



**A MOF Metamodel & UML Profile for
the Web Ontology Language (OWL)**
January 30, 2003

Elisa F. Kendall
CEO & Founder
ekendall@sandsoft.com
(650) 960-2456



- 1996-1998 DARPA Rapid Prototyping of Application Specific Signal Processors Program
- 1999 Initiated research into whether a UML approach to KR could support brokering across ontologies or facilitate agent communications
- 2000 Initiated work on the Visual Ontology Modeler and ontology brokering tools in parallel
- 2002 Beta VOM release, early customers involved in DARPA/DAML program and FIPA
- 2003 VOM 1.2 for Rational Rose release planned for Q2, port to XDE planned for release ~Q4

Common Ground



- **An ontology is a specification of a conceptualization.** – *Tom Gruber*
- **Knowledge engineering is the application of logic and ontology to the task of building computable models of some domain for some purpose.** – *John Sowa*
- **Knowledge representation means that knowledge is formalized in a symbolic form, that is, to find a symbolic expression that can be interpreted.** – *Klein and Methlie*
- **The Semantic Web is the abstract representation of data on the World Wide Web, based on the RDF standards and other standards to be defined. It is being developed by the W3C, in collaboration with a large number of researchers and industrial partners.** – *W3C*

An Ontology is ...



- An ontology specifies a rich description of the
 - Terminology, concepts, nomenclature
 - Properties explicitly defining concepts
 - Relations among concepts (hierarchical and lattice)
 - Rules distinguishing concepts, refining definitions and relations (constraints, restrictions, regular expressions)relevant to a particular domain or area of interest.

Core Applications Considered



- Ontologies provide a common vocabulary and definition of rules for use by independently developed services (*e.g.*, CORBA, web)
- Agreements among companies, organizations sharing common services can be made with regard to their usage and the meaning of relevant concepts can be expressed unambiguously
- By composing component ontologies, mapping ontologies to one another and brokering terminology among participating resources and services, independently developed systems, agents and services can work together to share information and processes consistently, accurately, and completely.
- Ontologies also facilitate conversations among agents to collect, process, fuse, and exchange information.
- Improve search accuracy by enabling contextual search using concept definitions and relations among them instead of/in addition to statistical relevance of keywords.

Component-Based Approach



- Since 1997, component-based ontology construction has been essential to our methodology and technology
- Key to interoperability solutions for RASSP – separation of interface vocabulary from domain vocabulary to limit breakage when CAD/CAM tools were retired, others upgraded or introduced
- Hierarchical approach to engineering and manufacturing process representation influenced layered, hierarchical approach to ontology modeling
- Need to consider configuration, version management for all process, metadata and ontology models for RASSP also influenced approach

Kinds of Ontologies



- An *upper* ontology defines base concepts supporting ontology development.
- A *domain, or classic*, ontology defines the terminology and concepts relevant to a particular topic or area of interest.
- A *process* ontology defines the inputs, outputs, constraints, relations, terms, and sequencing information relevant to business processes.
- An *interface* ontology defines the structure, content, messaging, and other restrictions relevant for a particular interface (*e.g.*, application programming interface (API), database access, scripting language).
- A *service* ontology defines a core set of constructs for describing the vocabularies and capabilities of services (grounding in CORBA/WSDL, support for Action Semantics, DAML-S)
- A *role* ontology defines terminology and concepts relevant for a particular end-user (person or consumer application).

Ontology Kind is currently implemented as a model property which will be used to further select top level ontology

Motivation for UML Approach



- Early potential customers showed interest in a UML approach
- Limited commercial tools for ontology development, none graphical
- Good fit with component and standards-based methodology, provides support for many KR constructs
- Small pool of experienced ontologists
- Large, growing population of UML engineers
- Need to make ontology modeling accessible to domain experts
- Growing interest from standards perspective (less motivating due to timing)

Metamodel Development Challenges



- KR research community has been relatively disconnected from OOAD and software engineering communities until very recently
- Representation languages continue to evolve independently, serve different purposes and communities - Knowledge Interchange Format (KIF), description logics (DLs), conceptual graphs (CGs), intelligent agent languages
- Mapping between KR concepts and software engineering concepts not always straightforward
- Granularity of ontologies and domain models varies greatly between organizations, applications

the Semantic Web



- Early 2002 – introduction to the DARPA/DAML effort by customers
- Created mapping to DAML+OIL and implemented code generation capabilities based on customer need
- Elected to maintain general approach with planned generation of KIF, conceptual graphs, DAML+OIL/OWL due to varying customer needs and planned support for additional logic capabilities (*e.g.*, modal logic for agent applications)
- Concern that OWL specification is immature, rule language has yet to be defined
- Lack of methodology or standards for ontology development, configuration management, registry support – for web services, ontology metadata management (instance data management, document linking), agent, or other applications

