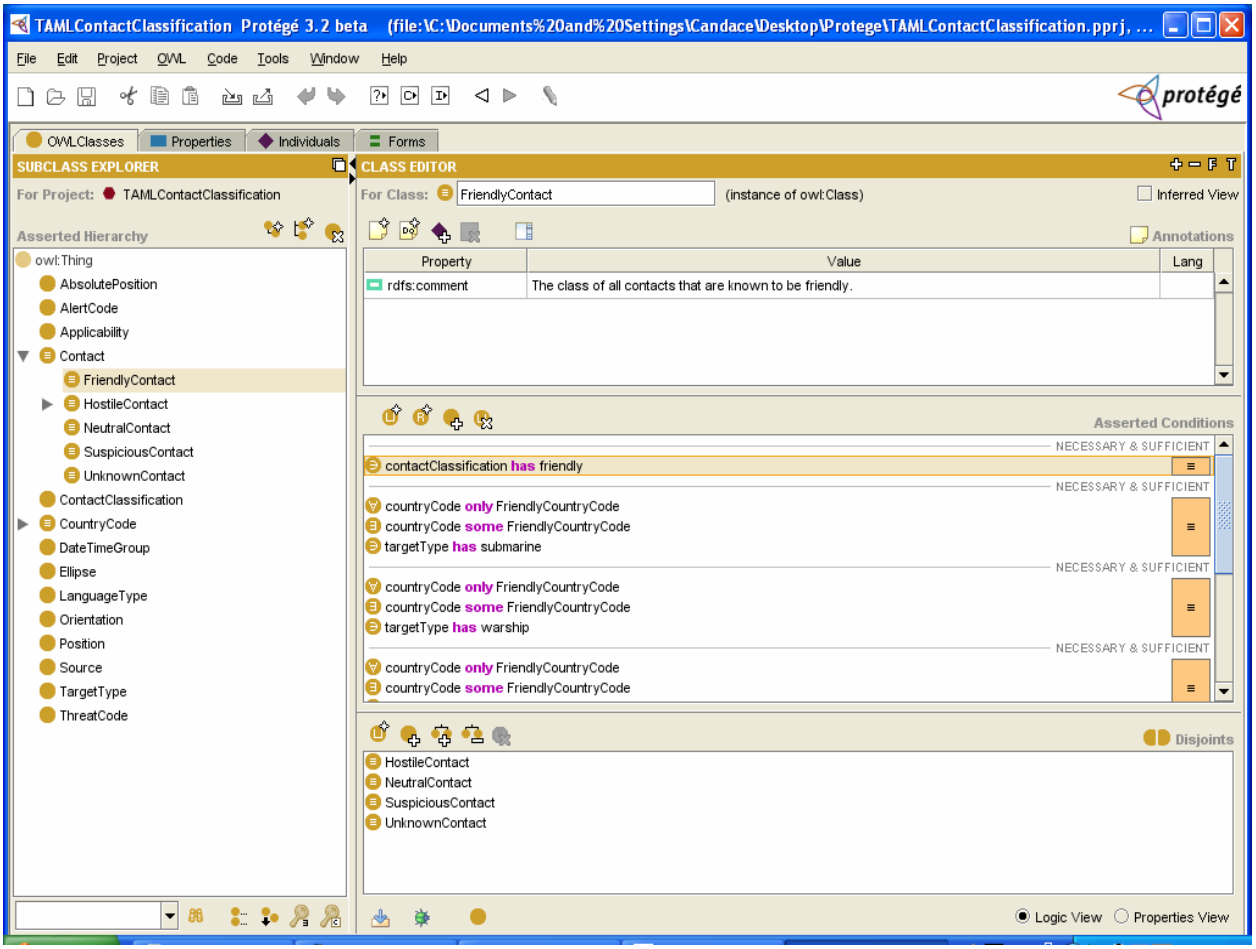


Figure 5. Screenshot of Protégé's OWL GUI interface.

The OWL plug-in architecture is made up of the OWL mappings to Protégé, an OWL API, and the user interface. The plug-in extends the Protégé model by mapping OWL constructs to existing frame-based constructs that make up Protégé. OWL components such as disjoint classes are a simple extension of Protégé's metamodel so most of these components are easy to map. However, there are significant semantic differences between OWL and frames which are more difficult to map (Knublauch, Fergerson, Noy, and Musen, 2004). The mappings are provided as part of the plug-in design and are transparent to the user.

The OWL plug-in also provides an API to access and edit OWL ontologies. The API is an extension of the core Protégé API with custom Java classes for the various OWL components. The OWL GUI provides the developer with an easy way

to edit OWL ontologies without dealing with the OWL syntax. The GUI provides five tabs for editing different aspects of the ontologies such as instances, classes, metadata, forms and properties. The GUI may be customized by writing tab or slot plug-ins. Providing an API also enables other applications to integrate such functionality.

The OWL expression editor uses traditional logic symbols to represent OWL elements as shown in Table 1. These symbols are easy to read and efficient to enter (Knublauch, Fergerson, Noy and Musen, 2004). The plug-in also provides the English prose meaning of an expression as a "tool tip" when the user's mouse is moved over the expression.

OWL Element	**Symbol**	**Key**	**Example expression in Protégé**
owl:allValuesFrom	∀	*	∀ *hasChildren* Female
owl:someValuesFrom	∃	?	∃ *hasHabitat* University
owl:hasValue	ə	$	*hasGender* ə male
owl:minCardinality	≥	>	*hasChildren* ≥ 1 (at least one value)
owl:maxCardinality	≤	<	*hasDegree* ≤ 5 (at most five values)
owl:cardinality	=	=	*hasGender* = 1 (exactly one value)
owl:intersectionOf	⋂	&	Student ⋂ Parent
owl:unionOf	⋃	|	Male ⋃ Female
owl:complementOf	¬	!	¬ Parent
owl:oneOf	{…}	{ }	{yellow green red}

Table 1. Table of the logical symbols used to represent OWL elements in the OWL plug-in [From Ref. Knublauch 2004].

The OWL plug-in provides an extensible user-friendly tool for developing, editing and maintaining OWL ontologies. The extensibility enables developers to customize the tool for particular domains. The user interfaces provide an easy and efficient way to create classes, properties and class definitions. The API allows developers to access and manipulate the ontology from an application. Protégé provides a well-documented and well-supported software code base for the OWL plug-in.

b. Visualization Plug-ins

Knowledge bases are often large and complex, making maintenance difficult. It is often helpful to visualize them at different levels of abstraction (Storey, Musen, and Silva, 2002). There are multiple visualization plug-ins for Protégé-OWL that allow developers to graphically view the models they build. The OWLViz tab and Jambalaya tab are particularly useful for visualizing class hierarchies and relationships between properties.

OWLViz is a tab plug-in for Protégé that enables the class hierarchies in an OWL ontology to be incrementally navigated (OWLViz Guide, 2004). It was created by Michael Horridge and is licensed as open source under the Lesser GNU Public license (LGPL) (http://www.gnu.org/licenses/lgpl.html). The plug-in uses algorithms from GraphViz software developed by AT&T. GraphViz must be installed along with the OWLViz plug-in to use the tool. The tool provides a graphical view of ontologies as illustrated in Figure 6.

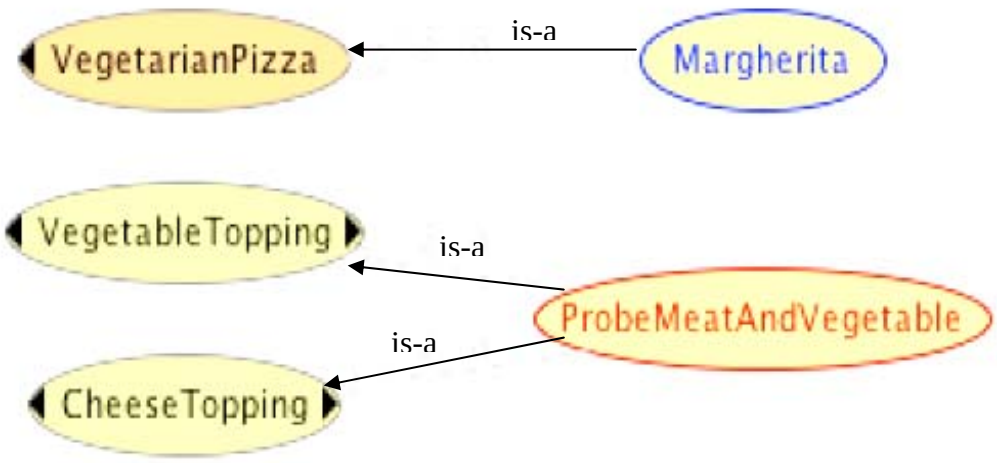


Figure 6. A graphical view of *VegetarianPizza* using the OWLViz plug-in [From Ref. OWLViz Guide 2004].

OWLViz allows a developer to see either a tree view of the asserted model or the inferred model of an ontology. The developer may use these two views to compare the original ontology to the provided ontology after a reasoning engine is run. The plug-

in is integrated into the OWL plug-in and uses the same color scheme as Protégé to make visualization more natural for Protégé users. For example, primitive classes are yellow, defined classes are orange, and inconsistent classes are red as shown in Figure 6. The class hierarchy may be incrementally navigated by using the expansion arrows to expand or collapse classes. This allows a developer to focus on one part of the ontology without the distraction of other elements. Graphs may be exported as image files for use in ontology documentation. OWLViz is a visualization tool designed to remove some of the complexity of viewing a text-based hierarchy for large ontologies.

Jambalaya is another visualization plug-in for Protégé that provides more powerful functionality than OWLViz for focusing on certain aspects of an ontology. The Jambalaya plug-in uses the Simple Hierarchical Multi-Perspective (SHriMP) visualization technique to enhance how users browse and interact with the ontology (Storey, Musen and Silva, 2002). SHriMP uses a nested graph to visualize the class hierarchy in the ontology. The tool allows users to change views in order to explore multiple perspectives from different levels of abstraction. Jambalaya is the plug-in that integrates SHriMP with Protégé. The plug-in provides the option to view classes in a tree, radial, or spring layout. Figure 7 shows a screenshot of the class tree view. Developers can change the level of abstraction by focusing on a domain/range, class tree, or nested class view. The tool enables developers to zoom in on specific classes of an ontology and export the views to an image format.

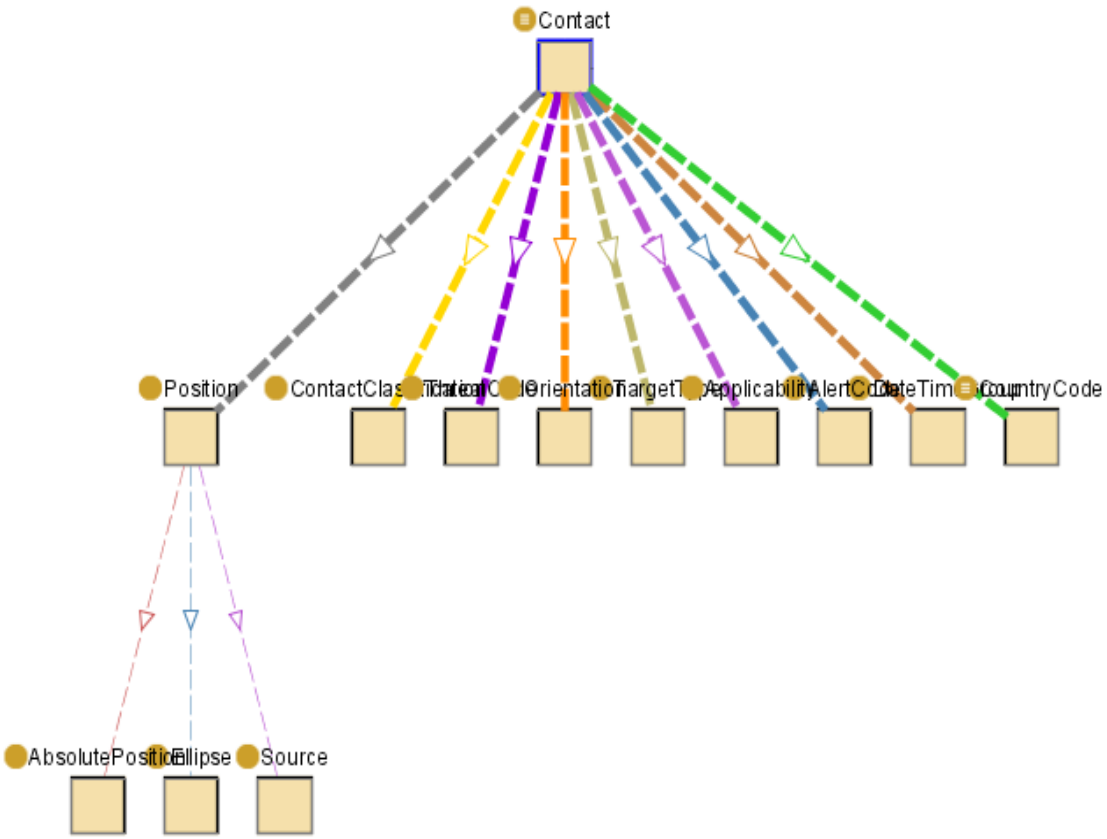


Figure 7. A graphical view of the *Contact* class from the Jambalaya plug-in.

Jambalaya provides a powerful visualization tools which allows a developer to choose from multiple views and abstractions, but the tool can be difficult to learn and use effectively. User guides and example walkthroughs are available online (http://www.thechiselgroup.org/jambalaya).

c. SWRL Editor Plug-in

The Semantic Web Rule Language (SWRL) editor is another extension of the OWL plug-in that provides a user interface for writing SWRL rules (http://protege.stanford.edu/plug-ins/owl/swrl). The editor is used to create new SWRL

rules, edit existing SWRL rules, or read SWRL rules. The editor is included with the Protégé-OWL full install and is accessed through the SWRLTab plug-in as shown in Figure 8.

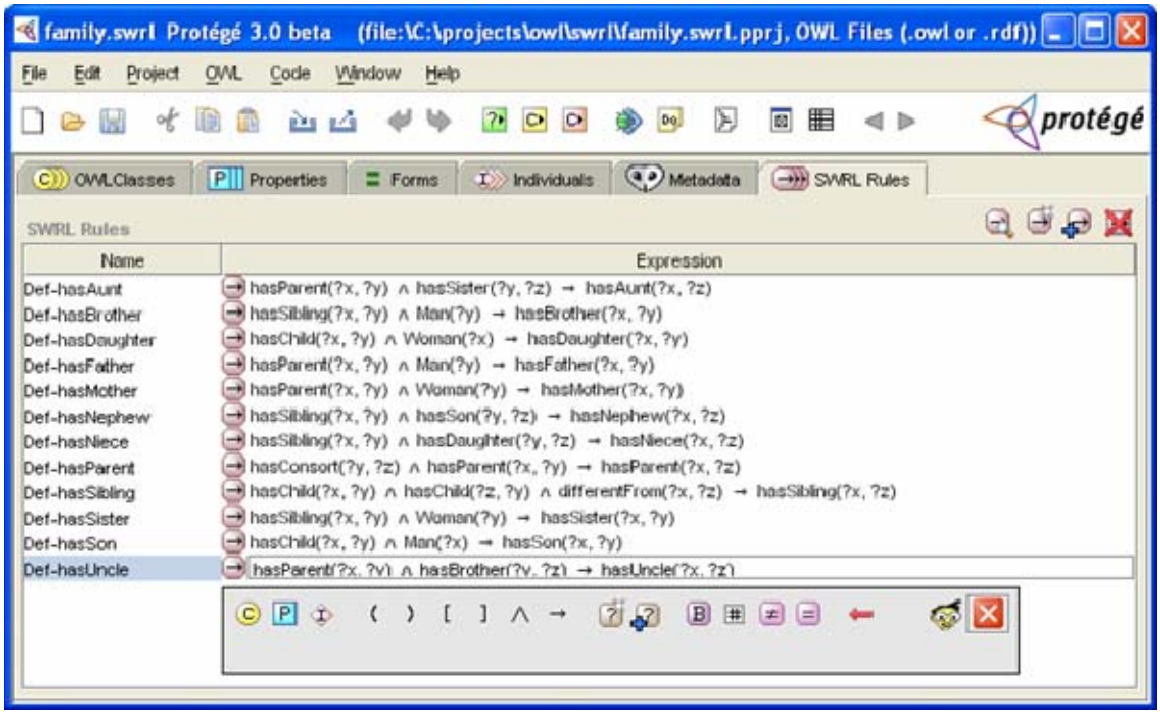


Figure 8. A screenshot of the SWRLTab plug-in [From Ref. http://protege.stanford.edu/plug-ins/owl/swrl].

The SWRL ontology must be imported from the Web in order for the SWRLTab to be activated. The SWRL ontology includes all of the built-in classes used to write SWRL rules. The SWRLTab provides a SWRL editor for developers in order to reduce the work involved in writing rules and to reduce errors in the rules. The editor includes two editing modes: in-table editing or multi-line editing. A rule may be edited in the table by left-clicking on the rule. A rule editor is then displayed as a separate popup as shown in Figure 8. The editor prevents erroneous rules from being entered by keeping the rule editor open until a correct rule is written. The multi-line editor changes the user interface for writing rules by opening a separate window for the current rule. The editor includes icons that allow developers to insert elements from their OWL ontology into the rule. Using this feature reduces errors caused by mistyping class or property names.

2. SemanticWorks

SemanticWorks is a commercial ontology editing tool available from Altova (http://www.altova.com/products_semanticworks.html). The tool provides graphical editing for RDF and OWL files as shown in Figure 9. The drag-and-drop nature of the tool makes ontology creation and editing easy. The tool also provides full syntax checking to ensure that produced files conform to the RDF/XML specification.

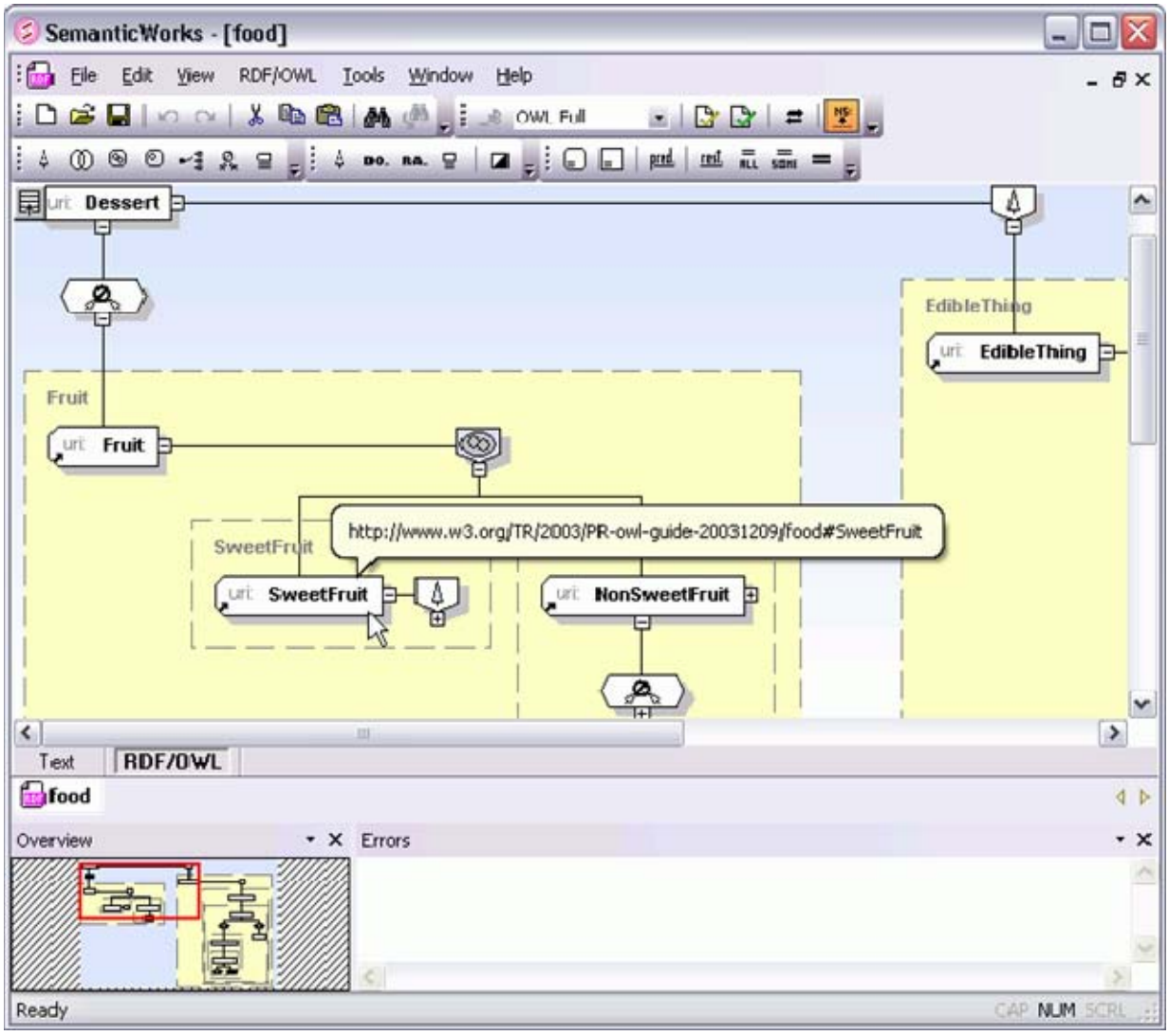


Figure 9. Screenshot of the Altova SemanticWorks graph editing tool [From Ref. http://www.altova.com/products_semanticworks.html].

SemanticWorks allows developers to design ontologies using any of the OWL subset languages. Once a subset language is chosen, the tool removes any functionality not included in the chosen subset. For example, if an ontology is declared as OWL-Lite then the disjoint and quantifier functionalities are unavailable during editing. The selection of OWL language subset used to define an ontology is easily changed via a drop-down language-selection menu. This functionality allows a developer to begin with an OWL-Lite ontology and only increase the expressivity if needed.

SemanticWorks also provides built-in consistency checking. If two class definitions contradict each other, an error message is displayed. The error message is linked to the location of the error, so a developer may easily address the problem. An option to view the source text of the ontology is also available, so that a developer may directly edit the text file.

3. RacerPro Version 1.9

RacerPro is an OWL reasoning engine which was first available in 2002 (http://www.sts.tu-harburg.de/~r.f.moeller/racer). RacerPro is a SW information repository with optimized querying and inferring capability (RacerPro User Guide, 2005). The reasoner has an interface developed by the Description Logic (DL) Implementation group (DIG) which it uses to communicate with and perform reasoning on Protégé ontologies as shown in Figure 10. The reasoner is able to handle large sets of data descriptions.

RacerPro processes OWL-Lite and OWL-DL ontologies, but does not support user defined XML datatypes. The reasoner provides consistency checking, instance classification, resource synonym discovery, and OWL Query Language (OWL-QL) resolution (RacerPro User Guide, 2005). Consistency checking is a function used to determine whether or not the class definitions in an ontology contradict each other. For example, if an ontology defines two classes as disjoint but states that instance A belongs to both classes, then RacerPro returns an inconsistency error. Consistency checking helps developers ensure their domain model makes sense. Instance classification is the function that provides the inferencing capability of the reasoner. RacerPro processes class

definitions and determines under which subclass a specific instance belongs. For example, if a *CheesyPizza* class is defined as a class whose members have only cheese as toppings and instance A has one topping that is cheese and no others, then the reasoner infers that A is a *CheesyPizza*. RacerPro also checks if classes are synonyms. For example, if two classes represent the same set of members then these two classes are synonyms; therefore, they are defined as OWL equivalent classes by RacerPro.

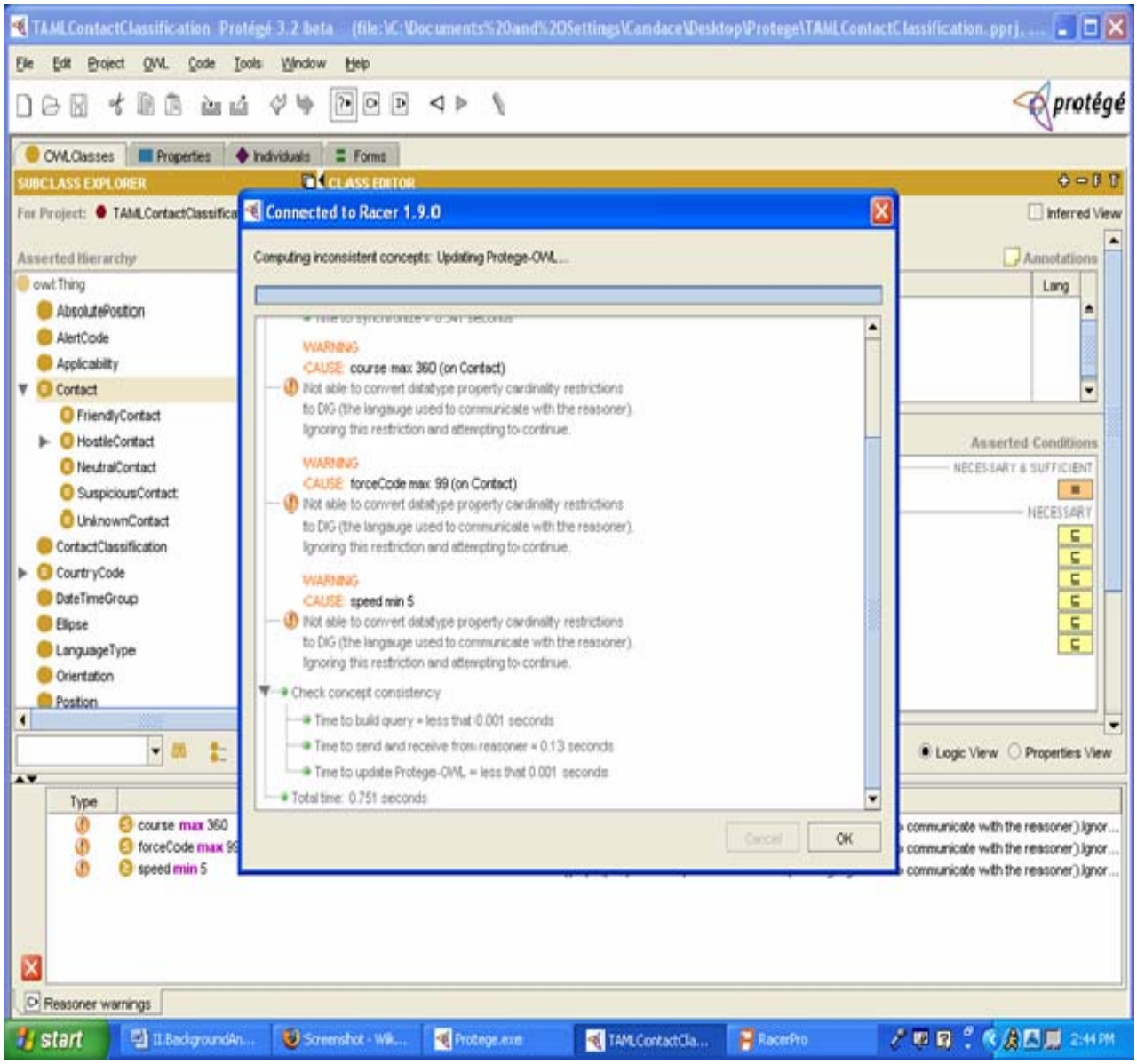


Figure 10. Screenshot of RacerPro running inside the Protégé ontology editor.

The reasoner also supports the OWL-QL. OWL-QL is a protocol for answering query dialogues for OWL ontologies (Fikes, Hayes and Horrocks, 2003) . OWL-QL is a formal language which specifies the semantic relationships between a query, the answer, and the knowledge base used to answer the query. The protocol is used to answer queries where answers are derived from multiple ontologies. RacerPro's support for OWL-QL is useful for queries with multiple answers where each answer resides in a different ontology.

RacerPro is commercially available from Franz Inc (http://www.franz.com/products). It provides many functions to support ontology development and ontology applications such as consistency checking and inferencing capabilities.

4. Pellet Reasoner

Pellet is an Java based open source OWL-DL reasoner developed by the Maryland Information and Network Dynamics Lab Semantic Web Agents Project (Mindswap). Pellet provides the ability to check ontology consistency, classify taxonomies, check entailments and answer a subset of RDQL queries. Pellet was developed using tableaux algorithms developed for DL languages. It supports the full expressivity of OWL-DL (http://www.mindswap.org/2003/pellet).

Pellet provides analysis and repair for OWL-DL ontologies. The OWL-DL languages imposes a number of restrictions on RDF graphs in order to ensure computational tractability. Ensuring that an ontology meets this restriction can be difficult for developers, so Pellet incorporates heuristics to detect OWL-Full constructs within an OWL-DL ontology and repair them.

Query optimization algorithms are implemented in the Pellet reasoner in order to ensure querying a knowledge base is efficient. These algorithms focus on carefully and more efficiently choosing candidates for variable matching. They exploit the dependencies between different variable bindings in order to reduce the total number of satisfiability tests, thus speeding up query processing (http://www.mindswap.org/2003/pellet).

Similar to the RacerPro reasoner, Pellet includes a DIG interface which allows communication between Pellet and Protégé. Pellet also includes an interface to support communication with Jena. Unlike RacerPro, Pellet is an open source reasoner thus it is freely available for any use. Pellet provides similar capabilities as RacerPro but does not yet support any SWRL constructs.

D. SEMANTIC WEB (SW) PROJECTS

1. Valued Information at the Right Time (VIRT)

The goal of command and control and other collaborative distributive systems is to provide the most valuable information to war-fighters, so they are able to make the best possible decisions. Information systems face three major challenges: networks are growing, available data from various sources is increasing, and the time available for decision making is decreasing (Hayes-Roth, 2005). There are more decision makers, more information to filter through, and less time to process information. A system is needed that automates the process of filtering information based on individual user needs and then selectively presents prioritized information to each user at the moment they need it. Current distributive system architectures do not promote the prioritization and filtering of information. VIRT is a proposed architecture for providing valuable information to the right people at the right time as shown in Figure 11.

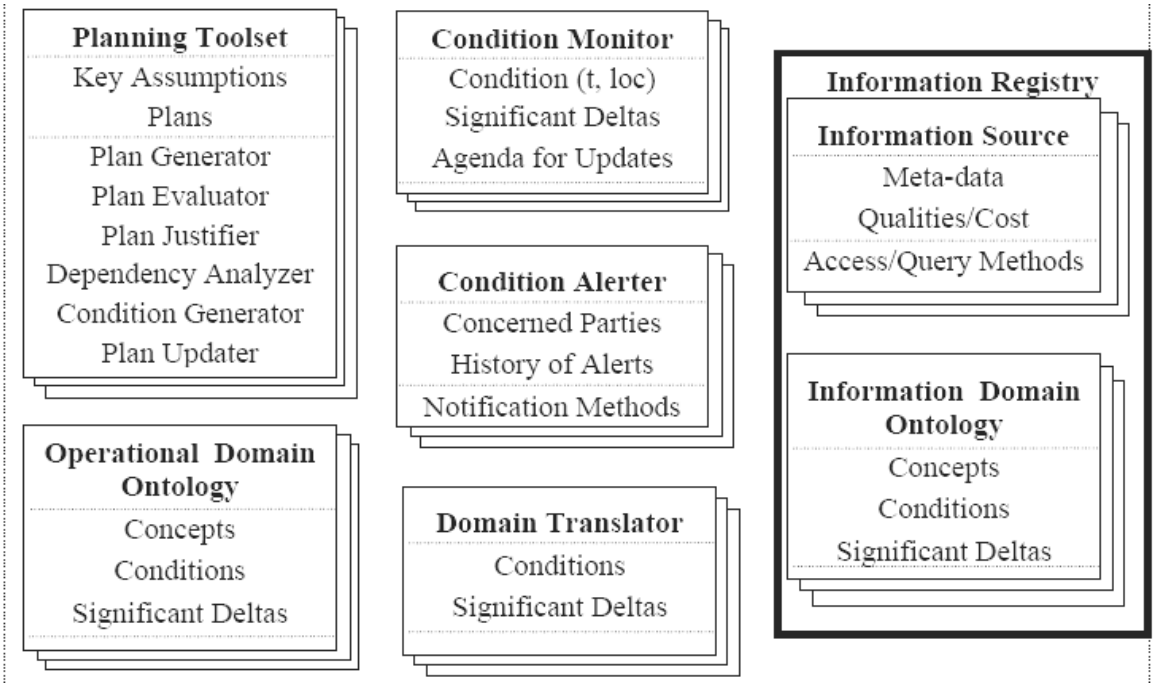


Figure 11. Visualization of the component-based architecture for VIRT [From Ref. Hayes-Roth 2005].

The VIRT architecture calls for a Planning Toolset, a Condition Monitor, a Condition Alerter, Domain and Information Ontologies and a Domain Translator (Hayes-Roth, 2005). The Planning Toolset allows planners to create plans, compare alternatives and justify selections. Similar planning tools already exist, but VIRT adds new capabilities. The Dependency Analyzer determines what variables a plan's outcome relies on, and a Condition Generator translates these variables into conditions that must be monitored. All of the conditions created are checked by the Condition Monitor, and when certain changes occur, the Condition Alerter notifies the users who are affected by the change. The Condition Monitor obtains changing situation data from the sources listed in the Information Registry. Each source provides dynamic information about data of interest and is itself described by pertinent meta data. Each source also provides a method for accessing its data, such as via a query language.

Ontologies are used to characterize the semantics of the data being exchanged in the VIRT system. One ontology specifies the user's domain and another ontology

specifies the information source. The Domain Translator translates variables and changes in data from the Information Ontology to the Domain Ontology using ontology mappings (Hayes-Roth, 2005).

VIRT thus proposes an architecture solution to automating the filtering and prioritizing of information for individual users through the use of ontologies and ontology mapping. The SW provides sufficiently capable tools and languages for creating expressive ontologies about particular domains and mapping them to other ontologies. The SW thus can provide the foundation for building a VIRT application. This challenge remains an important area for future work.

2. Suggested Upper Merged Ontology (SUMO)

SUMO is an upper ontology developed by an IEEE working group that includes collaborators from the fields of engineering, philosophy and informational science. An upper ontology defines concepts that are meta, generic, abstract or philosophical, so they are general enough to address several different domains at a high level. Upper ontologies such as SUMO provide a foundation for domain-specific ontologies by defining general purpose terms and acts (Niles and Pease, 2001).

SUMO was developed by merging publicly available ontology information into one single and comprehensive ontology represented in the Knowledge Interchange Format (KIF) language. KIF is a logically comprehensive KR language for data interchange (http://www-ksl.stanford.edu/knowledge-sharing/kif). KIF is more expressive than OWL but reasoning over the language is undecidable. The first step in the SUMO development was to identify all high-level ontological content that was available for free and translate it into KIF. The next step was to merge the differing semantics into a single, consistent and comprehensive framework (Niles and Pease, 2001). The ontologies were divided into high-level concepts and low-level concepts and then merged together. The high-level overview of SUMO concepts is shown in Figure 12.

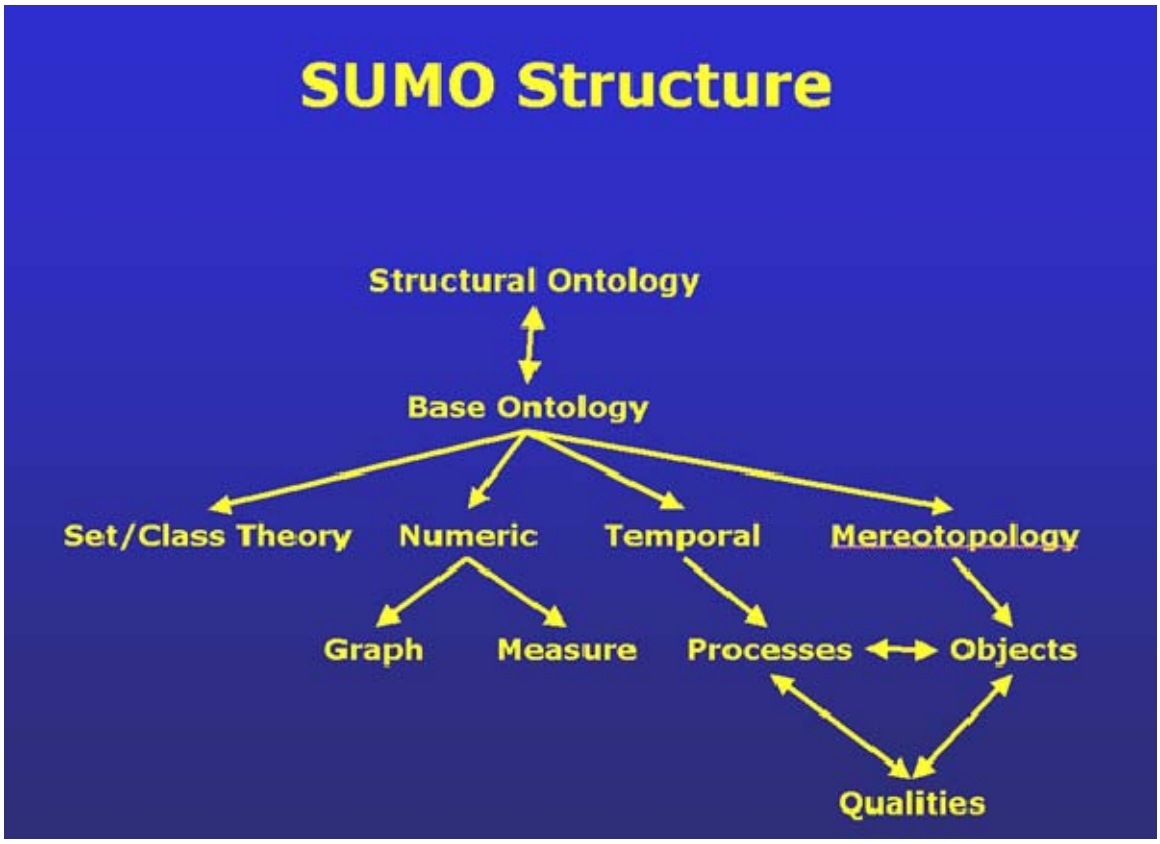


Figure 12. High-level view of the SUMO ontology [From Ref. http://ontolog.cim3.net].

A Mid-Level Ontology (MILO) is currently in development to bridge the abstract constructs of SUMO to the lower-level concepts in domain-specific ontologies. The layered set of ontologies separates levels of knowledge and allows ontology developers to determine the level of abstraction needed for their application.

SUMO is an open-source standards effort by the IEEE. The open-source nature of the ontology increases the user community and ensures the ontology is subjected to extensive peer review to ensure its correctness. The ontology focuses on pragmatic principles, so distinctions of philosophical interests are not represented in the ontology (Niles and Pease, 2001). SUMO has been mapped from KIF to OWL for use in SW applications. However, since OWL is less expressive than KIF, the ontology loses information in the translation. The resulting OWL translation may be less expressive but on the other hand can be imported into OWL ontologies for use on the SW. Furthermore, the resulting OWL ontology is more broadly interoperable and may have better defined

bounds and computational complexity, permitting usage with better expectations for tractability and completability.

SUMO provides a standard upper-level or abstract view of common concepts. It may be used as a translation point between domain-specific ontologies. If mappings to and from SUMO are provided from several domain-specific ontologies, then interoperability is achievable. For example, one developer may map an undersea warfare tactical ontology to SUMO and another developer may map an air warfare tactical ontology to SUMO (Sikora, 2006). If the developers want to map the USW ontology to the air warfare ontology, then each maps their respective ontologies to SUMO which then provides a correspondence. The mappings are still a challenge, but they offer a solution to semantic interoperability.

3. Cyc

Cyc is a multi-contextual knowledge base and inference engine (http://www.cyc.com/cyc/technology/whatiscyc). The development of the Cyc knowledge base has been ongoing since 1984, and the knowledge base contained 400,000 assertions in 1995 (Whitten, 1995). Cyc is an attempt to create a knowledge base that contains a foundation of basic "common sense" knowledge about the world. Cyc includes a knowledge base, an inference engine, the CycL representation language, a natural language processing subsystem, an integration bus, and developer toolsets as shown in Figure 13. Portions of Cyc are available as open-source, but large portions of the knowledge base are proprietary and licensed by Cycorp.

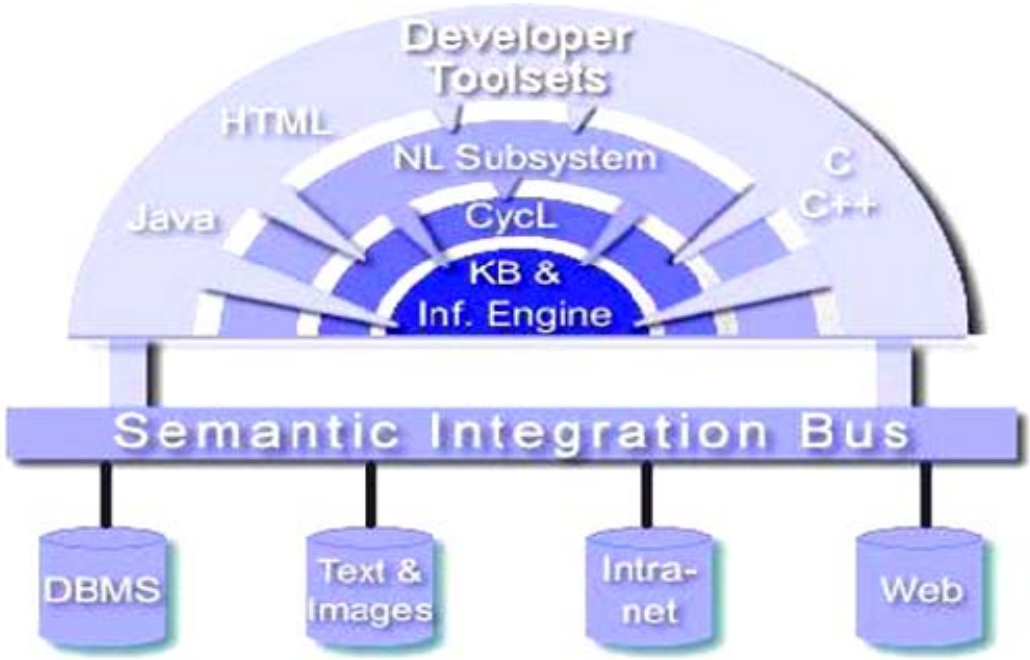


Figure 13. A diagram of the Cyc components
[From Ref. http://www.cyc.com/cyc/technology/whatiscyc].

The knowledge base includes facts, rules of thumb, and heuristics which machines use to reason about the world and everyday life. It is divided into microtheories, each of which has a set of assertions based on certain assumptions. Microtheories give context to the assertions. It allows Cyc to contain assertions that may contradict each other as long as they belong to different microtheories. For example, a microtheory about a day may contain an assertion that the sun is shining and a microtheory about a night may contain an assertion stating the sun is not shining. These two assertions contradict each other, but they are both valid within their microtheory or context.

The inference engine uses proprietary heuristics to perform logical deduction on the knowledge base (http://www.cyc.com). The reasoning performed includes modus ponens, modus tollens, universal and existential qualification, inheritance, and classification. Reasoning is optimized for each microtheory by restricting the search domain. The engine has over 500 microtheory-specific algorithms that ensure efficient reasoning over the large knowledge base (Reed and Lenat, 2002). The proprietary

methods developed by Cycorp for reasoning against Cyc are highly optimized for handling the large number of terms and assertions in the knowledge base.

The Cyc knowledge base is written in the CycL representation language developed by Cycorp. CycL is a large knowledge representation language used to express first-order logic (FOL) and some second order logic features (http://www.cyc.com/cycdoc/ref/cycl-syntax.html). Cyc includes a natural-language processor (NLP) that relies on a lexicon, a syntactic parser, and a semantic interpreter to remove ambiguities from natural language statements. The semantic interpreter uses the "common sense" knowledge encoded in Cyc to resolve ambiguous natural language statements. The semantic integration bus allows data stored in various formats such as databases, spreadsheets, and text documents to be converted into usable information for the knowledge base. The information from these sources are treated as implicit assertions. Cyc also provides software tools for developers so that they can browse, edit, extend and query the knowledge base. Thus Cyc is a knowledge base along with a set of tools and engines to edit, access and query the "common sense" information.

Cyc provides an expressive upper ontology. CycL is a sufficient language for mapping/merging and integrating domain-specific ontologies and Cyc. Furthermore CycL is expressive enough to allow axioms to be written when terms in two ontologies do not have a 1-to-1 correspondence (Reed and Lenat, 2002). Semantic Knowledge Source Integration (SKSI) provides declarative mappings between outside sources and the Cyc knowledge base. Once external sources are mapped to Cyc, the data from the sources is treated as part of the knowledge base and included in reasoning by the Cyc inference engine. Cyc thus provides a single access point for large sets of related data and a corresponding solution for integrating structured knowledge sources (http://www.cyc.com/cyc/cycrandd/areasofrandd_dir/sksi).

E. SUMMARY

The SW is supported by a wide variety of tools including user-friendly editors and powerful reasoning engines. Protégé is an open-source ontology editor which hides the details of the OWL language from developers creating domain-specific ontologies.

Protégé provides a plug-in architecture which makes it easy to extend and customize. Tools such as Protégé enhance the ability to design SW applications which access and manipulate OWL ontologies.

The SW also provides the possibility of semantic interoperability among knowledge-based information sources. Upper ontologies provide a common world model that may be refined to represent and correlate particular knowledge domains. Upper ontologies provide a intermediate step towards interoperability across broad domains. Ontology mappings are used to resolve differences between different world views. Such interoperability is expected to increase the power of data exchange and information sharing. Architectures such as VIRT which provide valuable information from various different sources support the ultimate goals of the SW.

III. DETAILED PROBLEM STATEMENT

> [We must] leverage information technology and innovative network-centric concepts of operations to develop increasingly capable joint forces. Our ability to leverage the power of information and networks will be the key to our success…
>
> -- Former Deputy Secretary of Defense Paul Wolfowitz, April 2002

A. INTRODUCTION

This chapter addresses the challenges that face attempts to establish information superiority. The data available to warfighters is increasing while time to process the data into useful information and make decisions is decreasing. This chapter proposes three possible solutions to increasing information superiority through automating data processing and identifies the SW as a promising approach. This chapter also introduces the capabilities of the SW which are explored by this thesis.

B. NATURE OF THE PROBLEM

The Global Information Grid (GIG) is the net-centric information environment of the future (Winters and Tolk, 2005). The GIG is an architectural platform which is expected to provide real-time valuable decision making information to all warfighters on demand in order to achieve information superiority. The goal of the GIG is to provide a global net-centric system for processing, storing, managing and transporting information to support the DoD in peace and during operations as illustrated in Figure 14. Information superiority is the capability to collect, process and analyze a flow of information while denying an enemy's ability to do the same. It is vital in current battlespaces, and surveillance and intelligence technologies have made data abundant. However, information superiority still faces three major challenges: how to combine data from different sources to provide a richer picture of the battlespace, how to analyze the overwhelming amount of data available, and how to customize data for each individual (Hayes-Roth, 2004).

Figure 14. Vision of the GIG for combining various data sources to create interoperability [From Ref. http://akss.dau.mil/dag/Guidebook/IG_c7.3.4.1.asp].

The first challenge facing information superiority is interoperability. Interoperability is the capability for computer systems to share information and perform tasks concurrently (Pitoura, 1997). Warfighters have multiple data sources at their disposal, but each of these sources structures data differently. The amount of information and the tactical view of the battlespace is improved by combining data collected and processed by different systems. Central to the strategy for enhancing system interoperability is the use of XML for data exchange throughout the environment. XML provides a framework for describing and structuring data (Hunter, 2004), so it can be interchanged across a network. The goal of XML is to allow mark-up of any data without restricting the terminology used to describe the data. It is a meta-language, which means it is a language used to define other languages. XML is flexible, and any XML vocabulary can be processed by an XML processor. TAML is an XML vocabulary that describes undersea warfare tactical information such as contact speed, course and position. TAML provides a standard XML tagset for exchanging tactical messages

between systems that were not developed with interoperability in mind (Brutzman and Grimley, 2005). The Undersea Warfare (USW) community uses various weapon and information systems to gather information about a battlespace to make tactical decisions. TAML enables these systems to exchange data but does not solve semantic differences in data. New technologies are needed to encode semantic information about a domain in a machine-interpretable language. Once syntactic and semantic interoperability are achieved, machines will be able to receive data from any available sources and combine the data into valuable information for warfighters.

Every warfighter has a need for customized information. Marines on the ground need terrain data and information about enemy movements in their area of operation, while sailors on a ship need information about the weather and surface threats in their area. The goal of the GIG is to provide valuable real-time information to every warfighter. Thus a computer that does not understand the meaning of data needs a way to determine which pieces of data are relevant to a particular warfighter. Current methods are insufficient because the information a warfighter needs and the information available are constantly changing whereas the methods used to represent that information are static. A more flexible and extensible method of collecting and analyzing data to extract and derive valuable information is necessary.

C. CANDIDATE SOLUTIONS

One approach to parsing and analyzing data is to create a hard-coded program for each structured data source, defining explicit instructions for parsing and handling the data. However, this disconnected approach leads to "stove-piped" systems that are unable to adapt to changes in assumptions about a domain and are not easily extended to handle new information sources. Interoperability among systems increases information superiority, but interoperability between stove-piped systems is difficult. An information exchange interface has to be coded for each one-to-one correspondence of systems to resolve differences between the systems. For example, System A may define the speed of a contact as feet per second and System B may define the speed as miles per hour. These differences must be resolved before the two systems can effectively exchange information and operate together. The large number of legacy systems and growing

number of new systems makes hard-coding such one-to-one correspondences unfeasible. The hard-coded approach presents another challenge because the interfaces are hard to change if one of the systems changes. For example, if System A changes its definition of speed then the interface no longer works and must be modified. Modifying programs is a difficult task, made even more difficult by the requirement to deploy such changes globally without losing capabilities. Modifying programs should be avoided if possible, thus a more flexible approach to interoperability and information analysis is needed.

XML offers another approach, but it only provides a partial solution. The language provides a framework for describing and structuring data (Hunter, 2004), so it can be interchanged across a network. The goal of XML is to allow mark-up of any data without restricting the terminology used to describe the data. XML is a meta-language, which means it is a language used to define other languages. It is the W3C standard for creating vocabularies that describe a domain.

Before XML, programmers structured their data in a variety of ways. Each programmer developed their own data definition and mark-up styles. For every different method of mark-up, there was a different method for parsing and pulling out needed information. This approach was both non-interoperable and inefficient, often dedicating a major portion of each program to simple input/output. Standardization was needed so data could be exchanged among multiple people without each person needing multiple tools to process the data. XML provides the basis for such data standardization and has been widely accepted throughout the Web community. No matter how data is structured in XML, any XML parser can retrieve the information and immediately check for possible errors. XML is platform independent and license free, so it can be used by anyone. For example, the TAML schema describes the rules for structuring TAML documents and parsers enable users to validate document correctness. Programs are written to extract the data from the documents, but the structure must be known because the meaning or semantics of the data is implicit and unknown to the machine processing the data. Therefore, any changes to the structure or schema of TAML documents requires changes to all the applications that process TAML. Thus XML alone is not enough to overcome the challenges of the frequently changing requirements in the information world.

The SW is an extension of XML which provides semantic interoperability in addition to syntactic interoperability. The original vision of the SW applies to creating a connected Web. The SW is expected to connect machine-interpretable data from independent domains on the Web to form one large web of understanding (Berners-Lee, 1999). Common ontologies will provide mappings to domain-specific information allowing machine agents to search the Web for information and apply that information to a problem. This same architectural approach can be applied outside the Web to create a net-centric environment for warfighters. Ontologies provide a filter for machines to process information and determine which data is important to each warfighter. The added semantics also enable machines to apply algorithms to data in order to infer valuable information. Thus the SW benefits from the syntactic standardization provided by XML and provides additional standard languages to express and evaluate meaning.

D. APPROACH CHOSEN

The SW offers a promising solution to all three challenges faced by requirements for information superiority. It is realized through several components including common syntax for data exchange, common semantic representation, ontologies, logic, proof and trust. Each component addresses different aspects of system interoperability and automated processing of information.

XML provides the foundation for interoperability through a common syntax for data exchange. The SW languages, RDF and OWL, build on this syntactic interoperability by providing a flexible standard semantic representation language. SW languages provide the framework and syntax for explicitly defining semantics, but allow experts to develop models which define their specific domains. This standardization enables machines to process RDF or OWL ontologies defined for any domain. Reasoners developed to process these languages are also able to derive new information from the data by applying reasoning patterns to the ontology. Since machines are capable of processing data more quickly than human operators, automation increases the information available to warfighters. Ontologies provide another key element for the net-centric battle field. They are not only used to model data sources, but can also explicitly

define the information that is valuable to a particular warfighter. These user ontologies are then mapped to domain ontologies, so a machine can process source data and translate it to a custom view of information.

