Representing Ontologies, Taxonomies, & Business Vocabularies in UML:
Best Practices & Lessons Learned of a Working Ontologist

Elisa Kendall
Thematix Partners LLC
No Magic World, 6 May 2014
Mars was photographed by the Hubble Space Telescope in August 2003 as the planet passed closer to Earth than it had in nearly 60,000 years. Image Credit: NASA, J. Bell (Cornell U.) and M. Wolff (SSI)

A sunset on Mars creates a glow due to the presence of tiny dust particles in the atmosphere. This photo is a combination of four images taken by Mars Pathfinder, which landed on Mars in 1997. Image credit: NASA/JPL

Recent images from instruments on board the Mars Reconnaissance Orbiter take much more detailed, narrower views of specific features of the Martian surface. Image credit: NASA/JPL

The Planetary Data Store (PDS) is a distributed repository of 40+ years’ imagery & data taken by a range of instruments on many diverse missions, available for scientific research.
Provenance/sources for tracking family members in the 19th century include early census data (often error prone), military records, passenger & immigration lists, online documents (e.g., county histories, church histories, etc.)

- **Historical/forensic research requires cross-domain search of a wide variety of resources within a given geo-spatial/temporal context**
- **Similar capabilities are essential for business intelligence, law enforcement, government applications – all require terminology reconciliation**
Merchandising

<table>
<thead>
<tr>
<th>Attribute 1</th>
<th>Attribute 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leg Room</td>
<td></td>
</tr>
<tr>
<td>Attribute 4</td>
<td>Attribute 5</td>
</tr>
<tr>
<td>Attribute 5</td>
<td></td>
</tr>
<tr>
<td>On-Off Access</td>
<td>Attribute 7</td>
</tr>
<tr>
<td>Attribute 8</td>
<td>Attribute 9</td>
</tr>
<tr>
<td>Media Options</td>
<td></td>
</tr>
<tr>
<td>Attribute 11</td>
<td>Attribute 12</td>
</tr>
<tr>
<td>Meal Options</td>
<td></td>
</tr>
<tr>
<td>Attribute 14</td>
<td>Attribute 15</td>
</tr>
<tr>
<td>Time of Day</td>
<td></td>
</tr>
<tr>
<td>Attribute 17</td>
<td>Attribute 18</td>
</tr>
<tr>
<td>Distance to Gate</td>
<td>Attribute 20</td>
</tr>
<tr>
<td>Attribute 21</td>
<td></td>
</tr>
<tr>
<td>Flight Characteristics</td>
<td>Attribute N</td>
</tr>
</tbody>
</table>

- Google, Yahoo!, Microsoft and other smaller search vendors have invested millions in schema.org, which is expanding daily
- More than 15% of websites now include schema-based mark-up
- Google’s Knowledge Graph group is also growing, gaining traction in a number of domains
Search Engine Optimization (SEO)

- Use search engines as a front door to transaction
- Dramatically increase relevance by inclusion of more meaningful, synonymic tags
- Enrich search results, adding ratings, video, prices, avail, amenities, locality etc.
Per FSB and Basel, global SIFIs must comply with Risk Data Aggregation (RDA) requirements by early 2016

- A bank should establish integrated data taxonomies across the banking group, which includes information on the characteristics of the data (metadata)
- Risk data must be complete and captured/aggregated across the enterprise
- Risk data must be accurate and the firm must be able to reconcile/validate reports

-- courtesy David Newman, Wells Fargo, & Mike Bennett, EDM Council
Common challenges for institutions

Current State of Business Data
- Data incongruity and fragmentation often found across silos
- Limited data standards
- Data rationalization problems
- Costly application program logic required to process data into concepts
- Brittle schemas are costly to change
- Rigid and limited taxonomies

Desired State of Business Data
- Data linkage and integration *despite* silos
- Open global *reusable* data standards
- Alignment based on *meaning*
- Highly expressive data schemas with built in *rules* that reflect *concepts*
- Flexible changeable schemas
- Rich multi-level taxonomies

-- courtesy David Newman, Wells Fargo, & Mike Bennett, EDM Council
Key use cases, work streams in Smart Regulation at OMG

Q4, 2012-Q1, 2013
Q2, 2013
Q3, 2013
Q4, 2013...

Active Business Use Cases, POCs

- **Front Running** (define the business challenge in implementing Front-running requirements, identify opportunity to apply SmartReg guidelines, recommended standards and tools, conduct POC etc.)
- **Open Symbology of Financial Instruments**: Proposal by Bloomberg to develop OMG standard for Financial Instrument Identifiers
- **Secondary Mortgage Market**: Leveraging mortgage industry standards, business natural language and decision modeling to comply with mortgage regulatory directives and data analytics
- **Capital Adequacy Proof of Concept**: Use of Financial Industry Regulatory Ontology for EU regulatory directives
- **Financial Industry Business Ontology (FIBO)**: Usage in Trading operations, support for LEI and business entity hierarchy analysis, beginning to support domain areas – Indicies & Indicators, Securities in 2014
- **SEC’s Consolidate Audit Trail (CAT)**: comprehensive system to track Equity and Options trading activity, a regulatory response to the "flash crash" of 2010 (opportunity to make CAT Smarter 😊)

Proposed Business Use cases, POCs

- **Financial Stability Board’s common data templates**: Counterparty exposure, systemic risk data to support Dodd-Frank; Basel Principles for Risk data aggregation (mapping of OMG, EDMC, other standards)
- **Data Act**: Accountability and transparency in Federal spending
- **AML, Cross-border investment flow fraud events/patterns modeling**
- **COREP, Call Reports...**
- **Innovation in ‘Makers to Consumers’ communication (smarter approach to comment letters?)**