MOF/UML Modeling Challenges



- MOF only supports binary Associations
- Associations in MOF are limited to simple associations and cannot contain features
- UML Association Classes can contain features, such as attributes, but must have defined Association Ends and are restricted to classes
- KR requires associating additional information (slots, facets, rules) with *classes, relations, functions and individuals*
- Layering between classes and objects in MOF/UML is distinct, whereas for certain privileged individuals in an ontology, it may be desirable to include both classes and individuals on the same diagram
- UML Attributes are not first class citizens, and therefore not well suited to modeling slots (used for some facets)
- OCL may not be sufficiently expressive for rule implementation (*e.g., limited variable support*)

Tool-related Implementation Issues



Limitations in tool support impacted implementation of ontology modeler add-in, and therefore impacted the profile itself

- Certain capabilities defined in the UML specification, such as n-ary associations, association inheritance are not supported by tools
- Association Classes are not supported
- Limited API accessibility to attributes, model properties, other model elements
- Representation of objects (individuals), classes on the same drawing prohibited

Basic Approach



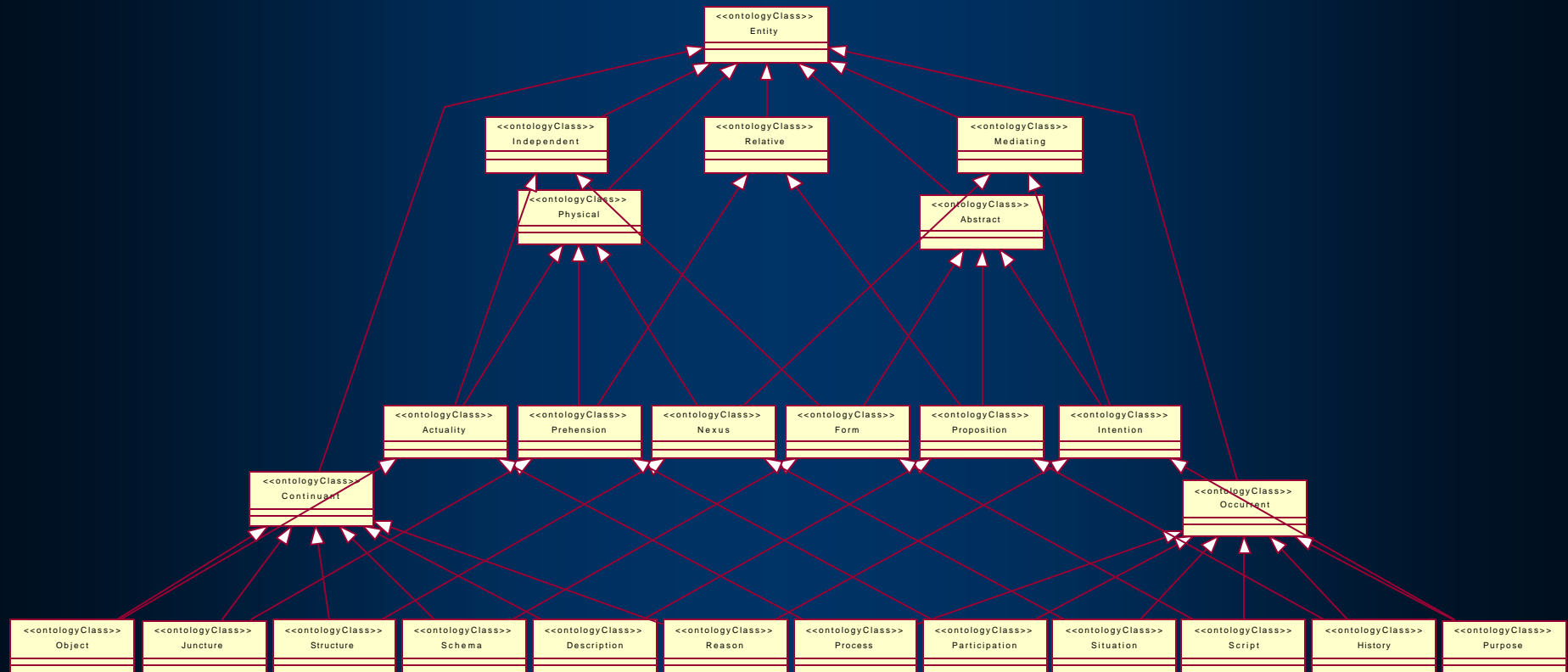
- Use of packages to emulate frames (class, relation and slot, function, individual)
 - Provides consistent mechanism for keeping details together (slots, facets, axioms, diagrams, DAML/OWL- related deployment details)
- Cleaving off ontologies and frames as separate components (controlled units) facilitates collaborative development, configuration management, ontology restructuring and maintenance
- Use of UML classes to represent KR model elements enables support for
 - Inheritance for relations, functions, slots, and individuals in addition to classes
 - n-ary relations
 - Relations among first class elements, including individuals, without requiring defined endpoints
 - Representation of all element types on the same diagram
 - Incomplete definition of individuals (common in KR)
 - Argument overloading for relations and functions