The SW increases the capabilities of data-processing machines by allowing them to quickly collect data, analyze it, infer new information and disseminate valuable information to the right person. This thesis assesses the capabilities of three SW languages: RDF, OWL and SWRL. The goal is to provide an analysis of the potential each language provides for increasing information superiority.

E. SUMMARY

Information superiority is vital to current warfighting; however, several challenges must be overcome. The amount of information available is increasing, but the amount of time to sort through the information and make decisions is decreasing in current battlespaces. The current approach of hard-coding programs to extract information from data is no longer adequate. Hard-coding creates stove-piped systems which must be correlated through one-to-one mappings in order to exchange data. Since the data being exchanged has no meaning to the machine, this approach is not flexible enough to deal with changes in the structure of the data or changes in the assumptions of the domain. XML provides a syntactic standard for message exchange but does not add any machine-interpretable meaning to the data. However, emerging SW standards and technologies enhance XML by providing a standardized method to explicitly define the meaning or semantics of data. The SW provides machines the ability to process the semantics of data (not just the bits) and to infer new information to aid warfighters. It is a promising technology for enhancing information superiority

IV. TACTICAL ASSESSMENT MARKUP LANGUAGE (TAML)

> XML is everywhere, and indeed that is what is amazing about it. XML is how we create web content, how we integrate computers, how we define vocabularies and languages for communicating….My non-XML friends always say the most important thing about XML is that it is everywhere.
>
> -- Dave Hollander, Founding Member
> XML Working Group, February 2003

A. INTRODUCTION

This chapter provides an overview of the TAML vocabulary designed to improve Undersea Warfare (USW) interoperability. The chapter describes the purpose of TAML, gives a description of the vocabulary, and discusses the process taken to design the schema. The chapter attempts to provide background information for TAML which is later used as the basis for a TAML Contact Classification Ontology.

B. PURPOSE

XML has become the standard that businesses, government and military organizations use to structure and exchange data. The Department of the Navy (DON) vision for XML is "to fully exploit XML as an enabling technology to achieve interoperability in support of maritime superiority" (XML Naming and Design Rules, 2005). The DoD sees a potential for the use of XML for improving information exchange between systems. There are several current projects working to convert data driven applications into XML format, so data interchange can be platform independent.

TAML is an effort to design a platform-independent XML vocabulary for use in exchanging information among USW systems. The TAML vocabulary is designed to represent tactical information about a battlespace. The goal of TAML is to improve interoperability in USW by providing a vocabulary for systematic data interchange (Brutzman and Grimley, 2005).

The USW community uses various information and weapon systems to gather information about a battlespace to make tactical decisions. Today the need for interoperability among these systems is recognized. The amount of information and the tactical view of the battlespace may be improved by combining information gathered and processed by different systems. However, different systems were developed at different times and in different computer languages. The current approach to creating interoperability between systems is to hard-code one-to-one mappings, but the number of mappings needed grows exponentially with the number of systems. As an example, Figure 15 illustrates that 56 mappings are needed to create interoperability among eight systems.

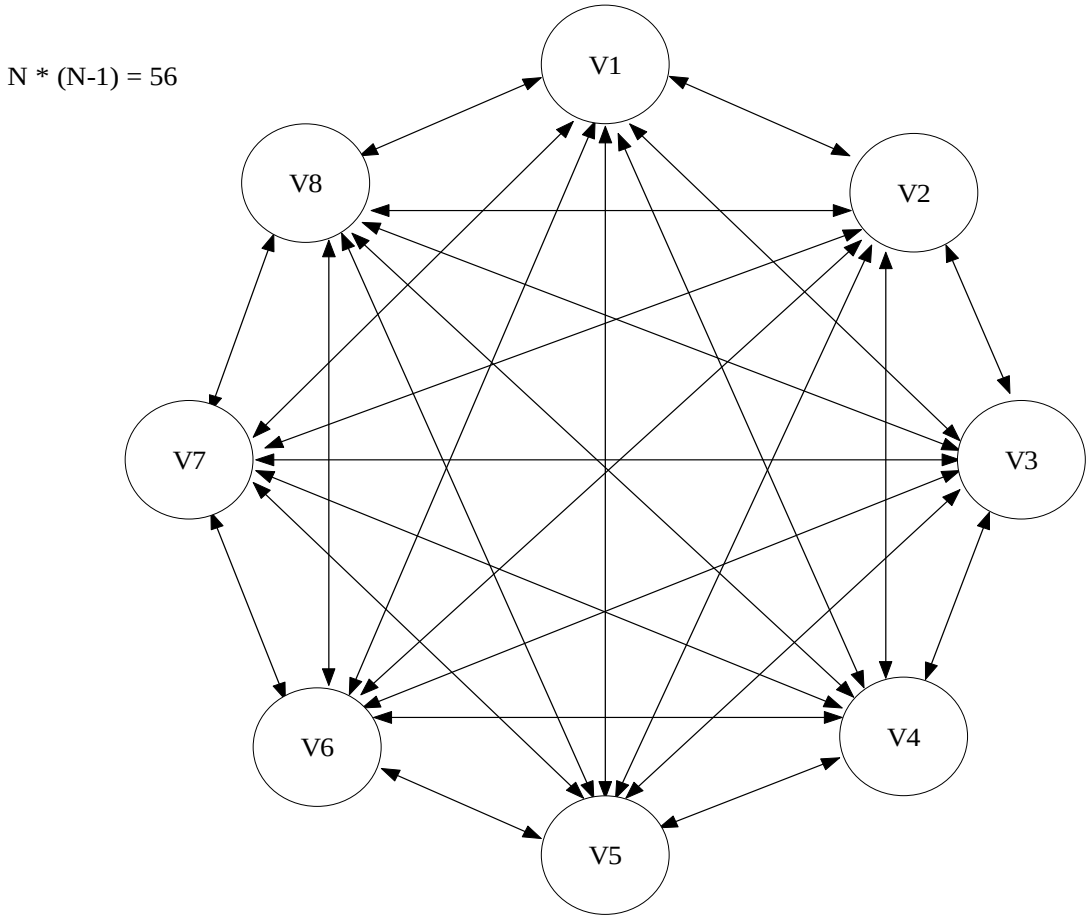


Figure 15. An illustration of the mappings needed to connect eight different systems for information exchange
[From Ref. Brutzman and Grimley, 2006].

A common format and external model is needed to reduce the number of mappings needed to achieve interoperability among systems. TAML provides a well-defined XML tagset for exchanging tactical messages between systems that might not have been originally developed with interoperability in mind. Data converters can be added to legacy systems to convert and wrap the inputs and outputs of each system to TAML. TAML acts as a common model in order enable information sharing among the systems. This approach reduces the mappings needed to connect the eight different systems as shown in Figure 16.

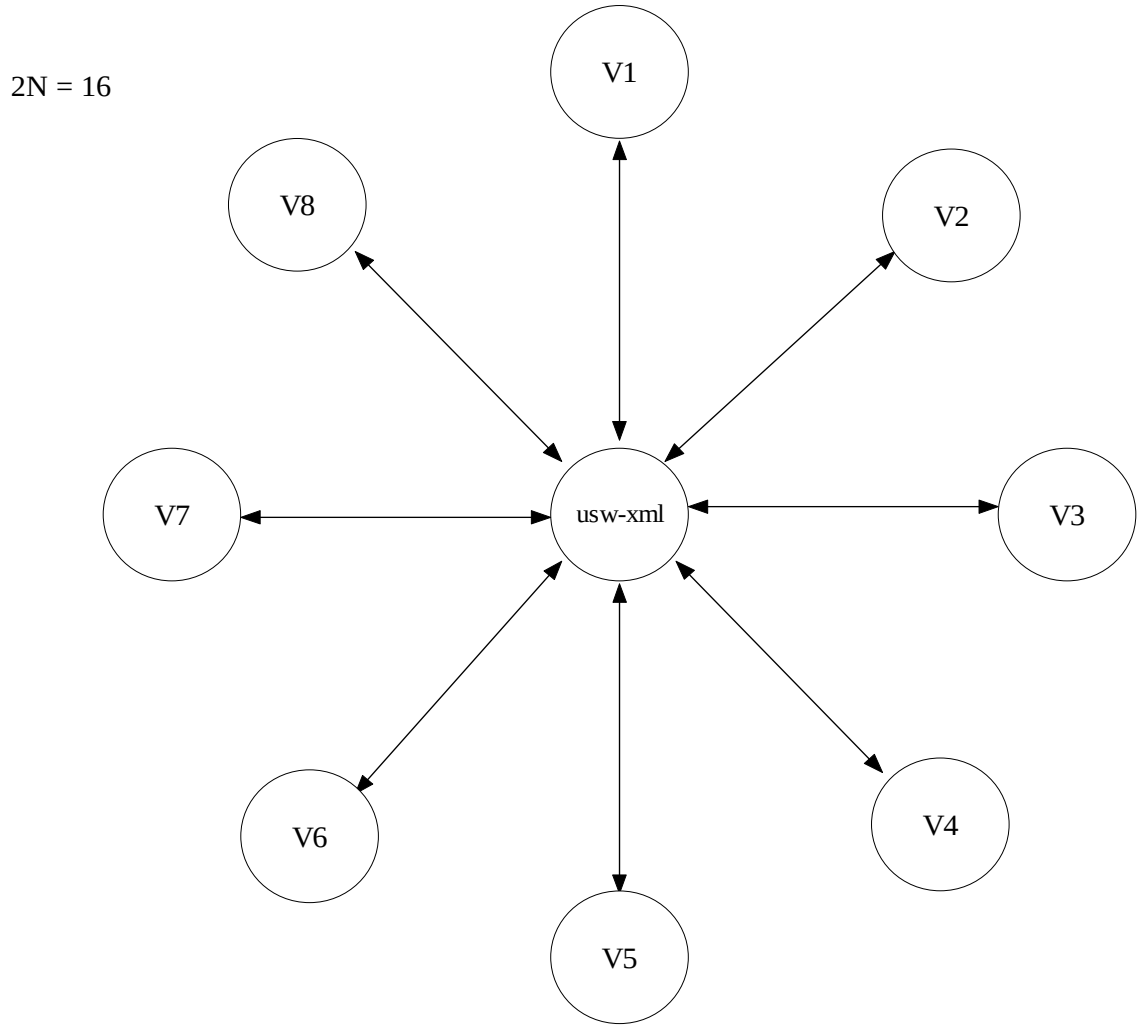


Figure 16. An illustration of the mappings needed to connect eight different systems when TAML is added as an external model between systems [From Ref. Brutzman and Grimley, 2006].

TAML also provides a common format for archiving data. Archiving data is increasingly important because legacy systems are increasingly not maintainable, and when they are removed from service their capabilities are typically lost. If data is archived in the output format of the system whose capabilities are lost then there is no longer a system to process or understand the information; therefore, the information is lost with the system. TAML provides a well-structured and well-defined format for storing information that allows the information to be accessible and usable by any system with a converter to the common external model provided by TAML (Brutzman and Grimley, 2005).

TAML provides a common information-representation model that supports interoperability among different systems. Both the syntax and semantics of the model are defined unambiguously in the TAML schema document. The support for interoperability provided by TAML enhances USW tactical data sharing for command and control, exercise assessment, operational analysis and simulation (Brutzman and Grimley, 2005).

C. DESCRIPTION

TAML was designed by the USW-XML working group as a possible solution to address expanding interoperability challenges. Full interoperability includes the ability for two systems to communicate, share information and perform tasks concurrently.

TAML is an XML vocabulary. Before XML developers designed their own proprietary markup style which led to an infinite number of ways to structure data. For every different method of mark-up, there was a different method for parsing and extracting needed information. Standardization is needed so data can be exchanged among multiple systems without each system needing multiple tools to process the data. XML provides this standardization and has been widely accepted throughout the Web community. Although XML is a standard language, it is extensible. A developer can mark-up data in any way using the language even if others are marking-up similar data differently. Each developer determines the name of the tags used to describe data and how to structure those tags and data. The flexibility of XML has contributed to its widespread adoption. XML is a key enabler of the Department of the Navy (DON) net-

centric data strategy (htpp://doncio.navy.mil). The standardization of XML ensures that TAML documents are platform independent. Any system can receive and process TAML documents. However, TAML only provides a partial solution since each system's output still needs to be mapped to and from the TAML external model.

The focus of TAML is representing own-ship and target track data (Brutzman and Grimley, 2005). The goal is to provide a common model to support multiple programs including the Anti-Submarine Tactical Assessment System (A-TAS), the Undersea Warfare Decision Support System (USW-DSS) and the Carrier Tactical Support Center (CVTSC). The integration of the language with these systems and the development of TAML exemplars are in progress (Brutzman and Grimley, 2005).

D. SCHEMA DESIGN

The Anti-Submarine Warfare Tactical Assessment System (A-TAS) was first utilized on deployment in 2004. During the initial tests of the system, the need for a standardized data transfer format became apparent. In 2005, collaboration began on a Navy wide schema for USW. The Naval Postgraduate School (NPS) coordinated the effort and TAML was the first schema produced. TAML is designed to provide format standardization for data collection and analysis systems. The first version focuses on own ship and target ship track information for Anti-Submarine Warfare (ASW).

TAML was designed in accordance with the DON XML Naming and Design Rules (NDR) v2.0 (htpp://doncio.navy.mil). The NDR document is an effort to comply with the FORCEnet requirement for a common structure and language to be used in information handling. The NDR provide developers with rules and guidelines in order to standardize schema development and increase interoperability. The NDR is necessary because the flexibility of the XML Schema standard developed by W3C allows the development of incompatible XML Schemas. Compatibility is vital for successfully exchanging data between organizations. The NDR document is a product of the DON XML Working Group which is composed of 13 Navy, Marine Corps, and Secretary of the Navy organizations. An XML Business Standards Council (BSC) is charged with ensuring the rules remain applicable and current. Applying the rules to schema design maximized interoperability between schemas and moves the Navy towards a net-centric

environment which is sustainable, responsive, and agile (Wennergren, 2005). Compliance with the NDR is necessary in order for schemas to be registered on the DoD Metadata Registry (http://metadata.dod.mil).TAML is the first tactical schema to be designed in compliance with NDR and registered on the DoD Metadata Registry (Burkley, 2006).

The design of TAML involved several stakeholders including Keyport, the information producer and Newport, the information consumer. The USW-XML Working Group was also involved in the design of TAML. Progeny Systems built a translator to facilitate harmonization between Keyport and Newport. Mappings between systems were encoded in XSLT and used by the translator. The USW-XML Working Group also provided standards guidance throughout the design process.

The TAML Schema was designed by Fred Burkley, Greg Sabatino and Mike Grimley using a top-down approach. The A-TAS database schema provided the domain knowledge for the TAML vocabulary (Burkley, 2006). A-TAS is a post-exercise analysis assessment toolkit used to judge how well operators perform in an exercise. The system provides a method to assess and monitor fleet Anti-Submarine Warfare (ASW) readiness at the platform, strike group, theater, and fleet level based on standardized metrics (Fleet Anti-Submarine Warfare Command, 2004). The TAML meta model describes the data needed for ASW exercise replay and analysis. Exercise replay provides the analyst the opportunity to observe events, interpret and score participants, and score an entire exercise. The scores are then archived back into the A-TAS database, so scores can be compared against other participants and commands (Burkley, 2006). TAML only includes a subset of the domain knowledge represented in the A-TAS database. Common data shared by A-TAS and CVTSC was chosen for the first version of TAML. The subset focuses on track information, but future versions of TAML are expected to encode more of the A-TAS domain knowledge. For example, future versions of TAML will better capture weapon and weapon system domain knowledge.

The A-TAS database schema provided the domain knowledge and influenced the element and attribute names defined in the TAML Schema. The domain knowledge was restructured into a hierarchical XML vocabulary by Fred Burkley, Greg Sabatino, and

Mike Grimley. They used their view of the domain knowledge to structure the schema in order to provide a human-readable XML vocabulary for track data (Burkley, 2006).

The root element of each document is the TAML tag. The TAML tag has four children: *Operation*, *Platform*, *Sonobuoy*, and *Threat*. *Operation* defines the organization and temporal parameters of an exercise such as start and stop time. The *Operation* element also defines a list of individual events which take place within the operation. Each *Event* is then further described through a set of child elements. *Platform* is a list of exercise participants which are identified by ship name and hull number. The platform's track, sensor configuration, search events, contact call and engagements are modeled as child elements. Each *Sonobuoy* is modeled as a platform which contains its own track and contacts. The *Threat* element is used to aggregate contact events from separate platforms into a single track (Burkley, 2006). The *Threat* element provides an overview of a *Contact* from the combined views of several platforms. The *Operation, Platform, Sonobuoy*, and *Threat* elements make-up the high-level design of the TAML Schema.

TAML provides a standard format for data exchange among data collection and analysis systems for naval exercises and operations. A-TAS can currently import and export data stored as TAML by means of a file download or through a web service. Several data collection systems and data analysis systems are currently planning to export data in TAML format.

E. SUMMARY

TAML provides an external model which increases interoperability by decreasing the amount of mappings needed to create a network of data exchange among multiple systems. The domain knowledge for TAML was influenced by the data contained in the A-TAS database. TAML focuses on structuring own ship and target ship track information in a hierarchical format which is easy for humans to interpret. The TAML Schema defines the syntax of the vocabulary and provides implicit semantics for the meaning of the elements; however, no explicit semantics are defined for machine interpretation.

TAML was designed in accordance with the DoD NDR guidelines and is registered in the DoD Metadata Registry. The original version of TAML is designed to model a subset of the information stored in the A-TAS database. Future version of TAML are expected better capture more data such as weapon and weapons system domain knowledge.

V. SEMANTIC WEB (SW) CONCEPTS

> The first step is putting data on the web in a form that machines can naturally understand, or converting it to that form. This creates what I call a Semantic Web - a web of data that can be processed directly or indirectly by machines.
>
> -- Tim Berners-Lee, *Weaving the Web*, 1999

A. INTRODUCTION

This chapter provides an overview of the SW, as well as the layered components that together bring the SW to realization. It is also reviews the technologies that make up the SW layers along with the implications of these technologies.

B. WHAT IS THE SEMANTIC WEB (SW)?

1. The Vision

Tim Berners-Lee, the founder of the SW, describes the SW as a powerful way of representing data on the web in order to add semantic meaning to the data that can be interpreted and used by machines (Berners-Lee, 1999). The current Web is a forum for humans to display, look up and interpret data, so the Web is structured to present information in a human-friendly manner (Berners-Lee, 1999). HTML provides a language for structuring data in a human-readable form, but does not provide any explicit meaning that can be read and used by machines. The vision of the SW is to provide an extension to the current Web where data is given well-defined meaning through its structure. The relationships between data are described explicitly by adding metadata to already existing data, therefore creating machine-interpretable content (Berners-Lee, 1999). Systems are then expected to be able to use this well-defined data to perform a variety of tasks that currently require human intervention, such as tasks that invoke and query web services.

Tim Berners-Lee also envisioned the SW creating a collaborative medium. The SW is expected to connect machine-interpretable data from independent domains on the web to form one large web of understanding (Berners-Lee, 1999). Common ontologies will provide mappings to domain-specific information, thereby allowing machine agents to search the web for information and apply that information to a problem. For example, an agent might be tasked with scheduling a family vacation including airline tickets, hotels and a rental car. The agent can search the SW for information from various airline, hotel, and rental car companies, find the correct services, and follow machine-interpretable rules to make purchases using semantically defined web services. The SW is expected to provide a more automated Web where systems can perform sophisticated tasks at computer speeds. The computer has the ability to process more information in a shorter amount of time than humans, so SW capabilities are likely to provide the ability to absorb and process more information before decisions are made. Such increased information processing and automated decision making are important in the Armed Forces where the amount of data being collected can be overwhelming, so the potential value of these approaches is great.

2. Semantic Web (SW) Layers

The SW is realized through multiple components including common syntax for data exchange, common semantic representation, ontologies, logic, proof and trust. Figure 17 illustrates the different stages of interoperability that must be achieved in order to create machine understanding and highlights the languages that provide this interoperability.

Broad machine interpretation of data will not be fully realized until full interoperability has been defined, developed and agreed upon by the World Wide Web Consortium (W3C). Working groups in the W3C organization are currently developing and approving new tools and languages for the SW. The W3C is defining and refining the standards that are expected to lead to interoperable exchange and understanding of data between machines. This section describes the basic components that are needed to create the SW in order to present a basic understanding of how the SW can be realized.

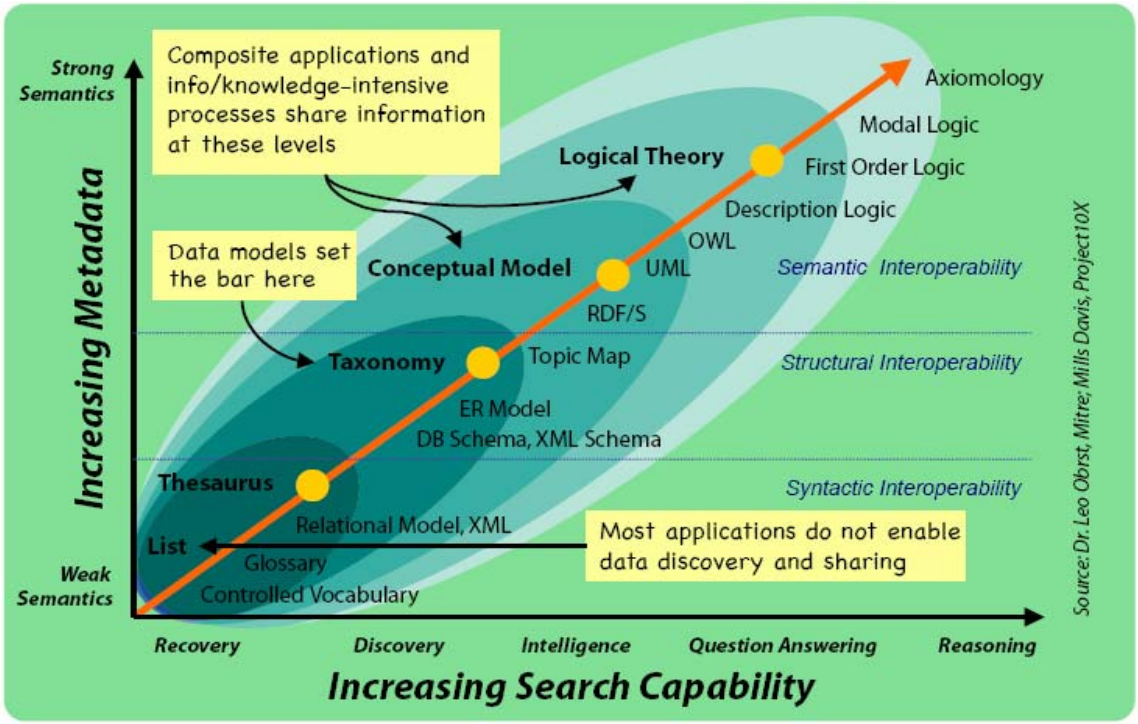


Figure 17. Illustration of the levels of interoperbility required for machine understanding of data across systems [From Ref. Obrst, 2006].

a. Semantic Layer

Portraying semantics in a machine-understandable format is the key to realizing the SW. The Merriam-Webster dictionary defines semantics "as of or relating to meaning" (http://www.m-w.com/dictionary/semantic). Explicit semantics are formal descriptions of concepts or terms and their relationships defined in a way which supports machine understanding (Lacy, 2005). Before the SW, semantics had to be hard-coded into software or database schemas. Hard-coded semantics describe data for a particular application but there is no common representation across differing applications or domains. SW technologies allow for the explicit definition of a domain using a common representation to reduce ambiguity and increase interoperability (Lacy, 2005). Explicit semantics can document such concepts and relationships through the use of models or graphs. Simple statements about the concepts in a domain are combined to create an overall semantic understanding of information relationships. Using the same model structure to create the simple statements describing a domain enables computers to process the semantics from any domain using the same tools.

b. Ontology Layer

Gruber defines an ontology as a "formal specification of a conceptualization" (Gruber, 1993). Figure 18 illustrates Gruber's definition. The conceptualization of a domain is defined by an ontology which in turn describes the domain.

Ontologies are used in the SW to formally describe a domain by describing classes, relationships between classes, class properties and constraints on relationships (Powers, 2003).

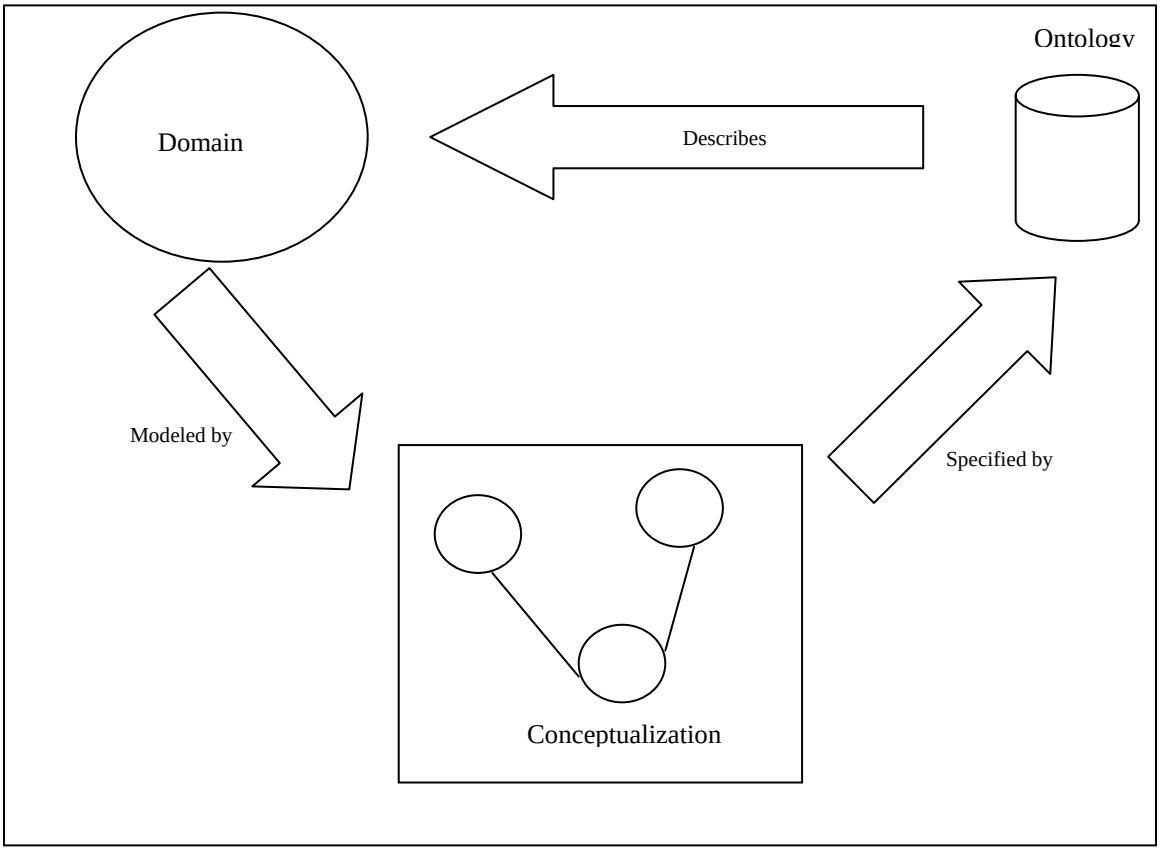


Figure 18. Definition of an ontology as a specification of a conceptualization and a description of a domain [After Ref. Gruber, 1993].

Ontologies combine the assertions made in the semantic layer with class, property and hierarchical relationships to increase the semantic meaning available. More-explicit semantics enhance information exchange and understanding, therefore ontologies are useful for domain interoperability, knowledge reuse, unambiguous data representation and automated data analysis.

Along with a standard syntactic representation of semantics, ontologies describe the objects in a domain in common terms that can be processed by both humans and machines (Noy and McGuinness, 2000). For example, suppose several different businesses contain inventory databases and provide e-commerce web services. If these businesses share a common ontological model describing the terms they use in their inventory databases, then software agents can collect and analyze data from each of these different databases. The agents can use the aggregated data collected to perform more powerful queries or to provide more powerful web services to customers.

Ontologies provide reuse of domain-specific knowledge (Noy and McGuiness, 2000). Once an ontology has been developed to represent a domain, that ontology can be imported into other domain ontologies. For example, an ontology that models location in longitude and latitude can be imported as part of a more general ontology describing a city map or an ontology describing enemy track locations. The imported ontology may need to be modified to reflect domain-specific differences, but the majority of the work for describing location has been completed.

Once a large database of common ontologies exists, creating new ontologies will consist of simply combining and modifying these ontologies. The DoD Metadata Registry may serve as an online collection of ontologies for DoD use. The DoD Metadata Registry is a collection of XML vocabularies, approved for DoD use and maintained by the Defense Information System Agency (DISA). The goal of the database is to promote interoperability and software reuse within the Global Information Grid (GIG) (http://diides.ncr.disa.mil). Reusing ontologies simplifies ontology creation and extends the idea of common data modeling which in turn supports further interoperability and the broader goals of a widely collaborative medium.

Ontologies remove ambiguity in data representation by making explicit assumptions (Noy and McGuiness, 2000). Ontologies define terms through the use of rules and properties. These extended definitions make it clear to machines what is being modeled. Explicit assumptions also make change easier. If underlying assumptions in a data model change, the ontology can be changed to reflect the new assumptions. Before the SW, such conceptual assumptions were hard-coded into programs and were hard to correctly modify.

Ontologies also provide a way to analyze data in a domain (Noy and McGuiness, 2000). Once the relationships and properties in a domain have been well defined through an ontology, data can be classified in that domain based on the definitions and properties defined. Ontologies that are exercised by reasoning engines can provide inferences about the classification of available data within the specified domain. These inference mechanisms provide machines with powerful abilities to process information and infer additional knowledge beyond the specific data input.

Ontologies also provide the SW a way to define knowledge and remove ambiguity (Noy and McGuiness, 2000). Ontologies formed for different domains that specify the same real-world object can be mapped together to provide semantic integration. Semantic integration is the identification and explanation of logical connections between classes, properties and individuals across different ontologies (Ushold and Menzel, 2005). Semantic integration enables semantic interoperability, which in turn allows machines to exchange information in a meaningful way. The uses of ontologies within the commercial world and the military sphere is rapidly growing.

c. Logic

The logic layer of the SW provides a way of incorporating deductive reasoning in these ontologies. Logic allows for deduction of one type from another type, consistency checking, and query resolution by converting unknown terms into known terms (Tim Berners-Lee, 1998).

The Merriam-Webster dictionary defines logic as “the science that deals with the principles of reasoning” (http://www.m-w.com/dictionary/logic). Reasoning is a capability mostly associated with the human mind. However, the goal of the SW is to give machines the ability to reason about data and make inferences about the data in a particular domain. Currently, such logic is typically hard-coded into a program which only works for a specific domain and application. SW technologies seek to provide a standardized common syntax, so that common reasoning engines might perform logic operations on any domain. The key to machine-reasoning power lies in defining relationships about data through simple assertions and rules that can be defined in shared machine-interpretable languages. These simple assertions and rules can then be combined into more complex relationships, which the computing power of machines can fully process.

Inference rules form the basis of logic. Propositional logic contains axioms that allow the derivation of conclusions from a set of assertions. Propositional logic is simple and only allows the expression of simple axioms (Daconta, Obrst, and Smith, 2003). Axioms are universally accepted rules or rules that can be proved.

For example the axiom of “modus ponens” allows us to state a rule and an assertion and then make an inference:

Rule: If A is TRUE, then B is TRUE.
Assertion: A is TRUE.
Inference: B is TRUE.

Propositional logic simply states facts about propositions, but does not state facts about individuals or instances. First-order logic (FOL) has a finer granularity which allows logic to be applied to instances. FOL provides a set of rules that allow the expression of set theory and near complete expression of mathematics (Wikipedia, 2005). More complex rules can be broken down into a series of simple rules like the “modus ponens” rule above.

Logic provides the ability to deduce relationships and classifications. After a class is named in an ontology, it can be further defined with FOL quantifiers. For example, if a class *Pizza* is created it can be defined as anything that has at least one pizza base from the class *PizzaBase* using the existential quantifier. In addition to classes,

properties can be defined using FOL. The property *hasAunt* can be defined as true if Brian has a mother named Donna, and Donna has a sister named Ann, because then Brian has an aunt named Ann. This rule is written in FOL as a set of antecedents that imply a consequence:

{*parent*(Brian, Donna) [AND] *sister*(Donna, Ann)}= *hasAunt*(Brian, Ann)

Once a set of rules are created to define a class and its properties, raw data can be fed into a reasoner which then applies the rules and classifies the data. A way to specify exceptions to rules is also important. For example, if the property *hasAunt* referred to a biological aunt and Brian was adopted, it might be important to determine that the property *hasAunt* does not apply to Brian.

Logical rules are also used to check the consistency of an ontology. Once logic is inserted into an ontology, there must be a mechanism to ensure that one rule does not contradict another. For example, if two classes, *CloudyDay* and *SunnyDay* are created and described as disjoint, then any rule stating that an instance of a class is both a *CloudyDay* and a *SunnyDay* is logically inconsistent. Unintended inconsistencies can easily be built into ontologies and domain-knowledge representations, but logic axioms (like the axiom that states the meaning of disjoint) can be used to catch inconsistencies. In this manner a sophisticated FOL rule base can be created that is guaranteed to be logically consistent.

Ontologies by themselves simply represent a knowledge base for a particular domain. They do not provide any means to take data and place it where it belongs in the ontology. This is where logic is important. Logic is the part of the SW that allows for inference, classification and powerful querying. Prior to the SW, logic had to be hard-coded into applications using programming languages like Java and C++. Logic might be programmed for a specific set of data through programming constructs such as if/then/else statements. Rewriting such programming logic is difficult, often error prone and may even be infeasible due to administrative reasons (such as licensing, recompilation, or validation requirements). When such logic is hard-coded into applications, the applications are only applicable to one specific representation of data.

With the SW, the logic is generic and can be applied to data from all sources. This is the power of the SW.

d. Proof

A way to verify that collected and inferred data is correct is essential to the adoption of the SW. This can be accomplished through the use of proof-checking mechanisms (Alesso and Smith, 2005). A proof is a sequential trace of information manipulations that generate a check on a query answer. These proofs provide justification and establish the credibility of the data being received by the machines. An answer to a query that is wrong is useless. For example, if a particular source describes a bird as blue and another source describes the same bird as red, which source can the machine believe? Proof-determination mechanisms are needed for verification of both initial data and derived results.

Proof mechanisms on the SW provide validity to statements collected. If a set of statements collected are proven to be correct, then any inferences formed from the statements through proper logic are expected to be correct. Note that proof mechanisms also have to consider context when determining validity. For example, if one source says x is true and another says x is false, they might both be right if each is describing x in a different context. The SW provides no value if the inferences and analysis of statements produce questionable results. Thus SW logical capabilities have been carefully scoped to provide provably consistent results.

e. Trust

The web hosts a lot of data with no central authority to ensure the validity or credibility of the data. This approach of having uncensored sources of information can provide an advantage because it allows anyone to post or consume data, but it can also pose challenges to machines and humans trying to use data found on the web to perform tasks. Machines and humans gathering information on the SW need a way to know whether data comes from a trusted source.

Therefore, the SW must eventually include the technology to build a "Web of Trust" (Hu, Se-Ting, and Min-Huei, 2005). The SW might approach this issue by allowing semantic statements of trust to be attached to statements as metadata. For example, if Document A is trusted by Bob, then Bob can attach a statement to Document A stating his trust. If Sue trusts Bob then she can trust Document A. Nevertheless, trust still raises many challenges. How does Sue trust that Bob's trust statement was actually attached by Bob? Practices such as Certificates and the Public Key Infrastructure (Khare and Rifkin, 1997) used in the modern Web are expected to benefit the SW and solve some of the challenges associated with trust.

The Web currently provides some mechanisms for humans to determine trust, but it does not provide mechanisms for machines to determine trust of data or web services. The SW must address the need for machine-interpretable trust rules and semantics. One approach to this problem is to create a trust ontology and trust rules to form a "Web of Trust". Trust in the SW is defined as an authentication, authorization and delegation-verification problem for agents collecting and using information (Hu, Se-Ting, and Min-Huei, 2005). The trusted SW is expected to incorporate well-defined trust ontologies and trust rules within agent protocols to allow agents to authenticate information, authorize the use of information and delegate tasks to proper web services.

A trust ontology provides a machine interpretation of the concept of trust. A trust ontology and trust rules are combined to provide a trust knowledge base. A trust knowledge base might consist of several combined ontologies such as a trust ontology, message ontology, role ontology and service ontology (Hu, Se-Ting, and Min-Huei, 2005). The trust ontology is used to describe a taxonomy of digital certificates that make up trusted information. The terms declared in this ontology are used as the basis for trust-verification rules. The message ontology describes the messages the agents use to interact with one another. Messages allow agents to share information, but the message structures that they share must include trust properties. The role ontology is used to classify the organizations issuing certificates for information authenticity. The service ontology provides a description of the trust mechanisms needed to ensure proper use of a web service. Together these ontologies may provide a machine-interpretable view of trust and provide the terms needed to create trust rules.

Trust rules build upon the terms defined in the trust ontology to provide the guidelines for determining whether a source can be trusted or for determining whether a source can be granted access to certain information. For example, if Agent A wants to access a Travel Web Service, he must satisfy all of the trust rules derived from the trust ontology for the Travel Web Service.

The SW can continue to use existing web-trust mechanisms such as digital certificates and Public Key Encryption, but needs to enhance them so machines can interpret their meaning and use them to determine trust. SW trust ontologies and rules might provide these needed enhancements to the SW. Such capabilities are just beginning to be demonstrated. Conceivably they might someday be applied as part of business-process transactions and even military communications between trusted coalition partners.

f. SW Agents

SW agents are the computer applications that collect, process, analyze and use data to perform sophisticated tasks involving the use of data from one or more domains (Alesso and Smith, 2005). SW agents are capable of understanding and processing information from many different domains that are described by different ontologies, since the ontologies are described in common machine understandable semantic formats.

SW Agents are expected to increase the power and role of machines. Agents are expected to more quickly accomplish time-consuming, repetitive tasks currently performed by humans. For example, Bob may have an agent that schedules meetings and appointments for him. If Bob needs to schedule a doctor's appointment, the agent can query several doctor sites on the SW, find the doctor closest to Bob covered by his insurance, and check that the doctor has an appointment available when Bob is free. At this point the agent can communicate with the doctor's agent to schedule an appointment. All Bob has to do is specify the parameters for the appointment he needs, and the agent automatically completes the remainder of the task and presents Bob with the results.

Agents might thus perform a wide variety of tasks with logical consistency and unlimited endurance. Agents can communicate with each other and share information through a common message ontology. Extending the example, Bob might have an overall management agent that controls his entire day including when and where he eats, setting business and personal appointments and scheduling leisure activities. Bob's appointment agent can communicate his scheduled appointments with Bob's overall management agent. Given SW interoperability, the sheer volume of data a machine agent will be able to process can surpass any human processing capability. Agents are the real application power of the SW that are expected to inspire its widespread adoption.

C. ENABLING TECHNOLOGIES

The W3C has approved specifications for several languages that contribute to realizing the SW. This section provides an overview of the technologies and languages that currently make up the SW as illustrated in Figure 19.

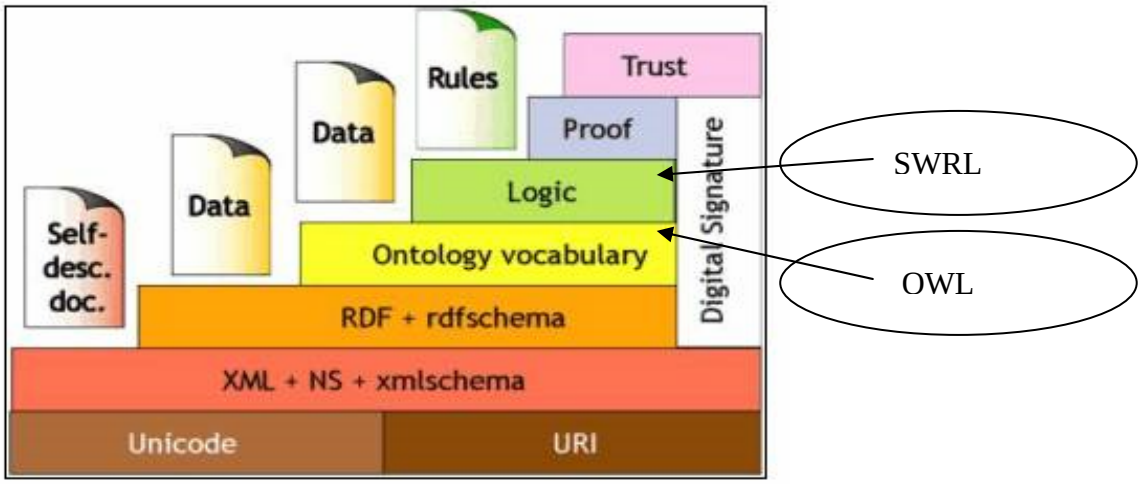


Figure 19. Modified Semantic Layer Cake showing the components that make up the SW and highlighting the technologies that enable those layers [After Ref. Berners-Lee, 2001].

1. Providing a Common Syntax Using XML

XML is the language that forms the syntactical foundation of the SW (Daconta, Obrst and Smith, 2003). Each SW language is built on top of XML so that syntactical interoperation remains at each later. XML provides a framework for describing and structuring data (Hunter, 2004) so it can be interchanged across a network. The goal of XML is to allow mark-up of any data without restricting the terminology used to describe the data. XML is a meta-language, which means it is a language used to define other languages. XML is the W3C standard for creating vocabularies to describe a domain.

XML is platform independent and license free, so it can be used by anyone. There are many tools that aid developers in creating, editing and using XML. All XML languages must follow certain rules in order to be considered well-formed, and only well-formed XML can be parsed for information by an XML parser.

Although XML is a standard language, it is extensible. A developer can mark-up data in any way using the language even if others are marking-up similar data differently. Each developer determines the name of the tags used to describe data and how to structure those tags and data. The flexibility of XML has contributed to its widespread adoption.

XML provides rules for creating languages. As long as developers follow these rules, their languages are well-formed and parseable by XML parsers. The most basic rule is that tags come in pairs.

Each start tag must be closed by a matching close tag. For example the following is well-formed XML:

```
<name>Bob</name>
```

The following is not well-formed XML:

```
<name>Bob
```

There are other rules for XML that make up the standard. As long as the rules are followed, the naming and structure of the elements are up to the developer. In this manner XML can describe any data set with meaningful tags. However, these tags are limited because they are only meaningful to the humans reading them.

XML is hierarchical in nature. This allows information to be broken up into subtopics that make the information more readable and understandable to the humans browsing the mark-up. Elements can be set-up in parent-child relationships to show relationships between them, but these relationships are only known to humans and are not explicitly defined by the XML language.

For example:

```
<name>
     <first>Bob</first>
     <last>Smith</last>
</name>
```

Humans reading XML understand the relationship implied by the hierarchy: a name is made up of a first and last name. Machines understand there is a parent-child relationship between the elements in the document graph, but they do not know the specifics of the relationship.

XML provides a common syntax for exchanging information, but also provides flexibility to ensure any domain can be described. However, XML does not provide any explicit semantic information that is interpretable by machines. The SW is not built on XML alone.

2. Resource Description Framework (RDF)

RDF is the language that provides the semantic foundation of the SW. RDF is used to make assertions about resources or add metadata to data (Hjelm, 2001). Assertions are statements that describe or point something out about a resource, and a resource is any object that can be described. Resources in RDF are identified by a unique Uniform Resource Identifier (URI) to avoid name conflicts. The RDF model creates a description or set of machine-interpretable statements about a resource. RDF is a W3C specification and does not require a license. RDF can be serialized in many formats including XML to ensure broad interoperability with other graphs.

a. Assertions

The RDF model used for representing assertions is known as a "triple" because each assertion has three parts (Hjelm, 2001). Each RDF statement is similar to a basic English sentence and is made up of a subject, predicate and object. Several basic assertions can be combined to create a full description of a resource. There are several notations for writing RDF statements.

The most basic syntax is the N-triple format which resembles but is not strict XML. For example the following N-triples describe a book with the URI http://www.resources.org/book:

```
<http://www.resources.org/book>
<http://purl.org/dc/elements/1.1/title><Practical RDF>.
<http://www.resources.org/book>
<http://purl.org/dc/elements/1.1/creator><Shelley Powers>.
```

The first statement states the book has the title *Practical RDF* and the second statement states the book's creator is Shelley Powers. Each of the three parts of the statement is enclosed in brackets. The first part is the subject followed by the predicate and then the object.

In this example, the resource is described by the defined Dublin Core predicates, *creator* and *title*. The Dublin Core is a metadata initiative created by librarians and can be imported into any RDF document (http://dublincore.org). The goal of the Dublin Core Metadata Initiative (DCIM) is to promote the widespread adoption of metadata standards as an effort to increase interoperability and enable more intelligent information discovery systems. The DCIM organization is independent, international and influenceable. The organization is open and there are no prerequisites for participation. The group aims for consensus among all participating organizations. The Dublin Core initiative demonstrates how domains can be reused, since Dublin Core predicates can be used within any document which has an author, title, etc.

N-triples can also be described by graphs. The N-triples from above form the graph in Figure 20.

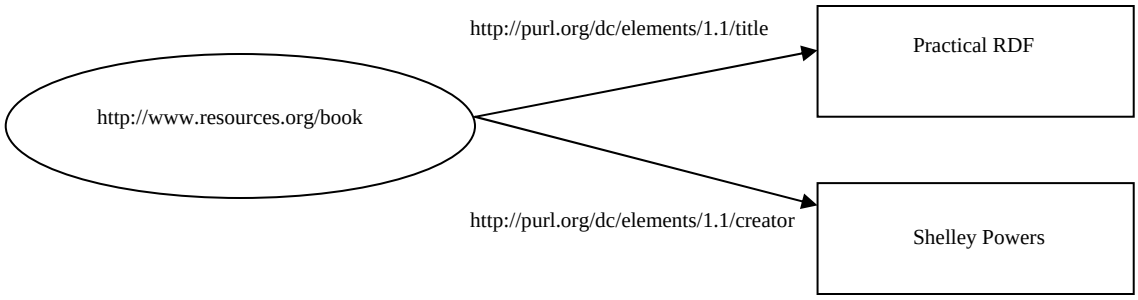


Figure 20. RDF Graph showing the graph representation of the above N-Triples

The subject of an RDF triple is the resource being described by the statement. Each subject must be identified by a unique URI to remove ambiguity. For example, if the subject “company” is being described, the meaning can be ambiguous. Does “company” refer to company as in a business or company as in friends coming for a visit? By denoting the resource with a URI, the resource is uniquely defined. Several assertions can then be made about any resource.

The predicate defines the relationship between the subject and the object. Predicates must also be defined by a unique URI. This removes ambiguity and also allows reusing the same predicate to define many subjects. For example, the predicate “title” can mean a movie title, book title, or a person’s title. Thus the predicate “title” needs to be further defined by its URI. The predicate title can also be reused by using the same URI, so the computer knows the same predicate relates book one to its title and also relates book two to its title.

The object of a triple is either a resource referred to by the predicate or a literal value. In the above statements the objects are the literals *Practical RDF* and Shelley Powers. An object can also be a resource related to the subject by the predicate. For example, if the book *Practical RDF* was a resource with its own URI, it might serve as the object of the statement or as the subject of its own assertions.

Assertions are the building blocks of machine-readable semantics. Assertions add power to data representation by explicitly defining relations between resources and literal values. XML allows for data to be described by meaningful tags but

does not provide a way to organize the tags to define relations. RDF adds this assertion capability on top of XML.

b. Containers

Containers allow RDF developers to make multiple literals or resources the object of a statement without having to create multiple triples. There are three defined types of containers: bag, sequence and alternate. When an RDF statement is created with multiple objects, the container becomes the object of the statement and holds (or contains) all the individual objects.

A bag contains an unordered list of objects and can contain duplicates. For example, if the object of the statement is apple pie, blueberry pie, and cherry pie and the order does not matter, then a bag is used as a list container for the objects. A sequence is used for an ordered set of objects where duplication is still allowed. If the object is a list of ordered steps to take for booting a computer then a sequence is used as the list container. The final container is an alternate container which is used to store a set of choices. An alternate container is used to constrain the object of a statement to a specific list of choices.

c. Reification

Reification is a property of RDF that allows developers to make statements describing other statements (Powers, 2003). In reification a statement becomes the object of another statement. Reification is useful for annotations of other's work or for implied assertions. For example, the statement Bob recommends the book titled *Practical RDF* includes two assertions. However, the assertion "Bob recommends the book" is a meta-statement supporting the assertion "the book is titled *Practical RDF"*. Reification allows representation of this type of relationship between statements. Reification is accomplished by defining the inner statement, "the book is titled *Practical RDF"*, as a resource with a unique identifier provided by the RDF parser. This inner statement then becomes the object of the outer statement, ""Bob recommends".

Reification takes RDF beyond simple statements by allowing the annotation of statements.

d. RDF/XML

In addition to the N-triple format, RDF can also be serialized as well-formed XML. RDF/XML serialization gives RDF documents the same advantages of XML discussed earlier. However, unlike most XML vocabularies, RDF is not hierarchical. RDF statements about a resource are listed below the resource in an *rdf:description* tag.

For example, the following N-triples:

```
<http://www.resources.org/book>
<http://purl.org/dc/elements/1.1/title><Practical RDF>.
<http://www.resources.org/book>
<http://purl.org/dc/elements/1.1/creator><Shelley Powers>.
```

are serialized as XML:

```
<rdf :RDF
     xmlns:rdf = "http://www.w3.org/1999/02/22-RDFSyntax-ns#"
     xmlns:ex = "http:// www.resources.org/"
     xmlns:dc= "http://purl.org/dc/elements/1.1″/>

      <rdf :Description rdf :about = "ex:book">
            <dc:title>Practical RDF</dc:title>
            <dc:creator> Shelley Powers </dc:creator>
      </rdf:Description>
 </rdf:RDF>
```

The subject is placed inside an *rdf:Description* tag as an *rdf:about* attribute. Inside the description tag, the predicates are defined as elements with the objects included as text nodes inside these elements. After all predicates have been defined for a resource, the description tag is closed. The entire RDF file is enclosed by an *rdf:RDF* tag. The non-hierarchical nature of RDF makes it much harder for a human to read and interpret, but the semantics are much more explicit which allows the data to be parsed and used by a machine.

In XML terms, RDF/XML must be well-formed but there is no way to validate RDF/XML documents because RDF is an open grammar (Daconta, Obrst and

Smith, 2003). Any namespace-qualified elements can be mixed into an RDF document making validation useless; therefore, RDF/XML documents do not use XML schemas or Document Type Definitions (DTD) for validation. An RDF/XML document must conform to the rules of XML and also to the rules of RDF. The RDF specification includes RDF tags that are used to define the parts of a statement. Some of these tags are the Description tag, subject tag, predicate tag, etc. Any RDF/XML document can be parsed into N-triple or graph format through a variety of tools. XML serialization has increased the adoption and usability of RDF, but RDF still struggles for full adoption because of its complex format.

e. Capabilities

RDF provides a way to record data in a machine-understandable format which allows for more powerful and effective data interoperability, searching, cataloging, navigation and classification (Powers, 2003). RDF adds metadata to documents that enhances machine understanding of the contents. RDF is a description for a data model that can be used to describe any specific data vocabulary. Like XML, RDF provides the flexibility to describe anything with its assertion structure.

Current search engines are based on keyword searching. Keyword searching is inaccurate because the engine is not interested in the context of the word. For example, searching for something like “bats”, which has more than one meaning returns data about the animal bat and also about baseball bats. With RDF, developers can describe the context of the data in the form of statements, and machines can use this metadata to perform smarter searches. RDF is also domain neutral which allows terms from different domains to be combined in one document. Combining domains is possible because each domain has a unique URI attached to its terms. The examples of RDF triples above demonstrate combining the Dublin Core domain and the domain represented by http:// www.resources.org. RDF not only provides the cornerstone for the SW, but also provides a way to add meaningful metadata to documents.

3. RDF Schema (RDFS)

RDFS is the language that forms the foundation for the ontology layer of the SW. RDFS is an extension of RDF which provides the ability to define classes and class properties to create a class model called an RDF Schema. The class model is created through a set of tags provided by the RDFS language which define classes and properties. RDF Schemas are written in RDF format as a series of RDF statements. Therefore, RDFS is a language that adds object-oriented features to RDF to create a taxonomy for a specific domain (Lacy, 2005).

