

# *The Role of Ontologies in Enterprise Modeling*

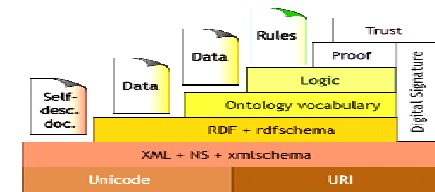
Prof. Dr. Colin Atkinson

University of Mannheim

VORTE 2005 Keynote Talk  
Enschede, 20 September 2005

# Automating and Connecting Enterprises

- two main visions of how enterprises can best be automated and connected :



## Model Driven Architecture

- “Platform independent” models (PIMs) of enterprise processes and information
- (semi) automatically transformed into executable, platform specific models
- allows domain experts to focus on describing salient domain and business properties

## Semantic Web

- formal representations of information, publishable and shareable on the web
- subject to automated checking and extension through “reasoning” engines
- allows domain experts to focus on describing salient domain and business properties

# *Different Histories – Common Future?*

- models (à la MDA) and ontologies (à la semantic web) have different heritages and initial goals
  - models originated from the software engineering world for the purpose of simplifying the description of software
  - ontologies originated from the artificial intelligence world for the purpose of precisely capturing knowledge
- BUT what does their future hold?
  - complete merger,
  - happy marriage,
  - uneasy truce,
  - all out war.
- If the title of this talk were “What is the role of models in Enterprise Modeling”
  - Would it, or should it, make a difference?



# *Some Common Accepted Wisdom*

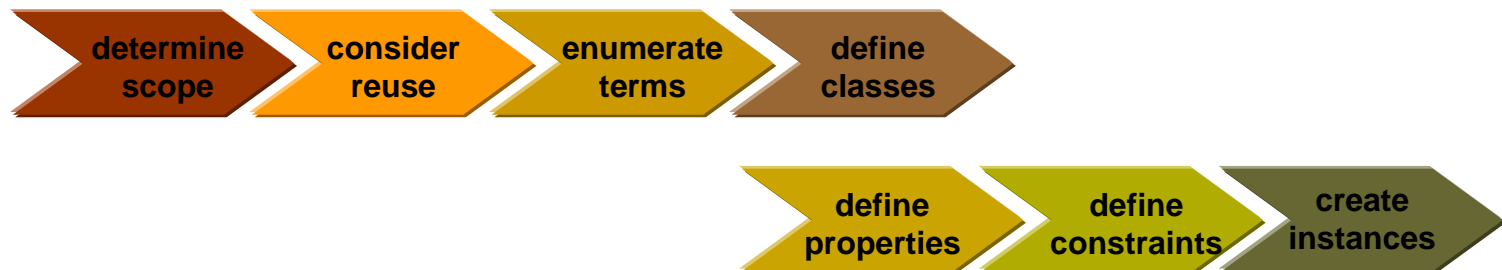
- Ontologies have formal semantics
- Ontologies are for knowledge representation
- Ontologies support reasoning
- (UML) models don't
- (UML) models aren't
- (UML) models don't

⇒ ontologies are "different" to models

- provides the rationale for recently "visions" such as
  - Ontology-Driven Software Development
  - Ontology-based MDA Framework
  - Ontology Driven Architectures
  - Ontology-based Modeling
  - Model Driven Semantic Web
  - .....

# A Guide to Creating your First Ontology (1/2)

- Highly influential report by Noy and McGuinness describing how to create a Wine Ontology
  - Step 1: Determine the domain and scope of the ontology
  - Step 2: consider reusing existing ontologies
  - Step 3: Enumerate important terms in the ontology
  - Step 4: Define the classes and the class hierarchy
  - Step 5: Define the properties of classes - slots
  - Step 6: Define the facets of the slots
  - Step 7: create instances



# A Guide to Creating your First Ontology (2/2)

- although the authors acknowledge that some of the ontology design ideas originated from object-oriented **design** (Booch and Rumbaugh) they make the following distinction :

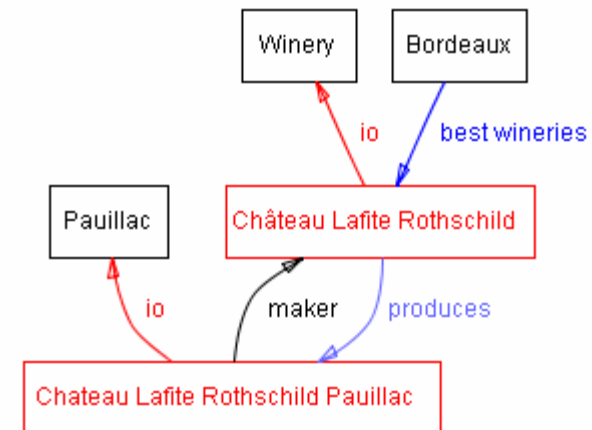
## Ontology development

- reflects structure of the world
- about structure of concepts
- actual physical representation is not an issue