Advocacy, Communication: Regulators, Private/Public Sector (ongoing)

--- courtesy Harsh Sharma, Citigroup

Copyright © 2014 Thematix Partners LLC
Historical Context

Knowledge Representation

- Cross-disciplinary field with historical roots in philosophy, linguistics, computer science, and cognitive science
- Goal is to represent the meaning of knowledge unambiguously, so that it can be understood, shared, and used by computational agents acting on behalf of people to accomplish some task

Philosophical origins

- Socrates questioning, Plato’s studies of epistemology – the nature of knowledge
- Aristotle’s shift to terminology, development of logic as a precise method for reasoning about knowledge
- Arguments for the existence of God dating back to Anselm of Canterbury
- Medieval theories of reference and of mental language, Scholastic logic
“Brain Cells for Grandmother”

- Neuroscientists continue to debate today how we store memories.
- One theory, documented as recently as this month in a Scientific American article, suggests that single neurons hold our memories, as concept cells.
- “Each concept – each person or thing in everyday experience – may have a set of corresponding neurons assigned to it.”
  - Rodrigo Quian Quiroga, Itzhak Fried and Christof Koch, February 2013 issue, Scientific American
- “…a relatively few neurons, numbering in the thousands or perhaps even less, constitute a “sparse” representation of an image.”
- “Our brain may use a small number of concept cells to represent many instances of one thing as a unique concept – a sparse and invariant representation … What is important is to grasp the gist of particular situations involving persons and concepts that are relevant to us, rather than remembering an overwhelming myriad of meaningless detail.”
- “The full recollection of a single memory episode requires links between different but associated concepts … If two concepts are related, some of the neurons encoding one concept may also fire to the other one.”
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

Artificial Intelligence can be viewed as the study of intelligent behavior achieved through computational means. Knowledge Representation then is the part of AI that is concerned with how an agent uses what it knows in deciding what to do. – Brachman and Levesque, KR&R

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 task of classifying all the words of language, or what's the same thing, all the ideas that seek expression, is the most stupendous of logical tasks. Anybody but the most accomplished logician must break down in it utterly; and even for the strongest man, it is the severest possible tax on the logical equipment and faculty. – Charles Sanders Peirce, letter to editor B. E. Smith of the Century Dictionary
What is an Ontology?

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.
Predicate logic is harder to read than the original English, but is more precise:

Every semi-trailer truck has at least 3 axles.

\[(\forall x)(((\text{SemiTrailerTruck}(x) \land (\exists y)(\text{SemiTrailer}(y) \land (\text{hasPart}(x,y)))) \land
(\text{SemiTrailerTruck}(x) \land (\exists z)(\text{TractorUnit}(z) \land (\text{hasPart}(x,z))))
\supset (\exists s)(\text{set}(s) \land (\text{count}(s, (\geq 3)))
\land (\forall w)(\text{member}(w,s) \supset (\text{Axle}(w) \land \text{hasPart}(x,w)))) \)].

Logic is a simple language with few basic symbols.

The level of detail depends on the choice of predicates – these predicates represent an ontology of the relevant concepts in the domain.

Different choices of predicates represent different ontological commitments.

Ontologies provide a *common vocabulary* for use by independently developed resources, processes, services.

*Agreements* among organizations sharing common services can be made with regard to their *usage*; the *meaning* of relevant concepts can be expressed *unambiguously*.

By *composing / mapping* ontologies and *mediating* terminology across participating events, resources and services, independently-developed services can work together to share information and processes consistently, accurately, and completely.

Ontologies also ensure:
- Valid conversations among agents to collect, process, fuse, and exchange information.
- Accurate searching by ensuring context using concept definitions and relations instead of/in addition to statistical relevance of keywords.
Analysis approaches

- Definitions range from high-level mind mapping and brainstorming ... to detailed collaboration, dialog, and information modeling to support knowledge sharing.

- Tools are equally diverse, from inexpensive brainstorming tools to sophisticated ontology and software model development environments.

- Common capabilities include:
  - “drawing a picture” that includes concepts and relationships between them.
  - Producing sharable artifacts, that vary depending on the tool – often including web sharable drawings.
Knowledge Representation / Management for Large Scale Applications

- Provide broad metadata, process, service & asset management facilities (including feedback/lessons learned…)
- Enable rich cross-domain, cross-process, cross organizational modeling supported by mapping & transformation services to provide maximum flexibility, interoperability
- Leverage standards and best practices in information architecture, metadata modeling, management, registration, and governance, and asset management & registration
- Provide incremental reasoning capabilities for model validation, transformation services

Repeatable, reusable, interoperable
Hypothetical Conceptual Model “EU-Rent”

- rental contract document
- request for pick-up
  - accepted request for pick-up
  - rejected request for pick-up
- contract
  - rental
    - rental period
    - rental duration
    - time unit: RTU
    - rental car
      - has possession of (rented car)
    - car movement
      - is assigned to
    - is responsible for
      - renter / customer (car rental responsibility)
    - optional extra
    - is manifested in
Classification techniques are as diverse as conceptual models; and generally include understanding:

- Level of Expressivity
- Level of Complexity / Structure
- Granularity
- Target Usage, Relevance
- Amount of Automation, Reasoning Requirements
- Prescriptive vs. Descriptive / Reliability / Level of Authoritativeness
- Design Methodology
- Governance
- Vocabulary Management, Metrics
Framework of dimensions