Unlike RDF, RDFS does not describe specific resources. Instead, RDFS describes concepts in a machine-interpretable format. The meta-vocabulary defined by RDFS is a collection of class and property definitions which define how an application ought to interpret the RDF statements inside an RDF document (Lacy, 2005). An RDF Schema ensures consistency by verifying that an RDF document is semantically and syntactically the same in various encoding formats and implementations (Powers, 2003). RDFS was created by the RDF Model and Syntax Working Group and has become the W3C standard for basic ontologies (http://www.w3.org/TR/rdf-schema).

RDF Schemas define the terms in a vocabulary and define which terms represent classes and which terms represent properties. The schema also defines which properties describe a specific class and defines the range and datatype for each property. A class represents a concept in the domain. Instances of classes are what become the resources or subjects of RDF statements. RDFS has several built in properties used to define classes such as *rdfs:Class*, *rdfs:subClassOf*, and *rdfs:Comment*. RDFS also defines the properties used to describe the classes within the domain. The properties are the characteristics that describe the classes, and they are represented by the predicate in the RDF triple. Within the RDF Schema the properties are defined, data type information is provided, human-readable comments can be added for clarity, and the relationship between the properties and classes are defined. Properties are defined in terms of the classes to which they apply. The classes that a property can be used to describe are defined through the use of the *rdfs:domain* and *rdfs:range* attributes. The class or classes that a property characterizes is the domain of the property. The set of possible classes or literal values that can be the

object of a property are the range of the property. RDFS is limited to creating hierarchies of classes and specifying minimal constraints on the properties. The main focus of RDFS is to define the terms used in the vocabulary and define the simple semantics of a concept.

Unlike XML schemas, RDF Schemas do not place constraints on the structure of a document. RDF Schemas just provide extra information to machines about how to process and interpret statements given in RDF documents. For this reason, RDF Schemas do not provide validation of RDF files. RDF Schemas inherit the use of datatypes from XML but do not provide this constraint themselves. However, RDF Schemas do enhance the machine's ability to understand and process data. RDF Schemas have the expressive power to build basic ontologies but not enough power to express many constraints or logical rules for the classes and properties (Passin, 2004).

4. Web Ontology Language (OWL)

OWL is an XML vocabulary which extends RDFS to create a fuller ontology representation of a domain. OWL is the W3C standard for creating ontologies for use with the SW (Lacy, 2005). Classes and properties can be defined with cardinality constraints, quantifiers and other property constraints. Reasoning engines are able to enhance OWL ontologies by improving the ontology classification, classifying instances serialized as RDF/XML and checking for semantic consistency. There are three dialects of OWL which offer varying degrees of expressiveness and reasoning capability. A fourth form of OWL is a subset used to describe associated web services. Each of the OWL languages is described below beginning with the least expressive subset.

a. OWL-Lite

OWL-Lite is a strict subset of the OWL language that provides minimum expressiveness and constraint capabilities but provides computationally efficient reasoning. OWL-Lite provides some of the benefits of an OWL ontology without the logical complexity and labor investment necessary for building a full ontology. OWL-

Lite is useful for building taxonomy type ontologies that do not have many properties with restrictions. The tools used to parse and reason on OWL-Lite ontologies are easier to develop and are likely to have a quicker processing time.

OWL-Lite uses the *owl:Class* tag to define a concept or class. For example, the XML below creates a class called “Ship”.

```
<owl:Class rdf:ID="Ship"/>
```

Classes in OWL can also be defined using the *rdfs:Class* tag. The *rdfs:subClassOf*, *rdf:Property*, and the *rdfs:subPropertyOf* are still used to define a class model for the ontology.

OWL-Lite adds the ability to express equality and inequality, property characteristics, some limited property restrictions and limited cardinality constraints on top of RDFS. Classes or properties can be defined as equivalent to each other, but they cannot be explicitly defined as different from each other within OWL-Lite. Classes that are equivalent have the same instances. Instances are individuals that belong to a particular class. For example, a 2005 Honda Accord would be an individual of the class *Cars*. If the class *Cars* is defined as equivalent to the class *Automobile* then any instance belonging to *Car* would also belong to *Automobile*, so a reasoning engine could classify the 2005 Honda Accord as an *Automobile*. The same equivalency relationships can be defined for properties.

Properties can optionally be described as functional, inverse functional, transitive, the inverse of another property, or symmetric with OWL-Lite constructs. Functional properties are properties with only one value per instance. For example, if a pizza is defined with the functional property *hasBase*, then any pizza instance can only have one pizza base. If a property is transitive, and relates x to y and y to z, then the property relates x to z. The property *hasAncestor* can be defined as transitive. If "Bob has an ancestor Sally" and "Sally has an ancestor Patrick", then "Bob has an ancestor Patrick" (Horridge, Knublauch, Rector, Stevens and Wroe, 2004). If a property is the inverse of another property, then the properties relate the same two classes but through a reverse relationship. For example the inverse property of *hasTopping* is the property *isIngredientOf*. Thus "the pizza has topping tomato" and "tomato is an ingredient of pizza" are inverse relationships. The symmetric characteristic defines a bidirectional

property. For example, if Bob has a brother Tom, then Tom has a brother Bob. Giving properties characteristics further defines the domain and allows a reasoning engine to use the characteristics to derive relationships that are not explicitly stated.

Property restrictions in OWL-Lite limit how properties can be used by an instance of a class. Properties are restricted with the *owl:allValuesFrom* and the *owl:someValuesFrom* constructions. The *owl:allValuesFrom* tag places a range restriction on a property which a reasoning engine can use to classify the object of the property. For example, if the property *hasBase* has a restricted range of the class *PizzaBase*, then all objects of the property must be instances of the class *PizzaBase*. If an assertion is made that states *CheesyPizzaOne* hasBase *CheesyBaseOne*, then a reasoning engine can infer that *CheesyBaseOne* is an instance of the class *PizzaBase*. The *owl:someValuesFrom* tag states that at least one object of a property has to be from a specific class. For example, the *owl:someValuesFrom* tag can be used to restrict the *hasTopping* property for the class *CheesyPizza*, to ensure that at least one object of the *hasTopping* property is an instance of the *CheesyTopping* class. These constructs are equivalent to the existential and universal quantifiers in first-order logic. They create more complex definitions for properties by restricting their range.

OWL-Lite allows developers to place limited cardinality restraints on properties. Cardinality restraints restrict the minimum and maximum number of times that a property can be defined for a particular class. OWL-Lite cardinality restrictions are limited to cardinalities of zero or one. Arbitrary cardinality restraints are not permitted with the OWL-Lite language in order to keep computational complexity tractable.

b. OWL-Description Logic (OWL-DL)

OWL-DL provides more expressive power than OWL-Lite, but still supports reasoning applications. OWL-DL includes the full OWL language but places restrictions on the use of some of the description tags. These restrictions ensure that the computations formed by a reasoning engine on OWL-DL ontologies can be completed in a finite amount of time (Lacy, 2005). OWL-DL is the compromise between OWL-Full and OWL-Lite and is the most-used dialect for representing domain ontologies for the

SW. OWL-DL provides a vast amount of inference capabilities, but it is still somewhat limited in the constraints that can be placed on property and class definitions.

OWL-DL includes constructs called class expressions which are used to create complex classes. Class expressions describe class membership criteria through a combination of enumerations, Boolean expressions, and property restrictions. The *owl:oneOf* tag is used to define an enumerated class. Enumerated classes are classes with a predefined, known and finite-member set. They can be defined by listing all the members. For example, a class called *DaysOfTheMonth* can have an enumerated list of thirty-one members. Boolean expressions apply the intersection, union, and complement operators to existing classes to define new classes. OWL-DL adds a new property restriction type with the *owl:hasValue* tag. This tag is used to identify membership in a class based on the properties it has and the value of the properties. Complex class definitions add the ability for a reasoning engine to classify individuals as members of a particular class through inference. OWL-DL also removes the cardinality restrictions placed on OWL-Lite. Properties defined by OWL-DL can have any integer value as the minimum and maximum cardinality. Less restriction in OWL-DL can increase computation time for reasoning engines, but it also increases the machine's knowledge of the domain concepts and relationships.

OWL-DL constructs have restrictions on expressivity to ensure finite computation time. For example, cardinality cannot be defined for transitive properties. Therefore, the restrictions on expressivity must be weighed against the gain in computational tractability and responsiveness. The OWL dialect chosen for an ontology depends on the intended use of the ontology and the domain that needs to be expressed.

c. OWL-Full

OWL-Full is the complete OWL language with no restrictions on RDF files. However, reasoning computations on OWL-Full present challenges because potentially sophisticated computations cannot be guaranteed to complete in finite time. OWL-Full is useful for fully describing domains, but does not provide machines the ability to make useful inferences to be used in the SW. OWL-Full constructs might not be

processed and computed in finite time because of their potential complexity. OWL-Full remains an excellent mechanism for a full representation of a specific domain for vocabulary mapping purposes if automated reasoning is not required. Tools can identify which expressions in an OWL-Full definition exceed the scope of OWL-Lite and OWL-DL, potentially simplifying the construction of simpler ontologies based on more-complete ontologies.

d. OWL-Services (OWL-S)

OWL-S is the OWL language being developed as the technology to build web services on the SW. OWL-S creates models to define web service discovery, invocation, interoperation, composition, verification and execution monitoring in a machine-interpretable format. OWL metadata enables machines to perform automatic discovery of the web services needed to complete their task. Ontology representations of the web service entities allow a machine to call and use web services, and an advanced representation of such services can allow a machine to combine web services to perform new automated tasks (Alesso and Smith 2005). OWL-S is a powerful use of the OWL language that brings the SW closer to SW Agents.

5. Semantic Web Rule Language (SWRL)

SWRL is the language that provides the foundation for building FOL rules about an ontology. There are many rule languages that can be used to define logic for ontologies, but SWRL is becoming the W3C standard. SWRL rules are written in terms of OWL classes and properties, and they are used to reason about OWL instances (O'Connner, Knublauch, Tu and Musen, 2005). When matched with a rule reasoner, SWRL adds more inference power to OWL ontologies.

Logic rules consist of two parts: the antecedent and the consequence. In SWRL the antecedent is called the *rule body* and the consequence is called the *head.* The rule body and head are made up of one or more *atoms*. When all conditions stated in the rule

body are true then the head is implied and executed. Complex rules are created by combining simple FOL rules.

A SWRL rule can be used to define a predicate. For example, the following rule:

Person(Bob) ^ *hasSibling*(Bob, Joe) ^ *Man*(Joe)
→ *hasBrother*(Bob, Joe)

defines the property *hasBrother* in relation to the properties *Person*, *hasSibling*, and *Man*. This SWRL rule states that if Bob is a *Person* and Bob *hasSibling* Joe and Joe is a *Man* then Bob *hasBrother* Joe (O'Connner, Knublauch, Tu, & Musen, 2005). If an instance of Person called Bob is defined and has the properties *hasSibling*(Bob, Joe) and Joe has the property *Man*(Joe) then a SWRL reasoning engine can add the property *hasBrother*(Bob, Joe) to the instance Bob. The antecedent or rule body: *Person* (Bob) ^ *hasSibling* (Bob, Joe) ^ *Man* (Joe), implies the consequent or head *hasBrother* (Bob, Joe), when all the conditions are true. OWL inference is limited to classifying instances into classes, but the addition of SWRL allows new properties to be inferred for instances of a class.

SWRL also supports literals within the rules. For example, the following rule:

Person(Bob) ^ *hasSibling*(Bob, Joe) ^ *Man*(Joe) ^
hasAge(Joe, 40) → *has40YearOldBrother*(Bob, Joe)

states that if Joe is 40 years old then Bob *has40YearOldBrother*. Figure 21 shows the SWRL source for the rule defined above.

```
<swrl:Imp rdf:ID="Rule-1">
 <swrl:body>
   <swrl:AtomList>
     <rdf:first>
       <swrl:ClassAtom>
         <swrl:classPredicate rdf:resource="#Person"/>
         <swrl:argument1>
           <Person rdf:ID="Bob"/>
         </swrl:argument1>
       </swrl:ClassAtom>
     </rdf:first>
     <rdf:rest>
       <swrl:AtomList>
         <rdf:rest>
           <swrl:AtomList>
             <rdf:first>
               <swrl:ClassAtom>
                 <swrl:argument1 rdf:resource="#Joe"/>
                 <swrl:classPredicate rdf:resource="#Man"/>
               </swrl:ClassAtom>
             </rdf:first>
             <rdf:rest>
               <swrl:AtomList>
                 <rdf:first>
                   <swrl:DatavaluedPropertyAtom>
                     <swrl:argument1 rdf:resource="#Joe"/>
                       <swrl:argument2
                       rdf:datatype="http://www.w3.org/2001/XMLSchema#int">40
                     </swrl:argument2>
                     <swrl:propertyPredicate rdf:resource="#hasAge"/>
                   </swrl:DatavaluedPropertyAtom>
                 </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
               </swrl:AtomList>
             </rdf:rest>
           </swrl:AtomList>
         </rdf:rest>
         <rdf:first>
           <swrl:IndividualPropertyAtom>
             <swrl:argument2 rdf:resource="#Joe"/>
             <swrl:propertyPredicate rdf:resource="#hasSibling"/>
             <swrl:argument1 rdf:resource="#Bob"/>
           </swrl:IndividualPropertyAtom>
         </rdf:first>
       </swrl:AtomList>
     </rdf:rest>
   </swrl:AtomList>
 </swrl:body>
 <swrl:head>
   <swrl:AtomList>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     <rdf:first>
       <swrl:IndividualPropertyAtom>
         <swrl:argument2 rdf:resource="#Joe"/>
         <swrl:argument1 rdf:resource="#Bob"/>
         <swrl:propertyPredicate rdf:resource="#has40YearOldBrother"/>
       </swrl:IndividualPropertyAtom>
     </rdf:first>
   </swrl:AtomList>
 </swrl:head>
</swrl:Imp>
```

Figure 21. The SWRL source for a rule defining the property, *has40YearOldBrother*.

The power of supporting literals is increased with the support of built-in predicates like comparison operators. For example the following rule determines if Bob has an older brother:

hasBrother(Bob, Joe) ^ *hasAge*(Bob, 30) ^ *hasAge*(Joe, 40) ^ *swrlb:greaterThan*(40, 30) → *hasOlderBrother* (Bob, Joe)

The built-in property *swrlb:greaterThan* (40, 30) is evaluated to true along with the other antecedents, which makes the consequence that Bob has an older brother Joe true. SWRL includes built-in comparison operators; math operators such as add, divide, power, and round; string operators like concatenate and length; date and time operators; and many others. A full list of built-in operators is listed in the SWRL Built-in Specification (http://www.daml.org/rules/proposal/builtins.html).

SWRL provides a standard syntax for defining first-order rules in a machine-interpretable language. SWRL is currently in the proposal phase at the W3C and has not been defined as the standard SW rule language. However, SWRL is a powerful rule language that is used to further define OWL properties.

D. SUMMARY

The SW is a multi-layered technology that provides data exchange interoperability between domains, more accurate and efficient indexing and information retrieval, inference capabilities and automation through the use of Software Agents.

Data interoperability is achieved through ontology definitions and mappings. For example, if two airline companies represent their data with a different vocabulary, but both companies create an OWL ontology, then the two distinct ontologies can be interpreted and mapped to a common ontology. Software can then use the common ontology to query and analyze information from each company.

Semantic markup enables machine understanding of data, thus improving index and searching capabilities. Users can further define search keywords using assertions to avoid ambiguity. Machines use interpretable assertions to ensure that searches are precise

and results are accurate. Each of these uses can benefit the discovery, collection and analysis of tactical information in the military.

The ontology and logic layers of the SW add the capability for machines to infer knowledge about data that is not specifically stated. Rules and restrictions for domain knowledge are written in a standardized format, so a SW reasoning engine can interpret specific rules for handling data in a specific domain. Thus new tools do not have to be developed to parse and analyze every different domain.

The ultimate goal of the entire SW layer is to provide information to machines that allows them to collect, analyze, infer and use data to complete sophisticated tasks. SW Software Agents are programmed to import and analyze such ontologies to learn the specific vocabulary defining certain information. The information inside the ontology provides the agent the semantic definitions it needs to analyze and act on the structured input information it receives. The benefit of SW Agents is expected to be their ability to pull and use information from any domain defined with SW languages. SW Agents are not affected by changes in assumptions or data structures, because they can import the new ontology and use it to process data.

The SW offers a promising future for the retrieval and use of data from a wide variety of sources. Automation can increase the capabilities of the military by allowing larger amounts of data to be processed in shorter amounts of time. Thus bringing the Armed Forces closer to net-centric warfare. The growth of available data and the need to process the data quickly highlights the need for the automation provided by the SW.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SERIALIZING TAML DOCUMENTS AS RDF/XML

> RDF is a framework for supporting resource description, or meta-data (data about data), for the web. RDF provides common structures that can be used for interoperable XML data exchange.
>
> -- W3C Semantic Web Activity Statement, 2002

A. INTRODUCTION

This chapter provides an overview of the process used to serialize TAML documents into RDF/XML documents, and describes the advantages and disadvantages of an RDF/XML document over an XML document. This chapter also explores the querying capabilities available for RDF documents by illustrating sample queries. The queries are run against the RDF/XML documents created to demonstrate the differences between RDF query languages and the XQuery language created for XML documents.

B. WHY USE THE RESOURCE DESCRIPTION FRAMEWORK (RDF)?

1. Advantages

RDF is the language that converts semantic information about data into a machine understandable format. The triple format of RDF allows information to be written as statements with explicit semantics. Each RDF statement is similar to a basic English sentence and is made up of a subject, predicate and object. The explicit semantics of the language ensure each statement has only one meaning or interpretation. By removing ambiguity, the triples enable machines to interpret the meaning of the statements.

In XML, structure and order are important if a parser is expected to process the document; however, order does not matter in RDF. An RDF parser can process the statements or triples in any order and come up with the same graph representation. Thus additional triples can be added asynchronously at runtime without loss of logical generality and without incurring excessive computation burdens.

XML utilizes complex nested structures which have to be represented in memory as a treelike structure in order for the data to be queried. XML Query (XQuery) is the standard language used when querying XML instance documents. XQuery uses the XPath language to process the nodes in the XML tree. XPath follows a directory-structured command down the leaves of the tree in order to find the correct node, which requires prior knowledge of the document structure for querying.

RDF decreases the complexity of querying data, so that more sophisticated queries become feasible (Powers, 2003). The graph representation of RDF documents is flat, which means that the only prior knowledge requirement for a parser is knowledge of the triple format and the URI of the elements to be queried. The triples are laid out so that they can be processed without having to work around presentation and organizational constructs (Powers, 2003). The querying engine does not need to traverse the entire document to answer a query, which makes RDF querying more efficient than XML querying in some cases (Powers, 2003).

The explicit semantics of an RDF document often provides easier querying because humans typically understand the semantics of the data they are querying but may not necessarily know the structure of the data. RDF effectively removes the requirement to know the structure of the data, thus RDF querying is easier, more efficient, and more powerful than XML querying.

RDF/XML still maintains the interoperability advantages of XML since the RDF/XML format itself is well-formed XML. There are other serializations of RDF, but RDF/XML is the most widely used because it can be accepted and processed by any XML parser. Therefore, it is possible to exchange RDF/XML documents between machines that accept XML documents.

RDF enhances the interoperability of XML by allowing vocabularies to be combined within a document. Combining vocabularies enables a resource to be described by properties from two different domains. Data from two different vocabularies can therefore be joined together without having to resolve structural differences between them (Powers, 2003). Unique namespaces between the vocabularies ensures uniqueness and prevents name collision. For example, properties from another vocabulary like the

Dublin Core vocabulary might be inserted into TAML RDF documents to further describe the resources in the document. The named resources inserted from the Dublin Core vocabulary are preceded by the unique Dublin Core namespace. Therefore, if an element inserted from Dublin Core has the same name as a TAML element, the machine is able to distinguish each element based on their prefixed namespace. Furthermore, adding an RDF Schema to a document allows equivalency statements to be made. Equivalency statements define when two properties from different domains have the same semantic meaning. Equivalency statements are used to map domains to enable interoperability among them. RDF enhances the interoperability of XML vocabularies through structural flexibility and simplicity.

The amount of tactical data available in the armed forces is staggering and requires automation in order to be processed and queried quickly. RDF provides a powerful mechanism for cataloging, retrieving and querying data (Powers, 2003). The RDF data model provides increased automation for finding and querying data stored as RDF triples. TAML documents are also filled with data that can be archived as historical information, which may need to be queried from large stores of information. The 10g Database released by Oracle for grid computing includes an RDF data model to store RDF triples (Ayers, 2005). An RDF representation of TAML can thus provide automation and efficiency improvements even for large tactical databases.

2. Disadvantages

Although RDF is powerful, it does not provide the best characteristics for all situations. RDF/XML increases the size of documents, thereby increasing overhead for exchanging information. In addition, RDF/XML increases document complexity, making RDF/XML documents harder for a human user to understand.

Serializing data into RDF/XML increases the size of the document. The same information is represented in a machine-interpretable format, but the size of an RDF/XML document is almost double the size of the XML representation. TAMLExample.xml is a TAML instance document with a size of 12.4 kilobytes. When the same data contained in TAMLExample.xml is serialized as RDF/XML it becomes a

19.6 kilobyte file. The larger file size introduces overhead during message exchange over networks. In most networks limited bandwidth does not present an issue, but the bandwidth available for exchanging tactical messages between ships is quite limited. Specialized XML compression shrinks the size of the file, but the XML format of TAML is the better choice for the task of exchanging data. However, the RDF/XML format may be the better choice for storing the data for later retrieval and querying. It is possible to benefit from both serializations, since the RDF/XML information can be kept in a separate file. The Extensible Stylesheet Language for Transformations (XSLT) can be used to convert TAML from XML to RDF/XML, so the use of TAML documents can be optimized.

RDF/XML adds a layer of complexity to TAML documents. The complexity makes it difficult for humans to browse and understand the information portrayed in an RDF/XML document. Humans understand the hierarchical format of XML, so RDF/XML is a trade-off between human readability and machine readability. XML may be the better choice for exchanging messages intended to be processed by both humans and machines, since RDF/XML increases automation at the expense of human readability.

RDF/XML presents advantages and disadvantages that need to be weighed by developers. The context of information use should determine the form of information representation.

C. DESIGN

Trafalgar.xml and TAMLExample.xml are two TAML documents that were used to prototype the RDF/XML serialization of TAML. The resulting RDF/XML serialized files are TrafalgarExample.rdf and TAMLExample.rdf. A subset of TrafalgarExample.rdf is located in Appendix A. TAML was created as a message exchange format using a hierarchical XML format easily understood by humans. The properties and resources are not explicitly defined within TAML, so the information cannot be processed for meaning by machines. The information contained in the TAML documents is rewritten as

RDF/XML in order to add explicit meaning to the data for enhanced machine interpretation.

1. Resources

TAML documents are tactical messages that describe events like tracking operations and entities like platforms and contacts. The events and entities are described as elements within the TAML instance documents and become resources in the RDF/XML serialization. A resource within RDF is the object being described. Each resource in RDF is identified by a unique name to remove ambiguity. Within the TAML vocabulary, *id* attributes are used to uniquely identify entities or elements like *Operation, Event, Platform*, and *Contact* as shown in Figure 22.

```
<Operation id="T001">
      <Name>Trafalgar</Name>
</Operation>
```

Figure 22. Fragment of Trafalgar.xml illustrating the use of *id* attributes.

Unique names must be created for the information contained in the RDF/XML document from the existing information in the Trafalgar.xml document. Figure 23 shows a subset of TrafalgarExample.rdf which demonstrates how a unique name was created for operations, platforms, and other elements identified in TAML with an *id* attribute.

```
<rdf:Description
rdf:about="http://usw.xml.wg/Trafalgar.xml/OperationT001">
      <taml:Name>Trafalgar</taml:Name>
</rdf:Description>
```

Figure 23. Fragment of TrafalgarExample.rdf demonstrating how unique names are created for operations.

The entity type in the subset above is *Operation* and the *id* is T001. The unique name identifying the entity is a concatenation of the file pathname, the entity type and the

id number. The file pathname ensures the name is unique even if an operation in another TAML instance document has the same *id* attribute. The entity type adds information to the name about what is being described, and the inclusion of the *id* within the name ensures the uniqueness of the entity within the document. This pattern of forming entity names by concatenation is maintained for all elements identified with *id* attributes within the TAML vocabulary.

TAML does not use *id* attributes to uniquely identify all elements. For example, in Figure 24, *Configuration* and *Track* are child elements of element *Platform* and do not contain their own *id* attribute, relying on *Platform's id* instead. *Configuration* and *Track* are child elements of other elements (i.e. *Contact*) as well.

```
<Platform id="HMSFLAG">
      <Name>Victory</Name>
      <Configuration>
            <ConfigurationItem id="CFGFLAG">100
                  Guns</ConfigurationItem>
            <ConfigurationItem id="CMDFLAG">Capt. T. M.
                  Hardy</ConfigurationItem>
      </Configuration>
      <Track>...</Track>
</Platform>
```

Figure 24. Fragment of Trafalgar.xml showing child elements as properties of their parent element.

These child elements become objects of triples describing their parent element and become the subject of their own triples in the RDF/XML serialization. Since these elements are directly related to a particular resource with a unique name, that name is concatenated with the element name to form a unique name for the new resource as shown in Figure 25. The name of the track associated with *PlatformHMSFLAG* becomes *TrackPlatformHMSFLAG*. The unique name given to the track illustrates that it is semantically related to *PlatformHMSFLAG* and not to any other *Platform*.

```
<rdf:Description
      rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMSFLAG">
      <taml:Track
      rdf:resource="http://usw.xml.wg/Trafalgar.xml/TrackPlatformHMSF
      LAG" />
</rdf:Description>
```

Figure 25. Fragment of TrafalgarExample.rdf showing how unique names are created for elements without unique *id* attributes.

The patterns used to name the specific resources described within the TAML instance documents ensure the names are unique internally and externally to the document. The patterns also add consistency to the naming convention to make querying resources simpler for applications.

2. Predicates

Many of the elements within TAML are used to describe their parent elements. Any element that describes its parent becomes a predicate in the RDF/XML file. Figure 26 illustrates a series of child elements describing the element *Contact.*

The elements *DateTimeGroup*, *Position*, *Orientation*, *Course*, *Speed*, *Frequency*, *ContactDesignator* and *ReportingStatus* are characteristics of the Contact with an *id* of x20, so each of these elements becomes the predicate of a triple describing the resource *ContactIDx20* as shown in Figure 27. The *taml* prefix references the TAML namespace which is used to uniquely identify the TAML vocabulary elements. The prefix precedes each property, since the property terms are defined in the TAML vocabulary. The XML also includes the attribute *sensorCode*, which describes *ContactIDx20*, so it also becomes a predicate in the RDF/XML serialization.

```
<Contact id="x20" sensorCode="Active">
      <DateTimeGroup>
            <DateTime>2005-05-12T14:00:00Z</DateTime>
      </DateTimeGroup>
      <Position>
            <AbsolutePosition>
                  <Latitude>22.12345</Latitude>
                  <Longitude>-121.123456</Longitude>
            </AbsolutePosition>
      </Position>
      <Orientation/>
      <Course>0</Course>
      <Speed>0</Speed>
      <Frequency>110</Frequency>
      <ContactDesignator>T1234</ContactDesignator>
      <ReportingStatus>GAIN</ReportingStatus>
</Contact>
```

Figure 26. Fragment of Trafalgar.xml showing child elements which describe element *Contact.*

```
<rdf:Description
      rdf:about="http://usw.xml.wg/TAMLExample.xml/ContactIDx20">
     <taml:sensorCode>Active</taml:sensorCode>
     <taml:DateTime>2005-05-12T14:00:0Z</taml:DateTime>
     <taml:Position>Absolute Position</taml:Position>
     <taml:Latitude>22.12345</taml:Latitude>
     <taml:Longitude>-121.123456</taml:Longitude>
     <taml:Orientation/>
     <taml:Course>0</taml:Course>
     <taml:Speed>0</taml:Speed>
     <taml:Frequency>0</taml:Frequency>
     <taml:ContactDesignator>T1234</taml:ContactDesignator>
     <taml:ReportingStatus>GAIN</taml:ReportingStatus>
</rdf:Description>
```

Figure 27. Fragment of TrafalgarExample.rdf showing child elements as properties of their parent class *Contact*.

The above steps are duplicated to create the predicates for each resource within the RDF/XML document. The TAML vocabulary acts as a prototype for the RDF semantics. Since the semantics of the TAML vocabulary are not explicitly defined by the schema, the meaning of the elements has to be interpreted by a human and translated to a set of RDF triples. The RDF/XML file states the parent/child relationships of the TAML hierarchy as explicit property relationships, which can then be interpreted by machines.

Predicates or properties are the linking mechanism used to illustrate relationships between objects and other objects, or between objects and literal values.

3. Objects

The object of an RDF/XML triple is either a literal value or another resource with a unique identifier. The RDF/XML serialization of TAML contains both objects that are literals and objects that are resources. The objects that are also resources are further described within the RDF document by their own set of triples. When the document is parsed, these objects are linked to the statements describing them by creating a graph structure of logical chains where the object of one triple is the subject of another triple.

The TAML Schema defines datatypes for the data stored within certain elements. The elements that store text or data are redefined as properties describing their parent element in the TAML RDF/XML serialization. The data stored within the element is redefined as the object of the property with a literal value. Figure 28 illustrates a fragment of TAMLExample.xml converted to RDF/XML. The objects are highlighted in bold text to illustrate the consistency within the change. In the RDF/XML serialization below, the triple object remains a literal value.

```
<Event id="x2">
      <StartTime>2005-05-12T14:00:00Z</StartTime>
      <EndTime>2005-05-13T14:00:00Z</EndTime>
      <Name>Event x2</Name>
</Event>
```

```
<rdf:Description
rdf:about="http://usw.xml.wg/TAMLExample.xml/Eventx2">
      <taml:StartTime>2005-05-12T14:00:00Z</taml:StartTime>
      <taml:EndTime>2005-05-13T14:00:00Z</taml:EndTime>
      <taml:Name>Event x2</taml:Name>
</rdf:Description>
```

Figure 28. Comparison of XML serialization (above) and RDF serialization (below) of literal objects.

Within the TAML vocabulary, many elements describe other elements. In RDF the relationship between these elements is explicitly defined by a triple. The parent element is the subject of the triple and the child element is the object of the triple. The property of the triple is the relationship between the two elements.

The name of the child element describes the relationship; therefore, it becomes the predicate of the triple as shown in Figure 29. The triple in Figure 29 states that *EventX2* has a *PlatformRef* with the name http://usw.xml.wg/TAMLExample.xml/Platformx5. The object of the triple is the specific entity that is related to the subject. Since the object is a specific entity with a unique identifier, it is further defined within the document.

```
<Event id="x2">
      <PlatformRef idref="x5"/>
</Event>
```

```
<rdf:Description
      rdf:about="http://usw.xml.wg/TAMLExample.xml/Eventx2">
      <taml:PlatformRef
      rdf:resource="http://usw.xml.wg/TAMLExample.xml/Platformx5"/>
</rdf:Description>
```

Figure 29. Illustration of a resource as an object in TAMLExample.rdf.

The ability of RDF to define resources as objects of statements allows for linkage of entities throughout the document to create complete semantics. When the document is parsed the linkages are mapped out to allow quick processing, understanding and querying of the document.

4. Containers

The TAML vocabulary allows for multiple elements of the same name to be listed under one parent element as shown in Figure 30. In the example, the parent element *Track* has two child elements named *TimePosition.* Semantically this represents that a *Track* is composed of multiple *TimePosition* elements. Each *TimePosition* is a unique entity with its own characteristics like *DateTimeGroup* and *AbsolutePosition* as shown in

Figure 30. Each *TimePosition* is a resource with a unique name in the RDF serialization. The name is created using the same pattern described in the "Resources" section above. However, each *TimePosition* is more than a unique independent entity, since each one is related to the other through their relationship with the parent element *Track*.

```
<Track type="RAW" source="Inertial">
      <TimePosition id="x15">
            <DateTimeGroup>
                  <DateTime>2005-05-12T14:00:00Z</DateTime>
            </DateTimeGroup>
      </TimePosition>
      <TimePosition id="x16">
            <DateTimeGroup>
                  <DateTime>2005-05-12T14:15:00Z</DateTime>
            </DateTimeGroup>
            <Position>
                  <AbsolutePosition>
                        <Latitude>22.22345</Latitude>
                        <Longitude>-121.223456</Longitude>
                  </AbsolutePosition>
            </Position>
      </TimePosition>
```

Figure 30. Example showing several elements of the same name *TimePosition* under one parent element, *Track*.

Track is made up of several *TimePosition* elements in a specific sequence. The rdf:*Seq* property is used to express the relationship of composability in the TAML RDF/XML serialization. Figure 31 illustrates that *Track* is made up of a sequence or ordered list of *TimePosition* elements. Each *TimePosition* is a resource that is further defined within the document.

```
<rdf:Description
  rdf:about="http://usw.xml.wg/TAMLExample.xml/TrackPlatformx5">
     <taml:TimePosition>
       <rdf:Seq>
          <rdf:li
       rdf:resource="http://usw.xml.wg/TAMLExample.xml/TimePositionIDx15" />
       <rdf:li
       rdf:resource="http://usw.xml.wg/TAMLExample.xml/TimePositionIDx16" />
        </rdf:Seq>
      </taml:TimePosition>
</rdf:Description>
```

Figure 31. Example of the use of rdf:*Seq* in TAMLExample.rdf.

The TAML vocabulary defines a *Search* element to describe the contact searches done by a platform. Each search consists of one or more physical searches completed by different systems on the ship. The individual search results are not necessarily listed in any sequential order; therefore, the rdf:*Bag* property is used to signify a collection relationship with no particular order as shown in Figure 32.

```
<taml:Search>
   <rdf:Bag>
       <rdf:li
       rdf:resource="http://usw.xml.wg/TAMLExample.xml/Platformx5Search1"/>
       <rdf:li
       rdf:resource="http://usw.xml.wg/TAMLExample.xml/Platformx5Search2/>
   </rdf:Bag>
</taml:Search>
```

Figure 32. Example of the use of rdf:*Bag* in TAMLExample.rdf

Containers make it possible to collect objects together in ordered or unordered lists. The ability to express a collection relationship among resources increases the machine-understandability of TAML semantics. For example, if each *TimePosition* had to be the object of a separate triple then there is no semantic indication that the set of *TimePosition* elements together make up the *Track* entity. With RDF containers, this type of collective semantic relationship is explicitly defined.

5. Reification

Reification is a property of RDF that allows the language to make statements describing other statements (Powers, 2003). In reification a statement becomes the object of another statement. Reification is useful for annotations of other's work or for implied assertions. Reification takes RDF beyond simple statements by allowing the annotation of those statements.

The TAML instance documents do not contain data that required the use of reification. Reification is used when the comments made about the statement are not facts but abstract ideas. For example, reification is used to express abstract characteristics about statements like confidence levels or recommendations for a resource. The information contained in TAML documents consists of facts about events and resources. There are no confidence or trust levels associated with the data, but reification could provide a way to add these abstract concepts to the data. Adding confidence levels to data can increase the richness of semantic expressions and enable software applications to use data with certain levels of trust. If an application found conflicting information about the same resource within different TAML documents, the information might be resolvable by processing confidence levels; thus reification can add valuable information to TAML documents. This is an important area for future work.

6. Development Tools

XMLSpy, an Altova product, was used to create and edit the RDF/XML serialization of the TAML instance documents (http://www.altova.com/products/xmlspy/xml_editor.html). XMLSpy is an XML editing tool with a built-in XML parser that ensures a file is well-formed XML.

SemanticWorks, another Altova product, was used to create a graphical representation of the RDF/XML files. After creating TrafalgarExample.rdf and TAMLExample.rdf, a graphical view of the files was created. The graphical view displays the triples in a human-friendly manner, which improved the checking and correction process. The graphical view also shows the links made throughout the document when resources were listed as objects of triples and then further defined later in

the document as shown in Figure 33. SemanticWorks provides a graphical view of each resource or predicate independently and shows how they connect to other resources.

The W3C RDF online parser (http://www.w3.org/RDF/Validator) was used to verify that the RDF/XML files were valid RDF. The W3C parser can also convert RDF/XML to N-Triple format as shown in Figure 34. The simpler N-Triple format allows developers to check their work and ensure their statements portray the intended semantics.

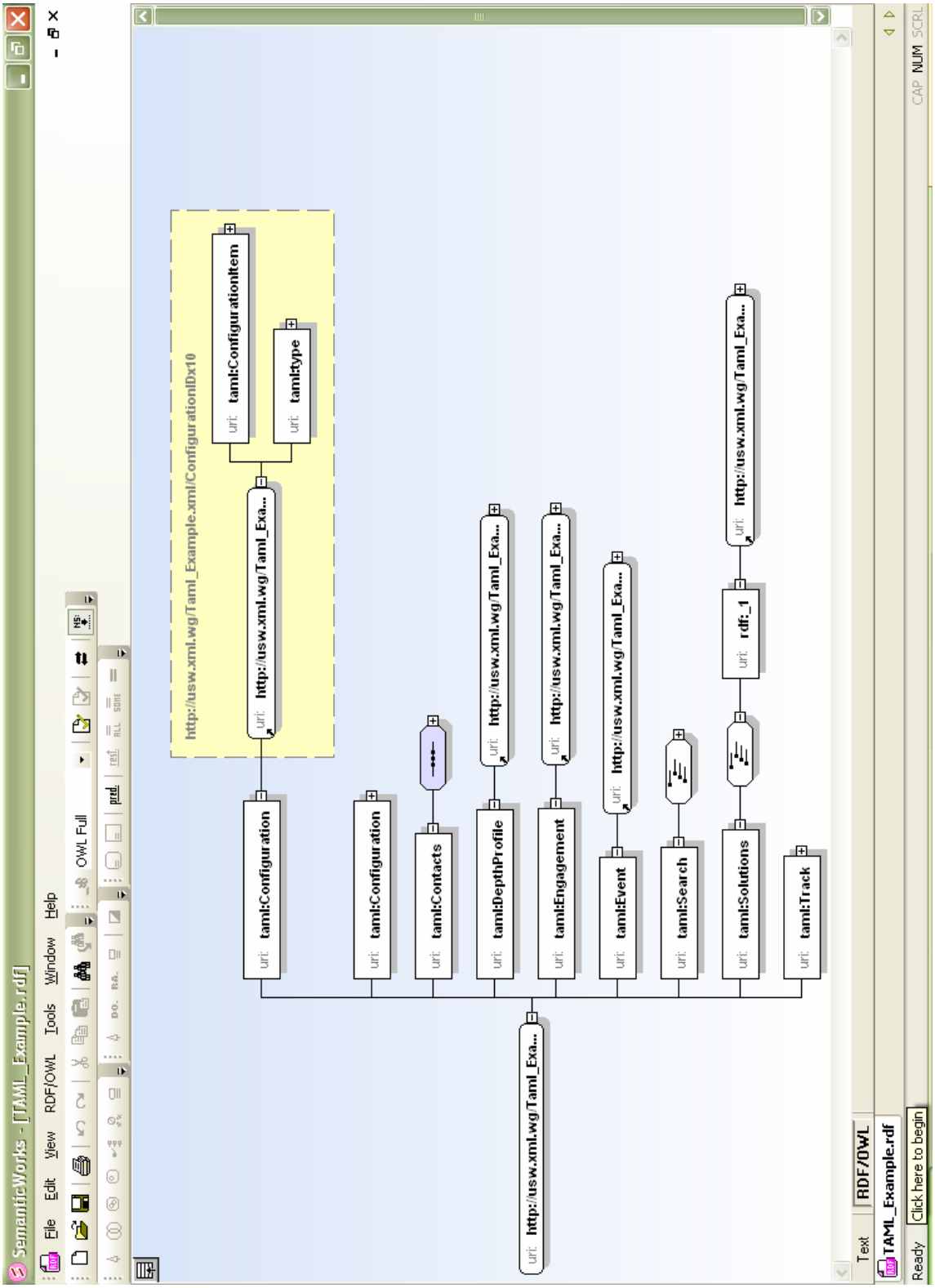
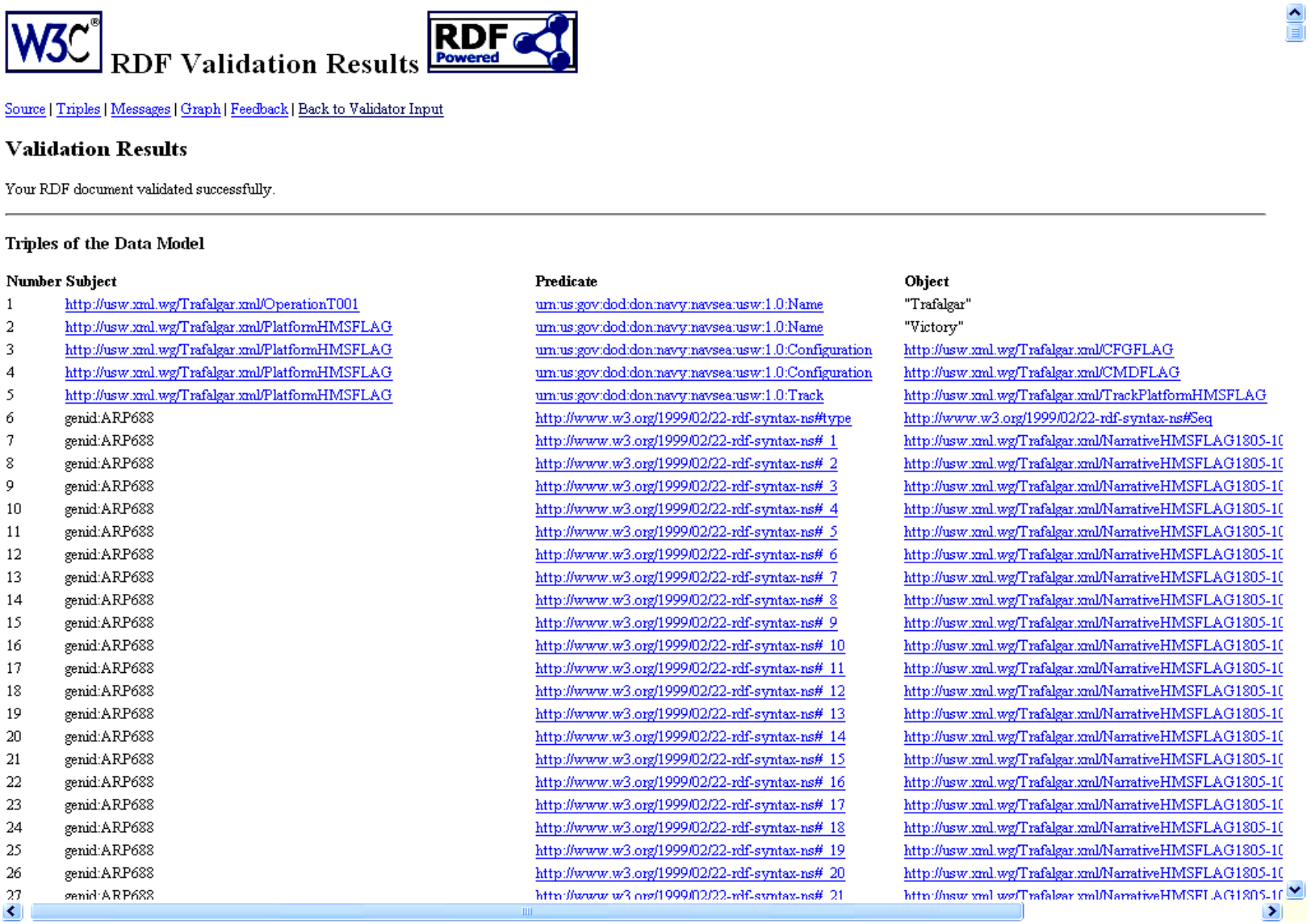


Figure 33. Screenshot of TAMLExample.rdf graph from Altova SemanticWorks.



RDF Validation Results

Source | Triples | Messages | Graph | Feedback | Back to Validator Input

Validation Results

Your RDF document validated successfully.

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://usw.xml.wg/Trafalgar.xml/OperationT001	urn:us:gov:dod:don:navy:navsea:usw:1.0:Name	"Trafalgar"
2	http://usw.xml.wg/Trafalgar.xml/PlatformHMSFLAG	urn:us:gov:dod:don:navy:navsea:usw:1.0:Name	"Victory"
3	http://usw.xml.wg/Trafalgar.xml/PlatformHMSFLAG	urn:us:gov:dod:don:navy:navsea:usw:1.0:Configuration	http://usw.xml.wg/Trafalgar.xml/CFGFLAG
4	http://usw.xml.wg/Trafalgar.xml/PlatformHMSFLAG	urn:us:gov:dod:don:navy:navsea:usw:1.0:Configuration	http://usw.xml.wg/Trafalgar.xml/CMDFLAG
5	http://usw.xml.wg/Trafalgar.xml/PlatformHMSFLAG	urn:us:gov:dod:don:navy:navsea:usw:1.0:Track	http://usw.xml.wg/Trafalgar.xml/TrackPlatformHMSFLAG
6	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq
7	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_1	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
8	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_2	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
9	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_3	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
10	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_4	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
11	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_5	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
12	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_6	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
13	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_7	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
14	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_8	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
15	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_9	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
16	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_10	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
17	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_11	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
18	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_12	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
19	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_13	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
20	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_14	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
21	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_15	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
22	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_16	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
23	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_17	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
24	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_18	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
25	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_19	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
26	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_20	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10
27	genid:ARP688	http://www.w3.org/1999/02/22-rdf-syntax-ns#_21	http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10

Figure 34. Screenshot of the W3C online parser used to validate RDF/XML. Available online at http://www.w3.org/RDF/Validator.

The tools used to develop the RDF/XML serialization of TAML documents ensure that the RDF document is well-formed XML, well-formed RDF and has well-formed semantics.

D. QUERY LANGUAGES

Query languages enable users to extract data from documents that match defined criteria. Data is useless if it cannot be extracted from large documents. Different query languages offer different capabilities. This section compares the capabilities of two query languages: XQuery and SPARQL.

1. XQuery

XQuery, released in 2002, is the W3C Recommendation query language for processing XML data (Bothner, 2002). The use of XML to mark up data is expanding along with the different methods of storage such as relational databases, file storage and native XML databases. The varying storage systems require a common method for access or XML loses its advantage of syntactic interoperability (Hunter, 2004). XQuery is that method because it works independently of the storage system and removes the need for XML developers to develop proprietary methods for data access. Any XML data can be queried using the XQuery syntax and tools that implement XQuery parsing.

The goal of XQuery is to provide a standard language for querying large data sources. XQuery is designed and optimized for retrieval of data from large collections or stores of XML data (Hunter, 2004). Query optimization and XML indexing are provided based on the document schema in order to handle the large sizes. Error handling is also strict to ensure that XML provides capabilities similar to large databases. XQuery takes XML as input and filters the XML source tree to create a result. The language includes a library of built-in functions and allows developers to define their own functions for filtering data. XQuery is not written using XML syntax, so it is easier to use with other programming languages (Hunter, 2004).

Queries written in XQuery require a knowledge of the internal structure established by the XML schema defining the data being queried. Queries are written using XPath expressions. XPath is a language used to locate data within a document by providing a class path for an element. For example the XPath expression /Book/Chapter[@number=1] returns the value "This is the first chapter" (Hunter, 2004) when processed on the XML fragment shown in Figure 35.

```
<Book>
     <Chapter number="1">This is the first chapter</Chapter>
     <Chapter number="2">This is the second chapter</Chapter>

     <Chapter number="3">This is the third chapter</Chapter>
     <Chapter number="4">This is the fourth chapter</Chapter>
<Book>
```

Figure 35. XML describing the make-up of a book [After Ref. Hunter, 2004].

XPath parses through the XML tree representation of a document to retrieve data. In the above example the expression lays out the path for the parser to take and returns the value when the end of the path is reached. Queries written in XQuery rely on XPath expressions to define where the requested data can be found within the document. Figure 36 illustrates a query for all book elements and the results returned from the associated XML code.

Querying XML documents with XQuery requires prior knowledge about the structure of the documents being queried. Thus queries over multiple documents defined by multiple schemas may become too complex to write, maintain and understand. Some users may not have prior knowledge to the structure of data they are trying to query and the syntax of XQuery is too complex to be used for random data querying. XQuery is best for writing queries which are expected be used over and over, so the overhead of writing the code is outweighed by its repeated use. Thus an alternative method for querying data is sometimes desirable for building query applications where users without knowledge of the data structure are able to write queries for retrieving information.

```
<bib>
      <book year="1994">
            <title>TCP/IP Illustrated</title>
      </book>
      <book year="1992">
            <title>Advanced Programming in the Unix
                  Environment
            </title>
      </book>
      <magazine year="2000">
            <title>Scientific America</title>
      </magazine>
</bib>
```

```
<Books>
{doc ("SimpleBooks.xml")/bib/book}
<Books>
```

```
<Books>
      <book year="1994">
            <title>TCP/IP Illustrated</title>
      </book>
      <book year="1992">
            <title>Advanced Programming in the Unix
                  Environment</title>
      </book>
<Books>
```

Figure 36. An illustration of SimpleBooks.xml (top), a query to retrieve all book elements (middle), and the result XML (bottom) [After Ref. Hunter, 2004].

2. SPARQL Protocol and RDF Query Language (SPARQL)

SPARQL is a query language and data-access protocol for RDF and OWL documents (http://www.w3.org/TR/RDFSparql-query). The language is defined in terms of the RDF data model, so it can be used to query any data source that is mapped to RDF. The W3C Data Access Working Group (DAWG) is developing the SPARQL recommendation which has reached the last-call phase. SPARQL consists of a query language specification, a query results XML format (http://www.w3.org/TR/RDFSparql-XMLres) and a data access protocol (http://www.w3.org/TR/RDFSparql-protocol). The query results XML format specification defines a format for displaying SPARQL results as XML, so the results may be further processed and manipulated with XML tools. The data-access protocol also defines HyperText Markup Language (HTML) and Simple Object Access Protocols for querying RDF documents from remote locations.

SPARQL provides a common query mechanism backed by a flexible data model so that data can be merged and queried (Dodds, 2005). A common query language enhances interoperability across different data sources. Unlike XQuery, SPARQL is not path based. In RDF, the order of triples does not matter; therefore, more emphasis is placed on the semantics of data and less on the structure. SPARQL users need knowledge about the names of resources and properties but do not need to know how these elements are structured. The SPARQL syntax is similar to the Structured Query Language (SQL) for relational databases. In addition to basic querying, SPARQL provides the capability to extract information based on rules defined by a user inside the query using the SELECT, DESCRIBE, and ASK query forms. However, SPARQL does not currently allow modification of documents being queried (Dodds, 2005).

E. USING SPARQL TO QUERY TAML RDF/XML DOCUMENTS

SPARQL is a W3C recommendation for querying RDF documents. It provides a standard syntax with multiple implementations, so developers and end users are able to write queries across a wide range of data (W3C SPARQL Recommendation). The language allows end users to extract information or RDF subgraphs from RDF documents. This section explores the capabilities of the SPARQL language by examining currently available SPARQL tools and by querying sample TAML RDF/XML documents.

1. Jena

Jena is an open-source Java API for building SW applications (http://jena.sourceforge.net). It includes Another RDF Parser (ARP), an API for writing and accessing triples in N-triple or RDF/XML format and support for several query languages including SPARQL. Jena's architecture focuses on the RDF model which is the set of statements that comprise an RDF document (Powers, 2003). In Jena, an RDF document is created by instantiating a model class and adding triples. Jena provides a tool that creates an application for instantiating and filling RDF documents based on a prototype document. This tool speeds up the process of creating multiple similar

documents. Jena also provides an API for querying languages to access documents and retrieve specific values, and also provides a command line utility for querying from within the Jena framework (McCarthy, 2005). RDF triples can be stored in a machine's memory or in a relational database. The Jena API provides the foundation for building applications to access and manipulate RDF documents.

a. Twinkle

Twinkle is a GUI application for writing queries in SPARQL and viewing the results. Twinkle provides an interface for ARQ, which is a Jena module and query engine that supports SPARQL (http://jena.sourceforge.net/ARQ). The ARQ module provides a processor library for executing SPARQL and RDF Query Language (RDQL) queries (McCarthy, 2005). The tool parses and queries documents in N-triple or RDF/XML format. Twinkle also offers various options for viewing query results including plain text, table and XML format. Figures 37-39 provide screenshots of an example query in the various result formats displayed by Twinkle. Twinkle also enables queries and results to be saved for later use.