Primary Profile Constructs



Frame-Based KR (Ontology) Element	MOF Metamodel Element(s)	UML Metamodel Element(s)	Stereotype
Ontology	Package	Package	ontology
Class (frame)	Package	Package	classFrame
	Class	Class	ontologyClass
Relation (frame)	Package	Package	relationFrame
	Class	Class	relation
Function (frame)	Package	Package	functionFrame
	Class	Class	function
	Association	Association	hasFunction
	Operation	Operation	function
Individual (frame)	Package	Package	individualFrame
	Class	Class	individual
	Association	Association	individualOf, typeOf
Slot (frame)	Package	Package	slotFrame
	Class	Class	slotRelation
	Association	Association	hasOwnSlot, hasTemplateSlot
Facet	Association	Association	valueType, hasValue, hasDefaultValue
	Attribute	Attribute	facet
Axiom	Class	Class	axiom
	Association	Association	-- depends on the axiom
	Operation	Operation, External File	axiom

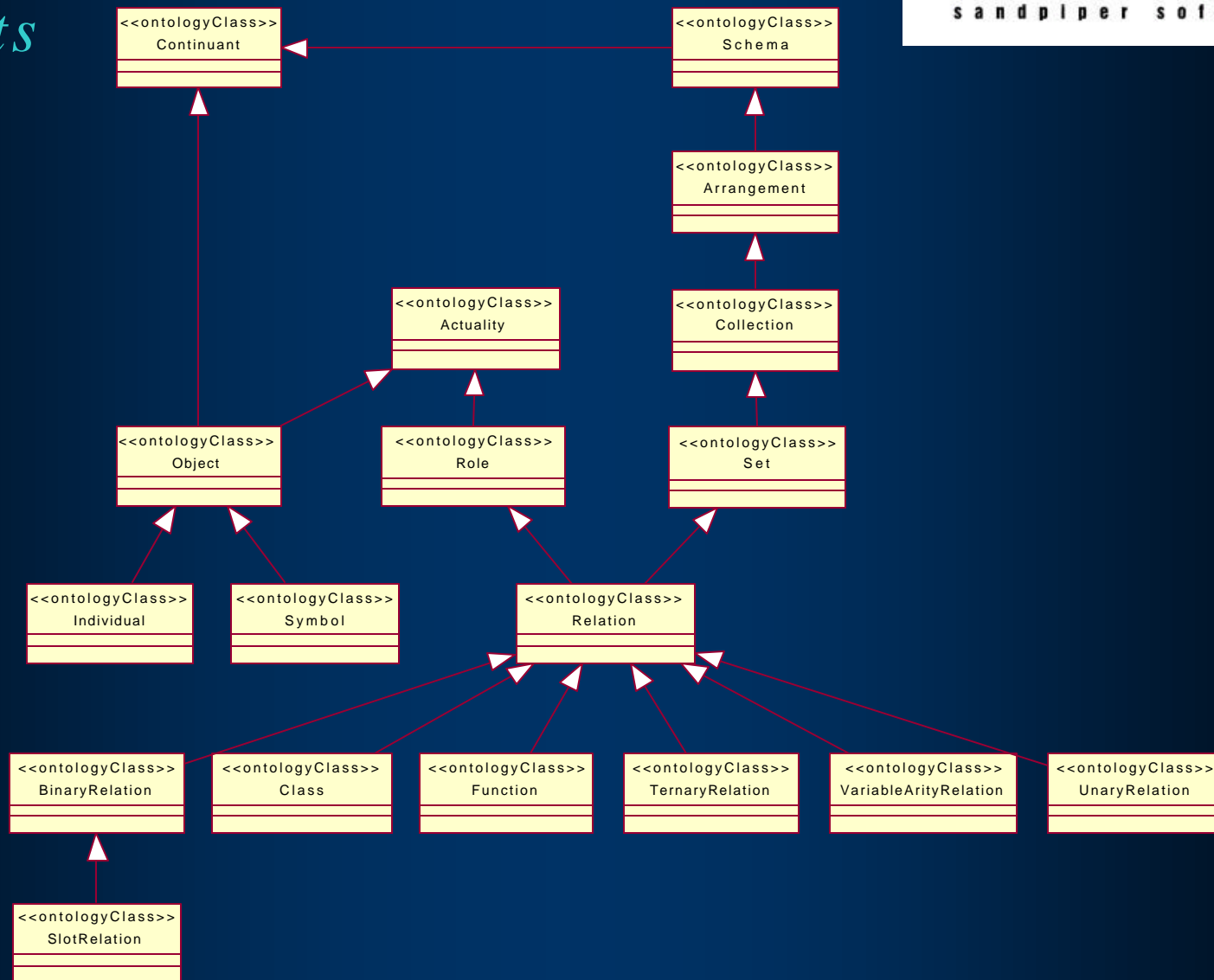
Top-Level Metamodel



References:

- Knowledge Representation: Logical, Philosophical, and Computational Foundations, John F. Sowa
- ISO/JTC1/SC32/WG2 Foundations of Logical Languages -- Part 1: Knowledge Interchange Form (draft)
- IEEE Standard Upper Ontology (Standard Upper Merged Ontology) -- <http://ontology.teknowledge.com/>