- Semantic Dimensions
  - **Expressiveness**: represents how well a KR language addresses increasingly complex semantics
  - **Structure**: represents how well an ontology encodes semantics, with the same or less expressivity than the KR language
  - **Granularity**: represents the level of detail specified in an ontology

- Pragmatic Dimensions
  - **Intended use**: the original use case(es), or purpose for developing a particular ontology
  - **Automated reasoning**: the extent to which the ontology is designed to be used for automated reasoning
  - **Prescriptive vs. Descriptive**: the extent to which an ontology was intended to be used for descriptive purposes vs. normative prescriptive use (i.e., with high degree of concern for correctness)

Considerations

- Intended use of ontologies, including domain requirements (e.g., scientific and engineering apps require formulas, units of measure, computations that may be challenging to represent)
- Intended use of KRSs that implement them, including reasoning requirements, questions to be answered
- For distributed environments, the number and kinds of resources, processes, services requiring ontologies – how distributed, how unique, developed collaboratively or independently, dynamic community participation or static
- What kinds of transformations are required among processes, resources, services to support semantic mediation
- Ontology and KRS alignment / de-confliction / ambiguity resolution requirements
- Ontology and KRS composition requirements, dynamic vs. static composition, in what environment and under what constraints
- Performance, sizing, timing requirements of target environment
Requirements, domain & use case analysis are critical
- Develop initial source/reference material
- Focus on system or application requirements
- Iterative development starting with a “thread” that covers basic capabilities can ground the work and prioritize decisions

Need to understand and communicate
- Architectural trade-offs, cost & technical benefits
- The nature of the information & kinds of questions that need to be answered drive the architecture, approach, and ontology scoping and design

Use discipline from formal domain analysis and use case development in UML to
- document and explain requirements
- identify requisite information sources and ontologies needed
- limit scope creep

Reuse standards and well-tested, available ontologies whenever possible

Copyright © 2014 Thematix Partners LLC
What to look for

- A controlled vocabulary
  
  FAA & IATA airport codes, ACRISS car codes, …

- A hierarchical or taxonomic structure (for query expansion)
  
  Vehicle, Ground-based Vehicle, Wheeled Vehicle, Powered Vehicle, Automobile, Sedan…

- Knowledge supporting structured queries
  
  Find all available hybrid SUVs or hybrid sedans that can seat four adults within a reasonable taxi ride of Planet Hollywood, Las Vegas for 3-4 days the week of June 20th

- Efficient inference (i.e., limited expressive power) vs. increased expressivity (potentially expensive or resource bounded computation)

- Custom reasoning for temporal relations, geospatial, dynamic algorithm / equation evaluation, process-specific, conditional operations

- Computational tractability
Using use cases to gather requirements

- A good summary for every use case should include:
  - A description of the basic business requirement / need the use case is intended to support
  - Primary goals
  - Scope – identify any known boundaries as a starting point
  - Pre-conditions and post conditions – any assumptions you know about the state of the “system”/world before and after
  - Actors and interfaces – identify primary actors, information sources, interfaces to existing or new systems
  - Triggers – what kicks off the use case, any particular series of events, situation, activity, etc., and any that affect the flow
  - Performance requirements – including any sizing or timing constraints, “ilities”, etc.

- Outline the major process steps for both the “normal”, or primary scenario, and alternative flows, such as if things don’t go well

- Use case and activity diagrams – typically done in UML, but could be visio, power point, or whatever tools your team is comfortable with

- Usage scenarios – you should have at least two narrative “stories” that describe how one of the main actors would experience the use case, with the intent of identifying additional requirements

- Competency Questions – identify as many of the questions you want the ontology / knowledge base to answer as possible

- Resources – describe any known contributing knowledge bases, other external resources that will be participating in the use case to the degree possible
Start with canonical definitions

- General concepts as well as domain-specific knowledge
- Basic starting point – cross-domain definitions
  - Namespace definitions, metadata, naming conventions, governance policies
  - Commonly used structures & vocabularies, such as domain-specific vocabulary & messaging standards, international country & language codes (ISO), national postal addressing or other government standards, industry best practices
  - Common metadata for ontology & schema management (e.g., Dublin Core, for documents & models, ISO 1087 for synonyms & similar relations, MIME media types for images & multimedia, etc.)
- Domain vocabularies must be prioritized, selected based on business requirements, clear ROI
- Common early targets include
  - Smart search (pull); customer experience & cross-sell / upsell (push)
  - Richer interoperability among trading partners
  - Service registration, description, discovery & management
  - Asset/artifact repository search & retrieval
  - Automated verification
Capturing definitions

- Layout a high-level architecture for key ontologies and ontology elements
- Identify the relationships among elements – roles, domain, interface, process, utility
- Define an approach for gathering content from subject matter experts, possibly based on IDEF5 (Integrated Definition Methods) Ontology Capture Method Analysis, that includes
  - Understanding and documenting source materials
  - An interview template
  - Traceability back to your use cases
- For each ontology element
  - Describe its domain and scope, how it will be used
  - Identify example questions and anticipated/sample answers for the application(s) it will support
  - Identify key stakeholders, ownership, maintenance, resources for instance knowledge
  - Describe anticipated reuse/evolution path
  - Identify critical standards, resources that it must interoperate with, dependencies
- Resources
Terminology analysis