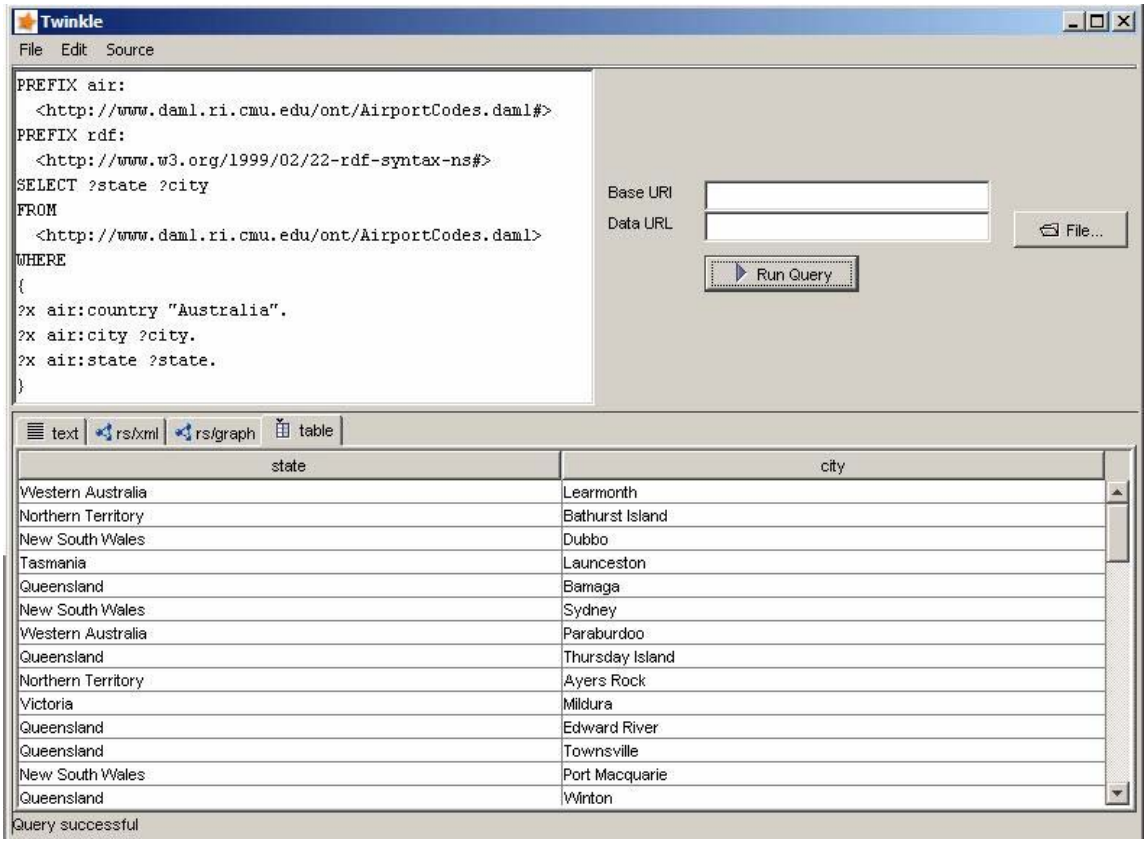


Figure 37. A screenshot of the results of an example query displayed as a table within the Twinkle GUI [From Ref. http://www.ldodds.com/projects/twinkle].

Figure 38. A screenshot of the results of an example query displayed as text within the Twinkle GUI [From Ref. http://www.ldodds.com/projects/twinkle].

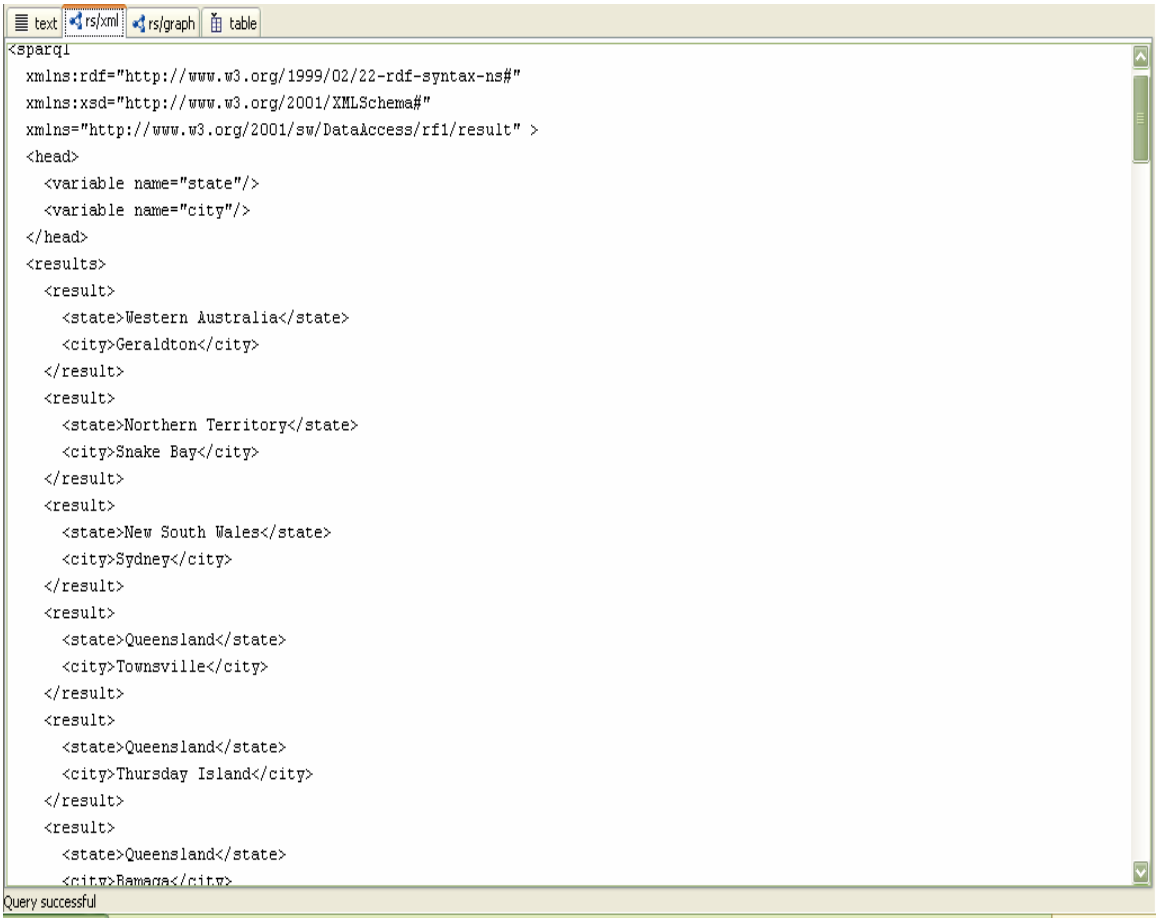


Figure 39. A screenshot of the results of an example query displayed as rs/xml within the Twinkle GUI [From Ref. http://www.ldodds.com/projects/twinkle].

2. Examples

SPARQL enables users to query RDF documents for specific data using commands such as SELECT, UNION, FILTER and ASK. This section introduces example queries run against TrafalgarExample.rdf and TAMLExample.rdf.

A SPARQL query is built on the concept of the semantic triple pattern which consists of a subject, predicate and object (Dodds, 2005). These triple patterns may contain variables in place of the subject, predicate or object. Variables are preceded by the ? symbol and represent the items of interest that are bound and returned by the query. For example, if the subject of a SPARQL query triple is replaced by a variable then all the subjects matching the specified predicate and object are returned. There are no limitations on the number of triple components that may be replaced by a variable. In

SPARQL all possible bindings or pattern matches are considered, so if more than one subject matches the specified predicate and object, then multiple triple-based results are returned (Dodds, 2005).

The most basic query that SPARQL provides is the SELECT/WHERE query which returns values that match triples given by the WHERE clause of the query. The structure consists of the PREFIX, SELECT, FROM and WHERE keywords as shown in Figure 40.

```
PREFIX taml: <urn:us:gov:dod:don:navy:navsea:usw:1.0:>

SELECT  ?x ?foundLatitude ?foundLongitude

FROM <file:TrafalgarExample.rdf>

WHERE
{
     ?x taml:Longitude ?foundLongitude.
     ?x taml:Latitude ?foundLatitude.
}

ORDER BY ?foundLongitude
```

Figure 40. SPARQL query requesting all resources with a *Longitude* and *Latitude* property.

The PREFIX keyword associates a label with a URI. Using the PREFIX keyword is similar to declaring an XML namespace. It is used as a shortcut so each resource does not have to be preceded by a full URI (Dodds, 2005). The SELECT keyword identifies the data items that are expected to be returned by the query. The query in Figure 40 returns the resource name and the value of the *Latitude* and *Longitude* for all the resources with latitude and longitude values. The FROM keyword identifies the documents the query is run against. SPARQL queries may be run against several documents, so there may be multiple FROM lines. The WHERE keyword identifies the triples that the query engine uses to match against the document being queried. There is no limit to the number of triples that can be listed, but only triples in the source document which match all of the triples identified by WHERE are returned in the result. When the query engine encounters a

variable, it tries all possible bindings to the variable and subsequently returns all triples which match. The query in Figure 40 returns all subjects which have a *taml:Longitude* and *taml:Latitude* property as shown in Figure 41. The OPTIONAL keyword may be added to indicate that a triple pattern in the WHERE clause is optional. This is useful when some data may be missing in an RDF document being queried. The ORDER BY keyword is optional and identifies which value is used to sort the results of a query.

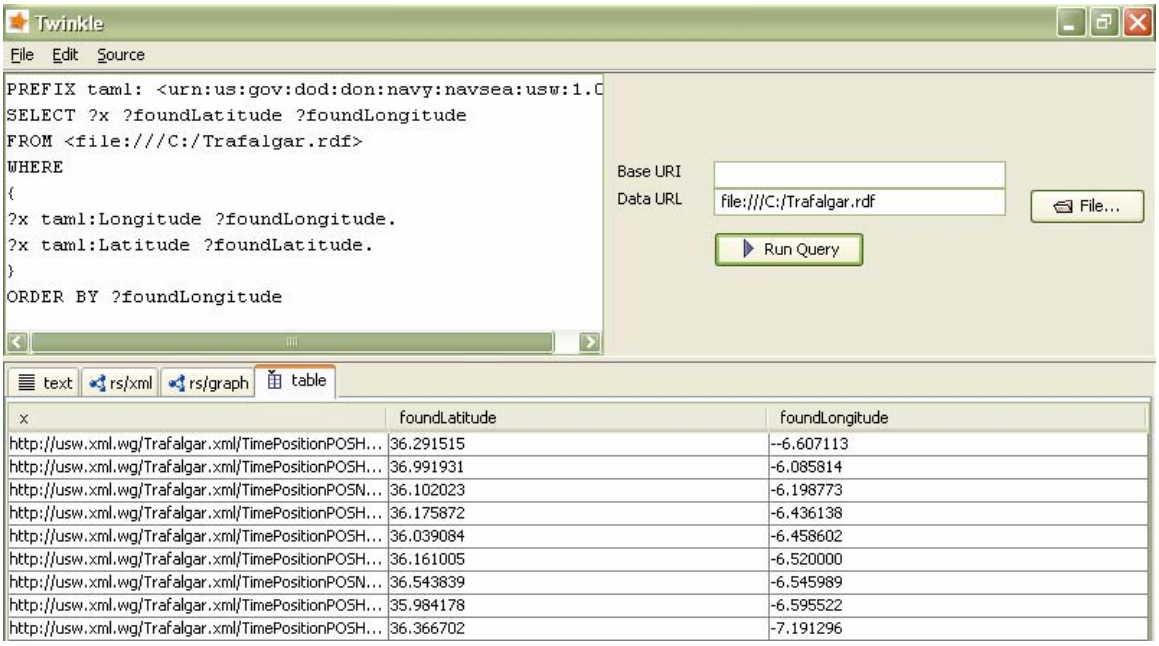


Figure 41. Results of SPARQL SELECT/WHERE query when run against Trafalgar.rdf. Results are sorted by *Longitude* and displayed in table format.

The FILTER keyword adds constraints to the triple patterns declared in the WHERE clause as shown by the query in Figure 42. Logical operators and arithmetic operators are used within the FILTER clause to define rules about the results returned. The query in Figure 42 returns all resources which have a *Speed* equal to ten and a *Course* equal to 45.0 as shown in Figure 43.

```
PREFIX taml: <http://usw.xml.wg/>

SELECT  ?x ?targetSpeed ?targetCourse

FROM <file:TAMLExample.rdf>

WHERE
{
?x taml:Speed ?targetSpeed.
?x taml:Course ?targetCourse.
FILTER (?targetSpeed = "10" && ?targetCourse= "45.0")
}
```

Figure 42. SPARQL query which returns all subjects who have *Speed* equal to ten and *Course* equal to 45.0.

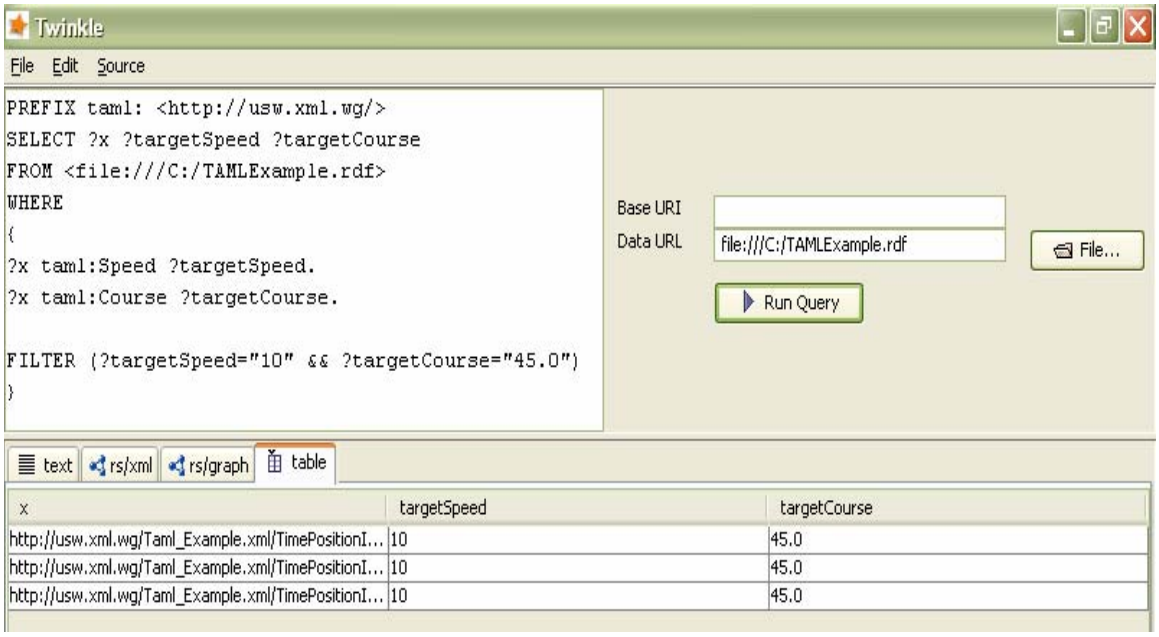


Figure 43. Results of query defined in Figure 42 when run against TAMLExample.rdf. Results are displayed in table format.

The UNION keyword is used to match and collect alternatives (Dodds, 2005). The results of a UNION query are all of the triples in a document that match any of the triple sets separated by the UNION keyword. Figure 44 is a sample UNION query which returns

all resources that have a *Course* property or a *Longitude* property. The results are displayed in Figure 45.

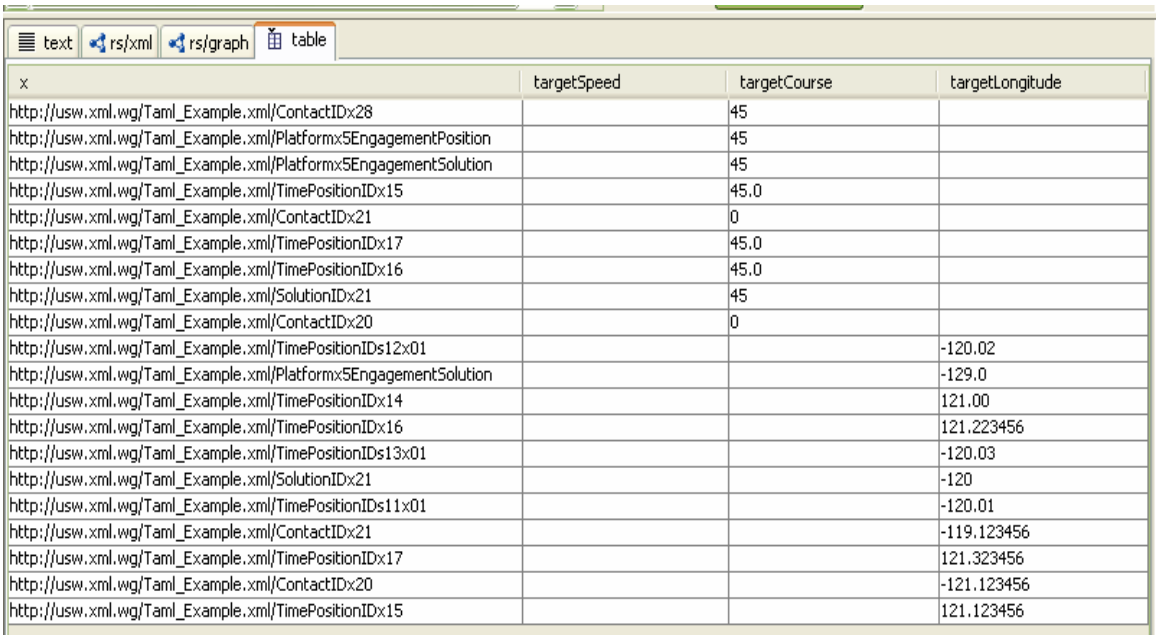
```
PREFIX taml: <http://usw.xml.wg/>

SELECT  ?x ?targetSpeed ?targetCourse ?targetLongitude

FROM <file:TAMLExample.rdf>

WHERE
{
  {
     ?x taml:Course ?targetCourse.
  }
UNION
  {
     ?x taml:Longitude ?targetLongitude.
  }
}
```

Figure 44. SPARQL query demonstrating the use of the UNION keyword.



x	targetSpeed	targetCourse	targetLongitude
http://usw.xml.wg/Taml_Example.xml/ContactIDx28		45	
http://usw.xml.wg/Taml_Example.xml/Platformx5EngagementPosition		45	
http://usw.xml.wg/Taml_Example.xml/Platformx5EngagementSolution		45	
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx15		45.0	
http://usw.xml.wg/Taml_Example.xml/ContactIDx21		0	
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx17		45.0	
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx16		45.0	
http://usw.xml.wg/Taml_Example.xml/SolutionIDx21		45	
http://usw.xml.wg/Taml_Example.xml/ContactIDx20		0	
http://usw.xml.wg/Taml_Example.xml/TimePositionIDs12x01			-120.02
http://usw.xml.wg/Taml_Example.xml/Platformx5EngagementSolution			-129.0
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx14			121.00
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx16			121.223456
http://usw.xml.wg/Taml_Example.xml/TimePositionIDs13x01			-120.03
http://usw.xml.wg/Taml_Example.xml/SolutionIDx21			-120
http://usw.xml.wg/Taml_Example.xml/TimePositionIDs11x01			-120.01
http://usw.xml.wg/Taml_Example.xml/ContactIDx21			-119.123456
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx17			121.323456
http://usw.xml.wg/Taml_Example.xml/ContactIDx20			-121.123456
http://usw.xml.wg/Taml_Example.xml/TimePositionIDx15			121.123456

Figure 45. Results of query defined in Figure 44 when run against TAMLExample.rdf.

The ASK keyword is used to ask and answer true/false questions. The result of an ASK query is a Boolean value. The query in Figure 46 asks if there are any resources with a *Longitude* value equal to "-6.085814" and a *Latitude* value equal to "36.991931". The answer to the query is yes (i.e. true) as shown in Figure 47.

```
PREFIX taml: <urn:us:gov:dod:don:navy:navsea:usw:1.0:>

ASK {?x taml:Latitude "36.991931"; taml:Longitude "-6.085814"}
```

Figure 46. A sample ASK query.

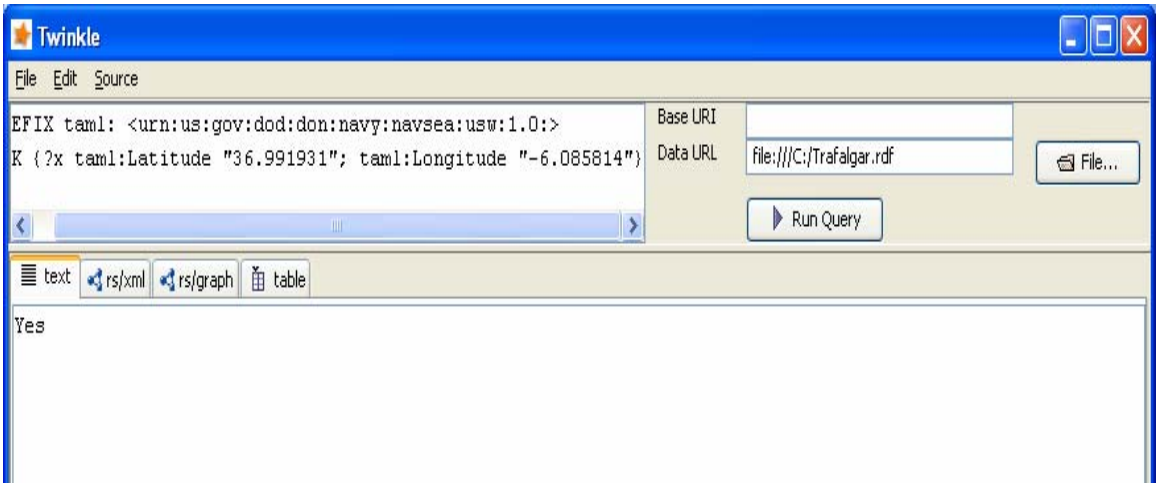


Figure 47. Result of the ASK query defined in Figure 46 when run against TrafalgarExample.rdf.

The DESCRIBE keyword is another powerful keyword used in SPARQL queries. It returns the RDF description of the resources matching the triple. Figure 48 defines a sample query asking for a description of the resource with the name Africa. The result is shown using RDF/XML format in Figure 49.

```
PREFIX taml: <urn:us:gov:dod:don:navy:navsea:usw:1.0:>

DESCRIBE ?x

WHERE{?x taml:Name "Africa"}
```

Figure 48. A query asking for a description of the resource with taml:*Name* equal to "Africa".