## Object-oriented **programming**

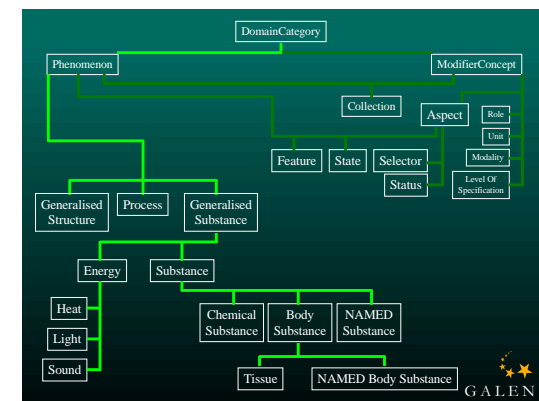
- reflects structure of data and code
- is usually about behavior (methods)
- describes physical representation of data (long int, char, etc.)

- the described process is 100% the same as that for creating a domain (conceptual) class diagram
- the stored information is 100% the same as that which would be stored in a domain class diagram



# What is an Ontology?

- “a theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system”  
(*Dictionary*)
- “An ontology is an explicit specification of a conceptualization”  
(*Tom Gruber*)
- “In information science, an ontology is the product of an attempt to formulate an exhaustive and rigorous conceptual schema about a domain”.  
(*Wikipedia*)
- An ontology is an explicit description of a domain in terms of:
  - concepts
  - properties and attributes of concepts
  - constraints on properties and attributes
  - individuals
- An ontology defines
  - a common vocabulary
  - a shared understanding



# What is a Model?

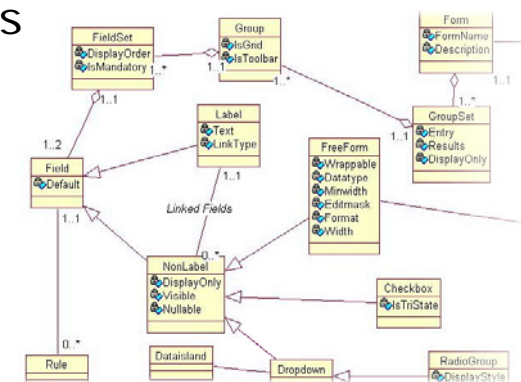
- "A model of a system is a description or specification of that system and its environment for some certain purpose"

*(MDA Guide)*

- "A domain model can be thought of as the conceptual model of the system.....The domain model is created to understand the key concept of the system and to familiarize with the vocabulary of the system "

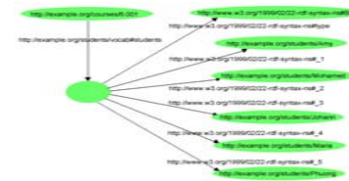
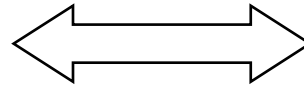
*(Wikipedia)*

- A domain model is an explicit description of a domain in terms of:
  - concepts
  - properties and attributes of concepts
  - constraints on properties and attributes
  - individuals
- A domain model defines
  - a common vocabulary
  - a shared understanding



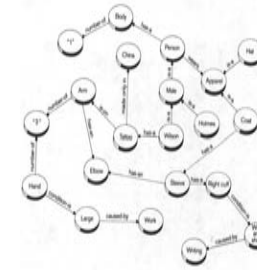
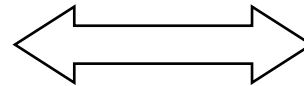
# Many Possible Levels of Comparison?