- ISO 704, Principles & Methods for Terminology Work, provides a methodology for describing concepts & terms
  - Uses ISO 1087 for terminology
  - Uses ISO 860 for terminology “harmonization” (alignment) methods
  - Basis for typical methods used for taxonomy development today
- Describes how to flesh out definitions
- Recommendations strategies for relating terms to one another using standard vocabulary
- ISO 1087 – great resource for language to describe kinds of relationships, acronyms & other designations, preferred vs. deprecated terms, etc.
- ISO 860 augments this with recommendations for vocabulary comparison
ISO 1087 terminology organization
Key methodology issues

- Naming conventions and versioning policies are critical for every organization
  - Namespace definitions for ontologies are critical, along with policies for their management that are well understood
  - `http://<authority>/<subdomain>/<topic>/<date, in YYYYMMDD form>/filename.extension` is common practice in some organizations, with use of GRDDL to support content negotiation to the latest version
  - Levels of hierarchy may be added for large organizations or modularized ontologies
  - Use “id.name.ext” or “ontology.name.ext” vs. “www.name.ext,” for the authority component of the URL is increasing in some communities
  - Namespace prefixes (abbreviations) for individual modules, especially where there are multiple modules, can be important
  - For model elements –
    - These vary by “content community” – data modelers often use underscores at word boundaries, spaces in names – which semantic web tools may not handle well, semantic web practitioners use camel case
    - Guidelines should be provided to development teams for naming classes or class-like elements (SBVR concepts), properties or property-like elements (SBVR roles, for example), individuals (objects in UML) so that conceptual models developed in UML or a UML profile can be exported consistently for reuse
Modularity

Guidance on modularity is limited

- When is *any* model “too big”?
- Can individual parts evolve independently, and if so, shouldn’t that dictate module boundaries?
- Can individual parts be used independently, and if so, shouldn’t that also dictate module boundaries?
- For ontologies, particularly OWL ontologies, individuals should be managed separately from “schema” to facilitate reasoning during development (*i.e.*, if you add an individual that creates a logical inconsistency, most reasoners won’t load the ontology at all)

Current practices in the semantic web community address modularity either statically or dynamically

- Static approaches include adherence to “DL-Safe” rules and suggestions in papers by Alan Rector (University of Manchester)
- Dynamic approaches include use of tools that can assist in determining whether or not an ontology is appropriately modularized, using reasoning to perform rewriting with respect to soundness and completeness for a given ontology
Separation of concerns

- Critical dimensions aid in determining module boundaries –
  - separate business-related content from technical detail
  - aspects of the business content, such as branding, from others describing distribution rights
  - separate disciplines into independent modules

- Other considerations include separation based on
  - back-end store / source repositories
  - application boundaries, system interfaces
  - distributed resources
  - the need to reason over some parts of the knowledge base but not others to answer sets of critical questions
  - performance requirements, for reasoning, query answering, etc.
  - asserted vs. inferred content
Best practices in namespace development

- **Availability** – people should be able to retrieve a description about the resource identified by the URI from the network (albeit internally)

- **Understandability** – there should be no confusion between identifiers for networked documents and identifiers for other resources
  - URIs should be unambiguous
  - URIs are meant to identify only one of them, so one URI can't stand for both a networked document and a real-world object
  - Separation of concerns in modeling “subjects” or “topics” and the objects in the real world they characterize is critical, and has serious implications for designing and reasoning about resources

- **Simplicity** – short, mnemonic URIs will not break as easily when shared for collaborative purposes, and are typically easier to remember

- **Persistence** – once a URI has been established to identify a particular resource, it should be stable and persist as long as possible
  - Exclude references to implementation strategies, as technologies change over time (e.g., do not use ‘.php’ or ‘.asp’ as part of the URI scheme), and organization lifetime may be significantly shorter than that of the resource

- **Manageability** – given that URIs are intended to persist, administration issues should be limited to the degree possible
  - Some strategies include inserting the current year or date in the path so that URI schemes can evolve over time without breaking older URIs
  - Create an internal organization responsible for issuing and managing URIs, and corresponding namespace prefixes
Best practices in publishing vocabularies

- Provide readable documentation, about the vocabulary or model, including the terms, their definitions, proper usage patterns and scenarios, and clear examples

- Articulate your maintenance policies, so that those depending on the information model can understand how stable it is, how to provide feedback to its stewards, and so forth

- Identify versions and allocate URIs to the unique information models so that they can be referenced
  - Every version of a defined or imported XML schema module, information model, document, etc., aside from locally defined, internal modules, must have a unique namespace.
  - This allows us to align schema versioning with namespace versioning, as appropriate for some applications.

- Publish the formal schema (or ontology)

- Provide a means for your community of users to provide feedback
Metadata should be standardized at both the model level and the element level for every model (not just ontologies)

- Model level metadata can reuse properties from the Dublin Core Metadata Terms, from the Simple Knowledge Organization System (SKOS), ISO 11179 Metadata Registry standard with ISO 1087 Terminology support, emerging W3C work on provenance and others.

- Most annotations should be optional at the element level, but a minimal set, including names, labels, and formal, text definitions, is important for reusability & collaboration.

- Consistent use of the same annotations (properties, tags) improves readability, facilitates automated documentation generation, and enables better search over ontology repositories.

- Model level metadata may reuse organization-specific taxonomies to enable better search through RDFa tagging, for example.