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-RDFSyntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/RDFSchema#"
    xmlns:taml="urn:us:gov:dod:don:navy:navsea:usw:1.0:" >
  <rdf:Description
      rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS011">
    <taml:Name>Africa</taml:Name>
    <taml:Configuration
      rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG011"/>
    <taml:Configuration
      rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD011"/>
  </rdf:Description>
```

Figure 49. Result of the DESCRIBE query defined in Figure 46 when run against TrafalgarExample.rdf.

As these examples demonstrate, SPARQL queries provide a powerful way to extract information from RDF and OWL documents. A combination of keywords enables users to define constraints and rules about the data requested by the query. SPARQL includes built-in arithmetic functions and logical operators which increase the capability of the language by allowing more complex and restrictive queries. The language includes a large library of keywords which are described in the specification document. However, one significant limitation remains. SPARQL does not enable users to update or modify information in documents.

F. SUMMARY

TAML is an XML vocabulary which defines a tactical messaging format. The vocabulary provides syntactic interoperability for machines exchanging tactical data messages but does not provide SW mechanisms for processing the semantics or meaning

of the data. An RDF/XML serialization of TAML documents adds explicit meaning to the data while maintaining an XML format for exchange between systems. Querying RDF/XML documents is more powerful and efficient than querying XML documents. Users need prior knowledge of an XML document's structure before writing queries using XQuery, but users only need to know the element names to query RDF/XML documents using SPARQL. Querying over RDF/XML is more natural for users who are not familiar with the internal idiosyncrasies of the structured data. SPARQL is the standard query language for querying RDF documents.

VII. TAML OWL ONTOLOGY

> The OWL web ontology language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of web content than that supported by XML, RDF, and RDFS by providing additional vocabulary along with formal semantics.
>
> -- W3C Semantic Web Activity Statement, 2004

A. INTRODUCTION

This chapter provides an overview of three possible use cases for a TAML ontology: semantic validation, classifying instances and querying over multiple domains. The chapter also discusses the design and testing of a TAML ontology which classifies contacts. The advantages and limitations of OWL for this particular application and these use cases are explored. An extensive set of properties and class definitions are developed using the Protégé tool and associated plug-ins.

B. USE CASES

This section provides an overview of different applications for a SW ontology. A use case diagram and a discussion is provided for each type of application explored in this thesis. These use cases include using an ontology to provide semantic validation, instance classification and to query over multiple domains.

1. Semantic Validation

XML schemas are a format for describing XML documents (Hunter, 2004). XML schemas allow users to specify the datatype of the text allowed in certain elements and attributes and can also specify a range for those values. For example, a schema constraint can specify that the element *Age* should contain only values of the type integer and the value must be between 0 and 150. XML schemas are also used to list the names and ordering of elements and attributes within a document. For example, an XML schema can require that a document have a *Name* element with the child elements: *First, Middle,* and

Last. XML schemas focus on defining rules for the syntactic components of an XML document such as naming, order and value types.

XML Schemas allow users to validate the syntactic components of an XML document, but they do not provide support for checking the consistency of the semantics. For example, the TAML schema is limited to validating syntactic components of TAML documents. Checking semantic consistency of TAML documents is a proposed use case of a TAML ontology as seen in Figure 50. A short example follows.

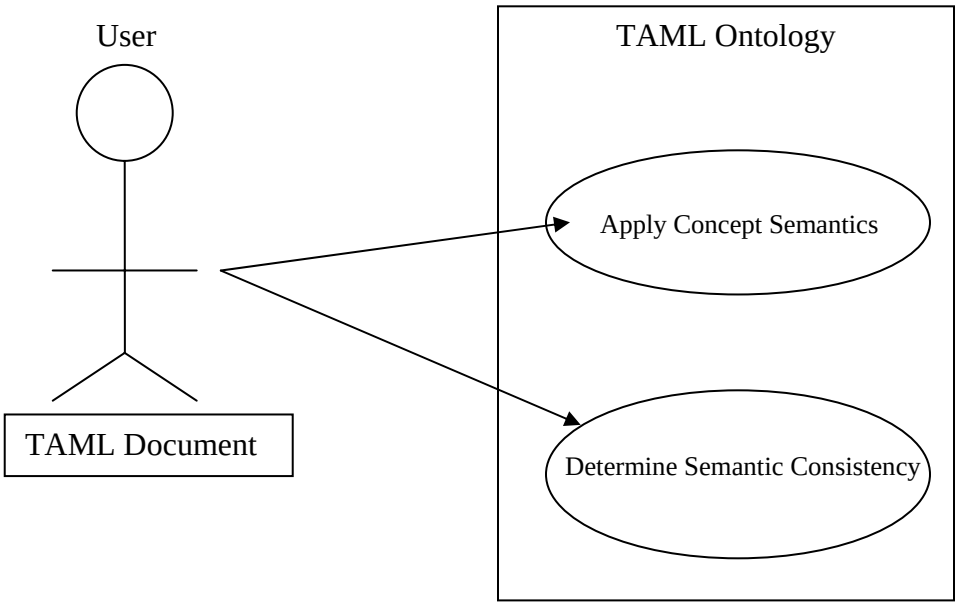


Figure 50. Use-case diagram for applying concept semantics and validating semantic consistency of TAML documents.

Ontologies define semantic definitions for a specific domain which can be used to determine whether an instance is consistent. Reasoners are applied to the knowledge base to determine whether the definitions make sense when applied to each instance. If an instance returns an error and the ontology is known to be complete and correct, then the problem may be with the data stored in the instance. For example, the TAML ontology states that members of the *FriendlyContact* class have a *threatCode* value of FRD (friendly) and members of the *HostileContact* class have a *contactClassification* value of hostile. These classes are defined as disjoint in the ontology, which means a particular instance of *Contact* may belong to either *FriendlyContact* or *HostileContact* but not both.

Therefore, if a TAML instance document contains a contact with both a valid *threatCode* value of FRD and a valid *contactClassification* value of hostile, then a reasoner produces an inconsistency error when the document is run against the ontology. The reasoner is unable to classify the contact as either friendly or hostile because there are conflicting semantic statements within the TAML document. Thus ontologies are a way to provide semantic validation for XML instance documents.

2. Classifying Instances

Ontologies provide a machine-interpretable representation of information. Ontologies written in the OWL, RDF and other standard languages are interpreted and processed by reasoners. Any description-logic (DL) reasoner is capable of interpreting an OWL ontology and inferring information about the knowledge base.

If TAML is represented as an OWL ontology, a reasoner may be run on TAML documents and the reasoner may infer new information based on the class definitions written for TAML elements in the ontology. This creates automated reasoning without having to hard-code rules or procedures in another high-level programming language. All of the class definitions written in OWL are well-formed XML. As an example, the RacerPro reasoner may process and infer information from any source that has an associated OWL ontology. There is no longer a need to create a separate program to process information for each domain. A classification use case for TAML documents is shown in Figure 51.

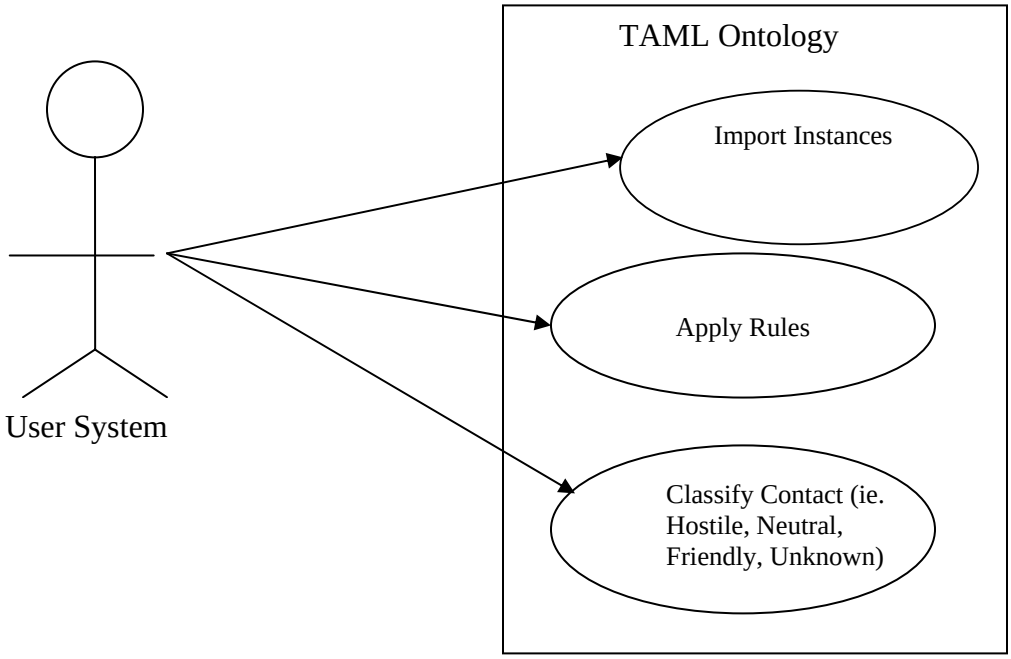


Figure 51. Use-case diagram for classifying TAML contact instances.

In this use case, a user or application populates the contact instances within the TAML ontology with contact data from TAML documents. The user runs the reasoner against the instances and then uses the given class definitions to subclass the contacts as members of the *HostileContact*, *FriendlyContact*, *NeutralContact*, *UnknownContact* or *ThreatContact* class. For example, if the class definition for *FriendlyContact* defines members of the class as contacts with *threatCode* equal to FRD, then any instance class that the user inputs with *threatCode* equal to FRD is inferred as a *FriendlyContact* by the reasoner. Interestingly, such reasoners have the ability to classify large numbers of instances more quickly than human operators.

3. Querying Over Multiple Domains

Different ontologies may model knowledge representations for specific domains in different ways (W3C OWL Recommendation, 2004). The OWL language provides tags for defining equivalent terms in separate ontologies so they may be mapped as corresponding together. More complicated mappings may be accomplished by mapping two lower-level ontologies to an upper-level ontology like the Suggested Upper Merged

Ontology (SUMO) (http://ontology.teknowledge.com). Ontology mapping allows a user to make queries over different domains, as illustrated in Figure 52.

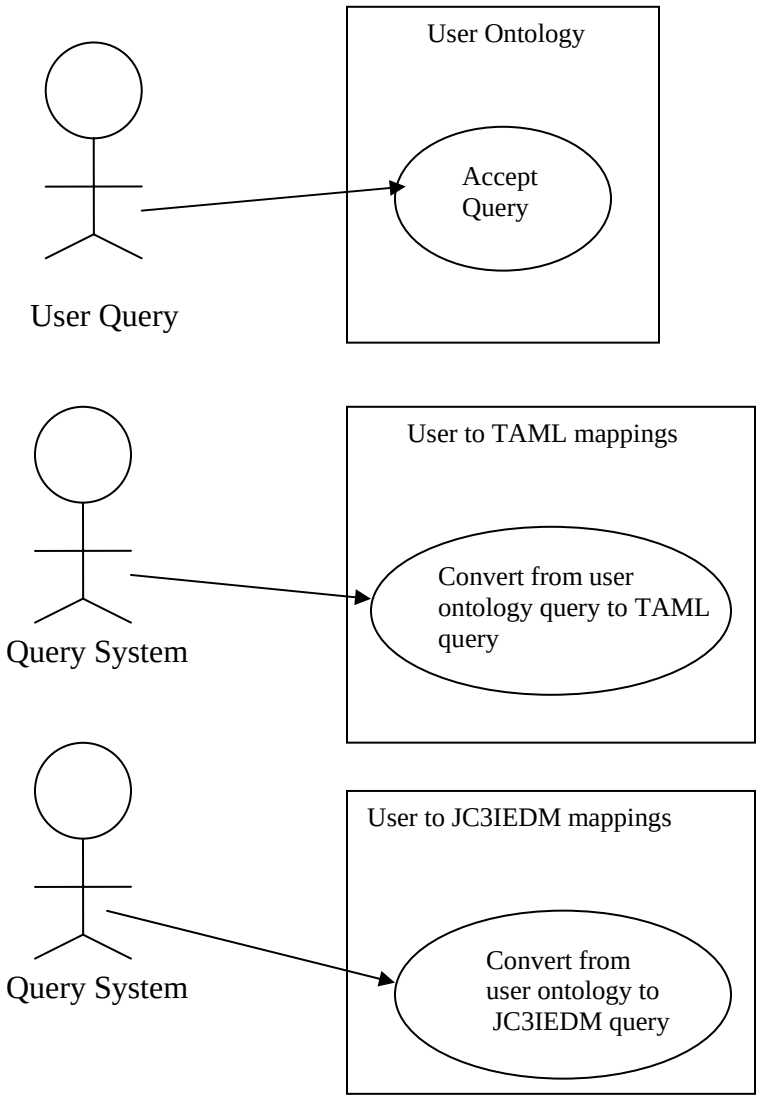


Figure 52. Use-case diagram for querying over multiple domains.

A query is entered by the user based on the user's view of the world or ontology. A query system uses ontology mappings to convert the query to a TAML query and a Joint Consultation Command & Control Information Exchange Data Model (JC3IEDM) query. The results are mapped back to the user ontology, so they are displayed in a format the user understands. For example, a user could query TAML documents and JC3IEDM documents for any ships on the Homeland Security watch list. Only one query is written by the user, but results are returned from two different knowledge bases.

This capability is significant. As shown in Figure 16, the ability to map diverse systems to a common system reduces interoperability requirements from exponential to linear. Employing a compatible ontology across multiple domains can similarly reduce conversion requirements, allowing queries to occur across different data domains without requiring the massive conversion of XML datasets from one form into another. Such conversions usually occur via XSLT, which typically embeds the relationships between tagsets as elaborate rules for mapping different XML element-attribute conversions. Shifting from exhaustive data conversion to mutually compatible SW constructs expressed in RDF/OWL greatly improves the ability to express and maintain semantically meaningful expressions across a wide variety of similar (but syntactically different) datasets.

C. ONTOLOGY LANGUAGE CHOICE FOR TAML APPLICATIONS

An ontology is an explicit definition of the semantic properties and relationships which exist in a given domain of knowledge. Thanks to seven decades of effective research in Artificial Intelligence (AI), there are many languages available for describing an ontology, and each language has advantages and disadvantages. The languages may be divided into three different types: first-order logic (FOL), frame and description logic (DL). The expressiveness, complexity and reasoning capability varies over each type of language. Once a type of language is chosen, a specific language must be chosen based on the applications that may use the ontology. Historically such languages have offered many choices that each applied to a narrow range of applicability.

FOL provides a formal language for expressing well-formed statements, and also allows developers to specify when the statements are true with respect to a certain world (http://en.wikipedia.org/wiki/First_order_logic). The language allows a developer to list propositions but does not clarify the relationships between the propositions. FOL is an expressive language that may be used to define almost anything. However, the expressive power of FOL makes automated reasoning over the language much harder. It is possible to express many constructs in FOL such that reasoning becomes a computationally intractable (i.e. undecidable) problem. Since FOL is not always decidable, it is often not the best language to use for defining ontologies where the goal is to reliably infer new

data from existing data. Since the goal of the TAML Contact Classification ontology is to infer the classification of contacts based on instance data in the ontology often applied within the context of real-time (possibly life-or-death) decision making, FOL languages are not the best choice for modeling the ontology.

Frame systems are another way to build an ontology that explicitly defines a domain (Nebel, 1999). Frames are a specialization of FOL where semantic constructs lose some expressiveness, but the reasoning problem is decidable. Frame systems encourage a hierarchical view of classes where classes encapsulate meaningful sets of data. These classes are assigned properties with constraints. Frames are limited because they do not express negation, disjunction between classes, or quantification. The types of reasoning done on frame systems is limited to similarity inheritance, comparison and default reasoning. Inheritance reasoning states that subclasses inherit all of the properties of their superclasses. Default reasoning is the assignment of default values to properties that may be overwritten. The depth of reasoning capability associated with frame systems is not powerful enough to solve the contact classification problem; therefore, it is not the best choice for modeling the TAML Contact Classification Ontology.

Description Logic (DL) languages are another type of language used to define ontologies. DL languages are more expressive than frames and reasoning is also decidable over DL ontologies. DL languages add the ability to define classes with quantifiers, disjoint properties and negation. Reasoners are then used to infer subsumption, classification of instances, and inconsistency in class definitions. Classification of instances is the type of reasoning the TAML Contact Classification Ontology seeks to perform; therefore, a DL language is the correct choice for defining the TAML ontology. OWL-DL is the specific language chosen because it is the SW standard for defining ontologies.

D. OWL ONTOLOGY DESIGN FOR TAML

The TAML schema identifies the important terms in the undersea warfare tactical domain, but aside from terminology definitions provided by XML schema annotations, the semantics of the terms are left for the application to determine. The TAML Contact

Classification Ontology is designed to explicitly state class definitions, class properties and their characteristics, and define a class hierarchy in the machine-interpretable language OWL-DL.

Ontology development is similar to object-oriented design but differs in a fundamental way. Object-oriented design centers around methods on classes, so a designer makes decisions based on the operational properties of a class. In ontology development, the designer makes decisions based on the structural properties of a class. The TAML Contact Classification Ontology is designed according to the steps in "Ontology 101: A Guide to Creating Your First Ontology" (Noy and McGuiness, 2000).

1. Ontology Editor

The Protégé-OWL version 3.2 beta application was used to create and edit the TAML Contact Classification Ontology. Protégé-OWL is a free open-source frame ontology editor with an OWL plug-in. Protégé-OWL may be used to edit ontologies, access reasoners and acquire instances for semantic mark-up (Knublauch, 2004). The editor benefits from a large and active community of users, a reusable library and a flexible architecture.

Protégé supports a plug-in architecture which makes it easy to extend (Knublauch, 2004). Developers can easily write plug-in applications that communicate with the core Protégé through the Protégé API. For example, if a developer wants to export their ontology in a custom format, then there is no need to modify the source code. A plug-in is written and packaged as a jar file which is placed in the Protégé plug-ins folder (Noy and Sintek, 2001). The plug-in is then invoked by changing the project configuration. This ability to rapidly innovate is a benefit offered by the Protégé plug-in architecture.

Figure 53 shows the architecture of Protégé-OWL. The OWL plug-in was added as an extension to the frame-based Protégé in order to create Protégé-OWL.

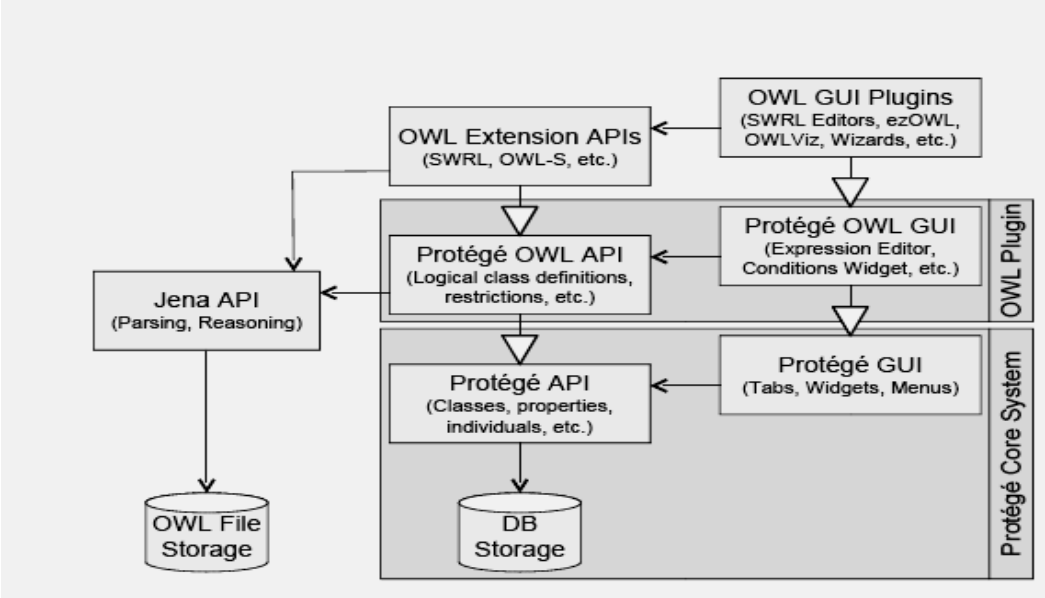


Figure 53. A diagram of the Protégé-OWL architecture [From Ref. Knublauch, 2004].

The OWL plug-in supports RDFS, OWL-Lite, OWL-DL and some parts of OWL-Full (Knublauch, 2004). Protégé-OWL is backwards compatible with Protégé and provides an open-source Java API for developing further applications. The editor hides the details of writing an OWL ontology from the user by providing a set of user friendly interfaces for creating classes, properties and instances. The editor supports importing and exporting ontologies in many forms such as CLIPS, XML, RDF and the default OWL. Protégé is a powerful, free and user-friendly tool for creating, editing and maintaining ontologies.

2. Domain and Scope

Tactical undersea warfare is the domain of the TAML Contact Classification Ontology. The domain describes concepts and properties that are common in a tactical picture of a battlespace. It deals with concepts like platform, contact, weapon, position, etc. The domain concepts and terms are listed in the TAML schema. The ontology is an extension of the schema that adds semantic information about classes and properties.

The TAML Contact Classification Ontology focuses on the concept of a contact within the TAML domain. The goal of the ontology is to classify contacts as hostile, friendly, neutral, unknown, or threat contacts based on the class definitions. The scope of

the ontology is limited to one application to demonstrate the type of reasoning that may be performed on instance data associated with an ontology. Other concepts within the TAML domain are not yet modeled because they do not have any relevance to classifying contacts. Although somewhat restricted with respect to TAML capabilities, the domain of contact classification is quite general and of broad interest throughout all the military services.

3. Ontology Reuse

The first step in the design process is to look for ontologies that may be imported and modified for use in the ontology under development. Reuse of ontologies reduces overhead in development and may be useful for ontologies that need to interact with other applications that have committed to certain ontologies or vocabularies (Noy and McGuinness, 2000). Protégé-OWL includes an import feature which imports ontologies in many different forms and converts them to the OWL language.

The TAML domain is specific to the USW community. Using the terms and concepts developed in the TAML schema is an important design goal of the TAML Contact Classification Ontology. This design goal ensures that converting TAML documents to OWL instances is as easy as possible; however, it makes ontology reuse less ideal. A better design decision is to use the TAML schema as a guide for developing the ontology. There may be a need in the future to import other ontologies such as ontologies that model time or geospatial locations. These existing ontologies may increase the TAML domain model and allow the developer to write more complex class definitions and perform more extensive reasoning on TAML data.

For this work, the DoD Metadata Registry was examined for possible ontologies of relevant use but none were found. In fact, TAML itself is among the first tactical tagsets to be submitted to the DoD Metadata Registry for Navy-wide use. These areas remain an important area for future work.

4. Class Hierarchy

The next step in the ontology development process is to define major concepts and create a class hierarchy. There are three basic approaches to developing a class hierarchy. A top-down approach starts by identifying the most general concepts and then specializing the concepts into subclasses. Bottom-up development takes the opposite approach and begins with quite specific classes and then generalizes them. A developer may also combine the two approaches by defining salient concepts first and then specializing and generalizing them (Noy and McGuinness, 2000). The TAML Contact Classification Ontology was designed using the top-down approach.

The concepts in the TAML domain are listed as elements in the TAML schema. Part of the design process was to go through the TAML schema and separate the elements that refer to objects from the elements that characterize those objects. The main concept of the TAML Contact Classification Ontology is a contact. The goal of the ontology is to model contacts, contact properties and definitions for different contact classifications. Therefore, the contact concept was the central focus of the design and the *Contact* class is the top of the hierarchy under the generic class owl:*Thing*. Everything is a subclass of owl:*Thing* which is a generic class included in all OWL ontologies. The *Contact* class is specialized into five different subclasses as illustrated in Figure 54. Capitalizing class names is the preferred naming convention in OWL, and that convention was also adopted in the design of the TAML Contact Classification Ontology. Table 2 lists and describes all of the classes in the TAML Contact Classification Ontology.

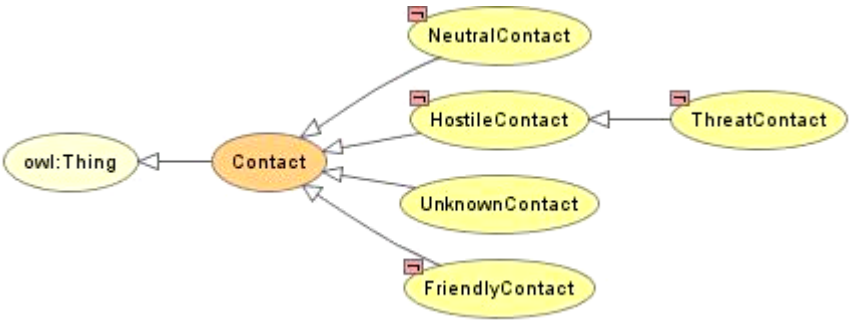


Figure 54. View of *Contact* hierarchy from the Protégé Ontoviz plug-in.

Class	SuperClass	Instances	Description
AbsolutePosition	owl:*Thing*	N/A	Position in latitude, longitude and altitude
AlertCode	owl:*Thing*	HIT, NSP, SUSP, TGT	Enumerates the level of the contact alert
Applicability	owl:*Thing*	live_training, real_world, simulated	Enumerates real world, training or simulation
Contact	owl:*Thing*	N/A	Defines a contact event
ContactClassification	owl:*Thing*	friendly, hostile, neutral, unknown	Defines classification
CountryCode	owl:*Thing*	N/A	Defines platform country by code
DateTimeGroup	owl:*Thing*	N/A	Date and time of an event
Ellipse	owl:*Thing*	N/A	Defines uncertain position as an ellipse
FriendlyContact	*Contact*	N/A	Defines a friendly contact
FriendlyCountryCode	*CountryCode*	CA, DE, FR, GB, US, IL	Defines the codes for friendly countries
HostileContact	*Contact*	N/A	Defines an enemy contact
HostileCountryCode	*CountryCode*	AF, IQ, IR	Defines the codes for hostile countries
NeutralContact	*Contact*	N/A	Defines a neutral contact
NeutralCountryCode	*CountryCode*	BE, CH, ES	Defines the codes for neutral countries
Orientation	owl:*Thing*	N/A	Defines pitch, roll, yaw and heading of a contact
Position	owl:*Thing*	N/A	Defines position of contact
Source	owl:*Thing*	GPS, RLGN, SINS	Source of position report
SuspiciousContact	*Contact*	N/A	Defines a suspicious contact
SuspiciousCountryCode	*CountryCode*	CN, EG, KH, KR, SO	Defines the codes for suspicious countries
TargetType	owl:*Thing*	merchant, smallboat, warship, militaryAircraft, civilianAircraft, submarine	Type of vessel or aircraft
ThreatCode	owl:*Thing*	AFD, FAKER, FRD, HOS, JOKER, NEU, PND, SUS, UNK	Enumerates the threat of the contact
ThreatContact	*HostileContact*	N/A	Defines a contact that is an immediate threat
LanguageType	owl:*Thing*	N/A	Defines language ID
UnknownContact	*Contact*	N/A	Defines a contact with unknown classification

Table 2. List and description of all the classes in TAMLContactClassification.owl.

There are other classes besides *Contact* in the TAML Contact Classification Ontology which act as the range of object properties. For example, the property *contactClassification* is an object property, so it has values from the *ContactClassification* class. The instances of *ContactClassification* are the various values that the property may hold.

5. Properties

Once a class hierarchy has been defined, properties are assigned to the classes. Properties are characteristics of the members of a class which define the internal structure of the concept (Noy and McGuinness, 2000). For example, a property of the class *Pizza* might be *hasTopping* since pizzas generally have toppings. The properties in the TAML Contact Classification Ontology are incorporated from the TAML schema document. The elements that describe the main concept of *Contact* in the TAML vocabulary are defined as properties in the ontology. Beginning property names with a lowercase letter is the accepted naming convention in OWL, thus this convention was adopted in the design of TAMLContactClassification.owl.

The OWL language allows the following characteristics for object properties to be defined: functional, inverse functional, symmetric and transitive. Datatype properties can also be characterized as functional (meaning one-to-one or several-to-one mappings). All of the properties in the TAML Contact Classification Ontology are functional because each contact has only one value for each property. For example, a contact has only one speed; therefore, the property *speed* is functional. The property *position* is both functional and inverse functional. Inverse functional indicates that two instances may not have the same range value for the property. By defining *position* as inverse functional the ontology explicitly states that two different contacts may not occupy the same position. For example, if two *Contact* instances have the same *position* then these two instances must be the same contact. In an ontology, properties are used to create class definitions, so a reasoner may infer information. However, OWL-DL is limited because it does not allow a reasoner to infer the value of properties. The rule language, SWRL adds this capability and is discussed in the next chapter.

There are two different kinds of properties: datatype properties and object properties. Datatype properties link a specific class member to an XML datatype value. For example, the value of the property *hasName* is a string; therefore, the property is a datatype property. The current OWL-DL specification only requires reasoners to support the xsd:string and xsd:int datatypes. For this reason, the TAML Contact Classification Ontology only uses the xsd:int datatype to restrict properties with numerical values. Many of the properties in the TAML schema are defined by user-defined datatypes. The current OWL specification also does not include direction on how to represent user-defined datatypes and does not require reasoners to support user-defined datatypes. Due to these restrictions, some of the semantics implied by the TAML Schema are lost. The OWL 1.1 Recommendation is expected to include support for other datatypes including user-defined datatypes. Thus the precise data typing of the TAML vocabulary will be better preserved in future versions. Table 3 lists and describes all of the datatype properties describing the class *Contact* in the TAML Contact Classification Ontology.

Name	OWL Datatype	Characteristics	Description
categoryType	xsd:string	functional	Enumerates contact categories
class	xsd:string	functional	Enumerates class
contactDesignator	xsd:string	functional	Temporary designation for contact
course	xsd:int	functional	Direction of motion in true degrees
depthElevationAngle	xsd:int	functional	Azimuth - positive elevation
forceCode	xsd:int	functional	Force Code
frequency	xsd:int	functional	Frequency of the contact
id	xsd:string	functional	Contact ID
range	xsd:int	functional	Contact distance from detecting platform
reportingStatus	xsd:string	functional	Reporting status of contact
sensorCode	xsd:string	functional	Contact sensor code
shipControlNumber	xsd:int	functional	USN control number of a ship
speed	xsd:int	functional	Speed of the contact
trackType	xsd:string	functional	Time and position
unitIdentificationCode	xsd:int	functional	Unit ID of contact

Table 3. List and description of the datatype properties in TAMLContactClassification.owl.

Object properties link a class member to another class member. Object properties have a defined domain and range. The domain defines the class the individual being described belongs to and the range defines the class that includes the member that the object is being linked to. For example, the property *alertCode* has a domain *Contact* and a range *AlertCode*, so the property links instances of *Contact* to instances of *AlertCode*. Table 4 lists and describes all the object properties describing the class *Contact* in TAMLContactClassification.owl.

Name	Domain	Range	Characteristics	Description
alertCode	Contact	AlertCode	functional	Level of contact alert
applicability	Contact	Applicability	functional	Enumerates real life, training, or simulation
contactClassification	Contact	ContactClassification	functional	Describes as friend or foe
countryCode	Contact	CountryCode	functional	Platform country by code
dateTimeGroup	Contact	DateTimeGroup	functional	Date and time of an event
orientation	Contact	Orientation	functional	Defines pitch, roll, yaw, and heading of a contact
position	Contact	Position	functional, inverse functional	Contact position
targetType	Contact	TargetType	functional	Type of vessel
threatCode	Contact	ThreatCode	functional	Enumerates threat of contact

Table 4. List and description of the object properties describing *Contact* in TAMLContactClassification.owl.

6. Class Definitions

Class definitions are a set of assertions or axioms applied to a class which define members of the class. Class definitions allow reasoners to determine if instances are incorrectly assigned to a class and also allow them to classify instances as members of classes. Class definitions include quantifier, cardinality and has value constraints. There are two types of class definitions in OWL: (1) necessary or (2) necessary and sufficient. Necessary definitions define the conditions that every member of the class must follow. Necessary and sufficient definitions define the conditions that unambiguously make an instance a member of a class. Necessary and sufficient definitions are the conditions that a reasoner uses to determine whether a particular instance may be classified as a member of a class.

The first step in defining classes is to list a set of axioms that describe the parent class, *Contact*. All definitions written for *Contact* are inherited by all its subclasses. The description of *Contact* is a list of assertions that define the properties that every contact must have. According to the TAML schema every contact must have a *dateTimeGroup*, *position*, *orientation*, *course*, *speed*, *id* and *sensorCode*. The cardinality for each of these properties is explicitly defined as one. This means that each contact must have one (and only one) value for the property. These properties are listed under the "necessary" tab of Protégé instead of the "necessary and sufficient" tab which indicates *Contact* is a primitive class instead of a defined class (Rector and Drummond, 2004). A reasoner will not classify instances as a *Contact* solely based on these necessary conditions. This is a desirable approach since every instance with these properties is not necessarily a *Contact*. For example, an instance of *Platform* may have the same properties as a *Contact* but cannot itself be a *Contact*. Table 5 lists and defines the axioms that describe *Contact.*

Assertion	**Description**
course = 1	Each *Contact* must have one and only one *course*.
dateTimeGroup = 1	Each *Contact* must have one and only one *dateTimeGroup*.
orientation = 1	Each *Contact* must have one and only one *orientation*.
position = 1	Each *Contact* must have one and only one *position*.
id = 1	Each *Contact* must have one and only one *id*.
sensorCode = 1	Each *Contact* must have one and only one *sensorCode*.
speed = 1	Each *Contact* must have one and only one *speed*.

Table 5. List and description of axioms defining the class *Contact*.

Protégé provides a visual separation between "necessary" conditions and "necessary and sufficient" conditions. Figure 55 illustrates that the descriptions of *Contact* are listed under the Necessary tab in Protégé.

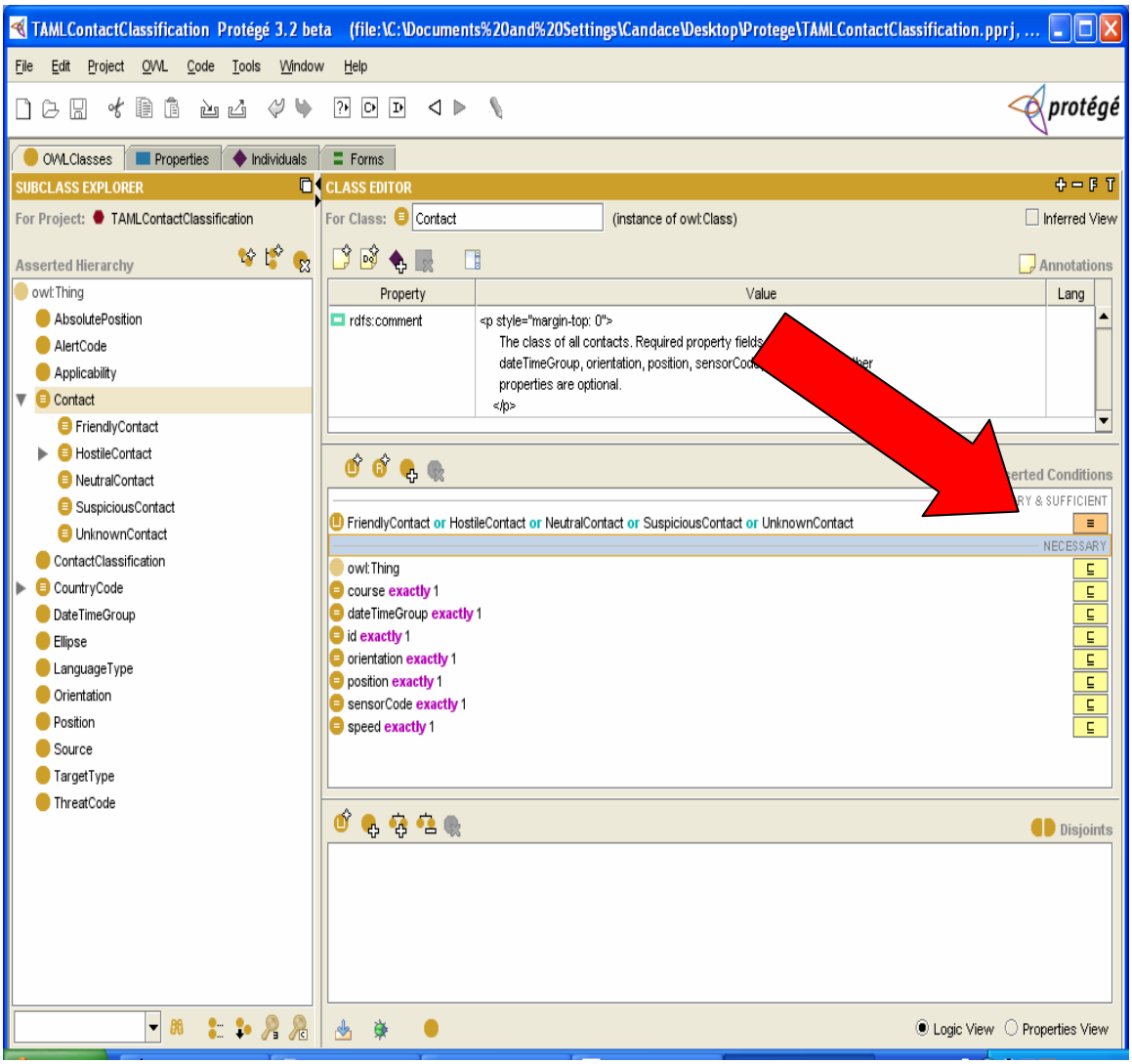


Figure 55. This screenshot illustrates the defintion for *Contact* asserted in Protégé with an emphasis on Necessary conditions.

After defining the necessary conditions of *Contact*, a covering axiom was added. The covering axiom asserts that every instance of *Contact* must also belong to at least one of its subclasses. This is true in the TAML Contact Classification Ontology because the *UnknownContact* class includes any contacts that are not classified; therefore, the

subclasses of *Contact* create an exhaustive decomposition of the class. A covering axiom asserts this fact by defining *Contact* as the union of all its subclasses. This axiom is a helpful prerequisite to ensure that classification definitions operate efficiently.

In order to add flexibility to the ontology, new subclasses of *CountryCode* are also added at this step. The *CountryCode* class has country abbreviations as its members. These abbreviations can be subclassed into the following categories: *FriendlyCountryCode*, *NeutralCountryCode*, *SuspiciousCountryCode* and *HostileCountryCode*. The subclasses are defined as disjoint from each other since a country code may only belong to one of the classes. Defining *CountryCode* this way adds flexibility to the design since the classification of countries is subject to change. When a change is made the corresponding country code is moved from one subclass to another, and the definitions for the contact subclasses do not need to be modified. The definitions for the contact subclasses are generic and reference the corresponding *CountryCode* to determine if a code indicates friendly or hostile. For example, the class *FriendlyContact* is defined as any contact that has the property *countryCode* where the value is a member of *FriendlyCountryCode*. If the code GB (Great Britain) is a member of *FriendlyCountryCode* and instance A has the value GB for its *countryCode*, then the instance is expected to be classified as a member of *FriendlyContact*. If the GB enumeration is modified to be a member of *NeutralCountryCode*, then instance A is no longer classified as a *FriendlyContact*. Such flexibility is an important design feature, since ontologies need to be easy to update when changes occur within the domain.

The next step in the design process is to define each of the subclasses of *Contact*. In OWL, it is assumed that classes may overlap unless they are explicitly defined as disjoint. This increases the expressiveness of OWL by allowing the developer to decide whether the classes are disjoint. The subclasses of *Contact* are asserted to be mutually disjoint. For example, a *FriendlyContact* cannot also be a *HostileContact*. This relationship holds between each subclass of *Contact*. Each subclass of *Contact* also inherits the necessary assertions defined for *Contact*.

The next step is to list the necessary and sufficient conditions for each subclass of *Contact* as shown in Figure 56. These conditions are listed so the reasoner may first

determine which subclass an instance of *Contact* belongs to, and then classify that instance. For example, a necessary and sufficient condition for *FriendlyContact* is any contact that has the property *contactClassification* and the value is equal to friendly. Therefore, if an instance of *Contact* has a *contactClassification* value of friendly the reasoner infers that the instance is a member of FriendlyContact.

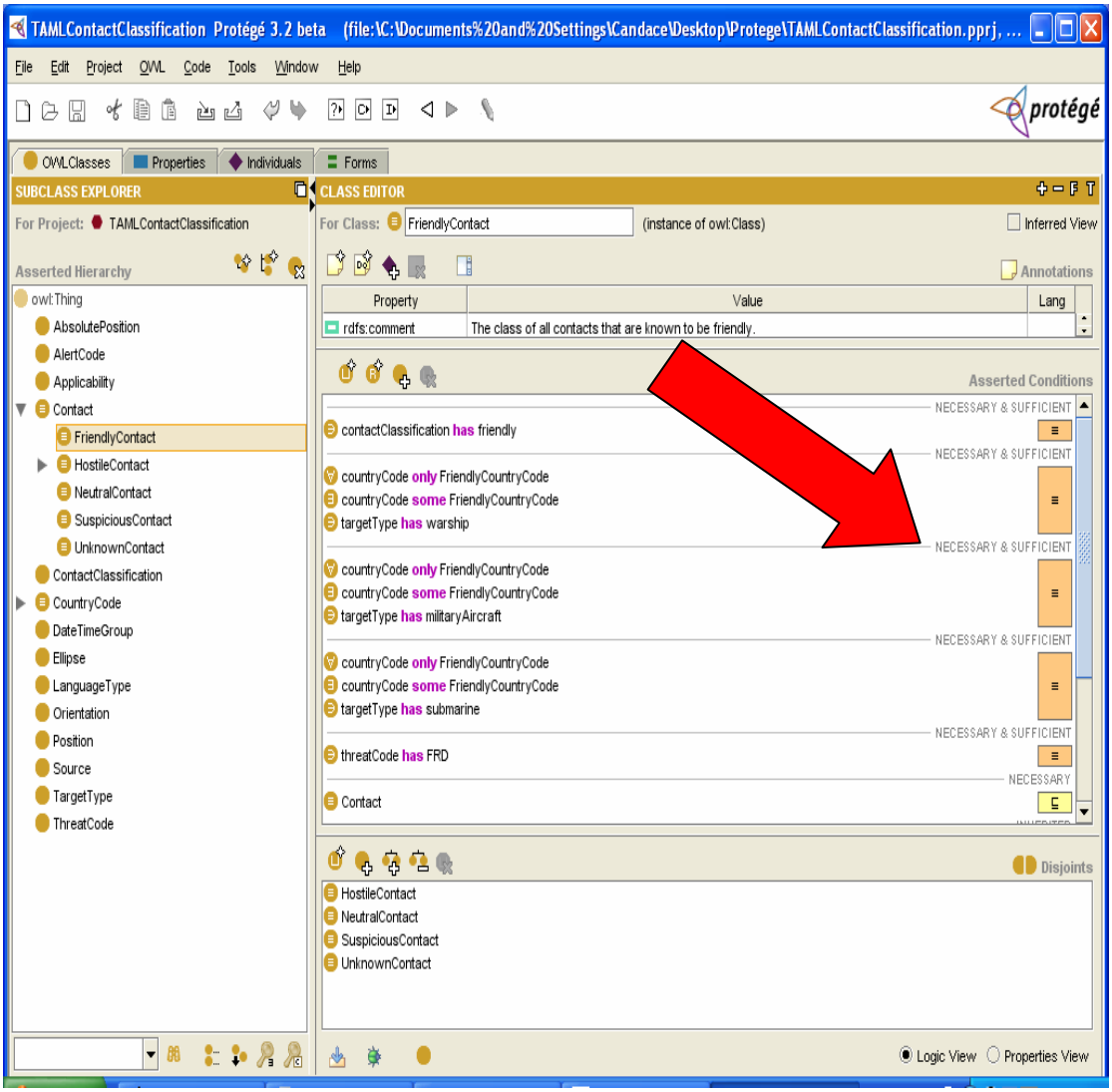


Figure 56. This screenshot illustrates the necessary and sufficient OWL definitions for *FriendlyContact* asserted in Protégé.

Table 6 lists each of the necessary and sufficient definitions for *FriendlyContact*. If an instance meets any one of those definitions, then the reasoner is able to classify the instance as a friendly contact.

Necessary and Sufficient Definition	**Meaning**
ɜ *contactClassification* friendly	A *FriendlyContact* is a contact with a contact classification of friendly.
ɜ *threatCode* FRD	A *FriendlyContact* is a contact with a threat code equal to FRD.
∀*countryCode* FriendlyCountryCode ∧ ∃*countryCode* FriendlyCountryCode ∧ ɜ *targetType* militaryAircraft	A *FriendlyContact* is any contact which, amongst other things, has at least one friendly country code and no other type of country code and is also a military aircraft.
∀*countryCode* FriendlyCountryCode ∧ ∃*countryCode* FriendlyCountryCode ∧ ɜ *targetType* warship	A *FriendlyContact* is any contact which, amongst other things, has at least one friendly country code and no other type of country code and is also a warship.
∀*countryCode* FriendlyCountryCode ∧ ∃*countryCode* FriendlyCountryCode ∧ ɜ *targetType* submarine	A *FriendlyContact* is any contact which, amongst other things, has at least one friendly country code and no other type of country code and is also a submarine

Table 6. List and description of axioms defining the class *FriendlyContact*.

OWL has an open-world assumption which means that if something cannot be proven as false, then it is assumed that it may be true. Therefore, the only statements that are false are the ones that are stated as false or that may be derived as false from the information in the ontology. Due to the open-world assumption, a closure axiom is often needed when definitions are written using the existential quantifier. For example, a developer may define a friendly contact as any contact that has one or more friendly country codes using the existential quantifier. However, this statement does not say that a contact cannot have both a friendly country code and a hostile country code. A closure axiom needs to be added using the universal quantifier. The closure axiom states that all of a friendly contact's country codes must be friendly country codes. An alternative approach is to define the cardinality of the property countryCode as one. Then the closure

axiom is unnecessary since each Contact may only have one country code value. For the TAML Contact Classification Ontology, the closure axiom approach was taken as illustrated in Table 6.

HostileContact, *SuspectContact*, *UnknownContact* and *NeutralContact* are also defined by a set of necessary and sufficient conditions as shown in tables 7-10.

Necessary and Sufficient Definition	**Meaning**
∋ *contactClassification* hostile	A *HostileContact* is a contact with a contact classification of hostile.
∋ *threatCode* HOS	A *HostileContact* is a contact with a threat code equal to HOS.
∀*countryCode* HostileCountryCode ∧ ∃*countryCode* HostileCountryCode ∧ ∋ *targetType* militaryAircraft	A *HostileContact* is any contact which, amongst other things, has at least one hostile country code and no other type of country code and is also a military aircraft.
∀*countryCode* HostileCountryCode ∧ ∃*countryCode* HostileCountryCode ∧ ∋ *targetType* warship	A *HostileContact* is any contact which, amongst other things, has at least one hostile country code and no other type of country code and is also a warship.
∀*countryCode* HostileCountryCode ∧ ∃*countryCode* HostileCountryCode ∧ ∋ *targetType* submarine	A *HostileContact* is any contact which, amongst other things, has at least one hostile country code and no other type of country code and is also a submarine
∀*countryCode* HostileCountryCode ∧ ∃*countryCode* HostileCountryCode ∧ ∋ *targetType* smallboat	A *HostileContact* is any contact which, amongst other things, has at least one hostile country code and no other type of country code and is also a smallboat

Table 7. List and description of axioms defining the class *HostileContact*.

Necessary and Sufficient Definition	**Meaning**
϶ *threatCode* SUS	A *SuspectContact* is a contact with a threat code equal to SUS.
∀*countryCode* HostileCountryCode ∧ ∃*countryCode* HostileCountryCode ∧ ϶ *targetType* civilianAircraft	A *SuspectContact* is any contact which, amongst other things, has at least one hostile country code and no other type of country code and is also a civilian aircraft.
∀*countryCode* HostileCountryCode ∧ ∃*countryCode* HostileCountryCode ∧ ϶ *targetType* merchant	A *SuspectContact* is any contact which, amongst other things, has at least one hostile country code and no other type of country code and is also a merchant.

Table 8. List and description of axioms defining the class *SuspectContact*.

Necessary and Sufficient Definition	**Meaning**
϶ *contactClassification* neutral	A *NeutralContact* is a contact with a contact classification of neutral.
϶ *threatCode* NEU	A *NeutralContact* is a contact with a threat code equal to NEU.
¬(∃*countryCode* HostileCountryCode ∨ ∃*countryCode* SuspiciousCountryCode) ϶ *targetType* smallboat	A *NeutralContact* is a contact that is a small boat and does not have a hostile or suspicious country code.
¬(∃*countryCode* HostileCountryCode ∨ ∃*countryCode* SuspiciousCountryCode) ϶ *targetType* merchant	A *NeutralContact* is a contact that is a merchant ship and does not have a hostile or suspicious country code.
¬(∃*countryCode* HostileCountryCode ∨ ∃*countryCode* SuspiciousCountryCode) ϶ *targetType* civilianAircraft	A *NeutralContact* is a contact that is a civilian aircraft and does not have a hostile or suspicious country code.

Table 9. List and description of axioms defining the class *NeutralContact*.

Necessary and Sufficient Definition	**Meaning**
∋ *contactClassification* unknown	An *UnknownContact* is a contact with a contact classification of unknown.
∋ *threatCode* UNK	An *UnknownContact* is a contact with a threat code equal to UNK
¬ FriendlyContact ∧ ¬ HostileContact ∧ ¬ NeutralContact ∧ ¬ SuspectContact	An *UnknownContact* is any contact that is not a friendly contact, and not a hostile contact, and not a neutral contact, and not a suspicious contact.

Table 10. List and description of axioms defining the class *UnknownContact*.

The ontology also includes a *ThreatContact* class which does not include any axioms or assertions. The *ThreatContact* is a class defined by restrictions on its datatype properties (such as speed greater than four) which are not yet supported by the current OWL-DL version. In OWL, restrictions on datatype properties will be defined by creating user-defined datatypes. For example, a user-defined datatype could be *speedGreaterThanFour* which would include all integers greater than four. In the current version of OWL-DL, such user-defined datatypes are not supported (Pan, 2004). Nevertheless, SWRL provides another way to enforce this type of restriction on datatype properties, and that approach is explored in the next chapter.

The ability to define classes gives OWL a great deal of expressive power. Reasoners are able to place instances into the proper subclass based on given definitions. This additional classification often results in new inferred knowledge being produced about an instance.

Of interest as future work is that the definitions for classification may themselves change if enemy tactics are unconventional. For example, a friendly contact may be captured and rejoin the fight as a threat, or a neutral contact might be overcome and driven as a weapon. Significant variants are possible that may themselves require higher-level rules to define different sets of classification criteria. Alternatively, further definitions might be written that apply special cases and exceptions which can override default definitions. Thus ontology design can become quite sophisticated. Traceability of

definitions, rules, properties and reasoning criteria will remain an important requirement for these classification systems.

E. ONTOLOGY TESTING

Testing an ontology is an important step in the design process. Iterative testing throughout the process of development ensures that class definitions do not contradict one another. Once the ontology is designed, test cases are inserted as instances to test whether the desired classification is actually performed by the reasoner. Protégé and RacerPro provide a variety of testing aids that are described in the following section.

1. Built-in Ontology Tests

The Protégé-OWL tool includes a set of built-in tests to ensure that an ontology is following the rules of the OWL specification. The Protégé-OWL test suite performs four different types of test and returns a list of warnings or errors inside a dialog box as shown in Figure 57. The "Run ontology tests..." menu item is selected to begin the tests. A developer may choose which tests to run by changing the preferences under the OWL menu.

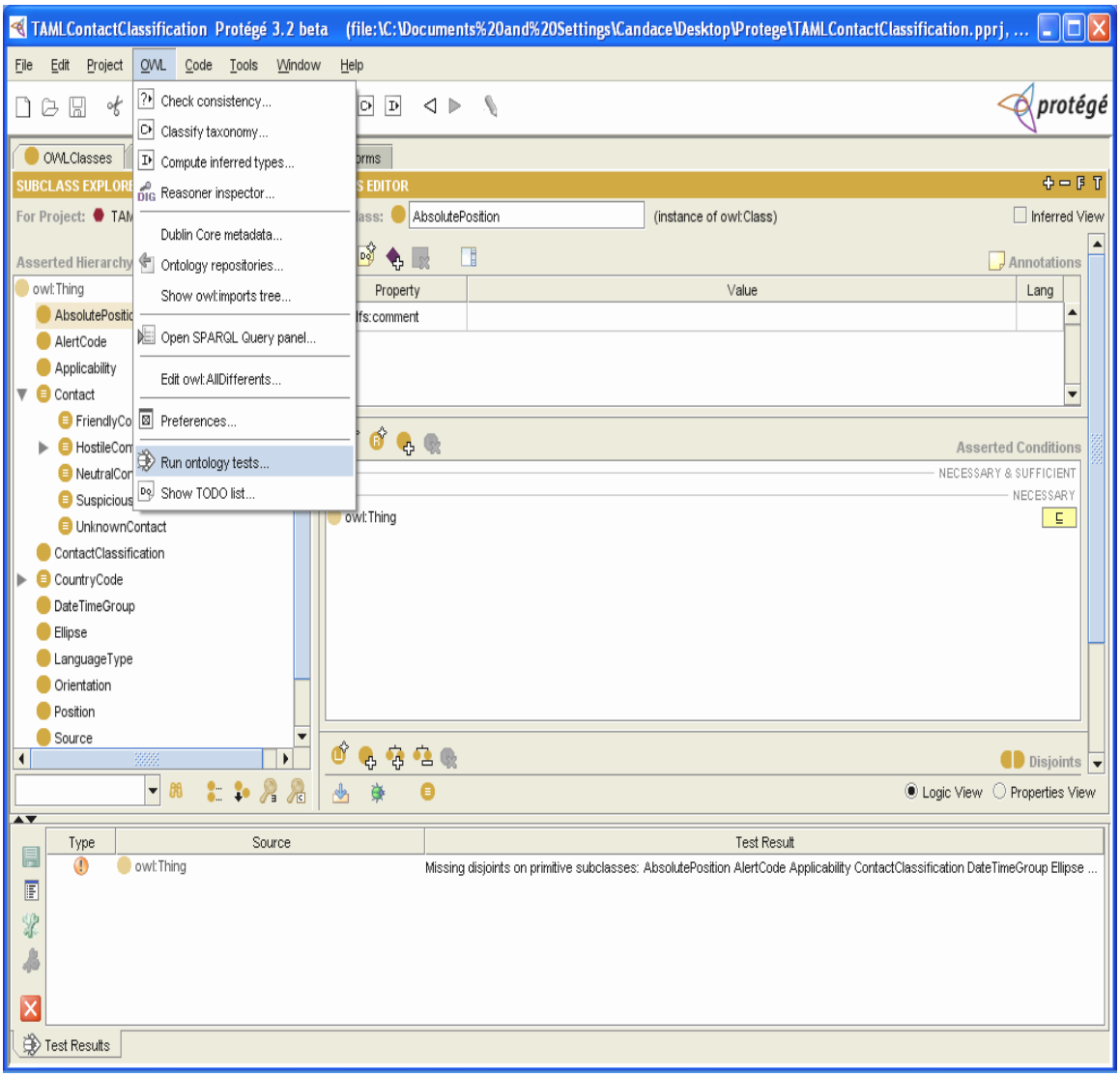


Figure 57. Invoking the built-in ontology test function in Protégé-OWL.

The test function also provides basic OWL-DL tests to ensure none of the classes or properties include OWL-Full constructs. These tests include checking to ensure there are no cardinality restrictions on transitive properties and transitive properties are not listed as functional. If a transitive property has either of these functions, then the test returns an error because the ontology is no longer OWL-DL. An extra validation check was done on the TAMLContactClassification.owl file using the WonderWeb OWL validator (http://phoebus.cs.man.ac.uk:9999/OWL/Validator) which indicated the file was valid OWL-DL.

The first set of tests that Protégé-OWL provides are maintenance tests. These tests look for classes that are suspicious or unnecessary. For example, if the ontology lists a class that does not have any children or any definitions, Protégé-OWL returns a warning that the class may not be needed.

Protégé-OWL also performs sanity tests to check for common modeling errors such as leaving the domain of a property empty. The last type of test is a style test which makes suggestions on how the ontology might be improved. These tests check for issues like duplicate restrictions on a subclass or a property with a minimum cardinality of zero. The results returned by these tests are suggestions and do not change the behavior of the model, but nevertheless may make the ontology more understandable.

The Protégé-OWL tests were run throughout the development of TAMLContactClassification.owl as incremental checks on the consistency of the ontology. At the end of development, a warning was returned that there may be missing disjoint definitions on some of the primitive classes as shown in Figure 57. The direct subclasses of owl:*Thing* all need to be disjoint since an instance may not belong to more than one of the classes. For example, an instance of *Applicability* may not also be an instance of *Contact*. Modeling the classes as disjoint is not necessarily important logically to the specific use case of classifying contacts. However, the disjoint property was added to improve the semantic clarity of the model and this addition may improve performance for some queries and some instance data. The TAML Contact Classification Ontology does not return any more warnings or errors when the built-in tests are run.

2. Reasoning Tests with RacerPro Reasoner

Reasoners are used to test OWL ontologies and to infer new information about a given OWL knowledge base. RacerPro was the reasoner first used to test and infer information over the TAML Contact Classification Ontology. RacerPro is a commercially available reasoner that includes an interface developed by the Description Logic Implementation Working Group (DIG) which enables it to receive and process information from Protégé-OWL. RacerPro was used to perform three types of reasoning

on the ontology: an ontology consistency check, a semantic consistency check on instances, and instance classification.

3. Testing for Ontology Consistency

RacerPro is used to check the consistency of the ontology. The reasoner returns an error if any class definitions contradict each other or do not make logical sense. For example, if the classes *FriendlyContact* and *HostileContact* are disjoint but both have a definition stating an instance belongs to their class if it has a *threatCode* value equal to UNK then an inconsistency error is thrown. A single instance may not belong to both classes; therefore, the definitions of the two classes may not overlap. Consistency checking is useful during definition development to ensure definitions convey the intended meaning. The TAML Contact Classification Ontology does not return any consistency errors.

4. Testing for Semantic Consistency

One potential use case for this ontology is to check for semantic consistency of *Contact* instances. For example, the ontology can ensure that a Contact instance does not contain conflicting data like a *threatCode* value of FRD and a *contactClassification* value of hostile. Semantic validation is useful since XML schemas are restricted to checking only syntactic correctness, not logical interdependencies among XML instance values.

Current reasoners do not place an emphasis on checking the semantic consistency of an instance. However, RacerPro does return an error when the inconsistency prevents classification of an instance. A test was run where an instance had a *threatCode* value equal to FRD and a *contactClassification* value of hostile. The reasoner returns an error as shown in Figure 58, but does not provide useful information about why the instance is incoherent, i.e. due to mutually incompatible logical constraints being simultaneously invoked in the OWL ontology.

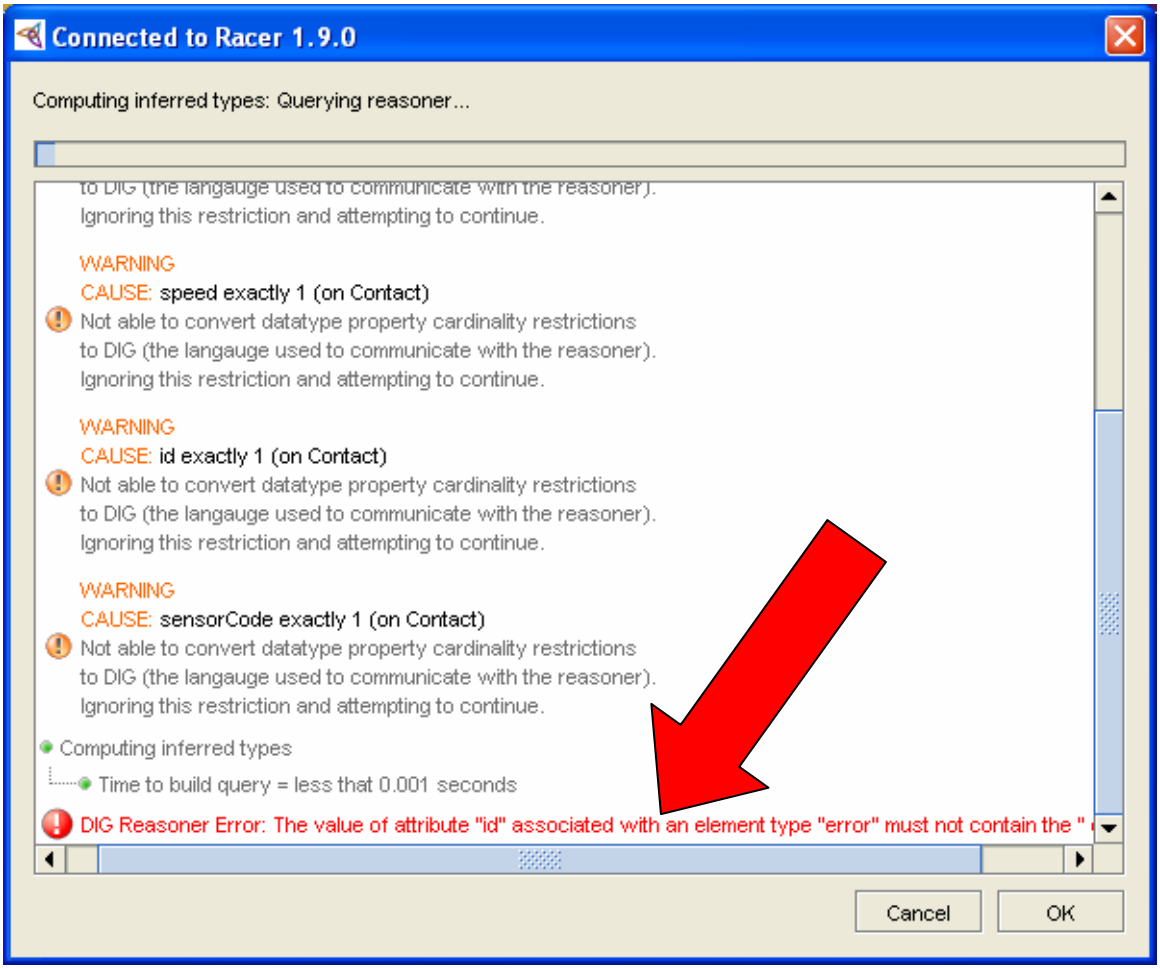


Figure 58. The reasoner returns an error when two properties contradict each other, but does not provide a useful reason for the error.

The ontology also defines the required fields of a contact. For example, every contact must have a position and orientation according to the necessary conditions of the *Contact* class. Protégé-OWL includes a built-in test for this type of consistency by highlighting in red the required property fields of an instance until a value is entered as shown in Figure 59. However, RacerPro does not report any error if a contact is entered without these necessary values. This is an implementation limitation of this reasoner when trying to use an ontology for semantic validation.

The internal DIG interface between Protégé and the reasoner does not support translating cardinality restrictions on datatype properties, so restrictions such as every *Contact* must have at least one and only one speed are ignored during the reasoning

process as shown in Figure 58. This is not a limitation of the OWL-DL language or even the RacerPro reasoner, but it is a representational limitation of the DIG interface.

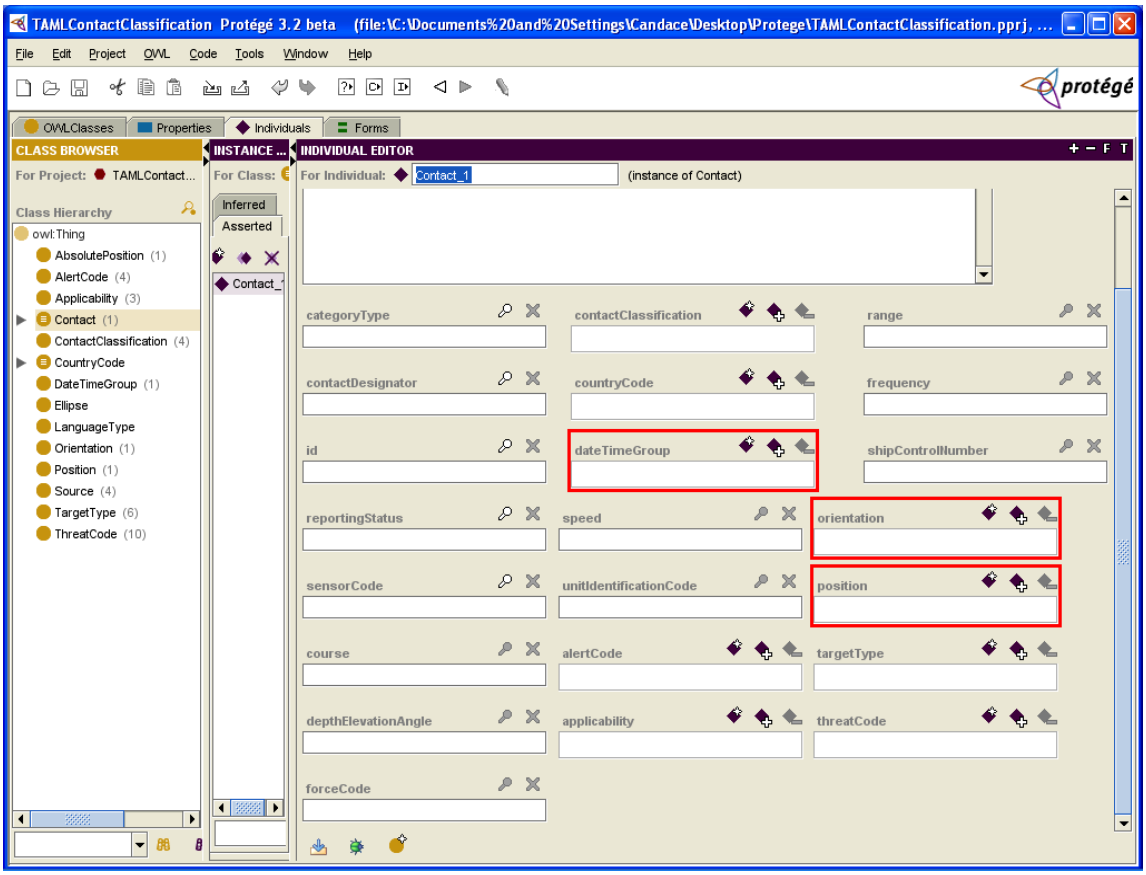


Figure 59. Protégé highlights required fields in red if they are missing a required value.

5. Testing the Classification Use Case

After the ontology is consistent and complete, the next step for the ontology developer is to test whether classifying instances returns the expected results. For example, does an instance with a *threatCode* equal to FRD actually get classified as a *FriendlyContact*. These tests were performed by creating test instances and running the instance classification capability of the reasoner.

Table 11 describes the test cases and the results. A test case was created for each definition within each class. The expected classification is indicated by the test name and only the property values that affected the test are shown in the table.

testName	countryCode	contactClassification	threatCode	targetType	speed	Inferred Classification
friendly1	N/A	friendly	N/A	N/A	N/A	FriendlyContact
friendly2	N/A	N/A	FRD	N/A	N/A	FriendlyContact
friendly3	US	N/A	N/A	submarine	N/A	FriendlyContact
friendly4	GB	N/A	N/A	warship	N/A	FriendlyContact
friendly5	CA	N/A	N/A	militaryAircraft	N/A	FriendlyContact
hostile1	N/A	hostile	N/A	N/A	N/A	HostileContact
hostile2	N/A	N/A	HOS	N/A	N/A	HostileContact
hostile3	IQ	N/A	N/A	militaryAircraft	N/A	HostileContact
hostile4	AF	N/A	N/A	submarine	N/A	HostileContact
hostile5	IR	N/A	N/A	smallboat	N/A	HostileContact
hostile6	IQ	N/A	N/A	warship	N/A	HostileContact
neutral1	N/A	neutral	N/A	N/A	N/A	NeutralContact
neutral2	N/A	N/A	NEU	N/A	N/A	NeutralContact
susp1	N/A	N/A	SUS	N/A	N/A	SuspiciousContact
susp2	IQ	N/A	N/A	civilianAircraft	N/A	SuspiciousContact
susp3	IQ	N/A	N/A	merchant	N/A	SuspiciousContact
unknown1	N/A	unknown	N/A	N/A	N/A	UnknownContact
unknown2	N/A	N/A	UNK	N/A	N/A	UnknownContact

Table 11. List and description of test cases used to test proper inference of Classification in TAMLContactClassification.owl.

A limitation of RacerPro was exposed during the instance classification testing. RacerPro is limited by speed when certain types of axioms are introduced. When the set of test cases is run against the ontology, the completion time is greater than two hours. To overcome this limitation, the ontology was reduced during testing. The definition of UnknownContact includes an axiom which states a *Contact* is unknown if it is not a *FriendlyContact, HostileContact, NeutralContact,* or *SuspectContact.* Surprisingly, when this definition is removed, the reasoner completion time is reduced to less than a second. This might limit the use of the ontology in real-time systems or limit the type of definitions given to classes. Since such reasoners have been developed for a variety of knowledge-representation languages over the past few decades, it is sensible to expect that highly efficient reasoners will become available in the near future as OWL and related recommendations reach final approval status.

F. SUMMARY

A TAML ontology provides a semantic model for contact classification in the USW tactical domain. The Contact Classification Ontology is a specialized ontology designed to answer a specific question. A full ontology of the entire TAML domain needs be developed to validate the semantics of TAML documents.

OWL-DL provides a powerful and standardized language for developing ontologies. Using a standardized language increases the capability of interoperability with other similar ontologies. A reasoner applied to an OWL ontology can add new information to the data stored in the ontology by classifying instances. The organization of the classes in the model and the class definitions determine how the reasoner classifies a particular instance.

Current reasoners have limitations. The largest limitation within the contact-classification ontology was the reasoner's inability to recognize and apply XML Schema datatype restrictions against datatype properties. For example, the reasoner ignores expressions that specify if the speed is less than or equal to five.

The OWL-DL language also has limitations. The language does not allow for rules that assign values to properties of instances. For example, a rule that specifies that if the *contactClassification* of a contact is equal to friendly then a corresponding *threatCode* is equal to FRD is not allowed. Expressions and definitions can be written for classes in OWL-DL but not for properties. The current specification of OWL-DL also limits the use of datatype properties in ontologies. User-defined datatypes are not supported which means special class definitions like "if speed is less than five" are not possible. Version 1.1 of the OWL specification is expected to include support for user-defined types.

This chapter demonstrates one powerful use of OWL-DL to model a specific tactical problem. It also highlights the limitations of ontology design and their practical impact. The next chapter explores ways to overcome some of the limitations exposed throughout the ontology design process.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. USING SWRL TO ADD RULES TO THE TAML CONTACT CLASSIFICATION ONTOLOGY

> Rules are everywhere. They are found in many domains, disciplines, and industries….Rules are a key element of the Semantic Web vision, allowing integration, derivation and transformation of data from multiple sources in a distributed, transparent and scalable matter.
>
> -- W3C Call for Participation, 2005

A. INTRODUCTION

This chapter provides an overview of the extension of the TAML Contact Classification Ontology by adding rules using the Semantic Web Rule Language (SWRL). The ontology addresses the same classification use case as the OWL-only version but overcomes some of the limitations by adding rules.

B. SEMANTIC WEB RULE LANGUAGE (SWRL)

Interoperability is one of the primary goals of the SW and rules play a key role in achieving that goal (O'Conner, 2005). SWRL is the proposed standard rule language for the SW. The language was formed by combining OWL-DL constructs with the Rule Markup Language (RuleML). SWRL is defined as an OWL ontology which can be imported into any other OWL ontology to provide another level of expressiveness. OWL is restricted to defining classes, but SWRL allows developers to define properties through rules. Although the SWRL ontology may be imported into any ontology, current SW reasoners are unable to interpret and execute SWRL rules due to its relative newness. Reasoner support is likely to improve rapidly.

SWRL is used to write rules in terms of the concepts and properties defined within an ontology. The language specification does not restrict how reasoning ought to be performed, so developers are free to use existing rule engines to interpret and reason with SWRL rules (O'Conner, 2005). For example, work is currently being done to integrate the Jess rule engine into the Protégé-OWL SWRL plug-in so that SWRL rules

can be executed locally and the results returned to the original ontology loaded in Protégé.

SWRL is a key component of the SW architecture. Rules add new capabilities to ontologies including rule-based reasoning and expressive-instance querying. The main goal of SWRL is to provide a semantically coherent way to create rule-bases that allow machines to reason about a domain defined by an ontology (Grosof, 2003). When SWRL is added to an ontology, SPARQL can be used to create expressive instance queries with respect to the terminology defined in the knowledge base. Thus the upper levels of the SW stack are beginning to interoperate in complementary ways.

1. SWRL Built-Ins

SWRL built-ins are a set of predefined predicates which are available for defining SWRL rules. Built-ins are predicates which accept several arguments and return a true or false value. The SWRL built-in specification reuses existing built-ins from XQuery and XPath which further enhances compatibility with other XML technologies. The SWRL built-ins were designed using a modular approach in order to provide extensibility and flexibility (http://www.daml.org/2004/04/swrl/builtins.html). Developers are able to choose which built-in implementations to include within their applications. The language is likely to be extended in future releases.

The built-ins are divided into several different types of functionality including comparison, math, Boolean, string, date/time/duration, URIs and list built-ins. Comparison built-ins are used to determine whether two values are greater than, less than, or equal to each other. Table 12 provides an overview of the format and definition of the comparison built-ins that are used in the TAML Contact Classification Ontology.

Built-In	Origin	Definition
swrlb:equal(x, y)	XQuery	Satisfied if and only if the value of x and y are the same.
swrlb:notEqual(x, y)	swrlb:equal	The negation of swrlb:equal.
swrlb: lessThan(x, y)	XQuery	Satisfied if x and y are both identified as (or mappable to) some implemented type and x is less than y.
swrlb:lessThanOrEqual (x, y)	swrlb:lessThan, swrlb:equal	Satisfied if x and y are both identified as (or mappable to) some implemented type and x is less than or equal to y.
swrlb:greaterThan(x, y)	XQuery	Satisfied if x and y are both identified as (or mappable to) some implemented type and x is greater than y.
swrlb:greaterThanOrEqualTo (x, y)	swrlb:greaterThan, swrlb: equal	Satisfied if x and y are both identified as (or mappable to) some implemented type and x is greater than or equal to y.

Table 12. List and description of the SWRL comparison built-in operators used to define rules within the TAML Contact Classification Ontology [After Ref. http://www.daml.org/2004/04/swrl/builtins.html].

Math built-ins are defined for numeric types and allow developers to incorporate rules using math operators such as add, subtract, multiply, divide, sin, cos, floor, etc. Table 13 describes the multiply built-in which was used within the TAML Contact Classification Ontology.

Built-In	Origin	Definition
swrlb:multiply (x, y, z)	XQuery	Satisfied if the x is equal to the product of y and z.

Table 13. Description of the SWRL multiply built-in used within the TAML Contact Classification Ontology [After Ref. http://www.daml.org/2004/04/swrl/builtins.html].

The SWRL built-in specification defines other types of built-ins which were not used in the TAML Contact Classification Ontology. An overview of other types of built-ins is provided in Table 14. SWRL built-ins are defined using the http://www.w3.org/2003/11/swrlb namespace.

Type	Example Built-In	Description of Example
Built-Ins for Boolean Values	swrlb:booleanNot (x, y)	Satisfied if and only if the first argument is true and the second argument is false, or vice versa.
Built-Ins for Strings	swrlb:stringLength (x, y)	Satisfied if and only if the first argument is equal to the length of the second argument.
Built-Ins for Date, Time and Duration	swrlb:subtractYearMonthDurations (x, y, z)	Satisfied if and only if the yearMonthDuration of the first argument is equal to the arithmetic difference of the second argument and the third argument
Built-Ins for URIs	swrlb:resolveURI (x, y, z)	Satisfied if and only if the URI reference of the first argument is equal to the value of the URI reference of the second argument resolved relative to the base URI (the third argument)
Built-Ins for Lists	swrlb:listIntersection (x, y, z)	Satisfied if and only if the first argument is a list containing elements found in both the second argument and the third argument

Table 14. Description of other SWRL built-ins not used in the TAML Contact Classification Ontology [After Ref. http://www.daml.org/2004/04/swrl/builtins.html].

2. SWRL Rule Format

SWRL is used to add Horn-like rules to an ontology. Horn logic is a formal language with a corresponding inference procedure. Horn logic enables the expression of rules and queries and includes inference capabilities to compute answers to queries based on rules. The Horn logic language has declarative semantics which can be used to express any computational function. This expressive power makes some queries in Horn logic undecidable (Mozetic, 2006). Horn logic rules are expressed using Horn clauses which include an antecedent followed by a consequence. SWRL rules are modeled using the same antecedent/consequence clause.

SWRL rules consist of two parts: the antecedent and the consequence. The antecedent is called the *rule body* and the consequence is called the *head.* The rule body and head are made up of one or more *atoms*. Atoms are predicates which evaluate to true or false (O'Conner, 2005). When all the atoms in the rule body evaluate to true, then the head is executed. Complex rules are created by combining multiple atoms.

SWRL rules can refer to individuals or concepts defined within the ontology. Variables are inserted to create general rules which in turn replace the variable with all possible instances when executed. Variables are preceded by the ? symbol. In the rule below, the variable ?x is defined as every instance of *HostileContact*. Thus the machine processes the rule for each *HostileContact* instance.

SWRL rules are used to define properties or set the value of properties for instances. For example, the following rule:

```
HostileContact(?x) ∧ speed(?x, ?speed) ∧ range(?x, ?range) ∧
swrlb:greaterThanOrEqual(?speed, 5) ∧
swrlb:lessThanOrEqual(?range, 20000)→ isThreat(?x, true)
```

sets the value of *isThreat* to true for all *HostileContact* instances having a speed greater than or equal to five and having a range less than or equal to 20000. Note that the patterns for *speed* and *range* are used to bind local variables to the values associated with hostile contact ?x.

SWRL provides a standard syntax for defining rules in a machine-interpretable language. SWRL is currently in the proposal phase at the W3C. SWRL is a powerful rule

language that is used to further define OWL properties which already adds great vale to SW capabilities.

3. Protégé-OWL SWRLTab Plug-in

The SWRLTab is a Protégé-OWL plug-in which supports the definition and editing of SWRL rules. The plug-in can be further extended to execute SWRL rules with the integration of a compatible rule engine.

The SWRLTab includes four software components: SWRL Editor, SWRL Factory, SWRL Bridge and SWRL Built-in Bridge. The SWRL Editor is an extension of the OWL plug-in that provides a user interface for writing SWRL rules (http://protege.stanford.edu/plug-ins/owl/swrl). The editor is used to create new SWRL rules, edit existing SWRL rules, or read SWRL rules. The editor is included with the Protégé-OWL full install. The SWRL Factory provides a programmatic API for creating and modifying SWRL rules inside an OWL knowledge base. The Factory is used to manipulate rules from within an application. The SWRL Bridge provides the framework for integrating rule engines into Protégé to execute rules. Once a rule engine is integrated, the SWRL rules can be interpreted to the corresponding rule language, executed and then the results are translated back into OWL constructs for insertion into the knowledge base. The SWRL Built-in Bridge is an extension of the SWRL Bridge which provides mechanisms to dynamically create definitions of SWRL built-ins and execute them within a rule engine (O'Conner, 2006).

The SWRLTab provides the framework for creating, editing, saving and executing SWRL rules. The tab is a Protégé plug-in which can be extended to integrate various rule engines and incorporate support for SWRL built-ins. Figure 60 is a screenshot of a SWRL rule defined in the Protégé SWRLTab multi-line editor.

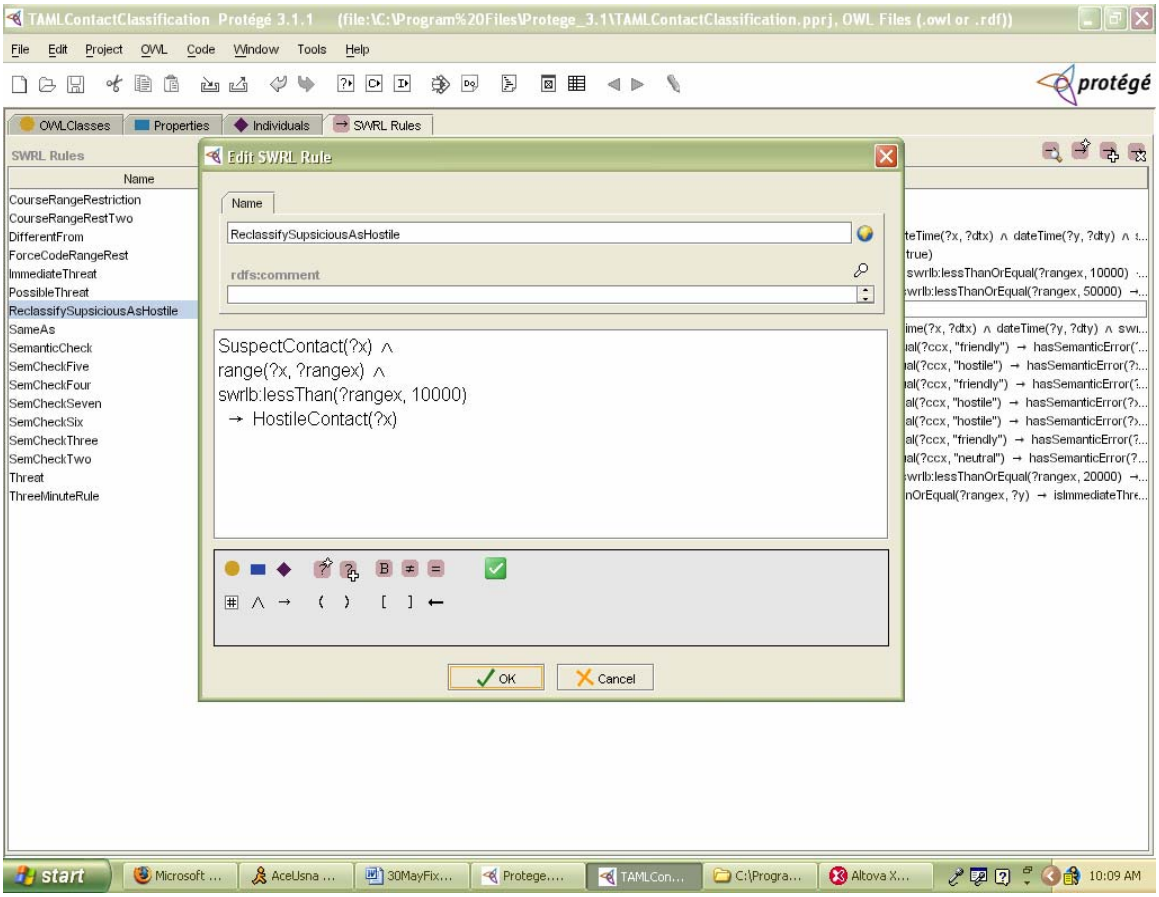


Figure 60. A screenshot of a SWRL rule inside the Protégé SWRL editor GUI. The rule is designed to reclassify instances of *SuspectContact* as instances of HostileContact when contact range is less than 10000.

The rule defined above is used to reclassify instances of *SuspectContact* as instances of *HostileContact* when their *range* is less than 10000. Figure 61 shows the source added to the OWL ontology by Protégé when the rule was defined using the SWRL editor. The rule is defined in terms of the SWRL ontology which was imported into Protégé from the Web.