MDA



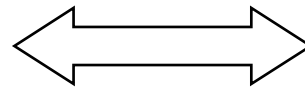
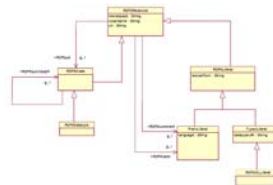
Semantic Web

Model



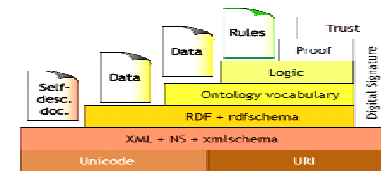
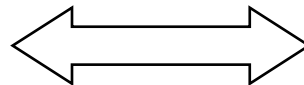
Ontology

UML



OWL

MDA  
Infra-  
structure



OWL/  
RDFS/  
RDF

# Three Schools of Thought

- UML (+OCL) **is** an ontology representation language
  - already can do everything that is needed
  - a UML model may be viewed as an ontology (depending on needed precision)
  
- UML is just a convenient syntax for ontologies
  - extend UML with necessary visualization features
  - at some point the represented information must be transformed into a “true” ORL such as OWL
  
- UML and OWL are, and will always be, different but a pragmatic interoperability solution using the UML/MOF’s extension mechanisms is desirable
  - University of Belgrade
  - University of Karlsruhe
  - Ontology Definition Metamodel (ODM) Partners

# Revised ODM Submission

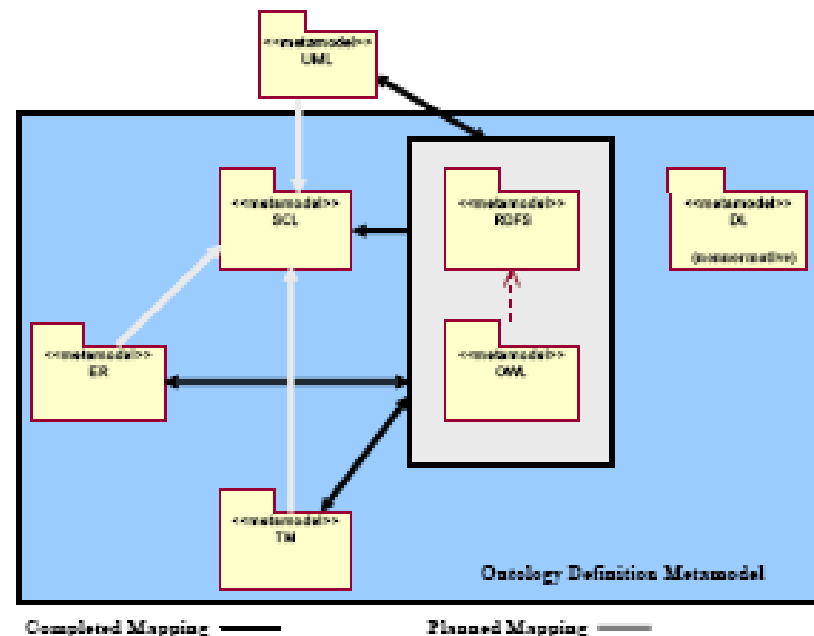
- collection of MOF metamodels and profiles for all the mainstream semantic web and data representation languages and their semantic formalisms

- Topic maps (TM)
- RDFS
- OWL Full
- ER Model
- Simple Common Logic
- Description logic

- collection of mappings between metamodels

- UML – OWL Full
- ER – OWL
- TM – Owl Full
- RDFS and OWL to SCL

- basis for interoperability between semantic web, MDA and database technologies

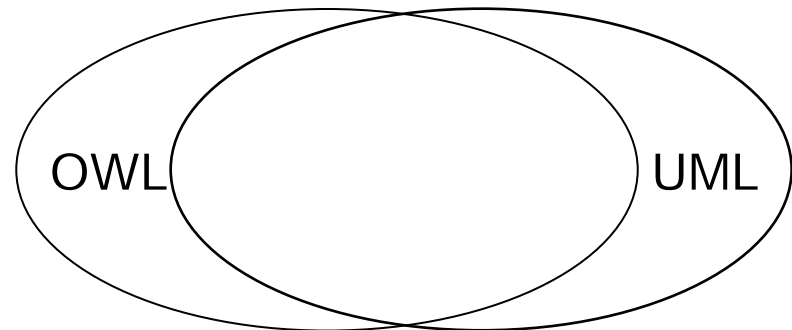


# Summary of Approach Characteristics

The approaches and their characteristics and feature fulfillment		Position of approach in paper (section reference)	Usability	Supported IRLs and interoperability	Use of UML-tool and language infrastructure	Utilization of MOF and MDA infrastructure	Easily translatable into (Semantic Web) ORLs	Instant access for business and application developers	Instant access for ontology engineers	Large experienced workforce	Access to OE-technology for modeling community	Access to MDA-technology for ontology engineering community
UML+OCL as ORL		2	+	-	+	+	o	+	-	+	+	o
UML-profile as modeling syntax		3	-	-	+	-	+	-	o	-	-	-
MOF-based abstract syntax and UML-based concrete syntax	Old ODM	4.1	-	o	+	+	+	-	+	-	-	o
	UML + OWL-Mix	4.2	o	o	+	+	+	o	o	o	o	+
	New ODM	4.3	o	+	+	+	+	-	o	o	+	+
Legend: + fully achieved, - not achieved, o partly achieved												