- Specifications up for review at the OMG March Technical Meeting, in Reston, including the Information Exchange Framework (IEF) Packaging Policy Vocabulary (IEPPV) and Financial Industry Business Ontology (FIBO), use this.
Change management

Change management and traceability is not well defined at the element level for ontologies, still a research topic

- Approach to element-level versioning to support reasoning is to track two parallel streams: one for additions to the ontology, one for retractions; often managed as two separate files

- MOF versioning suggests linking to a workspace; use of a default workspace would get the latest version, and tools such as MagicDraw© can provide an indication of the differences

- MOF versioning does not support reasoning about the differences so that users can determine the impact of applying the changes, which is frequently required in the semantic web community –
  - addition or deletion of axioms can change downstream reasoning results, especially where there are complex dependencies

- Mechanisms that preserve the versioning detail in generated artifacts, such as RDF/XML serialized OWL, are essential
  - Consider the use cases/justification for whatever level of versioning detail is needed on a project by project basis
  - Balance with usability/performance
A little review:

*Every semi-trailer truck has at least 3 axles.*

\[
(\forall x)(((\text{SemiTrailerTruck}(x) \land (\exists y)(\text{SemiTrailer}(y) \land (\text{hasPart}(x,y)))) \land \\
(\text{SemiTrailerTruck}(x) \land (\exists z)(\text{TractorUnit}(z) \land (\text{hasPart}(x,z)))) \\
\supset (\exists s)(\text{set}(s) \land (\text{count}(s,(\geq 3))) \\
\land (\forall w)(\text{member}(w,s) \supset (\text{Axle}(w) \land \text{hasPart}(x,w)))) )).
\]
XML angle brackets can be equally difficult to read.
Editors such as Protégé are better ...
As complexity increases, it can be difficult to follow relationships ...
And it’s worse with individuals
UML's well-known graphical notation is more accessible to many
A little more background …

- UML provides a graphical notation, but OWL & UML are very different languages
  
  - Language mapping from OWL to UML only covers a fraction of the UML language – to logical (class) diagrams
  
  - Key distinctions:
    - Properties in OWL are first-class citizens, second class in UML (meaning, it’s difficult to map OWL properties directly to UML properties or associations)
    - UML supports n-ary relations whereas in OWL, properties are typically binary
    - OWL uses true set theoretic concepts (intersection, union, complement, etc.), where UML has historically been less formal
    - Although UML 2.5 semantics are closer to OWL, many practitioners don’t use the set theoretic features

This is overly simplified – the mapping is not straightforward, but the benefits of having a graphical notation are acknowledged in the W3C OWL 2 community.
UML/MOF and KR Together

- MOF technology streamlines the mechanics of managing models as XML documents, Java objects, CORBA objects

- Knowledge Representation supports reasoning about resources
  - Supports semantic alignment among differing vocabularies and nomenclatures
  - Enables consistency checking and model validation, business rule analysis
  - Allows us to ask questions over multiple resources that we could not answer previously
  - Enables policy-driven applications to leverage existing knowledge and policies to solve business problems
    - Detect inconsistent financial transactions
    - Support business policy enforcement
    - Facilitate next generation network management and security applications while integrating with existing RDBMS and OLAP data stores

- MOF provides no help with reasoning

- KR is not focused on the mechanics of managing models or metadata

- Complementary technologies – despite some overlap
Ontology Definition Metamodel (ODM)


- A family of metamodels & profiles that enable model interchange, ontology development in UML 2

- Grounded in formal logic enabling reasoning engines to understand, validate, and apply ontologies developed using the ODM

- A number of key ontologies are being standardized at OMG
  - EDMC-FIBO – Financial Industry Business Ontology, includes coverage of critical concepts in describing financial contracts and instruments
  - FIGI – Financial Instrument Global Identifier, provides unique identifiers for instruments, mainly exchange traded, bonds, commodities, etc.
  - IEF/IEPPV – Information Exchange Framework and its Policy Vocabulary
  - VTW – Vocabulary for Terminology Work (ISO 1087) and MDR – Metadata Registries and Repositories (ISO 11179)
  - DTV – the Date Time Vocabulary, already an OMG standard, is being revised and the OWL ontology will be incorporated into the RTF’s next revision of the standard

- Mappings to other OMG standards are either in work or under consideration, including BPMN and the Business Motivation Model, Records Management, SysML to support ontologies in MBSE applications, etc.
Resource Description Framework (RDF)

- Describes relationships
- Uses URIs used for naming
- Language has
  - graph based model
  - RDF/XML serialization (exchange syntax)

- Specification, W3C presentations, tools are available at
  - Semantic Web: [http://www.w3.org/standards/semanticweb/](http://www.w3.org/standards/semanticweb/)
  - Linked Data: [http://www.w3.org/standards/semanticweb/data](http://www.w3.org/standards/semanticweb/data)
  - RDF: [http://www.w3.org/standards/techs/rdf#w3c_all](http://www.w3.org/standards/techs/rdf#w3c_all)

- Recent revisions to RDF include:
  - RDF 1.1 – cleans up a number of issues in the earlier specification, adds support for RDF “sources”, such as SPARQL endpoints, and “datasets”
  - New serialization standards – RDF 1.1 Turtle, RDF 1.1 N-Quads, in addition to RDF/XML
RDF Notation Options

Graph:

http://www.umlconference.com/NMW2014/NoMagicWorld.owl#Conference

http://www.umlconference.com/NMW2014/NoMagicWorld.owl#CourtyardDallasAllen

XML/RDF:

```xml
<rdf:Description rdf:ID="Conference"/>
<nmw:heldAt rdf:resource="#CourtyardDallasAllen"/>
</rdf:Description>
```

N3:

```
nmw:Conference  nmw:heldAt  nmw:CourtyardDallasAllen .
```
RDF Schema (RDFS)

- An RDF vocabulary that provides for identifying:
  - classes,
  - subsumption (inheritance) relations for classes,
  - subsumption (inheritance) relations for properties,
  - domain and range for properties
**RDF Schema (RDFS)**