```
<swrl:Imp rdf:ID="ReclassifySupsiciousAsHostile">
 <swrl:body>
   <swrl:AtomList>
     <rdf:rest>
       <swrl:AtomList>
         <rdf:first>
           <swrl:DatavaluedPropertyAtom>
             <swrl:argument1 rdf:resource="#x"/>
             <swrl:argument2 rdf:resource="#rangex"/>
             <swrl:propertyPredicate rdf:resource="#range"/>
           </swrl:DatavaluedPropertyAtom>
         </rdf:first>
         <rdf:rest>
           <swrl:AtomList>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
             <rdf:first>
               <swrl:BuiltinAtom>
                <swrl:builtin
                    rdf:resource="http://www.w3.org/2003/11/swrlb#lessThan"/>
                 <swrl:arguments>
                   <rdf:List>
                     <rdf:rest>
                      <rdf:List>
                       <rdf:first
       rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10000</rdf:first>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                 </rdf:List>
                     </rdf:rest>
                     <rdf:first rdf:resource="#rangex"/>
                   </rdf:List>
                 </swrl:arguments>
                 </swrl:BuiltinAtom>
             </rdf:first>
           </swrl:AtomList>
         </rdf:rest>
       </swrl:AtomList>
     </rdf:rest>
     <rdf:first>
       <swrl:ClassAtom>
         <swrl:classPredicate rdf:resource="#SuspectContact"/>
         <swrl:argument1 rdf:resource="#x"/>
       </swrl:ClassAtom>
     </rdf:first>
     </swrl:AtomList>
   </swrl:body>
   <swrl:head>
     <swrl:AtomList>
       <rdf:first>
         <swrl:ClassAtom>
           <swrl:classPredicate rdf:resource="#HostileContact"/>
           <swrl:argument1 rdf:resource="#x"/>
         </swrl:ClassAtom>
     </rdf:first>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     </swrl:AtomList>
   </swrl:head>
</swrl:Imp>
```

Figure 61. The OWL source for the rule defined in Figure 60.

C. RULE DESIGN FOR TAML

The exemplar rules included in the TAML Contact Classification Ontology have been designed to demonstrate the expressiveness of SWRL and do not form an exhaustive list of all the rules needed to classify and prioritize contacts. The rules are designed to illustrate how SWRL overcomes some of the limitations discussed in the previous chapter. These rules add range restrictions to some of the defined datatype properties, improve the ability to check semantic consistency, prioritize contacts that are a threat, determine which contact instances refer to the same (or different) real-world entities and also demonstrate how complex rules can be written using SWRL built-in rules.

1. Datatype Range-Restriction Rules

The current version of OWL does not provide support for restricting the range of datatype properties. For example, restrictions on course such as "course values in degrees must be between zero degrees (inclusive) and 360 degrees (exclusive)" cannot be expressed in OWL. Range restrictions are initially imposed within the TAML schema, but the semantic model or ontology of the domain also needs to reflect datatype range restrictions. Placing restrictions on the allowed values for properties is important for property modeling, and also for evaluating the semantics (or meaning) of a concept.

Future versions of OWL are expected to support user-defined datatypes which can be used to make property restrictions. Nevertheless, until new versions of OWL are released, SWRL can be used to impose range restrictions on datatype properties.

The TAML schema defines range restrictions on certain elements which are represented as datatype properties in the TAML Contact Classification Ontology. It is important to represent these restrictions in the ontology since they are part of the semantic definition for the concept and properties of *Contact*. For example, the element *forceCode* is defined as having an integer value less than 100, so the semantic model is incomplete without this explicit information. SWRL rules were written for the TAML Contact Classification Ontology in order to enforce range restrictions as shown in Table

15. A new Boolean property, *rangeViolationError*, was created to highlight when an instance violates one of the rules. When the rules are executed by a rule engine, the *rangeViolationError* property field for all instances with values outside the defined range is set to "true". Note that 'range' in this context refers to a value outside the allowed range of values, not to computed distance in the real world.

SWRL Rule	Meaning
Contact(?x) ∧ course(?x, ?course) ∧ swrlb:lessThan(?course, 0) → rangeViolationError(?x, true)	If x is a *Contact* and has a *course* which is less than zero, then set the *rangeViolationError* property for x to true.
Contact(?x) ∧ course(?x, ?course) ∧ swrlb:greaterThanOrEqualTo(?course, 360) → rangeViolationError(?x, true)	If x is a *Contact* and has a *course* which is greater than or equal to 360, then set the *rangeViolationError* property for x to true.
Contact(?x) ∧ forceCode(?x, ?forceCode) ∧ swrlb:greaterThan(?forceCode, 99) →rangeViolationError(?x, true)	If x is a *Contact* and has a *forceCode* which is greater than 99, then set the *rangeViolationError* property for x to true.

Table 15. List and description of the SWRL rules designed to indicate when an instance includes a range violation on a datatype property.

Future versions of OWL are expected to include a more direct approach for modeling range restrictions on datatype properties by allowing developers to create user-defined datatypes. For example, instead of creating a rule to set an error flag when a *Contact's course* is greater than 360, a developer can set the fundamental data range for the property *course* to be an integer between zero and 360 with a user-defined datatype. User-defined datatypes present a more direct approach so that a reasoner might provide the user an error message when the course is greater than 360 as opposed to just setting an error flag. Nevertheless, SWRL does provide a helpful alternate approach, through the use of properties designed to be error flags which in turn can be queried to return instances with errors.

2. Checking Semantic Consistency with SWRL

A use case for checking the semantic consistency of TAML documents using an OWL ontology was proposed in Chapter VII. The OWL ontology signaled inconsistencies in instance data when the reasoner was unable to classify an object.

However, the reasoner did not provide a valuable error message to help locate the causes of the logical inconsistencies. SWRL rules provide a method to flag instances with conflicting data or semantic inconsistencies.

In the TAML Contact Classification Ontology, a new Boolean property named *hasSemanticError* is defined to act as a settable flag when SWRL rules detect inconsistencies within particular instances. SWRL rules defining these error flags are shown in Table 16. For example, if instance A simultaneously has a *threatCode* value equal to "HOS" and a *contactClassification* value not equal to "hostile" then the *hasSemanticError* property for instance A is set to "true" when the corresponding consistency-checking rule is executed.

SWRL Rule	Meaning
`Contact(?x) ∧ threatCode(?x, ?fc) ∧ contactClassification(?x, ?cc) ∧ swrlb:equal(?fc, "HOS") ∧ swrlb:notEqual(?cc, "hostile") → hasSemanticError(?x, true)`	If x is *Contact* and has a *threatCode* equal to "HOS" and a *contactClassification* not equal to "hostile", then set the *hasSemanticError* property for x to true.
`Contact(?x) ∧ threatCode(?x, ?fc) ∧ contactClassification(?x, ?cc) ∧ swrlb:equal(?fc, "UNK") ∧ swrlb:notEqual(?cc, "unknown") → hasSemanticError(?x, true)`	If x is *Contact* and has a *threatCode* equal to "UNK" and a *contactClassification* not equal to "unknown", then set the *hasSemanticError* property for x to true.
`Contact(?x) ∧ threatCode(?x, ?fc) ∧ contactClassification(?x, ?cc) ∧ swrlb:equal(?fc, "NEU") ∧ swrlb:notEqual(?cc, "neutral") → hasSemanticError(?x, true)`	If x is *Contact* and has a *threatCode* equal to "NEU" and a *contactClassification* not equal to "neutral", then set the *hasSemanticError* property for x to true.
`Contact(?x) ∧ threatCode(?x, ?fc) ∧ contactClassification(?x, ?cc) ∧ swrlb:equal(?fc, "FRD") ∧ swrlb:notEqual(?cc, "friendly") → hasSemanticError(?x, true)`	If x is *Contact* and has a *threatCode* equal to "FRD" and a *contactClassification* not equal to "friendly", then set the *hasSemanticError* property for x to true.
`Contact(?x) ∧ threatCode(?x, ?fc) ∧ contactClassification(?x, ?cc) ∧ swrlb:equal(?fc, "SUS") ∧ swrlb:notEqual(?cc, "suspicious") → hasSemanticError(?x, true)`	If x is *Contact* and has a *threatCode* equal to "SUS" and a *contactClassification* not equal to "suspicious", then set the *hasSemanticError* property for x to true.

Table 16. List and description of the SWRL rules designed to indicate when an instance includes a semantic inconsistency.

SPARQL queries can also be included in an application to query the results of SWRL processing. For example, once a set of SWRL rules are executed on the TAML Contact Classification Ontology, then the SPARQL query shown in Figure 62 returns the *id* of all instances with the *hasSemanticError* property set to "true".

```
SELECT  ?id

FROM <file:TAMLContactClassification.owl>

WHERE
{
     ?x id ?id.
     ?x hasSemanticError ?hasSemanticError.
     FILTER (?hasSemanticError = "true")
}
```

Figure 62. SPARQL query which returns the *id* of all *Contact*s where the *hasSemanticError* property is equal to "true".

SWRL provides a mechanism for semantically validating documents by allowing a developer to explicitly write rules defining conflicting property values that indicate inconsistent semantics. The rules listed in Table 16 refer to one particular type of semantic inconsistency when evaluating the data associated with *Contacts.* A full ontology model of TAML needs to include rules for defining many types of inconsistencies and might provide a mechanism for semantically validating TAML documents. Further rules might be defined for prosecution tactics and even rules of engagement (ROEs). This is an important area for future work.

3. Prioritizing Hostile Contacts

SWRL rules are used in the TAML Contact Classification Ontology to prioritize hostile contacts based on their threat level. Three Boolean properties were added to the ontology to indicate three various levels of threat: *isPossibleThreat, isThreat,* and *isImmediateThreat*. The levels are determined by the *speed* and *range* of the *HostileContact* as illustrated in Table 17. The faster a *HostileContact* is moving and the

closer the *HostileContact* is to the detecting platform, the higher the threat level assigned to *HostileContact*. Note that *isThreat* and *isImmediateThreat* are not intended to be mutually exclusive. An immediate threat must also be considered a threat, but a contact being a threat does not automatically imply that the threat is immediate.

SWRL Rule	Meaning
`HostileContact(?x) ∧ speed(?x, ?speed) ∧ range(?x, ?range) ∧ swrlb:greaterThanOrEqual(?speed, 0) ∧ swrlb:lessThanOrEqual(?range, 50000) → isPossibleThreat(?x, true)`	If x is a *HostileContact* with a *speed* greater than or equal to zero and a *range* less than or equal to 50000 then set the property *isPossibleThreat* equal to true.
`HostileContact(?x) ∧ speed(?x, ?speed) ∧ range(?x, ?range) ∧ swrlb:greaterThanOrEqual(?speed, 5) ∧ swrlb:lessThanOrEqual(?range, 20000) → isThreat(?x, true)`	If x is a *HostileContact* with a *speed* greater than or equal to five and a *range* less than or equal to 20000 then set the property *isThreat* equal to true.
`HostileContact(?x) ∧ speed(?x, ?speed) ∧ range(?x, ?range) ∧ swrlb:greaterThanOrEqual(?speed, 10) ∧ swrlb:lessThanOrEqual(?range, 10000) → isImmediateThreat(?x, true)`	If x is a *HostileContact* with a *speed* greater than or equal to ten and a *range* less than or equal to 10000 then set the property *isImmediateThreat* equal to true.

Table 17. List and description of the SWRL rules designed to prioritize hostile contacts by their threat level.

The SWRL comparison built-in rules were used to design the threat level rules. For example, if instance A is a HostileContact with a speed greater than or equal to 10 knots and a range less than or equal to 10000, then when the corresponding SWRL rule is executed, the value of the *isImmediateThreat* property corresponding to instance A will be set to "true". Once the rules are executed, the results can be returned by the SPARQL query defined in Figure 63.

```
SELECT  ?id

FROM <file:TAMLContactClassification.owl>

WHERE
{
     ?x id ?id.
     ?x isImmediateThreat ?isImmediateThreat.
     FILTER (?isImmediateThreat = "true")
}
```

Figure 63. SPARQL query which returns the *id* of all *Contact*s where the *isImmediateThreat* property is equal to "true".

4. Same As/Different From Rules

SWRL includes predicates for indicating that two instances refer to the same real world entity and for indicating that two instances refer to different real world entities (O'Conner, 2005). These predicates are useful for determining if two instances detected from different platforms are referring to the same *Contact*. For example, if System A and System B provide separate TAML documents into the TAML Contact Classification Ontology, some of their instance data may overlap. For example, System A may have independently detected the same *Contact* as System B. The rules in Table 18 are used to indicate whether any instances in the knowledge base refer to the same or different entities. For example, if instance A and instance B are both Contacts with equal positions at the same time then the two instances are referring to the same entity because only one entity can occupy a position at a particular time.

SWRL Rule	Meaning
`Contact(?x) ∧ Contact(?y) ∧ position(?x, ?posx) ∧ position(?y, ?posy) ∧ swrlb:equal(?posx, ?posy) ∧ dateTime(?x, ?dtx) ∧ dateTime(?y, ?dty) ∧ swrlb:equal(?dtx, ?dty) → sameAs(?x, ?y)`	If x and y are instances of *Contact* with the same *position* at the same point in time, then the two instances refer to the same *Contact*.
`Contact(?x) ∧ Contact(?y) ∧ position(?x, ?posx) ∧ position(?y, ?posy) ∧ swrlb:notEqual(?posx, ?posy) ∧ dateTime(?x, ?dtx) ∧ dateTime(?y, ?dty) ∧ swrlb:equal(?dtx, ?dty) → differentFrom(?x, ?y)`	If x and y are instances of *Contact* with different *positions* at the same point in time, then the two instances do not refer to the same *Contact*.

Table 18. List and description of the SWRL rules *sameAs* and *differentFrom*, designed to indicate whether two instances refer to the same or a different real world Contact.

The current rules defined in Table 18 require the positions to be exactly the same. However, the rules can be enhanced by taking errors and small position/time differences into account if the accuracy of the detecting systems is known. The SWRL math built-ins provide the capability to write rules which take into account certain amounts of error. This is another valuable task for future work.

5. Adding Complex Rules

Several SWRL atoms can be combined with the "and" operator to create complex rules. The inclusive "or" operator cannot be used to exclusively separate atoms, but this restriction does not limit the expressivity of the language since two separate rules can be written to achieve the same effort, instead of one rule using the inclusive "or" operator. For example, if a developer wants to specify a condition whether the speed is greater than five "or" the range is less than 10000, then two rules are formed. The first rule specifies the condition if the speed is greater than five, and the second rule separately specifies the condition if the range is less than 10000. The consequence of each rule remains the same, so the execution of the two rules in combination forms the equivalent of the originally intended exclusive "or" statement.

The combination of several atoms is used in the TAML Contact Classification to write a rule based on the navigation rule of thumb known as the "three-minute rule". The three-minute rule states that the yards a *Contact* travels in three minutes is equal to the ship's speed multiplied by 100. The three-minute rule can be used inside the TAML Contact Classification Ontology to determine whether a *HostileContact* can reach the platform within three minutes. If the *Contact* is closing in quick enough to reach the platform in three minutes then it is labeled as an immediate threat as shown in Table 19.

SWRL Rule	**Meaning**
HostileContact(?x) $\wedge$ speed(?x, ?speed) $\wedge$ swrlb:multiply(?y, ?speed, 100) $\wedge$ range(?x, ?range) $\wedge$ swrlb:lessThanOrEqualTo(?range, ?y) $\rightarrow$ isImmediateThreat(?x, true)	This rule implicitly embeds the "three minute rule" to determine how soon a *Contact's range* will close with the detecting platform. If *Contact* x is a *HostileContact* and the *range* is less than or equal to the product of its *speed* and 100 then x can reach the detecting platform in less than three minutes, so declare x an immediate threat.
SuspectContact(?x) $\wedge$ range(?x, ?range) $\wedge$ swrlb:lessThan(?range, 10000) $\rightarrow$ HostileContact(?x)	If Contact x is a *SuspectContact* within 10000 yards then change the classification to *HostileContact*.

Table 19. List and description of SWRL rules added to the TAML Contact Classification Ontology to demonstrate how complex rules can be formed.

SWRL rules can also be used to reclassify instances of *Contact.* For example, if an instance of *SuspectContact* comes within 10000 yards of its detecting platform, it is reclassified as a *HostileContact* as shown in Table 19.

SWRL rules provide new expressive power for an OWL ontology. They are used within the TAML Contact Classification Ontology for several purposes including flagging semantically inconsistent instances, adding range restrictions to datatype properties, prioritizing instances of *HostileContact* and reclassifying *Contacts*.

D. SWRL LIMITATIONS

SWRL provides a standard rule language for the SW. One goal of the W3C is to make SW languages simple (and not necessarily complete) in order to increase adoption (Herman, 2004). This philosophy leads to some limitations in SWRL.

All SWRL atoms or predicates are boolean which means that they must evaluate to "true" or "false". This does not limit the expressivity of SWRL, although it can make writing some rules less natural and more cumbersome. The binary predicate nature of SWRL also affects built-ins. Built-ins are defined as axioms as opposed to functions (Matheus, 2005). For example, in the rule which uses the "three-minute rule" to determine whether a *HostileContact* is an immediate threat, it might have been more natural to write the rule as:

```
HostileContact(?x) ∧ speed(?x, ?speed) ∧
range(?x, ?range) ∧ ?range >= (speed x 100)→
isImmediateThreat(?x, true)
```

In the above rule the *range* of the Contact is compared to the value of the product of the *speed* and 100. The SWRL version from Table 19 is less intuitive because the value of the multiplication is stored in a variable and then the variable is compared against the *range* of the *Contact*. Nevertheless, the two rules are semantically equivalent.

The current SWRL specification does not include a function for asserting new entities. Developers are not able to write rules where a new instance of a class is created if certain conditions are met. For example, it might be useful to create an *Alert* class where the instances of the class are messages which alert a user to an immediate threat. If SWRL were someday to allow the assertion of new entities, then a rule might be written to create a new instance of *Alert* each time the property *isImmediateThreat* for an instance is set to "true".

OWL and SWRL are both based on an Open World Assumption (OWA) which restricts a machine from assuming something is false just because it is not asserted to be true (http://en.wikipedia.org/w/index.php?title=Open_World_Assumption&action). This is an important part of the OWL language since everything must be explicitly stated and not assumed. OWL ontologies assume that the data in the knowledge base is incomplete, a state of affairs that is often true in the tactical domain. However, the OWA can be

limiting when writing rules because it is often necessary to make decisions based on the assumption that the data available is complete (Matheus, 2005). Many rule languages assume a Closed World Assumption (CWA) where negation can be used to indicate failure (http://en.wikipedia.org/wiki/Negation_as_failure). With the CWA, if something is not proved as true then it is assumed to be false. In some situations there is a need for SWRL rules to use "negation as failure", but this feature is currently not available.

The SWRL syntax is difficult for humans to read and write. However, tools such as the Protégé SWRLTab reduce this problem by providing user-friendly GUI's for creating and editing rules. Another proposed approach has been to use a higher-level language like Prolog for creating SWRL rules, and then use a tool to convert them to SWRL or XML format (Matheus, 2005).

Another limitation of the SWRL language is the lack of support for SWRL rule execution within SW reasoners. Current reasoners provide little or no support for SWRL. Integrating support for a rule language into an existing Description Logic (DL) reasoner is a difficult process because the overlapping semantics of the two languages have to be mapped and aligned together. However, the integration of SWRL into OWL is important in order to create and reason with rule-bases that mention vocabularies specified by an ontology (Grosof, 2003). RacerPro currently supports rules that only mention specific instances and do not contain variables. Support for more general rules involving variables is expected to be available in the near future. SWRL is not a completed language, thus some of its limitations may be corrected in future versions.

E. USING RULE ENGINES TO EXECUTE SWRL RULES

An alternate approach to providing SWRL support within DL reasoners is to integrate existing rule engines such as Jess, Drools, or Algernon for executing SWRL rules, while allowing DL reasoners such as RacerPro and Pellet to perform the reasoning associated with OWL constructs. Figure 64 illustrates the translations which are needed to integrate a rule engine with the SWRLTab inside Protégé. Referring to Figure 64, SWRL rules (1) and the facts about the instances and classes defined in the ontology have to be translated into the language used by the rule engine (2). The rule engine runs the

rules and outputs the results of the reasoning (3) which must then be translated back into OWL and input into the ontology (4). The DL reasoner is used to check for any inconsistencies that may have been inserted into the ontology and also to reclassify the instances based on the inserted information (Plas 2006). Note that these approaches are required of tool designers, not tool users.

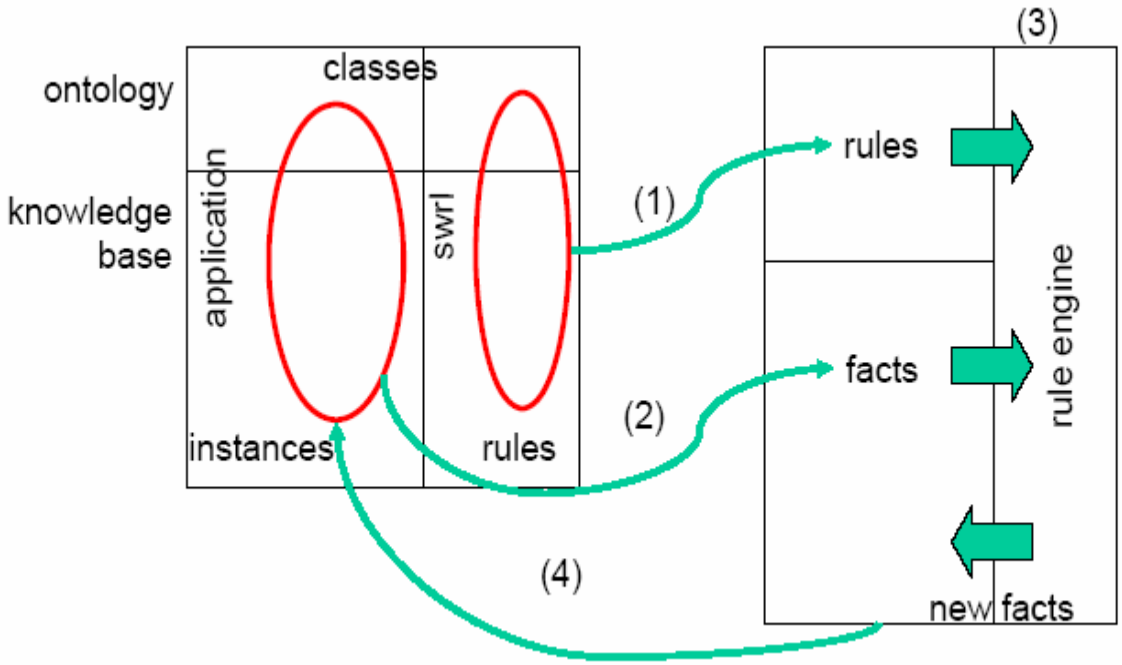


Figure 64. Multiple logical translations are needed in order to be able to execute SWRL rules using a rule engine integrated into Protégé [After Ref. Plas, 2006].

This approach requires the integration of both a DL reasoner and a rule engine in Protégé, but benefits from the existence of powerful rule engines which have already been implemented. A disadvantage to this approach is that it potentially allows the application-specific translations of SWRL rules to inadvertently introduce inconsistencies into the ontology (O'Conner, 2005). Once SWRL rules are executed, the results insert new information or facts into the ontology which may conflict with existing OWL constraints. Such potential inconsistencies are not revealed until the DL reasoner is run against the changed knowledge base.

A SWRLJessTab plug-in is currently under development for Protégé. The plug-in is being implemented into the SWRLTab using the SWRL Bridge infrastructure. The SWRLJessTab is designed to integrate the Jess rule engine into the Protégé SWRLTab in order to execute SWRL rules. Jess was chosen as the rule engine because, like Protégé, it is written in Java, has a large user base, is well documented and is easy to use and configure (O'Conner, 2005). The SWRLJessTab performs four main tasks: converts facts about OWL individuals to Jess facts, translates SWRL rules to Jess rules, performs inference using the rules, and translate the facts back to OWL to insert them into the knowledge base. The plug-in provides the translations and executes the rules for the user. An unfinished version of the plug-in is available (http://smi-web.stanford.edu/people/moconnor/swrl/SWRLJessTab.html). The current version does not support translating SWRL built-ins into Jess rules; therefore, in its current state it was not useful for testing and running the TAML Contact Classification Ontology. Conceivably a future version of Jess might support SWRL directly without the need for such rule conversions.

F. SUMMARY

SWRL is an important component within the SW architecture. Automated reasoning relies on rules layered on top of an ontology. SWRL rules add new semantic information to the TAML Contact Classification Ontology. The rules are used to overcome the limitations of OWL and to improve the reasoning power of the ontology. Current reasoners do not support the interpretation and execution of SWRL rules, so testing of the improved ontology was not possible. The Protégé team is working to integrate the Jess rule engine into the SWRLTab in order to execute rules and return results to the ontology. Once support for SWRL is readily available, the language can provide important additional semantic support for ontology development and use.

IX. EXPERIMENTAL RESULTS

> The idea behind the Semantic Web is to give data more meaning….What began seven years ago as a research project, which some critics said was moving too slow and would never see the light of day in products, has turned into a reality where companies such as Adobe and Sun Microsystems are employing it for content management.
>
> -- Clint Boulton, 2004

A. INTRODUCTION

This chapter provides an analysis of the advantages and limitations of XML, RDF, OWL and SWRL when used to model a tactical domain. The chapter uses the TAML Contact Classification Ontology developed in this work to compare the various levels of semantic expressiveness provided by each SW language. This chapter also explores the advantages of SPARQL over XQuery.

B. SEMANTIC WEB (SW) LANGUAGE ASSESSMENTS

The SW is designed to provide a machine-interpretable semantic definition of a domain by defining a vocabulary and an explicit model or ontology of the domain. An ontology explicitly defines concepts, and the relationships between them, to provide context for the data being processed by a machine. A rule-base is layered on top of an ontology to provide machine instructions for deriving information from data. The SW is composed of several XML vocabularies in a layered architecture. The layered architecture of the SW allows designers to choose the level of semantics needed to model their domains. Each layer or language compatibly extends the types of statements which can be made about the concepts and properties in a domain. Table 20 illustrates the features of each language. The layers are combined together in order to form a complete semantic model of a domain. The following sections assess the capabilities of each language, alone and together, when applied to tactical problems.

	Platform Independent	Supports User-Defined Datatypes	Provides Syntactical / Structural Interoperability	Provides Explicit Semantics	Support for Defining Concepts	Support for First-Order Logic (FOL) Rules
XML Schema	YES	YES	YES	NO	NO Annotations only with no processing support.	NO
RDF	YES	YES	YES	YES	NO	NO
OWL	YES	NO (Future versions will)	YES	YES	Limited (Provides quantifier, cardinality, and has value restrictions)	NO
SWRL	YES	NO	YES	YES	Limited (Does not support quantifier and cardinality restrictions)	YES

Table 20. An overview of the capabilities and limitations of SW languages.

1. Tactical Assessment Markup Language (TAML) Schema

An XML Schema is a document which describes the structure and syntax of XML documents. The TAML Schema defines the element and attribute names, the allowed datatypes and the ordering of elements within TAML documents. Schemas provide structural and syntactical interoperability for message exchange within a domain. Since XML is platform and domain independent, the language can be used to define a vocabulary for any domain. For example, the TAML Schema serves as an external model for data exchange between USW systems.

XML's flexibility allows developers to choose how to structure data for interoperability. There are no explicit semantics associated with the structure of XML documents. In an XML Schema the semantics are implicitly defined for human understanding through the structure or hierarchy of the elements and the inclusion of definitions as annotations. For example, Figure 65 shows two pieces of TAML with similar structure but different meaning. The semantic meaning of the relationship between the elements on the left is a part-of relationship. *Latitude* and *Longitude* are part-of the definition of an *AbsolutePosition*. The child elements, *course* and *speed* on the right are characteristics or properties of the *Platform* being defined but do not make up a *Platform*.

```
<AbsolutePosition>
 <Latitude>36.366702</Latitude>
 <Longitude>-7.191296</Longitude>
</AbsolutePosition>
```

```
<Platform id="POSHMSFLAG">
 <Course>90</Course>
 <Speed units="knots">3.0</Speed>
</Platform>
```

Figure 65. TAML code from Trafalgar.xml which illustrates the semantic ambiguity in TAML documents.

Implicit semantics are defined within the TAML Schema so that human users might understand the data being exchanged, but machines need explicitly defined semantics in order to interpret and reason on data. XML by itself does not provide a standardized capability to add explicitly defined semantics within schema-defined vocabularies such as these above.

2. TAML Serialized as RDF

RDF enhances XML by representing data with explicit semantics. Data in RDF is represented by a series of triples. Each triple has a subject, predicate or property, and an object. The triple explicitly states that there is a relationship between a subject and an object which is defined by the predicate. RDF triples can be serialized as XML so documents benefit from the platform independence and syntactic interoperability provided by XML. TAML documents were serialized as RDF/XML in order to demonstrate how RDF defines semantics as shown by Figure 66. TAML serialized as RDF/XML can be inserted into any document without losing any of its meaning because the meaning is no longer defined by the structure of the document.

```
<Event id="x2">
      <StartTime>2005-05-12T14:00:00Z</StartTime>
      <EndTime>2005-05-13T14:00:00Z</EndTime>
      <Name>Event x2</Name>
</Event>
```

```
<rdf:Description
rdf:about="http://usw.xml.wg/TAMLExample.xml/Eventx2">
      <StartTime>2005-05-12T14:00:00Z</ StartTime>
      < EndTime>2005-05-13T14:00:00Z</ EndTime>
      < Name>Event x2</Name>
</rdf:Description>
```

Figure 66. Comparison of XML serialization (above) and RDF serialization (below) of literal objects.

RDF/XML has little hierarchy since the data structures are relatively flat and the order of relationships definitions within an RDF/XML document does not matter. This makes RDF/XML documents more verbose and less human readable. It is difficult for a human to read through a TAML document represented as RDF/XML and understand all of the many relationships defined. However, a machine is able to quickly graph and traverse the relationships. Human readability does not have to be traded for machine readability since TAML itself can be represented in either format and then translated to the other format using the Extensible Stylesheet Language for Transformations (XSLT). This flexibility allows both approaches to be supported compatibly and automatically.

Modifying the structure of XML documents when there is a change in the assumption or view of a domain is difficult. If the schema is modified then all applications relying on that schema or hierarchy to process data must also be modified. The RDF model of data is more easily modified since the semantics and not the structure determine how the applications process data (Herman, 2004).

3. OWL-Only TAML Contact Classification Ontology

The Web Ontology Language (OWL) is an XML vocabulary used to add machine-understandable context to data by explicitly defining the concepts/classes and properties which make up a domain. OWL enables designers to define concepts using quantifiers, cardinality and has-value restrictions. Properties are defined by a range, domain and certain characteristics such as functional and symmetric. SW reasoners are then used to infer the class membership of instances based on the definitions defined in the ontology.

The TAML Contact Classification Ontology explicitly defines the concept of a *Contact* within the TAML domain. The ontology also defines the different classifications of a *Contact* such as hostile and friendly. The goal of the ontology is to accept *Contact* instance data from a TAML document and then use a reasoner to infer the classification of the Contact. OWL constructs were used to create simple definitions of classifications using quantifiers and has-value constraints. The reasoner was able to correctly classify instance data based on the definitions. However, several limitations of OWL and implemented reasoners were exposed during this development.

The current version of OWL-DL only supports two XML schema datatypes: xsd:int and xsd:string. OWL-DL does not support other XML schema (xsd) datatypes or user-defined datatypes (Pan, 2004). This lack of support creates a problem when trying to input TAML documents into the ontology, since TAML documents do contain other datatypes including floating-point and user-defined datatypes. The TAML Contact Classification Ontology defines the range of all numerical datatypes to be xsd:int and all other datatypes to be xsd:string in order to comply with the current OWL specification. Unfortunately, some of the data semantics implied of the TAML Schema are lost during

this process. OWL support for user-defined datatypes is expected to enable range restrictions on datatype properties. For example, a user-defined datatype might place the restriction that a *Contact*'s *course* must be greater than or equal to 0 and less than or equal to 360 by creating a datatype to represent all integers between 0 and 360 inclusive. Support for floating-point and user-defined datatypes is needed and is expected to be available in the next version of OWL.

OWL also does not support placing restrictions on datatype properties (Pan, 2004). For example, if a designer wants to map a distance in kilometers to miles, he needs to constrain the value of distance in miles to 1.6 times the value in kilometers, but OWL does not offer an expression for this type of constraint. More support is needed for datatype properties within the OWL specification.

Current DL reasoners and the DIG interface between Protégé and reasoners have limitations which affect the level of reasoning which is possible using the TAML Contact Classification Ontology. For example, constraints which use negation are not handled well by either the RacerPro or the Pellet reasoner. In the TAML ontology, the class definition for *UnknownContact* is any *Contact* which is not a *HostileContact*, not a *FriendlyContact*, not a *NeutralContact* and not a *SuspectContact*. Adding this condition on the *UnknownContact* class causes the reasoner to stall and fail to complete. OWL allows for constraints on classes using negation, but more efficient reasoners are needed to interpret these constraints for real-time systems.

Cardinality restrictions on datatype properties are ignored during reasoning due to a limitation in the DIG interface between Protégé and RacerPro. For example, the condition, " a *Contact* may only have one *speed"* is not known by the reasoner even though it is stated in the ontology. There is no protocol defined in the DIG interface for translating these constraints from OWL to the corresponding reasoner language. Therefore, a reasoner does not return an error if it receives an instance of a *Contact* with more than one speed. Such omissions are expected to improve over time as fully OWL-capable reasoners become available.

OWL has several limitations when used to model a problem without the use of SWRL rules. The use of OWL alone is limited to simple classification problems which

do not require restrictions on datatype properties and do not use mathematical functions to determine results. In the case of the TAML Contact Classification Ontology, SWRL is needed to overcome the limitations of OWL.

4. OWL/SWRL TAML Contact Classification Ontology

The Semantic Web Rule Language (SWRL) is used to explicitly define rules which are layered on top of an OWL ontology. SWRL rules increase a machine's ability to produce and process the properties which describe an ontology's concepts.

The SWRL rules added to the TAML Contact Classification Ontology overcome some of the limitations of OWL and enhance the overall ontology by adding the capability to prioritize instances of *HostileContact*. Since user-defined datatypes are not supported in the current version of OWL, SWRL is used instead to restrict the range of *course* and *forceCode* within the TAML Contact Classification Ontology. SWRL rules also provide a method for inserting Boolean properties for use as error flags, which are set when certain conditions defined by consistency-checking rules are met. For example, in the TAML Contact Classification Ontology, a Boolean property *hasSemanticError* acts as a flag to signal semantic inconsistencies such as an instance which has a *ThreatCode* value of "FRD" and a contrary *ContactClassification* such as "hostile".

Since SWRL supports FOL constructs and built-in arithmetic operators, rules were written to prioritize instances of *HostileContact* based on their *range* and *speed*. These rules enhance the level of semantic processing that a machine can conduct over a domain by explicitly defining properties and relevant conditions for an instance.

SWRL effectively overcomes some of the limitations encountered in the OWL-only TAML Contact Classification Ontology but software-implementation support for SWRL remains limited. SWRL is still undergoing final review at the W3C and so current SW reasoners have limited support for executing SWRL rules. However, a plug-in to integrate the Jess rule engine into Protégé is currently being developed and is expected to soon provide a method for executing SWRL rules and inserting results back into the knowledge base.

SWRL provides logic to the SW but there are some types of reasoning which are not yet possible such as associative thinking, spatial reasoning, recognition of images and complex decision procedures (Herman, 2004). The SW is a tool for adding context to data and is not intended to meet all requirements of Artificial Intelligence (AI). The SW needs to be scalable, thus simplicity is an important design goal (Alesso, 2005). Nevertheless a comprehensive range of new capabilities are now demonstrated and available, with further capabilities expected.

C. ASSESSMENT OF QUERY LANGUAGES

TAML documents can be queried using the XML Query Language (XQuery) but such queries require the user to have knowledge about the TAML schema. XQuery queries are written using XPath expressions which traverse a document using a directory-like command until the level holding the desired information is reached. The semantics of the query are implicitly coded in the XPath expressions (Mazzocchi, 2004). XQuery relies on the structure of a document to find data, so a query written for TAML documents is not necessarily portable to a document defined by a different schema.

Unlike XQuery, SPARQL (the RDF Query Language) is semantically portable because RDF defines the semantics associated with data explicitly (Mazzocchi, 2004). A SPARQL query is written using the names of properties or resources without needing information about where in the document they are stored. TAML data stored as RDF can be included in any document and still be queried using SPARQL, thus SPARQL queries are portable. The goal of SPARQL is to provide the capability to extract data from various sources using semantic queries (Dodds, 2005).

The SPARQL syntax is similar to database-oriented SQL which decreases the learning time necessary for users who are already familiar with database querying. SPARQL provides similar types of queries including SELECT/WHERE, UNION and CONSTRUCT. More powerful queries based on user-defined constraints can be written once an ontology or semantic model for a domain has been defined using OWL.

D. SUMMARY

The SW is defined by a layered architecture. Each layer or component adds to the capabilities of the layers below. This work explores the capabilities and limitations of the three main components currently defined: RDF, OWL and SWRL. Alone the components provide basic semantics for data in a machine-interpretable format: RDF provides explicit semantics for specific instances of data, OWL provides an explicit model or definitions of a domain, and SWRL enhances OWL by providing FOL rules or statements to the ontology. The layers are combined in order to form a more complete semantic model or knowledge base that includes specific instance data, context and rules for adding data. Once a complete knowledge base is formed, a machine can reason over given data and infer new data as demonstrated by the TAML Contact Classification Ontology.

THIS PAGE INTENTIONALLY LEFT BLANK

X. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The widespread adoption of XML for exchanging information has led to an increased interest in developing XML vocabularies like TAML for use in exchanging tactical information among military command and control systems. The increase in data and decrease in the time available to process it has led to a widespread problem of information overload. The need for machine automation to better support the tasks of gathering, processing and analyzing data from various sources is great. The SW provides a methodology for machine automation which can significantly enhance the information available to warfighters for making decisions. The SW is a standards-based extension of XML, giving it broad applicability. The SW adds explicit machine-interpretable meaning to data and provides context for processing the data through an ontology.

The United States Navy has taken a major step towards realizing a common Global Information Grid (GIG) by adopting XML as a data interchange language. The goal of the GIG is to provide valuable real-time information to warfighters. XML provides a method for standardizing data traffic between systems, but does not provide semantic information about the data to machines. SW languages are themselves XML vocabularies, so they are able to utilize and benefit from the family of XML technologies. The adoption of the SW is a natural extension to XML which allows for the addition of context to data. This approach demonstrably increases the ability for machines to analyze data and return valuable information in ways that currently require extensive human effort.

A main benefit of including context for data in an ontology (as opposed to a hard-coded program) is the adaptability it provides. SW applications are built around an ontology so that when the assumptions or view of the domain change, only the ontology needs to be modified. The applications refer to the ontology to receive information or context about the data they are processing. Thus, changes made in the ontology are reflected in the application, rather than requiring the attempted correction of internally captured logic. Several applications can refer to one ontology. For example, a full TAML

ontology may support other applications besides contact classification. Therefore, storing data about the domain in an ontology as opposed to a hard-coded program adds flexibility to applications.

SW languages and tools are still being developed and refined. However, the SW benefits from the widespread adoption of XML and a growing number of SW early adopters such as Oracle, Sun Systems, Hewlett Packard and Adobe. All of the SW languages are platform-independent, license-free standards that are being developed by a well-known standards organization, the W3C. A variety of other Knowledge Representation (KR) languages such as the Knowledge Interchange Format (KIF) are complete and available for use, but they do not benefit from the support for interoperability provided by XML nor from the support of the W3C.

The SW is a tool for adding context and meaning to data, but the SW is not a substitute for the use of AI-based technologies. The reasoning done using SW applications is limited and does not yet include things such as spatial reasoning, complex decision procedures and associative thinking (Herman, 2004). Nevertheless such capabilities are emerging.

B. RECOMMENDATIONS FOR FUTURE WORK

The following sections describe ways to complete, extend, enhance or improve this thesis work. The TAML Contact Classification Ontology serves as a proof of concept and not a deployable application. However, the ontology can provide the basis for more operable Command, Control, Communications, and Intelligence (C4I) applications using SW technologies.

1. Execute and Test SWRL Rules

This thesis includes a set of rules designed to enforce consistent semantics and to prioritize hostile contacts within the TAML Contact Classification Ontology. The currently incomplete state of SW reasoners prevented the full testing and use of the rules. Once reasoners are able to support SWRL, or once a plug-in is released to integrate a rule

engine into Protégé, then the SWRL rules can enhance the classification capabilities of the ontology.

Future work also needs to include the involvement of a domain expert to write more sophisticated class definitions and rules for classifying contacts. The current rules are simple and are provided as a proof of concept. The dedicated involvement of a domain expert can enhance and complete the semantic model needed to effectively classify contacts. This work can also continue to be designed to map to multiple data schemes used by a variety of C4I systems.

2. TAML Contact Classification Application

The TAML Contact Classification Ontology is designed as a semantic model which can be referenced and used by an application to determine the classification of contacts defined within TAML documents.

Protégé provides an API for applications to access and manipulate the semantics defined within an ontology. The API allows developers to design an application which inputs instance data into an ontology, calls a reasoner and returns the inferred information to the user.

The testing and demonstrations with the current TAML Contact Classification Ontology are done by manually inputting instance information into the knowledge base through the Protégé Graphical User Interface (GUI). This thesis did not explore the performance aspects of reasoners when large amounts of instance data are inserted for reasoning. An application which automatically populates the ontology with *Contact* instances from TAML documents and returns the inferred classifications ought to be developed and used to further study the performance aspects of SW applications.

3. Probabilistic-OWL (PR-OWL)

In the real world, many relationships hold with a certain probability and some facts are associated with an uncertainty (Fukushige, 2004). Current SW technologies do not provide constructs to represent and reason about uncertainty in a precise manner

which limits the development of many types of applications where data can be reasoned about based on accumulation of evidence and probable causation.

Current research being done on a probabilistic OWL is aimed at extending OWL so it can represent uncertainties and probabilities within an ontology. PR-OWL is based on Multi-Entity Bayesian Networks (MEBN) (http://www.pr-owl.org). MEBN is a logic for probabilistic reasoning in an Open World System. MEBN integrates FOL and probability theory to form a coherent foundation for open-world reasoning with random variables. Bayesian logic provides the ability to absorb new facts from the world and incorporate them into an existing fact base (Laskey, 2006).

MEBN logic is a key component for realizing Net-Centric Warfare. The tactical domain is full of ambiguous rules and incomplete data. MEBN provides mathematical techniques that can also help model human decision-making in ambiguous and complex situations (Costa, 2004). MEBN systems provide the ability to represent uncertainty and use recursive reasoning techniques to reason about entities. Combining MEBN ideas with OWL is likely to enable ontology and application developers to represent and reason about uncertainty in SW applications. In one sense probabilistic reasoning adds another important capability to the logic block in the Semantic Web "layer cake" shown in Figure 19. Adding probability and uncertainty concepts into the TAML Contact Classification Ontology is expected to allow for more precise classification of entities.

4. Domain-Specific Reasoners

SW reasoners such as RacerPro and Pellet are built to interpret and reason over any ontology, thus they are domain independent. Often domain-independent tools are not optimized for any one particular application or domain, thus they are not necessarily the most efficient tool to use in a real-time tactical application. Many real-time tactical systems require immediate results, but the results returned by RacerPro and Pellet can take hours or days if the ontology is large. Tactical applications may benefit from a domain-specific reasoner, optimized for the typical types of reasoning which need to be performed during C4I scenarios. The Jess tool in particular may be well suited for such adaptation and optimization.

5. Valued Information at the Right Time (VIRT)

VIRT is a proposed architecture for providing valuable information to the right people at the right time (Hayes-Roth, 2005). The architecture calls for domain and information ontologies to characterize the semantics of the data being exchanged within the system. Domain ontologies specify the data needed by users and information ontologies specify the data from the various information sources. A Domain Translator might then be used to translate data between the domain and information ontologies. The SW offers a promising method for defining domain and information ontologies using OWL. SW languages could also be used within the Domain Translator to provide translation information through OWL mapping constructs and SWRL rules.

6. Autonomous Vehicle Command Language (AVCL)

An important area for future work is the addition of SW capabilities to AVCL (Davis, 2006). The SW can be used to develop explicitly defined properties, relationships and rules for the tasking, control and reporting of autonomous robots and unmanned systems using AVCL. An OWL ontology can be used to explicitly define the concepts and relationships needed for autonomous vehicle tasking and rules might be added to aid machine automation. An AVCL Ontology might then be implemented into the AUV Workbench to provide the capabilities to automate mission planning and analysis.

7. Further TAML Ontology Development and Testing

The TAML Contact Classification Ontology should be submitted to the DoD Metadata Registry as an early exemplar tactical ontology. Further work using real-world applications needs to continue as part of the USW-XML Working Group. In addition to rules for contact classification and correlation, exploratory work needs to examine the feasability of expressing Rules of Engagement (ROEs) using SW technologies as a potential Tactical Decision Aid (TDA).

Upon further development, the TAML Contact Classification Ontology should be deployed in an actual tactical application and tested at sea, preferably during a Trident

Warrior annual exercise. At-sea testing is important for determining the reliability and performance of current SW applications in a real-time tactical situation.

APPENDIX A. RDF/XML SERIALIZATION OF TAML DOCUMENTS

A. DESCRIPTION

XML provides a standard vocabulary and syntax for exchanging documents. However, the semantics of an XML document are not apparent to a machine. RDF is a language that converts semantic information into a machine understandable format. The triple format of RDF allows statements to be written with explicit semantics. An example TAML document was translated by hand into RDF/XML using the XMLSpy editor. The document is an example TAML document which describes the Battle of Trafalgar. The design of the RDF document is documented in Chapter VI of this thesis and a subset of the resulting document is included in this appendix. These documents can be queried with SPARQL which provides a easier and more powerful query language than XQuery. These documents may also be read into an ontology as instance documents, so a reasoner can add more semantic information to the data.

B. TRAFALGAREXAMPLE.RDF

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-RDFSyntax-ns#"
    :rdfs="http://www.w3.org/2000/01/RDFSchema#":taml="urn:us:gov:dod:don:navy:navsea:usw:1.0
    :">

<!--Created by ENS Candace Childers 20Nov2005-->
<!--RDF assertions for Operation T001-->

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/OperationT001">
          <taml:Name>Trafalgar</taml:Name>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMSFLAG">
     <taml:Name>Victory</taml:Name>
     <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFGFLAG"/>
     <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMDFLAG" />
     <!--Track further defined on line 57-->
    <taml:Track rdf:resource="http://usw.xml.wg/Trafalgar.xml/TrackPlatformHMSFLAG"/>
    <taml:Narrative>
        <rdf:Seq>
      <!--Narratives defined line 130-->
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                21T06:10:00"/>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                21T06:13:00"/>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                21T06:22:00"/>
```

```
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T06:46:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T07:05:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T07:20:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T07:35:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T08:40:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T08:13:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T09:00:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T09:36:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T09:41:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T10:10:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-
                                  10-21T10:45:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-
                                  10-21T11:02:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T11:40:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T11:45:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T11:48:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T11:50:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T12:04:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T12:15:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T12:30:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T16:15:00"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                                  21T23:00:00"/>
        </rdf:Seq>
    </taml:Narrative>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFGFLAG">
    <taml:ConfigurationItem>100 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMDFLAG">
    <taml:ConfigurationItem>Capt. T.M. Hardy</taml:ConfigurationItem>
</rdf:Description>

<!--Track defined as a sequence of TimePositions...which are further defined below-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/TrackPlatformHMSFLAG">
    <taml:TimePositions>
        <rdf:Seq>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG001"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG002"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG003"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG004"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG005"/>
```

```
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG006"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG007"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG008"/>
            <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG009"/>
        </rdf:Seq>
    </taml:TimePositions>
</rdf:Description>

<!--TimePositions for TrackPlatformHMSFLAG further defined-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG001">
    <taml:DateTime>1805-10-19T09:30:00</taml:DateTime>
    <taml:Latitude>36.366702</taml:Latitude>
    <taml:Longitude>-7.191296</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG002">
    <taml:DateTime>1805-10-20T06:00:00</taml:DateTime>
    <taml:Latitude>36.991931</taml:Latitude>
    <taml:Longitude>-6.085814</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG003">
    <taml:DateTime>1805-10-20T16:00:00</taml:DateTime>
    <taml:Latitude>36.039084</taml:Latitude>
    <taml:Longitude>-6.458602</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG004">
    <taml:DateTime>1805-10-20T20:30:00</taml:DateTime>
    <taml:Latitude>36.291515</taml:Latitude>
    <taml:Longitude>--6.607113</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG005">
    <taml:DateTime>1805-10-21T04:00:00</taml:DateTime>
    <taml:Latitude>35.984178</taml:Latitude>
    <taml:Longitude>-6.595522</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG006">
    <taml:DateTime>1805-10-21T05:30:00</taml:DateTime>
    <taml:Latitude>36.161005</taml:Latitude>
    <taml:Longitude>-6.520000</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG007">
    <taml:DateTime>1805-10-21T06:00:00</taml:DateTime>
    <taml:Latitude>36.175872</taml:Latitude>
    <taml:Longitude>-6.436138</taml:Longitude>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG008">
    <taml:DateTime>1805-10-21T06:15:00</taml:DateTime>
    <taml:Course>67.5</taml:Course>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/TimePositionPOSHMSFLAG009">
    <taml:DateTime>1805-10-21T06:45:00</taml:DateTime>
    <taml:Course>90</taml:Course>
    <taml:SpeedUnits>knots</taml:SpeedUnits>
    <taml:Speed>3.0</taml:Speed>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T06:10:00">
   <taml:DateTime>1805-10-21T06:10:00</taml:DateTime>
   <taml:Text>To:General / Signal:Form the order of sailing in two columns</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T06:13:00">
   <taml:DateTime>1805-10-21T06:13:00</taml:DateTime>
   <taml:Text>To:General / Signal:Bear up and sail large on course ENE</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T06:22:00">
   <taml:DateTime>1805-10-21T06:22:00</taml:DateTime>
   <taml:Text>To:General / Signal:Prepare for battle</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
   21T06:46:00">
   <taml:DateTime>1805-10-21T06:46:00</taml:DateTime>
   <taml:Text>To:General / Signal:Bear up and sail large on course E</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T07:05:00">
   <taml:DateTime>1805-10-21T07:05:00</taml:DateTime>
   <taml:Text>To:Britannia / Signal:Take station as convenient without regard to the
   established order of sailing</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T07:20:00">
   <taml:DateTime>1805-10-21T07:20:00</taml:DateTime>
   <taml:Text>To:Prince and Dreadnaught / Signal:Take station as convenient without regard
   to the established order of sailing</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T07:35:00">
   <taml:DateTime>1805-10-21T07:35:00</taml:DateTime>
   <taml:Text>To:Euryalus, Naiad, Phoebe, and Sirius / Signal:Captain to come on board
   flagship</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T08:40:00">
   <taml:DateTime>1805-10-21T08:40:00</taml:DateTime>
   <taml:Text>To:Prince / Signal:Bear up and sail large on course steered by
   admiral</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T08:13:00">
   <taml:DateTime>1805-10-21T08:13:00</taml:DateTime>
   <taml:Text>To:General / Signal:</taml:Text>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T09:00:00">
   <taml:DateTime>1805-10-21T09:00:00</taml:DateTime>
   <taml:Text>To:Royal Soverign / Signal:Report if Tonnant cannot close: order other ships
   between</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T09:36:00">
   <taml:DateTime>1805-10-21T09:36:00</taml:DateTime>
   <taml:Text>To:Leviathan / Signal:Take station astern of Temeraire</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T11:02:00">
   <taml:DateTime>1805-10-21T11:02:00</taml:DateTime>
   <taml:Text>To:Defence/ Signal:Make all sail possible with safety to the mast</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
   21T11:50:00">
   <taml:DateTime>1805-10-21T11:50:00</taml:DateTime>
   <taml:Text>To:General/ Signal:Make all sail possible with safety to the mast</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T12:04:00">
   <taml:DateTime>1805-10-21T12:04:00</taml:DateTime>
   <taml:Text>Double volley towards stern of Buecentaure and bow of Redoutable</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T12:15:00">
   <taml:DateTime>1805-10-21T12:15:00</taml:DateTime>
   <taml:Text>To:Temeraire / Signal:Take station astern of Victory</taml:Text>
   <taml:Text>To:General / Signal:Engage the enemy more closely</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T12:30:00">
   <taml:DateTime>1805-10-21T12:30:00</taml:DateTime>
   <taml:Text>To:Africa/ Signal:Make all sail possible with safety to the mast</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T16:15:00">
   <taml:DateTime>1805-10-21T16:15:00</taml:DateTime>
   <taml:Text>To:Africa/ Signal:Come to the wind on starboard tack (end of chase and
   battle)</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMSFLAG1805-10-
                            21T23:00:00">
   <taml:DateTime>1805-10-21T23:00:00</taml:DateTime>
   <taml:Text>To:General/ Signal:Prepare to anchor after close of day</taml:Text>
</rdf:Description>
<!--End of PlatformHMSFLAG descriptions-->
```

```
<!--PlatformHMS001 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS001">
   <taml:Name>Temeraire</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG001" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD001" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG001">
   <taml:ConfigurationItem>98 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD001">
   <taml:ConfigurationItem>Capt. E. Harvey</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS001 descriptions-->

<!--PlatformHMS002 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS002">
   <taml:Name>Neptune</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG002" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD002" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG002">
   <taml:ConfigurationItem>98 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD002">
   <taml:ConfigurationItem>Capt. T. F. Fremantle</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS002 descriptions-->

<!--PlatformHMS003 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS003">
   <taml:Name>Conqueror</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG003" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD003" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG003">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD003">
   <taml:ConfigurationItem>Capt. I. Pellow</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS003 descriptions-->

<!--PlatformHMS004 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS004">
   <taml:Name>Leviathan</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG004" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD004" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG004">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD004">
   <taml:ConfigurationItem>Capt. H. W. Baynton</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS004 descriptions-->
```

```
<!--PlatformHMS005 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS005">
   <taml:Name>Britannia</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG005" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD005" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG005">
   <taml:ConfigurationItem>100 Guns</taml:ConfigurationItem>
</rdf:Description>
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD005">
   <taml:ConfigurationItem>Capt. C. Bullen</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS005 descriptions-->

<!--PlatformHMS006 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS006">
   <taml:Name>Ajax</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG006" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD006" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG006">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD006">
   <taml:ConfigurationItem>Lieut. J. Pilford</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS006 descriptions-->

<!--PlatformHMS007 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS007">
   <taml:Name>Orion</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG007" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD007" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG007">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD007">
   <taml:ConfigurationItem>Capt. E. Codrington</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS007 descriptions-->

<!--PlatformHMS008 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS008">
   <taml:Name>Agamemnon</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG008" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD008" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG008">
   <taml:ConfigurationItem>64 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD008">
   <taml:ConfigurationItem>Capt. Sir E Berry, Bt.</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS008 descriptions-->
```

```
<!--PlatformHMS009 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS009">
   <taml:Name>Minotaur</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG009" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD009" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG009">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD009">
   <taml:ConfigurationItem>Capt. C. J. Mansfield</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS009 descriptions-->

<!--PlatformHMS010 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS010">
   <taml:Name>Spartiate</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG010" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD010" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG010">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD010">
   <taml:ConfigurationItem>Capt. Sir F. Laforey, Bt.</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS010 descriptions-->

<!--PlatformHMS011 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS011">
   <taml:Name>Africa</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG011" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD011" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG011">
   <taml:ConfigurationItem>64 Guns</taml:ConfigurationItem>
</rdf:Description>
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD011">
   <taml:ConfigurationItem>Capt. H. Digby</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS011 descriptions-->

<!--PlatformHMS101 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS101">
   <taml:Name>Royal Soveriegn</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG101" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD101" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG101">
   <taml:ConfigurationItem>100Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD101">
   <taml:ConfigurationItem>Capt. E. Rotherham</taml:ConfigurationItem>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T08:45:00">
    <taml:DateTime>1805-10-21T08:45:00</taml:DateTime>
    <taml:Text>To:Lee Division / Signal:Keep on the larboard line of bearing though on the
    starboard tack</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T08:50:00">
    <taml:DateTime>1805-10-21T08:50:00</taml:DateTime>
    <taml:Text>To:Lee Division / Signal:Form the larbouard line of bearing</taml:Text>
    <taml:Text>To:Lee Division / Signal:Make more sail, leading ship first</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T09:15:00">
    <taml:DateTime>1805-10-21T09:15:00</taml:DateTime>
    <taml:Text>To:Belleisle and Tonnant/ Signal:Interchange places in the line</taml:Text>
    <taml:Text>To:Dreadnaught/ Signal:Make more sail</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T09:20:00">
    <taml:DateTime>1805-10-21T09:20:00</taml:DateTime>
    <taml:Text>To:Belleisle / Signal:Make more sail</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T09:30:00">
    <taml:DateTime>1805-10-21T09:30:00</taml:DateTime>
    <taml:Text>To:Belleisle / Signal:Take station bearing SW from Admiral</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T09:40:00">
    <taml:DateTime>1805-10-21T09:40:00</taml:DateTime>
    <taml:Text>
    :Revenge/ Signal:Take station bearing from Admiral as pointed out by compass signal (not
    logged)
    </taml:Text>
    <taml:Text>To:Revenge/ Signal:Make more sail</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T09:45:00">
    <taml:DateTime>1805-10-21T09:45:00</taml:DateTime>
    <taml:Text>
    :Lee Division / Signal:Take station bearing from Admiral as pointed out [not logged] and
    make more saill
    </taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
    21T11:45:00">
    <taml:DateTime>1805-10-21T11:45:00</taml:DateTime>
    <taml:Text>To:Lee Division / Signal:Make more sail</taml:Text>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS101-1805-10-
   21T12:00:00">
   <taml:DateTime>1805-10-21T12:00:00</taml:DateTime>
   <taml:Text>Double volley towards stern galleries of Santa Ana and bow of
   Fougueux</taml:Text>
</rdf:Description>
<!--End of PlatformHMS101 descriptions-->

<!--PlatformHMS102 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS102">
   <taml:Name>Belleisle</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG102" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD102" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG102">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD102">
   <taml:ConfigurationItem>Capt. W. Hargood</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS102 descriptions-->

<!--PlatformHMS103 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS103">
   <taml:Name>Mars</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG103" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD103" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG103">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD103">
   <taml:ConfigurationItem>Capt. G. Duff</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS103 descriptions-->

<!--PlatformHMS104 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS104">
   <taml:Name>Tonnant</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG104" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD104" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG104">
   <taml:ConfigurationItem>80 Guns</taml:ConfigurationItem>
</rdf:Description>
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD104">
   <taml:ConfigurationItem>Capt. C. Tyler</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS104 descriptions-->

<!--PlatformHMS105 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS105">
   <taml:Name>Bellerophan</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG105" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD105" />
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG105">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD105">
   <taml:ConfigurationItem>Capt. J. Cook</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS105 descriptions-->

<!--PlatformHMS106 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS106">
   <taml:Name>Colossus</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG106" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD106" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG106">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD106">
   <taml:ConfigurationItem>Capt. J. N. Morris</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS106 descriptions-->

<!--PlatformHMS107 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS107">
   <taml:Name>Achille</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG107" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD107" />
   <taml:Narrative>
      <rdf:Seq>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS107-1805-10-
          21T05:50:00"/>
      </rdf:Seq>
   </taml:Narrative>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG107">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD107">
   <taml:ConfigurationItem>Capt. R. King</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeHMS107-1805-10-
   21T05:50:00">
   <taml:DateTime>1805-10-21T05:50:00</taml:DateTime>
   <taml:Text>To:Nelson / Signal:Have discovered a strange fleet</taml:Text>
</rdf:Description>
<!--End of PlatformHMS107 descriptions-->

<!--PlatformHMS108 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS108">
   <taml:Name>Revenge</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG108" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD108" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG108">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD108">
    <taml:ConfigurationItem>Capt. R. Moorsom</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS108 descriptions-->

<!--PlatformHMS109 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS109">
    <taml:Name>Defiance</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG109" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD109" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG109">
    <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD109">
    <taml:ConfigurationItem>Capt. P. Durham</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS109 descriptions-->

<!--PlatformHMS110 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS110">
    <taml:Name>Prince</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG110" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD110" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG110">
    <taml:ConfigurationItem>98 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD110">
    <taml:ConfigurationItem>Capt. R. Grindall</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS110 descriptions-->

<!--PlatformHMS111 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS111">
    <taml:Name>Polyphemus</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG111" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD111" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG111">
    <taml:ConfigurationItem>64 Guns</taml:ConfigurationItem>
</rdf:Description>
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD111">
    <taml:ConfigurationItem>Capt. R. Redmill</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS111 descriptions-->

<!--PlatformHMS112 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS112">
    <taml:Name>Dreadnaught</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG112" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD112" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG112">
    <taml:ConfigurationItem>98 Guns</taml:ConfigurationItem>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD112">
    <taml:ConfigurationItem>Capt. J. Conn</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS112 descriptions-->

<!--PlatformHMS113 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS113">
    <taml:Name>Swiftsure</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG113" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD113" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG113">
    <taml:ConfigurationItem>98 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD113">
    <taml:ConfigurationItem>Capt. W. E. Rutherford</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS113 descriptions-->

<!--PlatformHMS114 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS114">
    <taml:Name>Thunderer</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG114" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD114" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG114">
    <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD114">
    <taml:ConfigurationItem>Lieut. J. Stockham</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS114 descriptions-->

<!--PlatformHMS115 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS115">
    <taml:Name>Defence</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG115" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD115" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG115">
    <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD115">
    <taml:ConfigurationItem>Capt. G. Hope</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS115 descriptions-->

<!--PlatformHMS201 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS201">
    <taml:Name>Euryalus</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG201" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD201" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG201">
    <taml:ConfigurationItem>36 Guns</taml:ConfigurationItem>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD201">
   <taml:ConfigurationItem>Capt. Hon. H. Blackwood</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS201 descriptions-->

<!--PlatformHMS202 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS202">
   <taml:Name>Naiad</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG202" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD202" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG202">
   <taml:ConfigurationItem>38 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD202">
   <taml:ConfigurationItem>Capt. T. Dundas</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS202 descriptions-->

<!--PlatformHMS203 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS203">
   <taml:Name>Phoebe</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG203" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD203" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG203">
   <taml:ConfigurationItem>36 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD203">
   <taml:ConfigurationItem>Capt. Hon. T. Capell</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS203 descriptions-->

<!--PlatformHMS204 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS204">
   <taml:Name>Sirius</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG204" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD204" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG204">
   <taml:ConfigurationItem>36 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD204">
   <taml:ConfigurationItem>Capt. W. Prowse</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS204 descriptions-->

<!--PlatformHMS301 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS301">
   <taml:Name>Canopus</taml:Name>
</rdf:Description>
<!--End of PlatformHMS301 descriptions-->
```

```
<!--PlatformHMS302 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS302">
    <taml:Name>Donegal</taml:Name>
</rdf:Description>
<!--End of PlatformHMS302 descriptions-->

<!--PlatformHMS303 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS303">
    <taml:Name>Queen</taml:Name>
</rdf:Description>
<!--End of PlatformHMS303 descriptions-->

<!--PlatformHMS304 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS304">
    <taml:Name>Spencer</taml:Name>
</rdf:Description>
<!--End of PlatformHMS304 descriptions-->

<!--PlatformHMS305 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS305">
    <taml:Name>Tiger</taml:Name>
</rdf:Description>
<!--End of PlatformHMS305 descriptions-->

<!--PlatformHMS306 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS306">
    <taml:Name>Zealous</taml:Name>
</rdf:Description>
<!--End of PlatformHMS306 descriptions-->

<!--PlatformHMS401 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS401">
    <taml:Name>Caesar</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG401" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD401" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG401">
    <taml:ConfigurationItem>80 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD401">
    <taml:ConfigurationItem>Capt. Sir Richard Strachan</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS401 descriptions-->

<!--PlatformHMS402 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS402">
    <taml:Name>Bellona</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG402" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG402">
    <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS402 descriptions-->

<!--PlatformHMS403 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS403">
    <taml:Name>Courageux</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG403" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD403" />
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG403">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD403">
   <taml:ConfigurationItem>Capt. R. Lee</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS403 descriptions-->

<!--PlatformHMS404 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS404">
   <taml:Name>Hero</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG404" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD404" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG404">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD404">
   <taml:ConfigurationItem>Capt. Hon. A. H. Gardner</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS404 descriptions-->

<!--PlatformHMS405 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformHMS405">
   <taml:Name>Namur</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG405" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD405" />
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG405">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD405">
   <taml:ConfigurationItem>Capt. L. W. Halstead</taml:ConfigurationItem>
</rdf:Description>
<!--End of PlatformHMS405 descriptions-->

<!--PlatformNPL501 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformNPL501">
   <taml:Name>Neptuno</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG501" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD501" />
   <taml:Analysis>
      <rdf:Seq>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP501-1805-10-
          21T23:59:00"/>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP501-1805-10-
          23T23:59:00"/>
      </rdf:Seq>
   </taml:Analysis>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG501">
   <taml:ConfigurationItem>80 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD501">
   <taml:ConfigurationItem>Capt. Don H. C. Valdes</taml:ConfigurationItem>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP501-1805-10-
    21T23:59:00" >
    <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
    <taml:text>Taken by Minotaur and Spartiate</taml:text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP501-1805-10-
    23T23:59:00">
    <taml:DateTime>1805-10-23T23:59:00</taml:DateTime>
    <taml:text>Retaken and escaped</taml:text>
</rdf:Description>
<!--End of PlatformNPL501 descriptions-->

<!--PlatformNPL502 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformNPL502">
    <taml:Name>Scipion (F)</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG502"/>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD502"/>
    <taml:Narrative>
       <rdf:Seq>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeNPL502-1805-10-
          21T08:45:00" />
       </rdf:Seq>
    </taml:Narrative>
    <taml:Analysis>
       <rdf:Seq>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL502-1805-10-
          21T23:59:00"/>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL502-1805-11-
          4T23:59:00"/>
       </rdf:Seq>
    </taml:Analysis>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG502">
    <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD502">
    <taml:ConfigurationItem>Capt. C. Bellanger</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeNPL502-1805-10-
    21T23:59:00">
    <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
    <taml:Text>Opened fire on Victory just after volley by Formidable</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL502-1805-10-
    21T23:59:00">
    <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
    <taml:text>Escaped</taml:text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL502-1805-10-
    23T23:59:00">
    <taml:DateTime>1805-11-4T23:59:00</taml:DateTime>
    <taml:text>Taken by Strachen</taml:text>
</rdf:Description>
<!--End of PlatformNPL502 descriptions-->
```

```
<!--PlatformNPL503 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformNPL503">
   <taml:Name>Intripide (F)</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG503"/>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD503"/>
   <taml:Analysis>
      <rdf:Seq>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP503-1805-10-
         21T23:59:00"/>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP503-1805-10-
         26T23:59:00"/>
      </rdf:Seq>
   </taml:Analysis>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG503">
   <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD503">
   <taml:ConfigurationItem>Capt. L. Infernet</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP503-1805-10-
   21T23:59:00" >
   <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
   <taml:text>Taken by Orion</taml:text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP503-1805-10-
   26T23:59:00"  >
   <taml:DateTime>1805-10-23T23:59:00</taml:DateTime>
   <taml:text>Burnt</taml:text>
</rdf:Description>
<!--End of PlatformNPL503 descriptions-->

<!--PlatformNPL504 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformNPL504">
   <taml:Name>Formidable (F)</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG504"/>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD504"/>
   <taml:Narrative>
      <rdf:Seq>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeNPL504-1805-10-
         21T12:35:00"/>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/NarrativeNPL504-1805-10-
         21T13:30:00"/>
      </rdf:Seq>
   </taml:Narrative>
   <taml:Analysis>
      <rdf:Seq>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL504-1805-10-
         21T23:59:00"/>
         <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL504-1805-11-
         4T23:59:00"/>
      </rdf:Seq>
   </taml:Analysis>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG504">
   <taml:ConfigurationItem>80 Guns</taml:ConfigurationItem>
</rdf:Description>
```

```
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD504">
    <taml:ConfigurationItem>Capt. J. Letellier</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeNPL504-1805-10-
    21T12:35:00">
    <taml:DateTime>1805-10-21T12:35:00</taml:DateTime>
    <taml:Text>Opened fire on Victory</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/NarrativeNPL504-1805-10-
    21T13:30:00">
    <taml:DateTime>1805-10-21T13:30:00</taml:DateTime>
    <taml:Text>To:Bucentaure/ Signal: How is my division to engage the enemy?</taml:Text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL504-1805-10-
    21T23:59:00" >
    <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
    <taml:text>Escaped</taml:text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNPL504-1805-10-
    23T23:59:00">
    <taml:DateTime>1805-11-4T23:59:00</taml:DateTime>
    <taml:text>Taken by Strachen</taml:text>
</rdf:Description>
<!--End of PlatformNPL504 descriptions-->

<!--PlatformNPL505 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformNPL505">
    <taml:Name>Duguay Trouin (F)</taml:Name>
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG505" />
    <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD505" />
    <taml:Analysis>
       <rdf:Seq>
           <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP505-1805-10-
           21T23:59:00"/>
           <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP505-1805-11-
           4T23:59:00"/>
       </rdf:Seq>
    </taml:Analysis>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG505">
    <taml:ConfigurationItem>74 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD505">
    <taml:ConfigurationItem>Capt. C. Touffet</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP505-1805-10-
    21T23:59:00" >
    <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
    <taml:text>Escaped</taml:text>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP50-1805-11-4T23:59:00">
    <taml:DateTime>1805-10-23T23:59:00</taml:DateTime>
    <taml:text>Taken by Strachan</taml:text>
</rdf:Description>
<!--End of PlatformNPL505 descriptions-->
```

```
<!--PlatformNPL506 Description-->
<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/PlatformNPL506">
   <taml:Name>Rayo (S)</taml:Name>
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CFG506" />
   <taml:Configuration rdf:resource="http://usw.xml.wg/Trafalgar.xml/CMD506" />
   <taml:Analysis>
      <rdf:Seq>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP506-1805-10-
          21T23:59:00"/>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP506-1805-10-
          24T23:59:00"/>
          <rdf:li rdf:resource="http://usw.xml.wg/Trafalgar.xml/AnalysisNP506-1805-10-
          26T23:59:00"/>
      </rdf:Seq>
   </taml:Analysis>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCFG506">
   <taml:ConfigurationItem>100 Guns</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar/ConfigurationCMD506">
   <taml:ConfigurationItem>Capt. Don E. Macdonel</taml:ConfigurationItem>
</rdf:Description>

<rdf:Description rdf:about="http://usw.xml.wg/Trafalgar.xml/AnalysisNP506-1805-10-
   21T23:59:00" >
   <taml:DateTime>1805-10-21T23:59:00</taml:DateTime>
   <taml:text>Escaped</taml:text>
</rdf:Description>
```