# Perceived Shortcomings of UML w.r.t OWL

- Support for synonyms
  - Extension equivalence
  - Sufficient conditions
  - Complex class constructors
  - Logical characterization of associations
  - Existential quantification
  - Value restriction
  - Global properties
  - Autonomous individuals
  - Class specific cardinality constraints
  - Universal concept "Thing"
  - Classes as objects
- BUT they all have "workarounds" with appropriate use of OCL
- far more similarities than differences

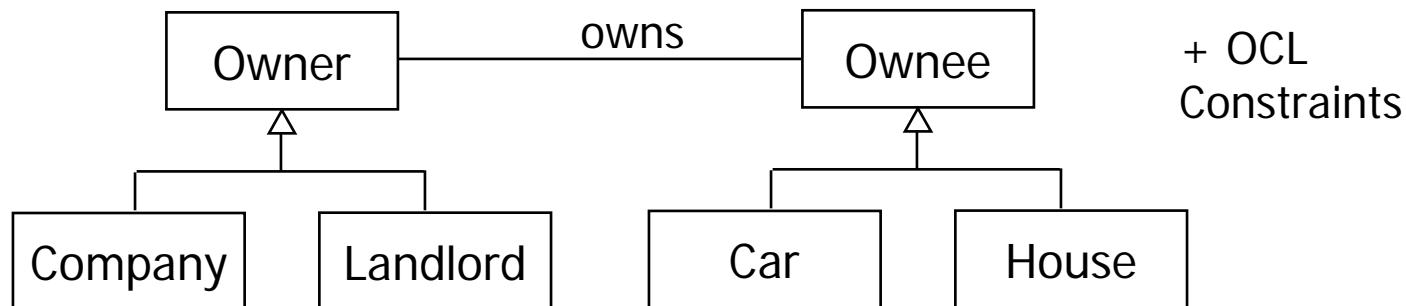


# Global Properties

- an often quoted shortcoming of the UML for the purpose of ontology representation -
  - “it is not possible to show that different incarnations of a relationship (e.g. “owns”) are somehow the same”



- but although association inheritance is not directly supported, it can be simulated with the help of OCL



# Functional Architecture of the Semantic Web

- semantic web is based on a hierarchy of increasingly powerful languages
  - may be mixed in arbitrary ways to represent information

## Metadata layer (RDF)

- basic concepts of resource and property to express meta information

RDF

## Schema layer (RDFS)

- uses the general features in the metadata layer to provides a basic ontology language