**XML/RDF:**

```xml
<rdf:Description rdf:ID="Hotel">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<nmw:ConferenceHotel rdf:ID="nmw:ConferenceHotel">
  <rdfs:subClassOf rdf:resource="#Hotel"/>
</nmw:ConferenceHotel>

<nmw:CourtyardDallasAllen rdf:ID="nmw:CourtyardDallasAllen"/>
```
The Web Ontology Language (OWL)

- Two languages emerged in parallel to address semantic web requirements
  - DAML-ONT, supported by the DARPA/DAML program
  - OIL (Ontology Inference Layer) developed by EU & US researchers
- Merged DAML+OIL was submitted to the W3C 2002, formed the basis for the WebOnt Working Group
- OWL extends RDF Schema
  - Has an RDFS based syntax and reuses some RDF vocabulary (e.g., subClassOf, domain, range)
  - Adds rich primitives and redefines others (transitivity, inverse, cardinality constraints, complex class definitions)
- Describes the structure of a domain in terms of classes and properties
- Uses RDFS for class/property membership assertions (ground facts), XML Schema Datatypes
- OWL specifications became W3C recommendations in February 2004
- OWL 2 specifications was adopted in October 2009, with minor revisions in December 2012
Class inheritance

- Classes are organized into subclass-superclass (or generalization-specialization) hierarchies

- True subclass relationships are the basis of a formal *is-a* hierarchy
  - Classes are “is-a” related if an instance of the subclass is an instance of the superclass
  - *is-a* hierarchies are essential for classification
  - Violation of true *is-a* hierarchical relationships can have unintended consequences & and cause errors in reasoning

- Class expressions should be viewed as sets, and subclasses as a subset of the superclass, in contrast with a collection of attributes as classes are sometimes specified in object oriented programming

- Examples
  - FloweringPlantType is a subclass of PlantType
    
    *Every flowering plant is a plant; every instance of a flowering plant (e.g., Azalea indica 'Alaska' (Rutherfordiana hybrid) is an instance of a flowering plant*
  
  - Azalea is a subclass of FloweringPlantType, Rhododendron
  
  - Monrovia is a company that breeds, grows, and sells flowering plants

* courtesy Monrovia web site

Copyright © 2014 Thematix Partners LLC
Example class hierarchy with other expressions

- Example from ISO 1087 combining subclass-superclass and union expressions
- Provides a more accurate representation of the relationships defined in the ISO standard than a strict is-a hierarchy would
Example class hierarchy with other expressions
Modeling considerations for class hierarchy development

- Practitioners tend to start
  - Top-down - define the most general concepts first and then specialize them
  - Bottom-up - define the most specific concepts and then organize them into more general classes
  - Combination (typical – breadth at the top level and depth along a few branches to test design)

- Class inheritance is transitive
  - A is a subclass of B (white wine, dessert wine are subclasses of wine)
  - B is a subclass of C (sauvignon blanc is a subclass of white wine, late harvest wine is a subclass of dessert wine)
  - therefore A is a subclass of C (late harvest sauvignon blanc is a subclass of white wine, dessert wine, & wine)
  - Start with a more formal is-a approach, tease out whether other expressions are more appropriate based on use cases, formal definitions when available, etc.
Class axioms

- Subsumption (necessary)
  - $A \subseteq B$ where $B$ is a class description
    - partial or primitive class

- Definition (necessary and sufficient)
  - $C \equiv D$ where $D$ is a class description
    - complete or defined class

Courtesy Evan Wallace, NIST
Disjoint classes

- Classes are disjoint if they cannot have common instances.
- Disjoint classes cannot have any common subclasses.
- If winery and wine are disjoint, then there is no instance that is both a winery and a wine; there is no class that is both a subclass of winery and a subclass of wine.
- Disjointness is often used to aid consistency checking.
- Disjointness is also helpful in teasing out subtle distinctions among classes across multiple ontologies.
- Equivalence is also often used to identify the same concepts across ontologies that may be named differently, or to name classes defined through class axioms.
Properties

- Properties describe characteristics, features, or attributes of the members of a class

*Every flowering plant has a bloom color, color pattern, flower form, petal form, etc.*

- Classes of properties
  - **intrinsic**: properties of the plant itself such as the optimal sunlight, soil conditions, expected height, Sunset climate zone, and so forth
  - **extrinsic**: properties imposed externally such as the grower and price
  - whole-part relations
  - geospatial, mereonomic relations: what microclimates are this plant well suited to, where in a particular historic garden will you find it

- Data and object properties
  - simple attributes typically have primitive data values (e.g., strings, numbers)
  - complex properties refer to other entities (e.g., an individual grower, such as Monrovia, or organization such as the American Orchid Society)
  - OWL reasoning requires strict separation of data and object properties
Domain & range properties

- In OWL and many other KR languages, relations (properties) are strictly binary.

- The domain & range represent the source & target arguments, respectively, for the property.

- Domain – the class (or classes) that may have the property – *Wine is the domain of the property* hasWineColor.

- Range – the class (or classes) defining valid property values – everything that fulfills the hasWineColor property is an instance of the enumerated class {red, white, rose}.