APPENDIX B. TAML CONTACT CLASSIFICATION ONTOLOGY SOURCE

A. DESCRIPTION

The TAML Contact Classification was designed to demonstrate the advantages and limitations of the Web Ontology Language (OWL). The goal of the ontology is to classify contacts as hostile, friendly, neutral, suspicious, or unknown. The definition of each classification of a *Contact* is explicitly defined using the OWL so a Semantic Web (SW) reasoner is able to automatically classify instances of *Contact*. The ontology was developed using Protégé-OWL beta 3.2. Figure 67 illustrates the properties defined for the Contact class and Figure 68 illustrates the basic structure of the FriendlyContact subclass definition within Protégé. The source OWL file is provided below.

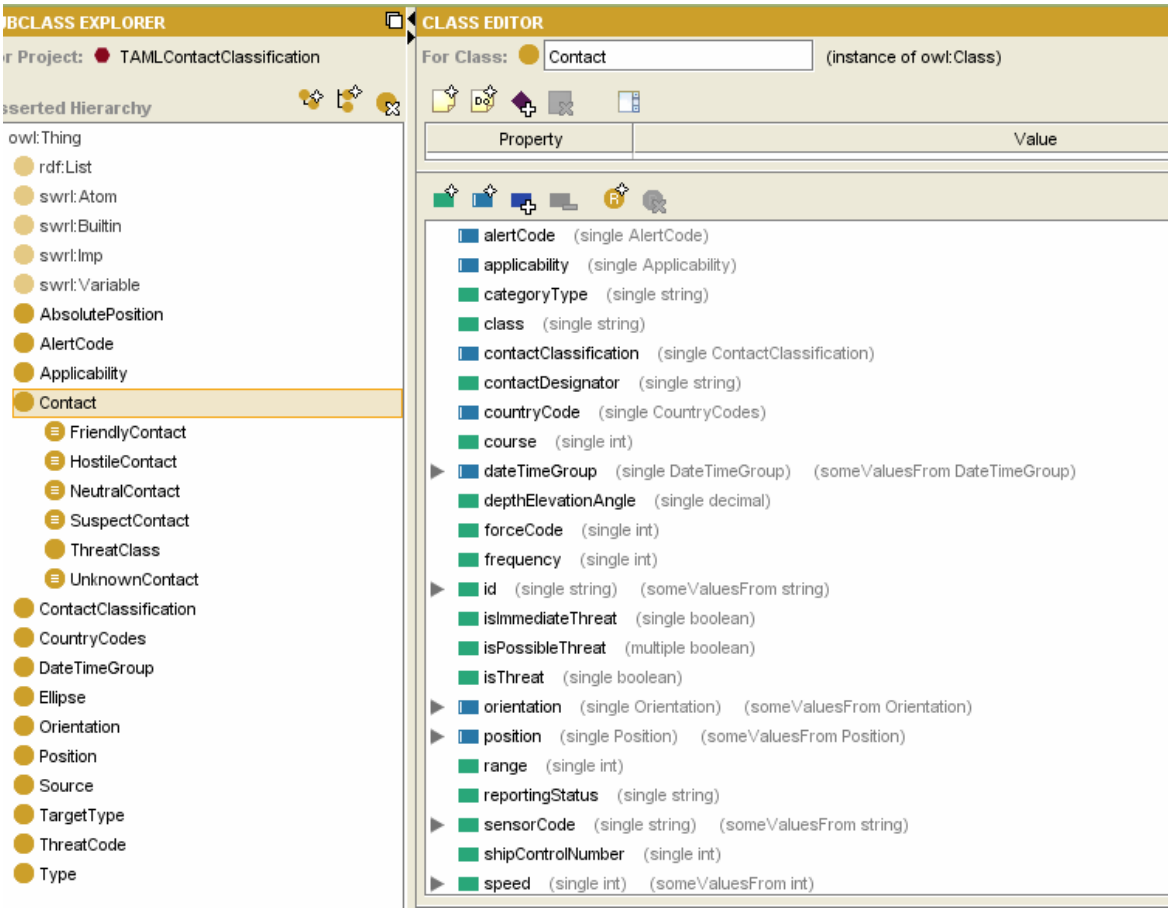


Figure 67. A view of the properties defined for *Contact* in the Protégé GUI.

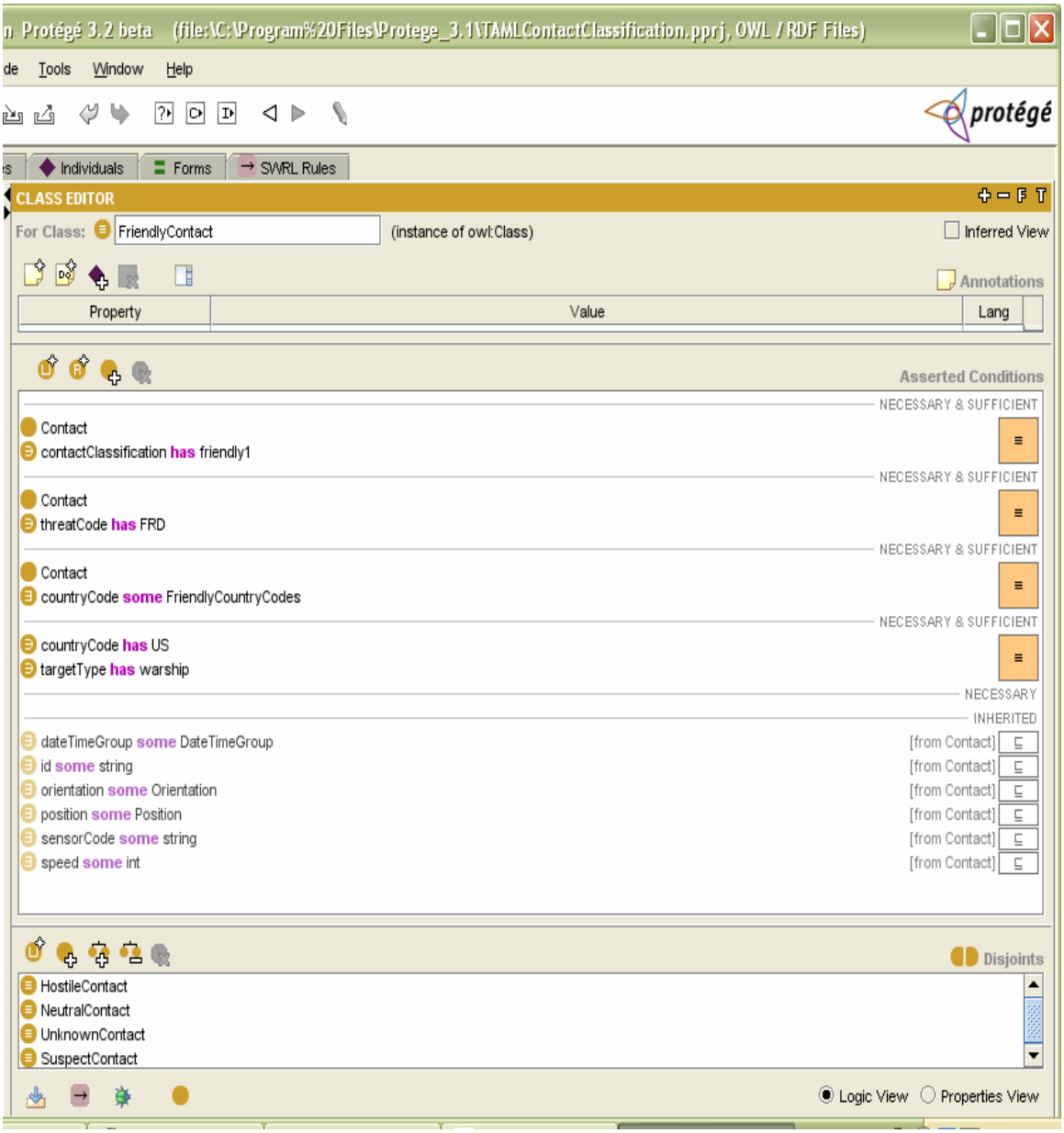


Figure 68. A view of the *FriendlyContact* class definition in the Protégé GUI.

B. TAMLCONTACTCLASSIFICATION.OWL SOURCE

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns="http://usw.xml.wg.TAMLContactClassification.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
 xml:base="http://usw.xml.wg.TAMLContactClassification.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="SuspiciousContact">
   <owl:equivalentClass>
     <owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
         <owl:Restriction>
           <owl:onProperty>
             <owl:FunctionalProperty rdf:ID="countryCode"/>
           </owl:onProperty>
           <owl:allValuesFrom>
             <owl:Class rdf:ID="HostileCountryCode"/>
           </owl:allValuesFrom>
         </owl:Restriction>
         <owl:Restriction>
           <owl:someValuesFrom>
             <owl:Class rdf:about="#HostileCountryCode"/>
           </owl:someValuesFrom>
           <owl:onProperty>
             <owl:FunctionalProperty rdf:about="#countryCode"/>
           </owl:onProperty>
         </owl:Restriction>
         <owl:Restriction>
           <owl:onProperty>
             <owl:FunctionalProperty rdf:ID="targetType"/>
           </owl:onProperty>
           <owl:hasValue>
             <TargetType rdf:ID="civilianAircraft"/>
           </owl:hasValue>
         </owl:Restriction>
       </owl:intersectionOf>
     </owl:Class>
   </owl:equivalentClass>
   <owl:disjointWith>
     <owl:Class rdf:ID="FriendlyContact"/>
   </owl:disjointWith>
   <owl:disjointWith>
     <owl:Class rdf:ID="UnknownContact"/>
   </owl:disjointWith>
   <owl:disjointWith>
     <owl:Class rdf:ID="NeutralContact"/>
   </owl:disjointWith>
   <owl:disjointWith>
     <owl:Class rdf:ID="HostileContact"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
     <owl:Class rdf:ID="Contact"/>
   </rdfs:subClassOf>
   <owl:equivalentClass>
     <owl:Restriction>
       <owl:hasValue>
         <ThreatCode rdf:ID="SUS"/>
       </owl:hasValue>
```

```
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="threatCode"/>
      </owl:onProperty>
    </owl:Restriction>
  </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#HostileCountryCode"/>
          </owl:allValuesFrom>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
          <owl:someValuesFrom>
            <owl:Class rdf:about="#HostileCountryCode"/>
          </owl:someValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:hasValue>
            <TargetType rdf:ID="merchant"/>
          </owl:hasValue>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#targetType"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="#Contact">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="sensorCode"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="id"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#FriendlyContact"/>
        <owl:Class rdf:about="#HostileContact"/>
        <owl:Class rdf:about="#NeutralContact"/>
        <owl:Class rdf:about="#SuspiciousContact"/>
        <owl:Class rdf:about="#UnknownContact"/>
```

```
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="course"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="speed"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >&lt;p style="margin-top: 0">
    The class of all contacts. Required property fields are id,
    dateTimeGroup, orientation, position, sensorCode, and speed. All other
    properties are optional.
  &lt;/p></rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="position"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="dateTimeGroup"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="orientation"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#FriendlyContact">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The class of all contacts that are known to be friendly.</rdfs:comment>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:hasValue>
        <ThreatCode rdf:ID="FRD"/>
```

```
    </owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#threatCode"/>
    </owl:onProperty>
  </owl:Restriction>
</owl:equivalentClass>
<owl:equivalentClass>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="contactClassification"/>
    </owl:onProperty>
    <owl:hasValue>
      <ContactClassification rdf:ID="friendly"/>
    </owl:hasValue>
  </owl:Restriction>
</owl:equivalentClass>
<owl:equivalentClass>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#countryCode"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="FriendlyCountryCode"/>
        </owl:someValuesFrom>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#countryCode"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#FriendlyCountryCode"/>
        </owl:allValuesFrom>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#targetType"/>
        </owl:onProperty>
        <owl:hasValue>
          <TargetType rdf:ID="warship"/>
        </owl:hasValue>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
<owl:equivalentClass>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:hasValue>
          <TargetType rdf:ID="submarine"/>
        </owl:hasValue>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#targetType"/>
        </owl:onProperty>
      </owl:Restriction>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#FriendlyCountryCode"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#countryCode"/>
```

```
          </owl:onProperty>
        </owl:Restriction>
        <owl:Restriction>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#FriendlyCountryCode"/>
          </owl:allValuesFrom>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith>
    <owl:Class rdf:about="#HostileContact"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#NeutralContact"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#Contact"/>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:someValuesFrom>
            <owl:Class rdf:about="#FriendlyCountryCode"/>
          </owl:someValuesFrom>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Restriction>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#FriendlyCountryCode"/>
          </owl:allValuesFrom>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#targetType"/>
          </owl:onProperty>
          <owl:hasValue>
            <TargetType rdf:ID="militaryAircraft"/>
          </owl:hasValue>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith rdf:resource="#SuspiciousContact"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#UnknownContact"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Applicability"/>
<owl:Class rdf:about="#NeutralContact">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:complementOf>
            <owl:Restriction>
```

```
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
            <owl:someValuesFrom>
              <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                  <owl:Class rdf:about="#HostileContact"/>
                  <owl:Class rdf:ID="SuspiciousCountryCode"/>
                </owl:unionOf>
              </owl:Class>
            </owl:someValuesFrom>
          </owl:Restriction>
        </owl:complementOf>
      </owl:Class>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#targetType"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#civilianAircraft"/>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
<owl:equivalentClass>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class>
        <owl:complementOf>
          <owl:Restriction>
            <owl:someValuesFrom>
              <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                  <owl:Class rdf:about="#HostileContact"/>
                  <owl:Class rdf:about="#SuspiciousCountryCode"/>
                </owl:unionOf>
              </owl:Class>
            </owl:someValuesFrom>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:complementOf>
      </owl:Class>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#targetType"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#merchant"/>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The class of all contacts that are known to be neutral.</rdfs:comment>
<owl:disjointWith rdf:resource="#SuspiciousContact"/>
<owl:disjointWith>
  <owl:Class rdf:about="#HostileContact"/>
</owl:disjointWith>
<owl:disjointWith>
  <owl:Class rdf:about="#UnknownContact"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="#FriendlyContact"/>
<owl:equivalentClass>
```

```
      <owl:Restriction>
        <owl:hasValue>
          <ThreatCode rdf:ID="NEU"/>
        </owl:hasValue>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#threatCode"/>
        </owl:onProperty>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Contact"/>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:complementOf>
              <owl:Restriction>
                <owl:someValuesFrom>
                  <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                      <owl:Class rdf:about="#HostileContact"/>
                      <owl:Class rdf:about="#SuspiciousCountryCode"/>
                    </owl:unionOf>
                  </owl:Class>
                </owl:someValuesFrom>
                <owl:onProperty>
                  <owl:FunctionalProperty rdf:about="#countryCode"/>
                </owl:onProperty>
              </owl:Restriction>
            </owl:complementOf>
          </owl:Class>
          <owl:Restriction>
            <owl:hasValue>
              <TargetType rdf:ID="smallboat"/>
            </owl:hasValue>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#targetType"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:hasValue>
          <ContactClassification rdf:ID="neutral"/>
        </owl:hasValue>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#contactClassification"/>
        </owl:onProperty>
      </owl:Restriction>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="#HostileCountryCode">
    <owl:disjointWith>
      <owl:Class rdf:about="#FriendlyCountryCode"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="NeutralCountryCode"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="CountryCode"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
```

```
    <owl:Class rdf:about="#SuspiciousCountryCode"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#FriendlyCountryCode">
  <owl:disjointWith rdf:resource="#HostileCountryCode"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CountryCode"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#NeutralCountryCode"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#SuspiciousCountryCode"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="ThreatCode"/>
<owl:Class rdf:ID="AlertCode"/>
<owl:Class rdf:about="#UnknownContact">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:hasValue>
        <ContactClassification rdf:ID="unknown"/>
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#contactClassification"/>
      </owl:onProperty>
    </owl:Restriction>
  </owl:equivalentClass>
  <owl:disjointWith rdf:resource="#NeutralContact"/>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#threatCode"/>
      </owl:onProperty>
      <owl:hasValue>
        <ThreatCode rdf:ID="UNK"/>
      </owl:hasValue>
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Contact"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#HostileContact"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#FriendlyContact"/>
  <owl:disjointWith rdf:resource="#SuspiciousContact"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The class of contacts where the classification is unknown.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Ellipse"/>
<owl:Class rdf:about="#HostileContact">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#targetType"/>
          </owl:onProperty>
          <owl:hasValue rdf:resource="#smallboat"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
```

```
        <owl:someValuesFrom rdf:resource="#HostileCountryCode"/>
      </owl:Restriction>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#HostileCountryCode"/>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#countryCode"/>
        </owl:onProperty>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
<owl:equivalentClass>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#contactClassification"/>
    </owl:onProperty>
    <owl:hasValue>
      <ContactClassification rdf:ID="hostile"/>
    </owl:hasValue>
  </owl:Restriction>
</owl:equivalentClass>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The class of all contacts that are known to be hostile.</rdfs:comment>
<owl:equivalentClass>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#targetType"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#warship"/>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#countryCode"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#HostileCountryCode"/>
      </owl:Restriction>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#HostileCountryCode"/>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#countryCode"/>
        </owl:onProperty>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
<owl:disjointWith rdf:resource="#UnknownContact"/>
<owl:equivalentClass>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#threatCode"/>
    </owl:onProperty>
    <owl:hasValue>
      <ThreatCode rdf:ID="HOS"/>
    </owl:hasValue>
  </owl:Restriction>
</owl:equivalentClass>
<owl:disjointWith rdf:resource="#FriendlyContact"/>
<owl:equivalentClass>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
```

```
            <owl:someValuesFrom rdf:resource="#HostileCountryCode"/>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#targetType"/>
            </owl:onProperty>
            <owl:hasValue rdf:resource="#submarine"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
            <owl:allValuesFrom rdf:resource="#HostileCountryCode"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#NeutralContact"/>
    <owl:disjointWith rdf:resource="#SuspiciousContact"/>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
            <owl:someValuesFrom rdf:resource="#HostileCountryCode"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#targetType"/>
            </owl:onProperty>
            <owl:hasValue rdf:resource="#militaryAircraft"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:allValuesFrom rdf:resource="#HostileCountryCode"/>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Contact"/>
  </owl:Class>
  <owl:Class rdf:ID="Position"/>
  <owl:Class rdf:ID="LanguageType"/>
  <owl:Class rdf:ID="ContactClassification"/>
  <owl:Class rdf:about="#NeutralCountryCode">
    <owl:disjointWith rdf:resource="#FriendlyCountryCode"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#SuspiciousCountryCode"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CountryCode"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#HostileCountryCode"/>
  </owl:Class>
  <owl:Class rdf:about="#CountryCode">
    <owl:equivalentClass>
```

```
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#FriendlyCountryCode"/>
          <owl:Class rdf:about="#HostileCountryCode"/>
          <owl:Class rdf:about="#NeutralCountryCode"/>
          <owl:Class rdf:about="#SuspiciousCountryCode"/>
        </owl:unionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="#SuspiciousCountryCode">
    <rdfs:subClassOf rdf:resource="#CountryCode"/>
    <owl:disjointWith rdf:resource="#HostileCountryCode"/>
    <owl:disjointWith rdf:resource="#FriendlyCountryCode"/>
    <owl:disjointWith rdf:resource="#NeutralCountryCode"/>
  </owl:Class>
  <owl:Class rdf:ID="Orientation"/>
  <owl:Class rdf:ID="TargetType"/>
  <owl:Class rdf:ID="AbsolutePosition"/>
  <owl:Class rdf:ID="DateTimeGroup"/>
  <owl:Class rdf:ID="Source"/>
  <owl:Class rdf:ID="ThreatContact">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The class of all hostile contacts that are known to be immediate
threats.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#HostileContact"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="applicability">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="#Applicability"/>
    <rdfs:domain rdf:resource="#Contact"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#contactClassification">
    <rdfs:range rdf:resource="#ContactClassification"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Contact"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="dateTime">
    <rdfs:domain rdf:resource="#DateTimeGroup"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="origin">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Ellipse"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="latitude">
    <rdfs:domain rdf:resource="#AbsolutePosition"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="reportingStatus">
    <rdfs:domain rdf:resource="#Contact"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="contactDesignator">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Contact"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="roll">
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Orientation"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="pitch">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Orientation"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="frequency">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Contact"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="axisOrientationAngle">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Ellipse"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="longitude">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#AbsolutePosition"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#course">
    <rdfs:domain rdf:resource="#Contact"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="shipControlNumber">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Contact"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="precision">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Position"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#speed">
    <rdfs:domain rdf:resource="#Contact"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="alertCode">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#AlertCode"/>
    <rdfs:domain rdf:resource="#Contact"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="unitIdentificationCode">
    <rdfs:domain rdf:resource="#Contact"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="forceCode">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Contact"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="range">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Contact"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
```

```
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="heading">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Orientation"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="ellipse">
  <rdfs:domain rdf:resource="#Position"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Ellipse"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="yaw">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Orientation"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="depthElevationAngle">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#threatCode">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="#ThreatCode"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="semiMajorAxis">
  <rdfs:domain rdf:resource="#Ellipse"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="categoryType">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#id">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="source">
  <rdfs:domain rdf:resource="#Position"/>
  <rdfs:range rdf:resource="#Source"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="trackType">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="absolutePosition">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Position"/>
  <rdfs:range rdf:resource="#AbsolutePosition"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#sensorCode">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#dateTimeGroup">
  <rdfs:range rdf:resource="#DateTimeGroup"/>
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#targetType">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#TargetType"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#countryCode">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#CountryCode"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#orientation">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="#Orientation"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="languageID">
  <rdfs:domain rdf:resource="#LanguageType"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="semiMinorAxis">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Ellipse"/>
</owl:FunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#position">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="#Position"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>
<Source rdf:ID="Source_9"/>
<NeutralCountryCode rdf:ID="ES">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Spain</rdfs:comment>
</NeutralCountryCode>
<FriendlyCountryCode rdf:ID="US">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >United States</rdfs:comment>
</FriendlyCountryCode>
<NeutralCountryCode rdf:ID="CH">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Switzerland</rdfs:comment>
</NeutralCountryCode>
<FriendlyCountryCode rdf:ID="GB">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Great Britain</rdfs:comment>
</FriendlyCountryCode>
<SuspiciousCountryCode rdf:ID="KR">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >North Korea</rdfs:comment>
</SuspiciousCountryCode>
<SuspiciousCountryCode rdf:ID="KH">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Cambodia</rdfs:comment>
</SuspiciousCountryCode>
<Source rdf:ID="RLGN"/>
<AlertCode rdf:ID="SUSP"/>
<FriendlyCountryCode rdf:ID="CA">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >Canada</rdfs:comment>
</FriendlyCountryCode>
<Orientation rdf:ID="contactOrientation">
  <pitch rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >5</pitch>
  <heading rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >210</heading>
  <roll rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >5</roll>
  <yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5</yaw>
</Orientation>
<AlertCode rdf:ID="NSP"/>
<SuspiciousCountryCode rdf:ID="EG">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Egypt</rdfs:comment>
</SuspiciousCountryCode>
<ThreatCode rdf:ID="FAKER"/>
<HostileCountryCode rdf:ID="IQ">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Iraq</rdfs:comment>
</HostileCountryCode>
<ThreatCode rdf:ID="ThreatCode_10"/>
<HostileCountryCode rdf:ID="IR">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Iran</rdfs:comment>
</HostileCountryCode>
<Applicability rdf:ID="simulated"/>
<FriendlyCountryCode rdf:ID="FR">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >France</rdfs:comment>
</FriendlyCountryCode>
<FriendlyCountryCode rdf:ID="IL">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Israel</rdfs:comment>
</FriendlyCountryCode>
<ThreatCode rdf:ID="PND"/>
<Applicability rdf:ID="real_world"/>
<Source rdf:ID="SINS"/>
<Position rdf:ID="Position_7">
  <absolutePosition>
    <AbsolutePosition rdf:ID="AbsolutePosition_8">
      <longitude rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >12</longitude>
      <latitude rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >36</latitude>
    </AbsolutePosition>
  </absolutePosition>
  <source>
    <Source rdf:ID="GPS"/>
  </source>
</Position>
<SuspiciousCountryCode rdf:ID="SO">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Somalia</rdfs:comment>
</SuspiciousCountryCode>
<AlertCode rdf:ID="TGT"/>
<Contact rdf:ID="Contact_1">
  <position rdf:resource="#Position_7"/>
  <dateTimeGroup>
    <DateTimeGroup rdf:ID="DateTimeGroup_5">
      <dateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
      >2006-04-01T00:00:00</dateTime>
    </DateTimeGroup>
```