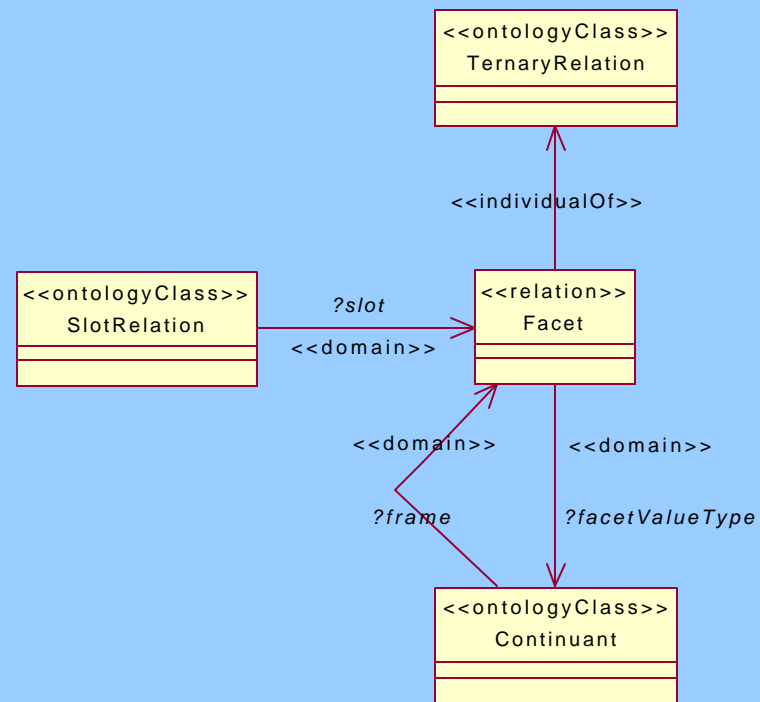
Metamodel Fragment for Profile Elements



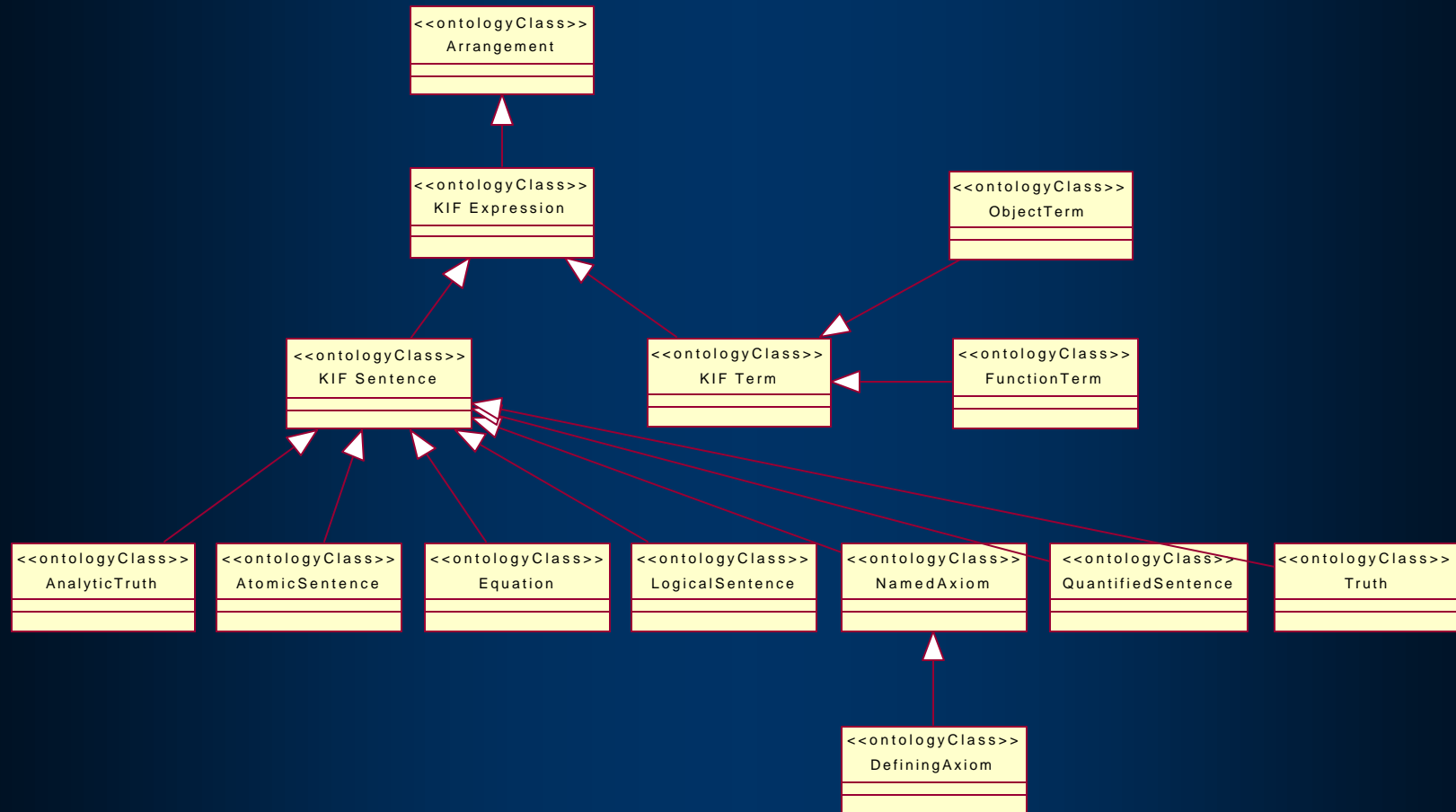
References:

- Knowledge Representation: Logical, Philosophical, and Computational Foundations, John F. Sowa
- *Ontolingua* (<http://ksl.stanford.edu>), Knowledge Interchange Format, Version 3.0, Reference Manual

Metamodel Definition of a Facet



Metamodel for KIF Expressions



References:

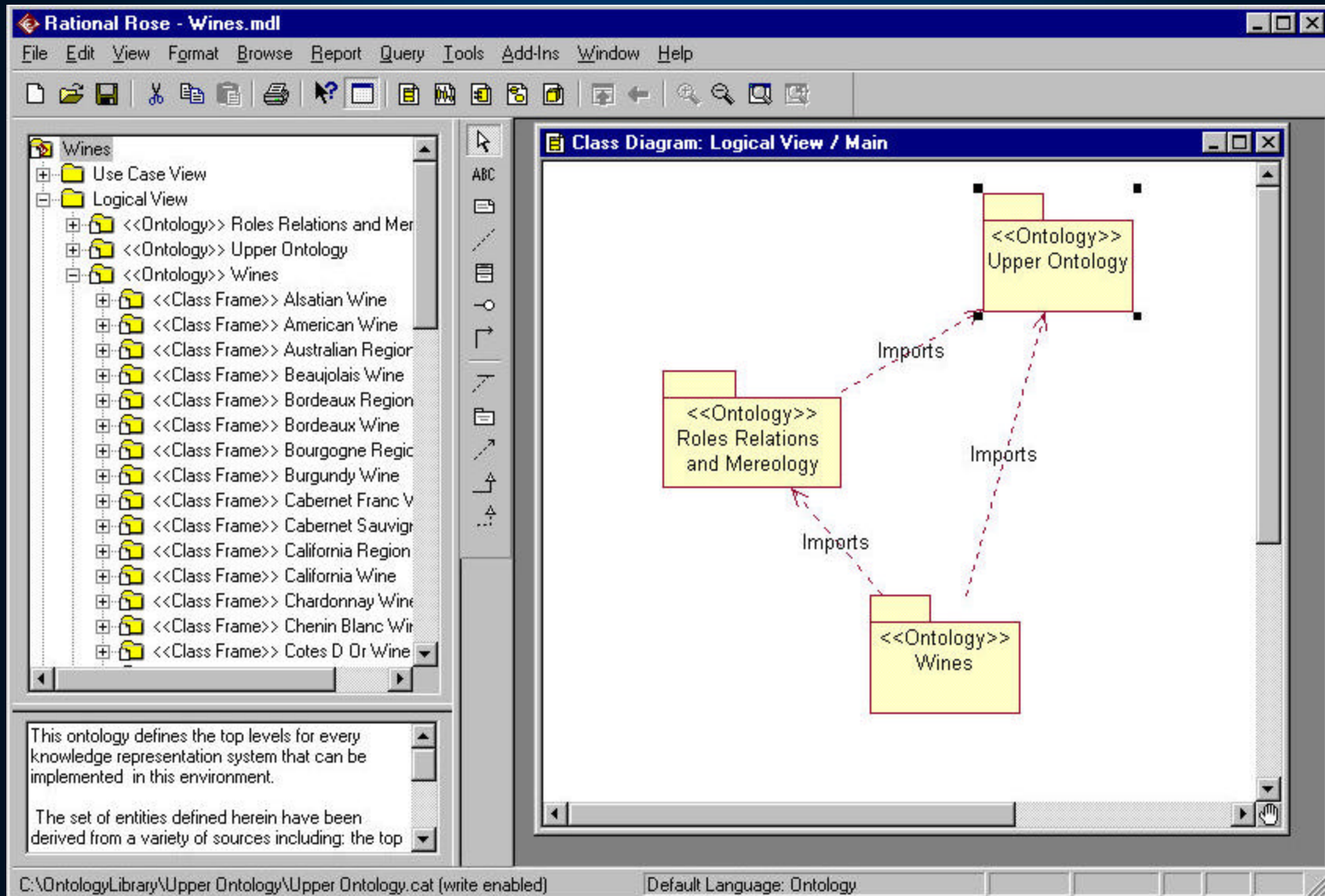
- ISO/JTC1/SC32/WG2 Foundations of Logical Languages -- Part 1: Knowledge Interchange Form (draft)
- Knowledge Interchange Format, Version 3.0, Reference Manual

Select DAML/OWL Features



DAML/OWL Logical Relation or Element	MOF Metamodel Element(s)	UML Metamodel Element(s)	Stereotype
complementOf	Association	Association	complementOf
disjointWith	Class Association	Class Association	disjointWith disjunct
domain	Association	Association	domain
hasValue	Association	Association	hasValue
import (ontology)		Dependency	<none used>
intersectionOf (frame)	Package Class	Package Class	intersectionOfFrame intersectionOf
inverseOf	Association	Association	inverseOf
Ontology		Property	<none used>
range	Association	Association	range
sameClassAs	Association	Association	sameClassAs
sameIndividualAs	Association	Association	sameIndividualAs
samePropertyAs	Association	Association	samePropertyAs
subClassOf (RDFS)		inheritance relation	<none used>
subPropertyOf (RDFS)		inheritance relation	<none used>
unionOf (frame)	Package Class	Package Class	unionOfFrame unionOf
versionInfo		Property	<none used>

Example: Import Dependencies



Example: *subPropertyOf*, *inverseOf*