- Some KR languages that inherently support n-ary relations, such as CL, do not make this distinction:
  - More flexible, intuitively more like mathematics, where functions have ranges (or return types) but not all relations are functions.
  - Requires additional relations to specify argument order, which can be critical for ontology alignment.
Class expressions that restrict property values

- Number restrictions describe or limit the number of possible values a particular property can have:
  - A language must be associated with at least one English name and at least one French name
  - A language may be associated with zero or more Indigenous names

- Cardinality – similar meaning to classical set theory, measures the number of elements in the set (restriction class):
  - Cardinality – cardinality $N$ means the class defined by the property restriction must have exactly $N$ values (individual or literal values)
  - Minimum cardinality - 1 means that there must be at least one value (required), 0 means that the value is optional
  - Maximum cardinality - 1 means that there can be at most one value (single-valued), $N$ means that there can be up to $N$ values ($N > 1$, multi-valued)
Simple number restriction examples

- Creating a class of individuals that have exactly one color & a class of individuals that have more than one color
- Note that the relationship between SingleColoredThing and the restriction class is modeled as an *equivalence relationship*, meaning that membership in the restriction class is a necessary and sufficient condition for being a SingleColoredThing.
OWL 2 provides the capability to further qualify cardinality by specific classes and data ranges.

Patterns that reuse a smaller number of significant properties, building up more complex class expressions that limit the set of valid values are common, useful in complex classification systems.
Qualified cardinality number restriction example

<owl:ObjectProperty rdf:about="&re;hasCharacteristic">
  <rdfs:range rdf:resource="&re;Characteristic"/>
</owl:ObjectProperty>

<owl:Class rdf:about="&re;BloomColor">
  <rdfs:subClassOf rdf:resource="&re;Characteristic"/>
  <rdfs:subClassOf rdf:resource="&re;Color"/>
</owl:Class>

<owl:Class rdf:about="&re;Characteristic"/>

<owl:Class rdf:about="&re;Color"/>

<owl:Class rdf:about="&re;FloweringPlantType">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&re;hasCharacteristic"/>
      <owl:onClass rdf:resource="&re;BloomColor"/>
      <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minQualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
Class expressions that restrict possible values by type

- Universal (allValuesFrom) and existential (someValuesFrom) quantification
  - allValuesFrom is the OWL equivalent for $\forall(x)$, and restricts the possible values for a property to members of a particular class or data range
  - someValuesFrom is the OWL equivalent for $\exists(x)$, and restricts the possible values for a property to at least one member of a particular class or data range

- Specific value (hasValue) restrictions
  - a single data value (e.g., the color property for a RedWine must be filled with the value “red”)
  - an individual member of a class (e.g., Winery is the value restriction on the hasMaker property on the class Wine)

- Enumerations – lists of allowable individuals or data elements

- Combinations of the above that include both restrictions and boolean class expressions (union – logical or, intersection – logical and, complement – logical not)
Azaleas are members of the set of flowering plants whose bloom color must be either an RHSColor (Royal Horticultural Society) or ASAColor (Azalea Society of America)
And the OWL for that ...

```xml
<owl:Class rdf:about="&re;Azalea">
  <rdfs:subClassOf rdf:resource="&re;FloweringPlantType"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&re;hasBloomColor"/>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <rdf:Description rdf:about="&re;ASAColor"/>
            <rdf:Description rdf:about="&re;RHSColor"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

* courtesy Azalea Society of America
Another typical pattern combining restrictions & boolean connectives

- Use of equivalence expressions like this, describing RHSFloweringPlantType as a flowering plant and something whose bloom colors must be from the set of colors defined by the Royal Horticultural Society is a common pattern.
And the corresponding OWL for that ...

```
<owl:Class rdf:about="&re;RHSFloweringPlantType">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="&re;FloweringPlantType"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="&re;hasBloomColor"/>
          <owl:allValuesFrom rdf:resource="&re;RHSColor"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```
Class expression restricting some & exact values for a given property
And the OWL for that ...

<owl:Class rdf:about="&re;Azalea">
   <rdfs:subClassOf rdf:resource="&re;FloweringPlantType"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="&re;hasBloomColor"/>
         <owl:allValuesFrom>
            <owl:Class>
               <owl:unionOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="&re;ASAColor"/>
                  <rdf:Description rdf:about="&re;RHSColor"/>
               </owl:unionOf>
            </owl:Class>
            <owl:allValuesFrom>
            </owl:Restriction>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="&re;hasBloomColorPattern"/>
            <owl:someValuesFrom rdf:resource="&re;ASAColorPattern"/>
         </owl:Restriction>
      </rdfs:subClassOf>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="&re;SingleColoredAzalea">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="&re;hasBloomColorPattern"/>
          <owl:hasValue rdf:resource="&re;Solid"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="&re;hasBloomColor"/>
          <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
    <rdfs:subClassOf rdf:resource="&re;Azalea"/>
  </owl:Class>
</owl:equivalentClass>
<owl:NamedIndividual rdf:about="&re;Solid">
  <rdf:type rdf:resource="&re;ColorPattern"/>
</owl:NamedIndividual>
Individuals and data ranges

- An individual (instance, object in other paradigms)
  - Any class that an individual is a member of, or is an individual of, is a type of the individual
  - Any superclass of a class is an ancestor of (or type of) the individual

- Specify property values for the individual
  - Property values should conform to the constraints such as range, value type, cardinality restrictions, etc.