```
    </dateTimeGroup>
    <targetType rdf:resource="#submarine"/>
    <countryCode rdf:resource="#CA"/>
    <orientation rdf:resource="#contactOrientation"/>
  </Contact>
  <FriendlyCountryCode rdf:ID="DE">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Germany</rdfs:comment>
  </FriendlyCountryCode>
  <NeutralCountryCode rdf:ID="BE">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Belgium</rdfs:comment>
  </NeutralCountryCode>
  <ThreatCode rdf:ID="AFD"/>
  <HostileCountryCode rdf:ID="AF">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Afghanistan</rdfs:comment>
  </HostileCountryCode>
  <Applicability rdf:ID="live_training"/>
  <SuspiciousCountryCode rdf:ID="CN">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >China</rdfs:comment>
  </SuspiciousCountryCode>
  <ThreatCode rdf:ID="JOKER"/>
  <AlertCode rdf:ID="HIT"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plug-in 2.2, Build 304)
http://protege.stanford.edu -->
```

APPENDIX C. TAML OWL / SWRL CONTACT CLASSIFICATION ONTOLOGY SOURCE

A. DESCRIPTION

The OWL language is limited to expressing basic definitions of classes by applying quantifiers, cardinality restrictions, and has-value restrictions. The Semantic Web Rule Language (SWRL) is used to write Horn-like rules expressed in terms of OWL concepts (O'Conner, 2005). Rules were added to the TAML Contact Classification Ontology in order to illustrate the advantages and limitations of SWRL. These rules enhance the classification ontology by adding range restrictions to datatype properties, defining range restrictions on datatype properties and by further prioritizing hostile contacts. Although SWRL adds more explicitly defined meaning to the classification ontology, current reasoners do not support the evaluation of these rules. SWRL support will be available in the future. The SWRL rules were added to the TAML Contact Classification Ontology using the SWRL editor plug-in for Protégé-OWL 3.1. Figure 69 provides a screenshot of the rules from the Protégé SWRLTab editor and the source file is contained in Section B.

Figure 69. A screenshot from the Protégé SWRLTab showing the SWRL rules defined for the TAML Contact Classification Ontology.

B. TAMLCONTACTCLASSIFICATIONSWRL.OWL SOURCE

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
 xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about="">
   <owl:imports rdf:resource="http://www.daml.org/rules/proposal/swrlb.owl"/>
   <owl:imports rdf:resource="http://www.daml.org/rules/proposal/swrl.owl"/>
 </owl:Ontology>
 <owl:Class rdf:ID="SuspiciousCountryCodes">
   <owl:disjointWith>
     <owl:Class rdf:ID="FriendlyCountryCodes"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
     <owl:Class rdf:ID="CountryCodes"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
     <owl:Class rdf:ID="NeutralCountryCodes"/>
   </owl:disjointWith>
   <owl:disjointWith>
     <owl:Class rdf:ID="HostileCountryCodes"/>
   </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="SuspectContact">
   <owl:equivalentClass>
     <owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
         <owl:Restriction>
           <owl:onProperty>
             <owl:ObjectProperty rdf:ID="targetType"/>
           </owl:onProperty>
           <owl:hasValue>
             <TargetType rdf:ID="smallboat"/>
           </owl:hasValue>
         </owl:Restriction>
         <owl:Restriction>
           <owl:onProperty>
             <owl:ObjectProperty rdf:ID="threatCode"/>
           </owl:onProperty>
           <owl:hasValue>
             <ThreatCode rdf:ID="UNK"/>
           </owl:hasValue>
         </owl:Restriction>
       </owl:intersectionOf>
     </owl:Class>
   </owl:equivalentClass>
   <owl:equivalentClass>
     <owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
         <owl:Restriction>
           <owl:hasValue>
             <ThreatCode rdf:ID="SUS"/>
           </owl:hasValue>
           <owl:onProperty>
```

```
            <owl:ObjectProperty rdf:about="#threatCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Class rdf:ID="Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:ID="countryCode"/>
          </owl:onProperty>
          <owl:hasValue>
            <HostileCountryCodes rdf:ID="IR">
              <rdfs:comment
     rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Iran
</rdfs:comment>
            </HostileCountryCodes>
          </owl:hasValue>
        </owl:Restriction>
        <owl:Class rdf:about="#Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:hasValue rdf:resource="#IR"/>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Class rdf:about="#Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith>
    <owl:Class rdf:ID="FriendlyContact"/>
  </owl:disjointWith>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:hasValue>
            <TargetType rdf:ID="warship"/>
          </owl:hasValue>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#targetType"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Restriction>
          <owl:hasValue rdf:resource="#UNK"/>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#threatCode"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
```

```
 </owl:Class>
 <owl:Class rdf:ID="AlertCode"/>
 <owl:Class rdf:ID="AbsolutePosition"/>
 <owl:Class rdf:ID="Ellipse"/>
 <owl:Class rdf:ID="Position"/>
 <owl:Class rdf:about="#Contact">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:FunctionalProperty rdf:ID="dateTimeGroup"/>
       </owl:onProperty>
       <owl:someValuesFrom>
         <owl:Class rdf:ID="DateTimeGroup"/>
       </owl:someValuesFrom>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
       <owl:onProperty>
         <owl:FunctionalProperty rdf:ID="speed"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:FunctionalProperty rdf:ID="sensorCode"/>
       </owl:onProperty>
       <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
       <owl:onProperty>
         <owl:DatatypeProperty rdf:ID="id"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="position"/>
       </owl:onProperty>
       <owl:someValuesFrom rdf:resource="#Position"/>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="orientation"/>
       </owl:onProperty>
       <owl:someValuesFrom>
         <owl:Class rdf:ID="Orientation"/>
       </owl:someValuesFrom>
     </owl:Restriction>
   </rdfs:subClassOf>
 </owl:Class>
```

```
 <owl:Class rdf:ID="Applicability"/>
 <owl:Class rdf:ID="HostileContact">
   <owl:equivalentClass>
     <owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
         <owl:Restriction>
           <owl:onProperty>
             <owl:ObjectProperty rdf:about="#threatCode"/>
           </owl:onProperty>
           <owl:hasValue>
             <ThreatCode rdf:ID="HOS"/>
           </owl:hasValue>
         </owl:Restriction>
         <owl:Class rdf:about="#Contact"/>
       </owl:intersectionOf>
     </owl:Class>
   </owl:equivalentClass>
   <owl:equivalentClass>
     <owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#Contact"/>
         <owl:Restriction>
           <owl:hasValue rdf:resource="#warship"/>
           <owl:onProperty>
             <owl:ObjectProperty rdf:about="#targetType"/>
           </owl:onProperty>
         </owl:Restriction>
         <owl:Restriction>
           <owl:onProperty>
             <owl:FunctionalProperty rdf:about="#countryCode"/>
           </owl:onProperty>
           <owl:hasValue>
             <HostileCountryCodes rdf:ID="IQ">
               <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
               >Iraq
</rdfs:comment>
             </HostileCountryCodes>
           </owl:hasValue>
         </owl:Restriction>
       </owl:intersectionOf>
     </owl:Class>
   </owl:equivalentClass>
   <owl:disjointWith>
     <owl:Class rdf:about="#FriendlyContact"/>
   </owl:disjointWith>
   <owl:disjointWith>
     <owl:Class rdf:ID="UnknownContact"/>
   </owl:disjointWith>
   <owl:disjointWith>
     <owl:Class rdf:ID="NeutralContact"/>
   </owl:disjointWith>
   <owl:equivalentClass>
     <owl:Restriction>
       <owl:hasValue>
         <ContactClassification rdf:ID="hostile1"/>
       </owl:hasValue>
       <owl:onProperty>
         <owl:FunctionalProperty rdf:ID="contactClassification"/>
       </owl:onProperty>
     </owl:Restriction>
   </owl:equivalentClass>
 </owl:Class>
```

```
<owl:Class rdf:ID="ContactClassification"/>
<owl:Class rdf:ID="TargetType"/>
<owl:Class rdf:ID="ThreatCode"/>
<owl:Class rdf:about="#FriendlyCountryCodes">
  <rdfs:subClassOf rdf:resource="#CountryCodes"/>
  <owl:disjointWith rdf:resource="#SuspiciousCountryCodes"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#NeutralCountryCodes"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#HostileCountryCodes"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#UnknownContact">
  <owl:disjointWith rdf:resource="#HostileContact"/>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:hasValue rdf:resource="#UNK"/>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#threatCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Class rdf:about="#Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith>
    <owl:Class rdf:about="#NeutralContact"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#FriendlyContact"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="ThreatClass">
  <rdfs:subClassOf rdf:resource="#Contact"/>
</owl:Class>
<owl:Class rdf:about="#NeutralContact">
  <owl:disjointWith rdf:resource="#HostileContact"/>
  <owl:disjointWith rdf:resource="#UnknownContact"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#FriendlyContact"/>
  </owl:disjointWith>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:hasValue>
            <ThreatCode rdf:ID="NEU"/>
          </owl:hasValue>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#threatCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Class rdf:about="#Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
```

```
            <owl:hasValue>
              <TargetType rdf:ID="merchant"/>
            </owl:hasValue>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#targetType"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#threatCode"/>
            </owl:onProperty>
            <owl:hasValue rdf:resource="#NEU"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:hasValue rdf:resource="#merchant"/>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#targetType"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Restriction>
            <owl:hasValue>
              <FriendlyCountryCodes rdf:ID="GB">
                <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Great Britain
</rdfs:comment>
              </FriendlyCountryCodes>
            </owl:hasValue>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#countryCode"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:hasValue>
              <ContactClassification rdf:ID="neutral1"/>
            </owl:hasValue>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#contactClassification"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Class rdf:about="#Contact"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="#NeutralCountryCodes">
    <owl:disjointWith rdf:resource="#SuspiciousCountryCodes"/>
    <owl:disjointWith rdf:resource="#FriendlyCountryCodes"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#HostileCountryCodes"/>
    </owl:disjointWith>
```

```
  <rdfs:subClassOf rdf:resource="#CountryCodes"/>
</owl:Class>
<owl:Class rdf:about="#HostileCountryCodes">
  <rdfs:subClassOf rdf:resource="#CountryCodes"/>
  <owl:disjointWith rdf:resource="#SuspiciousCountryCodes"/>
  <owl:disjointWith rdf:resource="#NeutralCountryCodes"/>
  <owl:disjointWith rdf:resource="#FriendlyCountryCodes"/>
</owl:Class>
<owl:Class rdf:ID="Source"/>
<owl:Class rdf:ID="Type"/>
<owl:Class rdf:about="#FriendlyContact">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Contact"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
          <owl:someValuesFrom rdf:resource="#FriendlyCountryCodes"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#targetType"/>
          </owl:onProperty>
          <owl:hasValue rdf:resource="#warship"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:hasValue>
            <FriendlyCountryCodes rdf:ID="US"/>
          </owl:hasValue>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#countryCode"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:hasValue>
            <ContactClassification rdf:ID="friendly1"/>
          </owl:hasValue>
          <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#contactClassification"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Class rdf:about="#Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith rdf:resource="#HostileContact"/>
  <owl:disjointWith rdf:resource="#NeutralContact"/>
  <owl:equivalentClass>
    <owl:Class>
```

```
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:hasValue>
            <ThreatCode rdf:ID="FRD"/>
          </owl:hasValue>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#threatCode"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Class rdf:about="#Contact"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <owl:disjointWith rdf:resource="#UnknownContact"/>
  <owl:disjointWith rdf:resource="#SuspectContact"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#orientation">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="#Orientation"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#targetType">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="#TargetType"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#position">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="#Position"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ellipse">
  <rdfs:range rdf:resource="#Ellipse"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Position"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#threatCode">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="#ThreatCode"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="isThreat">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasSemanticError">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="depthElevationAngle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="contactDesignator">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="frequency">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
```

```
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isPossibleThreat">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="class">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="trackType">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="origin">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdfs:domain rdf:resource="#Ellipse"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="semiMajorAxis">
  <rdfs:domain rdf:resource="#Ellipse"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="latitude">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AbsolutePosition"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="pitch">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Orientation"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="forceCode">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="semiMinorAxis">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdfs:domain rdf:resource="#Ellipse"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#id">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="rangeViolationError">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isImmediateThreat">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
```

```
  <rdfs:domain rdf:resource="#Contact"/>
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:about="#dateTimeGroup">
  <rdfs:range rdf:resource="#DateTimeGroup"/>
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="alertCode">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#AlertCode"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#countryCode">
  <rdfs:range rdf:resource="#CountryCodes"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="range">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="precision">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="unitIdentificationCode">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#speed">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="type">
  <rdfs:range rdf:resource="#Type"/>
  <rdfs:domain rdf:resource="#Position"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#contactClassification">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="#ContactClassification"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="categoryType">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="source">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Source"/>
  <rdfs:domain rdf:resource="#Position"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="shipControlNumber">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="longitude">
  <rdfs:domain rdf:resource="#AbsolutePosition"/>
```

```
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="roll">
  <rdfs:domain rdf:resource="#Orientation"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="reportingStatus">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="heading">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdfs:domain rdf:resource="#Orientation"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="dateTime">
  <rdfs:domain rdf:resource="#DateTimeGroup"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="axisOrientationAngle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Ellipse"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="precis">
  <rdfs:domain rdf:resource="#Position"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="yaw">
  <rdfs:domain rdf:resource="#Orientation"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="absolutePosition">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#AbsolutePosition"/>
  <rdfs:domain rdf:resource="#Position"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="languageLocaleID">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#token"/>
  <rdfs:domain rdf:resource="#Type"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#sensorCode">
  <rdfs:domain rdf:resource="#Contact"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="languageID">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Type"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#language"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="applicability">
  <rdfs:range rdf:resource="#Applicability"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Contact"/>
</owl:FunctionalProperty>
```

```
 <AlertCode rdf:ID="TGT"/>
 <swrl:Variable rdf:ID="speedx"/>
 <SuspiciousCountryCodes rdf:ID="EG">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Egypt
</rdfs:comment>
 </SuspiciousCountryCodes>
 <swrl:AtomList/>
 <swrl:Imp rdf:ID="SemCheckFour">
   <swrl:body>
     <swrl:AtomList>
       <rdf:rest>
         <swrl:AtomList>
           <rdf:first>
             <swrl:DatavaluedPropertyAtom>
               <swrl:propertyPredicate rdf:resource="#threatCode"/>
               <swrl:argument2>
                 <swrl:Variable rdf:ID="fcx"/>
               </swrl:argument2>
               <swrl:argument1>
                 <swrl:Variable rdf:ID="x"/>
               </swrl:argument1>
             </swrl:DatavaluedPropertyAtom>
           </rdf:first>
           <rdf:rest>
             <swrl:AtomList>
               <rdf:rest>
                 <swrl:AtomList>
                   <rdf:rest>
                     <swrl:AtomList>
                       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                       <rdf:first>
                         <swrl:BuiltinAtom>
                           <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                           <swrl:arguments>
                             <rdf:List>
                               <rdf:rest>
                                 <rdf:List>
                                   <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                   >suspicious</rdf:first>
                                   <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                 </rdf:List>
                               </rdf:rest>
                               <rdf:first>
                                 <swrl:Variable rdf:ID="ccx"/>
                               </rdf:first>
                             </rdf:List>
                           </swrl:arguments>
                         </swrl:BuiltinAtom>
                       </rdf:first>
                     </swrl:AtomList>
                   </rdf:rest>
                   <rdf:first>
                     <swrl:BuiltinAtom>
                       <swrl:arguments>
                         <rdf:List>
                           <rdf:first rdf:resource="#fcx"/>
                           <rdf:rest>
                             <rdf:List>
```

```
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                >SUS</rdf:first>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                              </rdf:List>
                            </rdf:rest>
                          </rdf:List>
                        </swrl:arguments>
                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                  </swrl:AtomList>
                </rdf:rest>
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                    <swrl:argument1 rdf:resource="#x"/>
                    <swrl:propertyPredicate
rdf:resource="#contactClassification"/>
                    <swrl:argument2 rdf:resource="#ccx"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#Contact"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#hasSemanticError"/>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:head>
  </swrl:Imp>
  <SuspiciousCountryCodes rdf:ID="SO">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Somolia
</rdfs:comment>
  </SuspiciousCountryCodes>
  <FriendlyCountryCodes rdf:ID="CA">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Canada</rdfs:comment>
  </FriendlyCountryCodes>
  <swrl:Variable rdf:ID="threatCodex"/>
  <FriendlyCountryCodes rdf:ID="FR">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >France</rdfs:comment>
  </FriendlyCountryCodes>
  <AlertCode rdf:ID="SUSP"/>
  <TargetType rdf:ID="unknown"/>
  <Source rdf:ID="SINS"/>
  <SuspiciousCountryCodes rdf:ID="CN">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >China
</rdfs:comment>
  </SuspiciousCountryCodes>
  <swrl:Variable rdf:ID="rangex"/>
  <ContactClassification rdf:ID="ContactClassification_4"/>
  <Contact rdf:ID="Contact_3">
    <contactClassification rdf:resource="#friendly1"/>
  </Contact>
  <AlertCode rdf:ID="NSP"/>
  <swrl:Variable rdf:ID="threatCodex"/>
  <SuspiciousCountryCodes rdf:ID="SaudiArabia"/>
  <swrl:Imp rdf:ID="Threat">
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
                            <rdf:first rdf:resource="#speedx"/>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                                >5</rdf:first>
                              </rdf:List>
                            </rdf:rest>
                          </rdf:List>
                        </swrl:arguments>
                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThanOrEqual"/>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:first rdf:resource="#rangex"/>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
```

```
                                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                                    >20000</rdf:first>
                                  </rdf:List>
                                </rdf:rest>
                              </rdf:List>
                            </swrl:arguments>
                          </swrl:BuiltinAtom>
                        </rdf:first>
                      </swrl:AtomList>
                    </rdf:rest>
                  </swrl:AtomList>
                </rdf:rest>
                <rdf:first>
                  <swrl:DatavaluedPropertyAtom>
                    <swrl:argument1 rdf:resource="#x"/>
                    <swrl:argument2 rdf:resource="#rangex"/>
                    <swrl:propertyPredicate rdf:resource="#range"/>
                  </swrl:DatavaluedPropertyAtom>
                </rdf:first>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:propertyPredicate rdf:resource="#speed"/>
                <swrl:argument2 rdf:resource="#speedx"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#HostileContact"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
            <swrl:propertyPredicate rdf:resource="#isThreat"/>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:head>
  </swrl:Imp>
  <swrl:Imp rdf:ID="ForceCodeRangeTest">
    <swrl:head>
      <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
```

```
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
            <swrl:propertyPredicate rdf:resource="#rangeViolationError"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#Contact"/>
          </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:propertyPredicate rdf:resource="#forceCode"/>
                <swrl:argument2 rdf:resource="#forceCodex"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:BuiltinAtom>
                    <swrl:arguments>
                      <rdf:List>
                        <rdf:rest>
                          <rdf:List>
                            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                            >99</rdf:first>
                            <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                          </rdf:List>
                        </rdf:rest>
                        <rdf:first rdf:resource="#forceCodex"/>
                      </rdf:List>
                    </swrl:arguments>
                    <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThan"/>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <swrl:Imp rdf:ID="SemCheckFive">
    <swrl:head>
      <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
        <rdf:first>
```

```
          <swrl:DatavaluedPropertyAtom>
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:propertyPredicate rdf:resource="#hasSemanticError"/>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Contact"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
                            <rdf:first rdf:resource="#fcx"/>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                >FRD</rdf:first>
                              </rdf:List>
                            </rdf:rest>
                          </rdf:List>
                        </swrl:arguments>
                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:first rdf:resource="#ccx"/>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                    >friendly</rdf:first>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                  </rdf:List>
                                </rdf:rest>
                              </rdf:List>
```

```
                        </swrl:arguments>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                  </swrl:AtomList>
                </rdf:rest>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:argument2 rdf:resource="#ccx"/>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:propertyPredicate
rdf:resource="#contactClassification"/>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:argument2 rdf:resource="#fcx"/>
            <swrl:propertyPredicate rdf:resource="#threatCode"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </rdf:rest>
  </swrl:AtomList>
 </swrl:body>
</swrl:Imp>
<swrl:Imp rdf:ID="CourseRangeRestriction">
 <swrl:head>
  <swrl:AtomList>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
   <rdf:first>
    <swrl:DatavaluedPropertyAtom>
     <swrl:argument1 rdf:resource="#x"/>
     <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >true</swrl:argument2>
     <swrl:propertyPredicate rdf:resource="#rangeViolationError"/>
    </swrl:DatavaluedPropertyAtom>
   </rdf:first>
  </swrl:AtomList>
 </swrl:head>
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:argument1 rdf:resource="#x"/>
     <swrl:classPredicate rdf:resource="#Contact"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:BuiltinAtom>
         <swrl:arguments>
          <rdf:List>
```

```
                        <rdf:rest>
                          <rdf:List>
                            <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                            >0</rdf:first>
                          </rdf:List>
                        </rdf:rest>
                        <rdf:first>
                          <swrl:Variable rdf:ID="coursex"/>
                        </rdf:first>
                      </rdf:List>
                    </swrl:arguments>
                    <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThan"/>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:argument2 rdf:resource="#coursex"/>
                <swrl:propertyPredicate rdf:resource="#course"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <Applicability rdf:ID="live_training"/>
  <swrl:Imp rdf:ID="ImmediateThreat">
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:propertyPredicate rdf:resource="#isImmediateThreat"/>
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
```

```
                        <rdf:first rdf:resource="#speedx"/>
                        <rdf:rest>
                          <rdf:List>
                            <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                              >10</rdf:first>
                          </rdf:List>
                        </rdf:rest>
                      </rdf:List>
                    </swrl:arguments>
                    <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThanOrEqual"/>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                                  >10000</rdf:first>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                              </rdf:List>
                            </rdf:rest>
                            <rdf:first rdf:resource="#rangex"/>
                          </rdf:List>
                        </swrl:arguments>
                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                  </swrl:AtomList>
                </rdf:rest>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#range"/>
                <swrl:argument2 rdf:resource="#rangex"/>
                <swrl:argument1 rdf:resource="#x"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#speed"/>
            <swrl:argument2 rdf:resource="#speedx"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
```

```
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#HostileContact"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <TargetType rdf:ID="hostile"/>
  <HostileContact rdf:ID="HostileContact_2">
    <speed rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >12</speed>
    <range rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >12000</range>
  </HostileContact>
  <FriendlyCountryCodes rdf:ID="DE">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Germany</rdfs:comment>
  </FriendlyCountryCodes>
  <swrl:Imp rdf:ID="SameAs">
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:argument1>
                  <swrl:Variable rdf:ID="y"/>
                </swrl:argument1>
                <swrl:classPredicate rdf:resource="#Contact"/>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="#position"/>
                        <swrl:argument2>
                          <swrl:Variable rdf:ID="posy"/>
                        </swrl:argument2>
                        <swrl:argument1 rdf:resource="#y"/>
                      </swrl:IndividualPropertyAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:first>
                                  <swrl:Variable rdf:ID="posx"/>
                                </rdf:first>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                    <rdf:first rdf:resource="#posy"/>
                                  </rdf:List>
                                </rdf:rest>
```

```
                                </rdf:List>
                              </swrl:arguments>
                            </swrl:BuiltinAtom>
                          </rdf:first>
                          <rdf:rest>
                            <swrl:AtomList>
                              <rdf:rest>
                                <swrl:AtomList>
                                  <rdf:rest>
                                    <swrl:AtomList>
                                      <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                      <rdf:first>
                                        <swrl:BuiltinAtom>
                                          <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                                          <swrl:arguments>
                                            <rdf:List>
                                              <rdf:rest>
                                                <rdf:List>
                                                  <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                                  <rdf:first>
                                                    <swrl:Variable rdf:ID="dty"/>
                                                  </rdf:first>
                                                </rdf:List>
                                              </rdf:rest>
                                              <rdf:first>
                                                <swrl:Variable rdf:ID="dtx"/>
                                              </rdf:first>
                                            </rdf:List>
                                          </swrl:arguments>
                                        </swrl:BuiltinAtom>
                                      </rdf:first>
                                    </swrl:AtomList>
                                  </rdf:rest>
                                  <rdf:first>
                                    <swrl:DatavaluedPropertyAtom>
                                      <swrl:argument2 rdf:resource="#dty"/>
                                      <swrl:argument1 rdf:resource="#y"/>
                                      <swrl:propertyPredicate
rdf:resource="#dateTime"/>
                                    </swrl:DatavaluedPropertyAtom>
                                  </rdf:first>
                                </swrl:AtomList>
                              </rdf:rest>
                              <rdf:first>
                                <swrl:DatavaluedPropertyAtom>
                                  <swrl:argument2 rdf:resource="#dtx"/>
                                  <swrl:argument1 rdf:resource="#x"/>
                                  <swrl:propertyPredicate
rdf:resource="#dateTime"/>
                                </swrl:DatavaluedPropertyAtom>
                              </rdf:first>
                            </swrl:AtomList>
                          </rdf:rest>
                        </swrl:AtomList>
                      </rdf:rest>
                    </swrl:AtomList>
                  </rdf:rest>
                  <rdf:first>
                    <swrl:IndividualPropertyAtom>
                      <swrl:argument1 rdf:resource="#x"/>
```

```
                    <swrl:propertyPredicate rdf:resource="#position"/>
                    <swrl:argument2 rdf:resource="#posx"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Contact"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:SameIndividualAtom>
            <swrl:argument2 rdf:resource="#y"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:SameIndividualAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
  </swrl:Imp>
  <SuspiciousCountryCodes rdf:ID="NOKO">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >North Korea
</rdfs:comment>
  </SuspiciousCountryCodes>
  <ThreatCode rdf:ID="JOKER"/>
  <Source rdf:ID="GPS"/>
  <swrl:Imp rdf:ID="SemanticCheck">
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#hasSemanticError"/>
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#Contact"/>
          </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
```

```
            <swrl:AtomList>
              <rdf:rest>
                <swrl:AtomList>
                  <rdf:rest>
                    <swrl:AtomList>
                      <rdf:first>
                        <swrl:BuiltinAtom>
                          <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#notEqual"/>
                          <swrl:arguments>
                            <rdf:List>
                              <rdf:first rdf:resource="#ccx"/>
                              <rdf:rest>
                                <rdf:List>
                                  <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                  <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                    >hostile</rdf:first>
                                </rdf:List>
                              </rdf:rest>
                            </rdf:List>
                          </swrl:arguments>
                        </swrl:BuiltinAtom>
                      </rdf:first>
                      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                    </swrl:AtomList>
                  </rdf:rest>
                  <rdf:first>
                    <swrl:BuiltinAtom>
                      <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                      <swrl:arguments>
                        <rdf:List>
                          <rdf:first rdf:resource="#fcx"/>
                          <rdf:rest>
                            <rdf:List>
                              <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                              <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                >HOS</rdf:first>
                            </rdf:List>
                          </rdf:rest>
                        </rdf:List>
                      </swrl:arguments>
                    </swrl:BuiltinAtom>
                  </rdf:first>
                </swrl:AtomList>
              </rdf:rest>
              <rdf:first>
                <swrl:IndividualPropertyAtom>
                  <swrl:argument2 rdf:resource="#ccx"/>
                  <swrl:propertyPredicate
rdf:resource="#contactClassification"/>
                  <swrl:argument1 rdf:resource="#x"/>
                </swrl:IndividualPropertyAtom>
              </rdf:first>
            </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:DatavaluedPropertyAtom>
```

```
              <swrl:argument1 rdf:resource="#x"/>
              <swrl:argument2 rdf:resource="#fcx"/>
              <swrl:propertyPredicate rdf:resource="#threatCode"/>
            </swrl:DatavaluedPropertyAtom>
          </rdf:first>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:body>
 </swrl:Imp>
 <NeutralCountryCodes rdf:ID="BE">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Belgium
</rdfs:comment>
 </NeutralCountryCodes>
 <swrl:AtomList>
   <rdf:rest>
     <swrl:AtomList>
       <rdf:rest>
         <swrl:AtomList>
           <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
           <rdf:first>
             <swrl:IndividualPropertyAtom>
               <swrl:argument1 rdf:resource="#x"/>
               <swrl:propertyPredicate rdf:resource="#contactClassification"/>
               <swrl:argument2 rdf:resource="#ccx"/>
             </swrl:IndividualPropertyAtom>
           </rdf:first>
         </swrl:AtomList>
       </rdf:rest>
       <rdf:first>
         <swrl:IndividualPropertyAtom>
           <swrl:propertyPredicate rdf:resource="#threatCode"/>
           <swrl:argument1 rdf:resource="#x"/>
           <swrl:argument2 rdf:resource="#threatCodex"/>
         </swrl:IndividualPropertyAtom>
       </rdf:first>
     </swrl:AtomList>
   </rdf:rest>
   <rdf:first>
     <swrl:ClassAtom>
       <swrl:argument1 rdf:resource="#x"/>
       <swrl:classPredicate rdf:resource="#Contact"/>
     </swrl:ClassAtom>
   </rdf:first>
 </swrl:AtomList>
 <HostileCountryCodes rdf:ID="AF">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Afghanistan
</rdfs:comment>
 </HostileCountryCodes>
 <Applicability rdf:ID="simulated"/>
 <swrl:Imp rdf:ID="ThreeMinuteRule">
   <swrl:body>
     <swrl:AtomList>
       <rdf:first>
         <swrl:ClassAtom>
           <swrl:argument1 rdf:resource="#x"/>
           <swrl:classPredicate rdf:resource="#Contact"/>
         </swrl:ClassAtom>
       </rdf:first>
       <rdf:rest>
```

```
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:BuiltinAtom>
                    <swrl:arguments>
                      <rdf:List>
                        <rdf:first rdf:resource="#y"/>
                        <rdf:rest>
                          <rdf:List>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                                >100</rdf:first>
                              </rdf:List>
                            </rdf:rest>
                            <rdf:first rdf:resource="#speedx"/>
                          </rdf:List>
                        </rdf:rest>
                      </rdf:List>
                    </swrl:arguments>
                    <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#multiply"/>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:DatavaluedPropertyAtom>
                        <swrl:argument1 rdf:resource="#x"/>
                        <swrl:argument2 rdf:resource="#rangex"/>
                        <swrl:propertyPredicate rdf:resource="#range"/>
                      </swrl:DatavaluedPropertyAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:first rdf:resource="#rangex"/>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                      <rdf:first rdf:resource="#y"/>
                                  </rdf:List>
                                </rdf:rest>
                              </rdf:List>
                            </swrl:arguments>
                          </swrl:BuiltinAtom>
                        </rdf:first>
                      </swrl:AtomList>
                    </rdf:rest>
                  </swrl:AtomList>
                </rdf:rest>
```

```
            </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:DatavaluedPropertyAtom>
              <swrl:argument2 rdf:resource="#speedx"/>
              <swrl:propertyPredicate rdf:resource="#speed"/>
              <swrl:argument1 rdf:resource="#x"/>
            </swrl:DatavaluedPropertyAtom>
          </rdf:first>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:body>
  <swrl:head>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
      <rdf:first>
        <swrl:DatavaluedPropertyAtom>
          <swrl:argument1 rdf:resource="#x"/>
          <swrl:propertyPredicate rdf:resource="#isImmediateThreat"/>
          <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</swrl:argument2>
        </swrl:DatavaluedPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:head>
 </swrl:Imp>
 <swrl:Imp rdf:ID="SemCheckThree">
  <swrl:head>
    <swrl:AtomList>
      <rdf:first>
        <swrl:DatavaluedPropertyAtom>
          <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</swrl:argument2>
          <swrl:propertyPredicate rdf:resource="#hasSemanticError"/>
          <swrl:argument1 rdf:resource="#x"/>
        </swrl:DatavaluedPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:rest>
            <swrl:AtomList>
              <rdf:first>
                <swrl:IndividualPropertyAtom>
                  <swrl:propertyPredicate
rdf:resource="#contactClassification"/>
                  <swrl:argument1 rdf:resource="#x"/>
                  <swrl:argument2 rdf:resource="#ccx"/>
                </swrl:IndividualPropertyAtom>
              </rdf:first>
              <rdf:rest>
                <swrl:AtomList>
                  <rdf:rest>
                    <swrl:AtomList>
```

```
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#notEqual"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                      >neutral</rdf:first>
                                  </rdf:List>
                                </rdf:rest>
                                <rdf:first rdf:resource="#ccx"/>
                              </rdf:List>
                            </swrl:arguments>
                          </swrl:BuiltinAtom>
                        </rdf:first>
                        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                      </swrl:AtomList>
                    </rdf:rest>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
                            <rdf:first rdf:resource="#fcx"/>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                  >NEU</rdf:first>
                              </rdf:List>
                            </rdf:rest>
                          </rdf:List>
                        </swrl:arguments>
                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                  </swrl:AtomList>
                </rdf:rest>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#threatCode"/>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:argument2 rdf:resource="#fcx"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Contact"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
```

```
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <ThreatCode rdf:ID="FAKER"/>
  <NeutralCountryCodes rdf:ID="CH">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Switzerland</rdfs:comment>
  </NeutralCountryCodes>
  <ThreatCode rdf:ID="AFD"/>
  <AlertCode rdf:ID="HIT"/>
  <swrl:Imp rdf:ID="CourseRangeRestTwo">
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Contact"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:argument2 rdf:resource="#coursex"/>
                <swrl:propertyPredicate rdf:resource="#course"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:BuiltinAtom>
                    <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThan"/>
                    <swrl:arguments>
                      <rdf:List>
                        <rdf:first rdf:resource="#coursex"/>
                        <rdf:rest>
                          <rdf:List>
                            <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                            >360</rdf:first>
                          </rdf:List>
                        </rdf:rest>
                      </rdf:List>
                    </swrl:arguments>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
```

```
          <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</swrl:argument2>
          <swrl:argument1 rdf:resource="#x"/>
          <swrl:propertyPredicate rdf:resource="#rangeViolationError"/>
        </swrl:DatavaluedPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
    </swrl:AtomList>
  </swrl:head>
 </swrl:Imp>
 <FriendlyCountryCodes rdf:ID="IL">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Israel
</rdfs:comment>
 </FriendlyCountryCodes>
 <ThreatCode rdf:ID="PND"/>
 <TargetType rdf:ID="neutral"/>
 <Applicability rdf:ID="real_world"/>
 <swrl:Imp rdf:ID="ReclassifySupsiciousAsHostile">
  <swrl:head>
    <swrl:AtomList>
      <rdf:first>
        <swrl:ClassAtom>
          <swrl:classPredicate rdf:resource="#HostileContact"/>
          <swrl:argument1 rdf:resource="#x"/>
        </swrl:ClassAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:first>
        <swrl:ClassAtom>
          <swrl:argument1 rdf:resource="#x"/>
          <swrl:classPredicate rdf:resource="#SuspectContact"/>
        </swrl:ClassAtom>
      </rdf:first>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:first>
            <swrl:DatavaluedPropertyAtom>
              <swrl:propertyPredicate rdf:resource="#range"/>
              <swrl:argument1 rdf:resource="#x"/>
              <swrl:argument2 rdf:resource="#rangex"/>
            </swrl:DatavaluedPropertyAtom>
          </rdf:first>
          <rdf:rest>
            <swrl:AtomList>
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
              <rdf:first>
                <swrl:BuiltinAtom>
                  <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThan"/>
                  <swrl:arguments>
                    <rdf:List>
                      <rdf:rest>
                        <rdf:List>
```

```
                        <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                        <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                        >10000</rdf:first>
                      </rdf:List>
                    </rdf:rest>
                    <rdf:first rdf:resource="#rangex"/>
                  </rdf:List>
                </swrl:arguments>
              </swrl:BuiltinAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </rdf:rest>
  </swrl:AtomList>
 </swrl:body>
</swrl:Imp>
<NeutralCountryCodes rdf:ID="ES">
 <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Spain
</rdfs:comment>
</NeutralCountryCodes>
<SuspiciousCountryCodes rdf:ID="KH">
 <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Cambodia
</rdfs:comment>
</SuspiciousCountryCodes>
<swrl:Imp rdf:ID="PossibleThreat">
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:classPredicate rdf:resource="#HostileContact"/>
     <swrl:argument1 rdf:resource="#x"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:first>
        <swrl:DatavaluedPropertyAtom>
         <swrl:argument2 rdf:resource="#rangex"/>
         <swrl:propertyPredicate rdf:resource="#range"/>
         <swrl:argument1 rdf:resource="#x"/>
        </swrl:DatavaluedPropertyAtom>
       </rdf:first>
       <rdf:rest>
        <swrl:AtomList>
         <rdf:first>
          <swrl:BuiltinAtom>
           <swrl:arguments>
            <rdf:List>
             <rdf:rest>
              <rdf:List>
               <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
               >0</rdf:first>
               <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
              </rdf:List>
```

```
                        </rdf:rest>
                        <rdf:first rdf:resource="#speedx"/>
                      </rdf:List>
                    </swrl:arguments>
                    <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThanOrEqual"/>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
                            <rdf:first rdf:resource="#rangex"/>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                                  >50000</rdf:first>
                                  <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                </rdf:List>
                              </rdf:rest>
                            </rdf:List>
                          </swrl:arguments>
                          <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
                        </swrl:BuiltinAtom>
                      </rdf:first>
                    </swrl:AtomList>
                  </rdf:rest>
                </swrl:AtomList>
              </rdf:rest>
            </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:DatavaluedPropertyAtom>
              <swrl:argument2 rdf:resource="#speedx"/>
              <swrl:propertyPredicate rdf:resource="#speed"/>
              <swrl:argument1 rdf:resource="#x"/>
            </swrl:DatavaluedPropertyAtom>
          </rdf:first>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:body>
  <swrl:head>
    <swrl:AtomList>
      <rdf:first>
        <swrl:DatavaluedPropertyAtom>
          <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</swrl:argument2>
          <swrl:argument1 rdf:resource="#x"/>
          <swrl:propertyPredicate rdf:resource="#isPossibleThreat"/>
        </swrl:DatavaluedPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
    </swrl:AtomList>
```

```
    </swrl:head>
  </swrl:Imp>
  <TargetType rdf:ID="friend"/>
  <Source rdf:ID="RLGN"/>
  <swrl:Imp rdf:ID="SemCheckTwo">
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:DatavaluedPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#hasSemanticError"/>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</swrl:argument2>
          </swrl:DatavaluedPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:argument2 rdf:resource="#fcx"/>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:propertyPredicate rdf:resource="#threatCode"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                    <swrl:argument1 rdf:resource="#x"/>
                    <swrl:argument2 rdf:resource="#ccx"/>
                    <swrl:propertyPredicate
rdf:resource="#contactClassification"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#notEqual"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                    >unknown</rdf:first>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                  </rdf:List>
                                </rdf:rest>
                                <rdf:first rdf:resource="#ccx"/>
```

```
                              </rdf:List>
                            </swrl:arguments>
                          </swrl:BuiltinAtom>
                        </rdf:first>
                      </swrl:AtomList>
                    </rdf:rest>
                    <rdf:first>
                      <swrl:BuiltinAtom>
                        <swrl:arguments>
                          <rdf:List>
                            <rdf:rest>
                              <rdf:List>
                                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                >UNK</rdf:first>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                              </rdf:List>
                            </rdf:rest>
                            <rdf:first rdf:resource="#fcx"/>
                          </rdf:List>
                        </swrl:arguments>
                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                      </swrl:BuiltinAtom>
                    </rdf:first>
                  </swrl:AtomList>
                </rdf:rest>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#Contact"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <swrl:Imp rdf:ID="DifferentFrom">
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:DifferentIndividualsAtom>
            <swrl:argument2 rdf:resource="#y"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:DifferentIndividualsAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:argument1 rdf:resource="#y"/>
                <swrl:classPredicate rdf:resource="#Contact"/>
              </swrl:ClassAtom>
```

```
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                    <swrl:propertyPredicate rdf:resource="#position"/>
                    <swrl:argument2 rdf:resource="#posx"/>
                    <swrl:argument1 rdf:resource="#x"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="#position"/>
                        <swrl:argument2 rdf:resource="#posy"/>
                        <swrl:argument1 rdf:resource="#y"/>
                      </swrl:IndividualPropertyAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:first rdf:resource="#posx"/>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                    <rdf:first rdf:resource="#posy"/>
                                  </rdf:List>
                                </rdf:rest>
                              </rdf:List>
                            </swrl:arguments>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#notEqual"/>
                          </swrl:BuiltinAtom>
                        </rdf:first>
                        <rdf:rest>
                          <swrl:AtomList>
                            <rdf:rest>
                              <swrl:AtomList>
                                <rdf:rest>
                                  <swrl:AtomList>
                                    <rdf:first>
                                      <swrl:BuiltinAtom>
                                        <swrl:arguments>
                                          <rdf:List>
                                            <rdf:rest>
                                              <rdf:List>
                                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                                <rdf:first rdf:resource="#dty"/>
                                              </rdf:List>
                                            </rdf:rest>
                                            <rdf:first rdf:resource="#dtx"/>
                                          </rdf:List>
                                        </swrl:arguments>
                                        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                                      </swrl:BuiltinAtom>
                                    </rdf:first>
```

```
                                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                  </swrl:AtomList>
                                </rdf:rest>
                                <rdf:first>
                                  <swrl:DatavaluedPropertyAtom>
                                    <swrl:argument1 rdf:resource="#y"/>
                                    <swrl:propertyPredicate
rdf:resource="#dateTime"/>
                                    <swrl:argument2 rdf:resource="#dty"/>
                                  </swrl:DatavaluedPropertyAtom>
                                </rdf:first>
                              </swrl:AtomList>
                            </rdf:rest>
                            <rdf:first>
                              <swrl:DatavaluedPropertyAtom>
                                <swrl:propertyPredicate
rdf:resource="#dateTime"/>
                                <swrl:argument1 rdf:resource="#x"/>
                                <swrl:argument2 rdf:resource="#dtx"/>
                              </swrl:DatavaluedPropertyAtom>
                            </rdf:first>
                          </swrl:AtomList>
                        </rdf:rest>
                      </swrl:AtomList>
                    </rdf:rest>
                  </swrl:AtomList>
                </rdf:rest>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:classPredicate rdf:resource="#Contact"/>
              </swrl:ClassAtom>
            </rdf:first>
          </swrl:AtomList>
        </swrl:body>
      </swrl:Imp>
    </rdf:RDF>

    <!-- Created with Protege (with OWL Plug-in 2.2, Build 304)
    http://protege.stanford.edu -->
```

LIST OF REFERENCES

Aamodt, A., Plaza, E. (1994). *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.* Retrieved on April 28, 2006 from http://www.iiia.csic.es/People/enric/AICom.html

Alesso, P., & Smith, C. (2005). *Developing Semantic Web Services.* Wellesey, MA. A K Peters, Ltd.

ARQ- A SPARQL Processor for Jena (2006). Retrieved on April 20, 2006 from http://jena.sourceforge.net/ARQ

Ayers, Danny. (2005). *Oracle Supports RDF.* Retrieved on April 26, 2006 from http://dannyayers.com/2005/07/12/oracle-supports-rdf

Berners-Lee, Tim. (1998). *Semantic Web Roadmap.* Retrieved December 1, 2005 from http://www.w3.org/DesignIssues/Semantic.html

Berners-Lee, Tim. (1999). *Weaving the Web.* New York: HarperCollins Publishers, Inc.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). *The Semantic Web.* Scientific America: May 17, 2001.

Bothner, Per. (2002). *What is XQuery.* Retrieved April 20, 2006 from http://www.xml.com/lpt/a/2002/10/16/xquery.html

Boulton, Clint. (2004). *W3C Wraps up Semantic Web Standards.* Retrieved on April 23, 2006 from http://www.internetnews.com/infra/article.php/3310831

Brutzman, Don & Grimley, Michael. (2005). *USW-XML: Next-Generation Undersea Warfare Information Interoperability.* Retrieved November 26, 2005 from http://usw- xml.cs.nps.navy.mil/UswXmlInformationInteroperability.2005 October18.pdf

Brutzman, Don & Grimley, Michael. (2006). *USW-XML: Next-Generation USW Interoperability using XML.* Retrieved on April 30, 2006 from http://usw-xml.cs.nps.navy.mil/usw-xml.html

Burkley, Fred. (2006). *Tactical Assessment Markup Language Design*. Personal email dated May 23, 2006.

Costa, P., Fung, F., Laskey, K., Pool, M., Takikawa, M., Wright, E. (2004). *MEBN Logic: A Key Enable for Network Centric Warfare.* Retrieved on May 23, 2006 from http://dodccrp.org/events/2005/10th/CD/papers/224.pdf

CYCorp. (2006). http://www.cyc.com. *Overview of Cycorp's Research and Development.* Retrieved on March 5, 2006.

Cycl Syntax. (2005). Retrieved on March 10, 2006 from http://www.cyc.com/cycdoc/ref/cycl-syntax.html

Daconta, M., Obrst, L., & Smith, K. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management.*. Indianapolis: Wiley Publishing, Inc.

Davis, Duane. (2006). PhD Dissertation NPS September 2006.

Defense Acquisition Guidebook. (2005). Retrieved on December 10, 2005 from http://akss.dau.mil/dag/Guidebook/IG_c7.3.4.1.asp

DoD Metadata Registry. (2005). http://diides.ncr.disa.mil

DON Naming and Design Rules. (2005). Retrieved on May 20, 2006 from http://www.doncio.navy.mil

Dodds, Leigh. (2005). *Introducing SPARQL: Querying the Semantic Web.* Retrieved on April 20, 2006 from http://www.xml.com/pub/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html

Dublin Core Metadata Initiative. (2005) http://dublincore.org, retrieved on March 4, 2006.

Fikes, R., Hayes, P., Horrocks, I. (2003). *OWL-QL - A Language for Deductive Query Answering on the Semantic Web.* Retrieved on April 2, 2006 from http://classweb.gmu.edu/kersch/infs770/Topics/SemanticWeb/KSL-03-14(OWL-QL).pdf

Fleet Anti-Submarine Warfare Command Press Release. (2004). *New ASW Tactical Assessment System Ready for Fleet Testing.* Retrieved on May 25, 2006 from http://www.news.navy.mil/search/display.asp?story_id=13600

Friedman-Hill, Ernest. (2003). *Jess in Action: Rule-Based Systems in Java.* Greenwich: Manning Publications Co.

Fukushige, Yoshio (2004) *Representing Probabilistic Knowledge in the Semantic Web.* Retrieved on May 23, 2006 from http://www.pr-owl.org

Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubezy, M., Eriksson, H., Noy, N., Samson, T. (2002). *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development.* Retrieved on March 25, 2006 from http://smi-web.stanford.edu/auslese/smi-web/reports/SMI-2002-0943.pdf

Gruber, Tom. (1993). *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition: Vol. 5.

Harold, Eliotte. (2003). *XML Quotes in 2003.* Retrieved on April 29, 2006 from http://www.ibiblio.org/xml/quotes2003.html

Hayes-Roth, F., Waterman, D., and Lenat, D., (1983). *Building Expert Systems*, Addison-Wesley.

Hayes-Roth, Rick. (2005) *Model-based Communication Networks and VIRT: Filtering Information by Value to Improve Collaborative Decision Making.* 10th International Command and Control Research and Technology Symposium.

Herman, Ivan. (2004). *Semantic Web Tutorial*. Retrieved on 23 May, 2006 from http://ebiquity.umbc.edu/blogger/2004/11/25/semantic-web-tutorial-by-ivan-herman

Hillman, D. (2003). *Semantic and Syntactic Interoperability for Learning Object Metadata.* Retrieved on May 16, 2005 from http://www.cancore.ca/semantic_and_syntactic_interoperability.html#Approaches%20to%20Syntactic%20Interoperability

Hjelm, Johan. (2001). *Creating the Semantic Web with RDF*. New York: John Wiley & Sons, Inc.

Horrocks, Ian; Patel-Schneider, Peter; Boley, Harold; Tabet, Said; Grosof, Benjamin; Dean, Mike. (2004). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML.* Retrieved on November 2, 2005 from http://www.w3.org/Submission/SWRL

Hunter, D., Cagle, K., Dix, C., Kovack, R., Pinnock, J., & Rafter, J. (2003). *Beginning XML* (3rd ed.). Indianapolis: Wiley Publishing Inc.

Hu, Y., Se-Ting, C., & Min-Huei, Y. (2005). *Trust on the Semantic Web Pyramid: Some Issues and Challenges.* Retrieved on December 20, 2005 from http://www.cs.nccu.edu.tw/~jong/pub/trust-sem.pdf

Jambalaya. (2004). http://www.thechiselgroup.org/jambalaya

Jena - A Semantic Web Framework for Java. Retrieved on April 20, 2006 from http://jena.sourceforge.net/

Khare, R., & Rifkin, A. (1997). *Weaving a Web of Trust.* Retrieved on November 23, 2005 from http://www.w3j.com/7/s3.rifkin.wrap.html

Knowledge Interchange Format (KIF). http://www-ksl.stanford.edu/knowledge-sharing/kif

Knublauch, Holger. (2003) *An AI tool for the Real World: Knowledge Modeling with Protégé.* Retrieved March 1, 2005 from http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html

Knublauch, H., Fergerson, R., Noy, N., Musen, M. (2004). *The Protégé OWL Plug-in: An Open Development Environment for Semantic Web Applications.* Retrieved on March 25, 2006 from http://www.knublauch.com

Knublauch, H., Rector, A., Stevens, R., Wroe, C. (2004). *A Practical Guide to Building OWL Ontologies Using the Protégé-OWL Plug-in and CO-ODE Tools.* Retrieved October 10, 2005 from http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

Lacy, Lee. (2005). *OWL: Representing Information Using the Web Ontology Language.* Canada: Trafford

Laskey, Kathryn. (2006). *MEBN: A Logic for Open-World Probabilistic Reasoning.* Retrieved on May 23, 2006 from http://ite.gmu.edu/~klaskey/papers/Laskey_MEBN_Logic.pdf

Lesser GNU Public License. (2005). Description of the Lesser GNU Public Licence. Retrieved on March 4, 2006 from http://www.gnu.org/licenses/lgpl.html

Mazzocchi, Stefano. (2004). *A No-nonsense Guide to Semantic Web Specs for XML People*. Retrieved on May 20, 2006 from http://www.betaversion.org/~stefano/linotype/news/78

Matheus, C., Kokar, M., Letkowski, J. (2005). *An Application of Semantic Web Technologies to Situation Awareness.* 2005 Semantic Web Conference.

Merriam Webster Dictionary (2005). *Semantic.* Retrieved November 20, 2005 from http://www.m-w.com/dictionary/semantic

McCarthy, John. (2004). *What is Artificial Intelligence?* Retrieved on April 29, 2006 from http://www-formal.stanford.edu/jmc/whatisai/whatisai.html

McCarthy, Philip. (2005). *Search RDF data with SPARQL*. Retrieved April 20, 2006 from http://www-128.ibm.com/developerworks/library/j-sparql

Morris, Bonnie. (1994). *Case-Based Reasoning.* Retrieved on April 29, 2006 from http://accounting.rutgers.edu/raw/aies/www.bus.orst.edu/faculty/brownc/aies/news-let/fall95/casebase.htm

Mozetic, Igor. (2006). *What is "Horn Logic"?* Retrieved on May 31, 2006 from http://www.w3.org/2005/rules/wg/wiki/Horn_Logic.

Mozilla Public License. (2006). Description of the Mozilla Public License. Retrieved on March 4, 2006 from http://www.mozilla.org/MPL/MPL-1.1.html

Nebel, Bernhard. (1999). *Frame-Based Systems.* Retrieved on May 16, 2006 from http://www.cs.umbc.edu/771/papers/nebel.html

Niles, I., Pease, A. (2001). *Towards a Standard Upper Ontology.* Retrieved on April 8, 2006 from http://home.earthlink.net/~adampease/professional/FOIS.pdf

Noy, N., & McGuinness, D. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology.* Retrieved November 1, 2005 from http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R., & Musen, M. (2001). *Creating Semantic Web Contents with Protege-2000.* Retrieved March 1, 2006 from http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-2001-0872.html

Obrst, L., & Davis, M. *Semantic Wave 2006.* (2006). Retrieved on March 2, 2005 from http://colab.cim3.net/file/work/SICoP/2006-02-09/Presentations/MDavis02092006.pdf

O'Conner, Martin. (2006). *SWRLTab.* Retrieved on May 1, 2006 from http://smi-web.stanford.edu/people/moconner/swrl/index.html

O'Conner, M., Knublauch, H., Samson, Tu., Grosof, B., Dean, M., Grosso, W., & Musen, M. (2005). *Supporting Rule System Interoperability on the Semantic Web with SWRL*. retrieved on December 1, 2005 from http://www.knublauch.com

Ontolog Collaborative Work Environment. http://ontolog.cim3.net

OWL. (2004). *Web Ontology LanguageOverview.* Retrieved on November 2, 2005 from http://www.w3.org/TR/owl-features

OWLViz Guide. (2004). Retrieved April 9, 2006 from http://www.co-ode.org/downloads/owlviz/OWLVizGuide.pdf

Pan, Jeff. (2004). *Description Logics: Reasoning Support for the Semantic Web.* Retrieved on March 1, 2006 from http://dl-web.man.ac.uk/~panz/Zhilin/pubc.php?sort=paper&id=thesis

Passin, Thomas. (2004). *Explorer's Guide to the Semantic Web.* Manning Publishers

Pellet Reasoner. (2003). *Maryland Information and Network Dynamics Lab Semantic Web Agents Project (Mindswap).* Retrieved on May 31, 2006 from http://www.mindswap.org/2003/pellet

Pitoura, E. (1997). *Providing Database Interoperability through Object-Oriented Language Constructs,* Journal of Systems Integration, Volume 7, No. 2, August 1997.

Plas, D., Verheijen, M., Zwaal, H., Hutschemaekers, E. (2006). *Manipulating Context Information with SWRL.* Retrieved on May 19, 2006 from https://doc.freeband.nl/dscgi/ds.py/Get/File-62333

Powers, Shelley. (2003). *Practical RDF.* Sebastopol, CA: O'reilly & Associates Inc.

PR-OWL. (2006). An overview of PR-OWL open research. Retrieved on May 23, 2006 from http://www.pr-owl.org

RacerPro Homepage. (2006). An overview of the RacerPro reasoner from Franz, Inc. http://www.racer-systems.com/products/racerpro/index.phtml

RacerPro User Guide. (2005). Retrieved on January 1, 2006 from http://www.racer-systems.com/products/racerpro/manual.phtml

RDF. (2005). *Resource Description Framework.* RDF Homepage. Retrieved on November 2, 2005 from http://www.w3.org/RDF

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Want, H., & Wroe, C. (2004). *OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors and Common Patters.* Retrieved March 20, 2005 from http://www.co-ode.org/resources/papers/ekaw2004.pdf

Reed, S., and Lenat, D. (2002). *Mapping Ontologies into Cyc*. In AAAI 2002 Conference Workshop on Ontologies for The Semantic Web, Edmonton, Canada, July 2002.

Riley, Gary. (2004). *CLIPS: A Tool for Building Expert Systems.* Retrieved on April 30, 2006 from http://www.ghg.net/clips/CLIPS.html

Sabatino, Greg. (2006). *TAML Summary*. Personal Email from Greg Sabatino dated May 24, 2006.

Sandahl, K. (1994). *Transferring Knowledge from Active Expert to End-user Environment.* Knowledge Acquisition, 6: 1-21.

SemanticWorks Homepage. (2006). Retrieved on December 1, 2005 from http://www.altova.com/products_semanticworks.html

Schmidt, Charles. (2005). *Logic, Knowledge, and Computation.* Retrieved on November 2, 2005 from http://www.rci.rutgers.edu/~cfs/472_html/Logic_KR/Adequacy.html

Sikora, Gary. (2006) *Applying Ontologies to Translate XML Schemas.* Progeny White Paper.

Storey, M., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N. (2002). *Jambalaya: Interactive Visualization to Enhance Ontology Authoring and Knowledge Acquisition in Protégé*. Retrieved April 9, 2006 from http://www.cs.uvic.ca/~mstorey/papers/kcap2001.pdf

SPARQL Protocol for RDF. (2006). Retrieved on April 20, 2006 from http://www.w3.org/TR/RDFSparql-protocol

SPARQL Query Language for RDF Recommendation. (2006). Retrieved on April 20, 2006 from http://www.w3.org/TR/RDFSparql-query

SPARQL Query Results XML Format Specification. (2006). Retrieved on April 20, 2006 from http://www.w3.org/TR/RDFSparql-XMLres

SUMO. (2005). An overview of the Standard Upper Merged Ontology (SUMO) Retrieved on January 3, 2006 from http://ontology.teknowledge.com

Standard Upper Ontology Working Group (SUO WG). (2003). Upper Ontology Effort Overview. Retrieved on February 4, 2006 from http://grouper.ieee.org/groups/suo/index.html

SWRL Built-Ins. (2005). Retrieved on February 3, 2005 from http://www.daml.org/rules/proposal/builtins.html

SWRL Editor FAQ's. (2005). Retrieved on December 2, 2005 from http://protege.stanford.edu/plug-ins/owl/swrl

Twinkle: A SPARQL Query Tool. (2006). Retrieved on April 20, 2006 from http://www.ldodds.com/projects/twinkle

University of Tokyo, Center for Collaborative Research. (2006). http://www.mpeg.rcast.u-tokyo.ac.jp/j/research/images/03-01/image010.gif&imgrefurl=http://www.mpeg.rcast.u-tokyo.ac.jp/j/research/03-01.html&h=221&w=468&sz=8&tbnid=_bcUFf5ak1mHoM:&tbnh=59&tbnw=125&hl=en&start=10&prev=/images%3Fq%3Drule-based%2Bsystems%2Barchitecture%26svnum%3D10%26hl%3Den%26lr%3D%26sa%3DG

Ushold, M., & Menzel, C. (2005). *Semantic Integration and Interoperability Using RDF and OWL.* Retrieved February 3, 2006 from http://www.w3.org/2001/sw/BestPractices/OEP/SemInt

Wennergren, D. (2005). *Extensible Markup Language (XML) Naming and Design Rules Official Release*. Retrieved on May 20, 2006 from http://doncio.navy.mil

Wikipedia: The Free Encyclopedia. (2006a). *CLIPS Programming Language.* Retrieved April 30, 2006 from http://en.wikipedia.org/wiki/CLIPS_programming_language

Wikipedia: The Free Encyclopedia. (2006b). *Description Logic.* Retrieved November 30, 2005 from http://en.wikipedia.org/wiki/Description_logic

Wikipedia: The Free Encyclopedia. (2006c). *Expert Systems.* Retrieved April 30, 2006 from http://en.wikipedia.org/wiki/Expert_System

Wikipedia: The Free Encyclopedia. (2005). *First-Order Logic.* Retrieved November 30, 2005 from http://en.wikipedia.org/wiki/First_Order_Logic

Wikipedia: The Free Encyclopedia. (2006). *Negation as Failure.* Retrieved May 19, 2006 http://en.wikipedia.org/wiki/Negation_as_failure

Wikipedia: The Free Encyclopedia. (2006). *Open World Assumption.* Retrieved May 19, 2006 http://en.wikipedia.org/wiki/Open_World_Assumption

Winters, L., & Tolk, A. (2005). *The Integration of Modeling and Simulation with Joint Command and Control on the Global Information Grid.* Retrieved October 31, 2005 from http://www.stormingmedia.us/17/1783/A178334.html

Wonder Web OWL Validator. (2006). http://phoebus.cs.man.ac.uk:9999/OWL/Validator

W3C OWL Recommendation. (2004). *OWL Web Ontology Language Overview*. Retrieved October 10, 2005 from http://www.w3.org/TR/owl-features

W3C RDF Validation Service. (2005). Tool for validating RDF documents http://www.w3.org/RDF/Validator

Young,Paul. *Heterogeneous Software System Interoperability Through Computer-Aided Resolution of Modeling Differences,* PhD dissertation, Naval Postgraduate School, June 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chief of Naval Research
   Office of Naval Research
   Arlington, VA

4. Dr. Peter Denning
   Naval Postgraduate School
   Monterey, CA

5. Dr. Don Brutzman
   Naval Postgraduate School
   Monterey, CA

6. Curt Blais
   Naval Postgraduate School
   Monterey, CA

7. CAPT Paul Young, USN
   United States Naval Academy
   Annapolis, MA

8. CAPT Richard Lee, USN Ret.
   Office of the Secretary of Defense
   Washington, D.C.

9. Dr. Rick Hayes-Roth
   Naval Postgraduate School
   Monterey, CA

10. Dr. Craig Smith
    Naval Postgraduate School
    Monterey, CA

11. Dr. Leonard Ferrari
    Naval Postgraduate School
    Monterey, CA

12. Dr. Dan Boger
    Naval Postgraduate School
    Monterey, CA

13. Dr. Neil Rowe
    Naval Postgraduate School
    Monterey, CA

14. Sue Higgins
    Naval Postgraduate School
    Monterey, CA

15. Shelley Gallup
    Naval Postgraduate School
    Monterey, CA

16. Dr. Robert McGhee
    Naval Postgraduate School
    Monterey, CA

17. CDR John Jorgensen, USN
    NAVSEA PEO IWS
    Washington, D.C.

18. Erik Chaum
    Naval Undersea Warfare Center
    Newport, RI

19. Fred Burkley
    Naval Undersea Warfare Center
    Newport, RI

20. CAPT Paul Rosbolt, USN
    NAVSEA PEO IWS
    Washington, D.C.

21. Dr. Ken Laskey
    MITRE
    McLean, VA

22. Dr. Mark Pullen
    George Mason University
    Fairfax, VA

23. Dr. Andreas Tolk
Old Dominion University
Norfolk, VA

24. Dr. Mike Hieb
George Mason University
Fairfax, VA

25. Dr. Kathryn Laskey
George Mason University
Fairfax, VA

26. Dr. Eric Miller
Semantic Web Activity for W3C
Cambridge, MA

27. Dr. Francisco Loaiza
Institute for Defense Analyses
Alexandria, VA

28. Lee Lacy
Dynamics Research Corporation
Orlando, FL

29. Bob Green
DON CIO
Washington, D.C.

30. Gary Sikora
Progeny Systems
Manassas, VA

31. Skip Miller
Progeny Systems
Manassas, VA

32. Mike Hess
Fleet ASW Command (FASWC)
San Diego, CA

33. Mike Grimley
NUWC
Newport, RI

34. Greg Sabatino
NUWC
Newport, RI

35. Peter Lyon
CACI
Washington, D.C.

36. Chris Gunderson
World Wide Consortium for the Grid (W2COG)

37. CAPT Scot Miller, USN
SPAWAR Assessment & Experimentation
San Diego, CA

38. Dallas Meggitt
Sound and Sea Technology
Edmonds, WA

39. Adolf Neumann
21st Century Systems
Crystal City, VA

40. John Davis
University of Massachusetts Dartmouth
North Dartmouth, MA

41. Undersea Warfare Announcements List
Naval Postgraduate School
Monterey, CA

42. USW-XML Mailing List
Naval Postgraduate School
Monterey, CA

43. Waterside-Protection Mailing List
Naval Postgraduate School
Monterey, CA