-Enumerated classes & data ranges are commonly used to specify the complete set of valid values for a particular property
Example enumerated class of individuals
Aggregating datatype restrictions via an intersection to specify the average size of an Azalea in the landscape.
Limiting scope

- An ontology should not contain all the possible information about the domain
  - No need to specialize or generalize more than the application requires
  - No need to include all possible properties of a class
    - Only the most salient properties
    - Only the properties that the application requires

- Ontologies of wine, food, and their pairings probably will not include details such as:
  - Bottle size (half bottle, full bottle, magnum, …)
  - Label color
  - Wine bottle color (green, amber, …)
  - Individual plants and their location in a historic garden
  - **Azalea indica 'Alaska'** (Rutherfordiana hybrid) might be a class or might be an individual, depending on the use case and application requirements
Syntax checking

For RDF & OWL Ontologies

- RDF syntax checking, graph visualization
  - W3C RDF Validator (http://www.w3.org/RDF/Validator/)
  - Jena API & Toolkit (http://jena.sourceforge.net/)

- OWL syntax checking, OWL dialect determination
  - OWL Consistency Checker (http://clarkparsia.com/pellet/, also provides full OWL 2 DL reasoning capabilities)
  - OWL 2 Validator (Univ. of Manchester, http://owl.cs.manchester.ac.uk/validator/)
  - Protégé OWL (http://protege.stanford.edu/)
  - Jena API & Toolkit (http://jena.sourceforge.net/)

Every tool provides unique capabilities; sophisticated projects may require multiple approaches

Tools listed are open source, commercial options are also available
Consistency checking and analysis

- Requirements are typically application specific
- RDF vocabularies should be checked by an RDF validator or rule engine
- OWL Ontologies should be run through a consistency checking reasoner
  - Pellet (open source, originally from Mindswap, supported by Clark & Parsia, LLC) – [http://clarkparsia.com/pellet/](http://clarkparsia.com/pellet/)
  - FaCT++ – [http://owl.man.ac.uk/factplusplus/](http://owl.man.ac.uk/factplusplus/)
  - VIStology’s ConsVISor OWL Consistency checker – [http://68.162.250.6:8080/consvisor/](http://68.162.250.6:8080/consvisor/)
- OWL instance data may also be run through checking tools
  - TW Instance data evaluation - [http://onto.rpi.edu/demo/oie/](http://onto.rpi.edu/demo/oie/)
- OOPS! (OntOlogy Pitfall Scanner!) ontology engineering tool available from the School of Computer Science at Universidad Politécnica de Madrid, UPM – [http://oeg-lia3.dia.fi.upm.es/oops/index-content.jsp](http://oeg-lia3.dia.fi.upm.es/oops/index-content.jsp) can also provide helpful hints
- OntoClean methodology for ontology analysis – see [http://en.wikipedia.org/wiki/OntoClean](http://en.wikipedia.org/wiki/OntoClean) for links to the seminal papers on this
Ontologies should be checked to ensure consistency, limit potential for invalid conclusions.
A new ontology

Wine ontology

Case 1: semantic inconsistency caused by a new class.

Case 2: semantic inconsistency caused by a new instance.
Use Cases

- Call Center Operations to identify conflicting Service Advisories
- Intelligence Analysis - supporting research operations
- Semantically enhanced Fraud Detection
- IP Content Publication & Management for Media
Organization and Management of Data to Support Automation

Information Architecture Standards Framework

- Modeling Standards
  - Unified Modeling Language (UML)
  - Ontology Definition Metamodel (ODM)
  - Information Management Metamodel (IMM)
  - UML Profile & Metamodel For Services (SoapL)

- Metadata Standards
  - Dublin Core Metadata Elements
  - ISO 11179 Metadata Registration
  - ISO 19763 Model Registration

Metadata Architecture

Reusable Cross-Domain Models
- Requirements, Software, System Description Models, Service Models, Process/Workflow Models
- Namespace definitions, related metadata, registration, governance
- Commonly used structures & vocabularies: messaging, events, service description, process terminology
- Common metadata for asset management & reuse

Reusable Domain Models
- Range resource allocation, monitoring & analysis terminology - recipes, resources, usage metrics, etc.
- Test design terminology - recipes, malware, vulnerabilities, adversarial models, OpFor strategies, etc.
- Test execution terminology - mapping, monitoring, pausing/continuing, restarting, etc.

Domain Vocabulary Layer
- Targeted information model development, using domain and cross-domain vocabularies, enables metadata extraction, artifact generation, transformation services, search, navigation & repository models, for service discovery & monitoring, capture of lessons learned, etc.

Includes terms describing resources, configuration patterns, requirements, mission objectives, terms that are specific to particular tests
Model Driven Service Provisioning

GE Water and Process – InSight Remote Monitoring & Diagnostics application

- **Cross-industry** – Beverage, Chemical Processing, Commercial, Food, General Industrial, Hydrocarbon Processing, Life Sciences, Medical Dialysis, Microelectronics, Mining & Mineral Processing, Municipal, Pharmaceutical, Power, Primary Metals, Residential, Transportation

- **Cross-domain** – Boiler Water Treatment, Cooling Water Treatment, Feedstock Flexibility Solutions, Fuel Flexibility Solutions, Mobile Water, Petrochemical Process Solutions, Process Separations, Water Recovery, Water Scarcity Relief

- Common hardware, software, process foundational components

- Common goals for service offerings, provisioning, reporting across industries & solutions – what was done and why, how the services were accomplished, how did the customer benefit, metrics

- Solution includes conceptual information models representing concepts, services, processes, relationships across them, large domain-specific knowledge bases, web-based service deployment & data management

- Assists GE customers in reducing water, energy & labor costs, improves water and process treatment programs

- Within 6 months of deployment – over 200+ plants were licensed users
Visual Ontology Modeling is a new and valuable method for the development of commercial and scientific ontologies.

Multiple diagrammatic “points of view” simplify understanding and maintenance of large, complex ontologies.

It bridges the gap between the Semantic Web and the world of UML modeling.

Come see us at our booth, or contact us at sales@thematix.com