RDFS

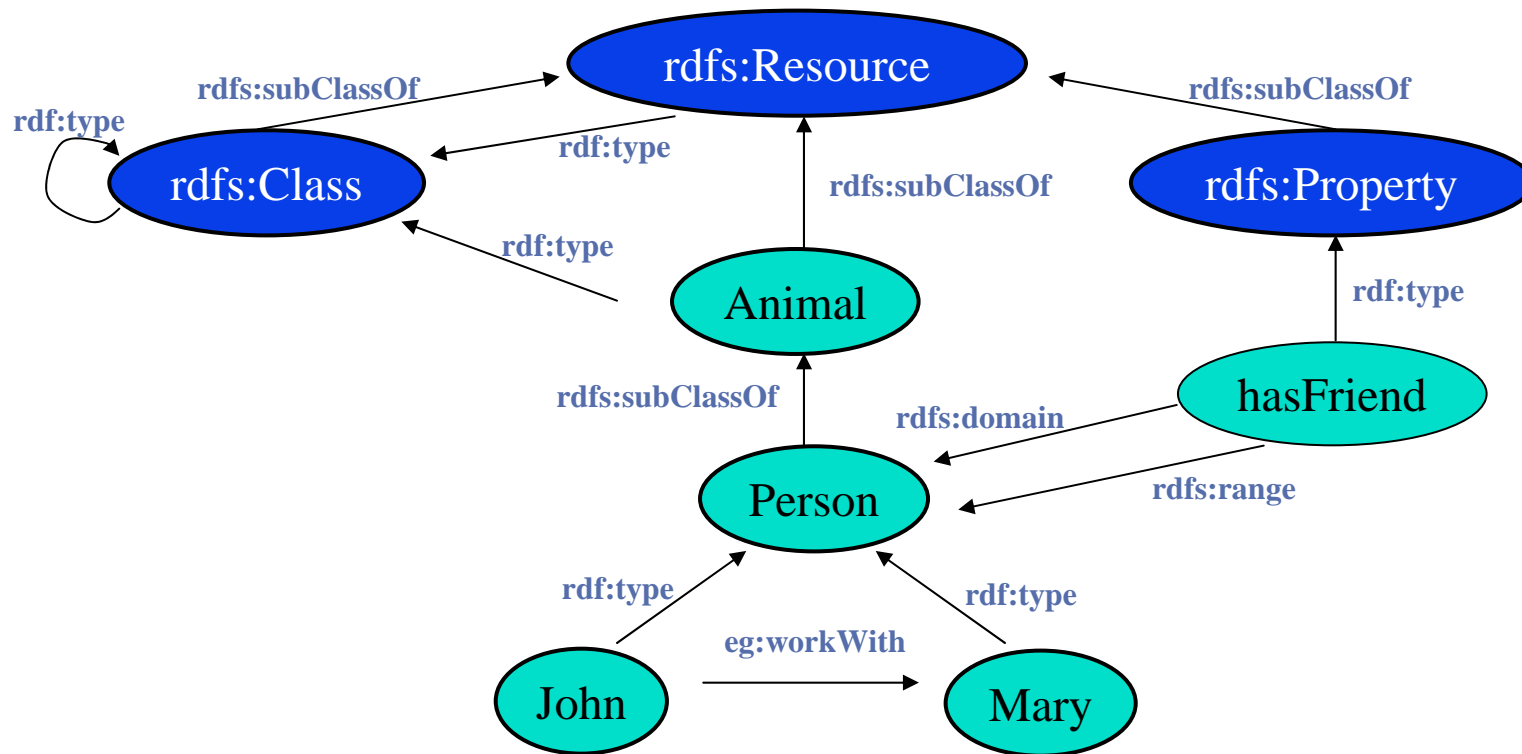
## Logical layer (OWL, DAML+OIL)

- powerful Web ontology languages that define a much richer set of modeling primitives that can be mapped to Description Logics

OWL, DAML + OIL

# All is Not well in the Standard Architecture

- “Dual Role” problem
  - multiple modeling primitives are represented by the same RDFS primitive (`rdf:type`, `rdfs:subClassOf`)
  - `Rdfs:subClassOf` is used to define ontologies as well as other RDF modeling primitives



# *Semantic Issues*

- the class `rdfs:Class` is an instance of itself
  - in RDFS, all classes (including `rdfs:Class`) are instances of `rdfs:Class`
  - suspiciously by close to Russell paradox.
- the class `rdfs:Resource` is a superclass and an instance of `rdfs:Class` at the same time
  - the super set (`rdfs:Resource`) is a member of the subset (`rdfs:Class`).
- the properties `rdfs:subClassOf`, `rdf:type`, `rdfs:range` and `rdfs:domain` are used to define both the other RDFS modeling primitives and the ontology
  - makes their semantics unclear and
  - makes it very difficult to formalise RDFS
- as a result, RDFS has no clear semantics

# *RDFS(FA) [Pan and Horrocks 2001]*

- “RDFS with Fixed layer meta-modeling Architecture”
  - a sub-language of RDF(S)
  - description logic style semantics
  - motivated by the “strict” interpretation of the UML four layer infrastructure
- divides universe of discourse into a series of “strata”
  - user defined facts and RDF/OWL built-in vocabulary are in different strata
  - each modeling primitive belongs to a certain stratum labeled with a stratum-specific prefix
- supports use of meta-classes and meta-properties
  - in the stratum above classes and properties

# RDFS(FA) Layers

- RDFS modeling primitives are mapped to the primitives in corresponding metamodeling layers
  - `rdfs:Class` is mapped to `rdfsfa:MClass` and `rdfsfa:LClass`
  - `rdfs:Property` is mapped to `rdfsfa:MProperty` and `rdfsfa:Lproperty`

**Stratum 3 (Meta-Language Layer)**

**fa:MResource,  
fa:MClass  
fa:MProperty ...**

**Stratum 2 (Language Layer)**

**fa:LResource,  
fa:LClass  
fa:LProperty ...**

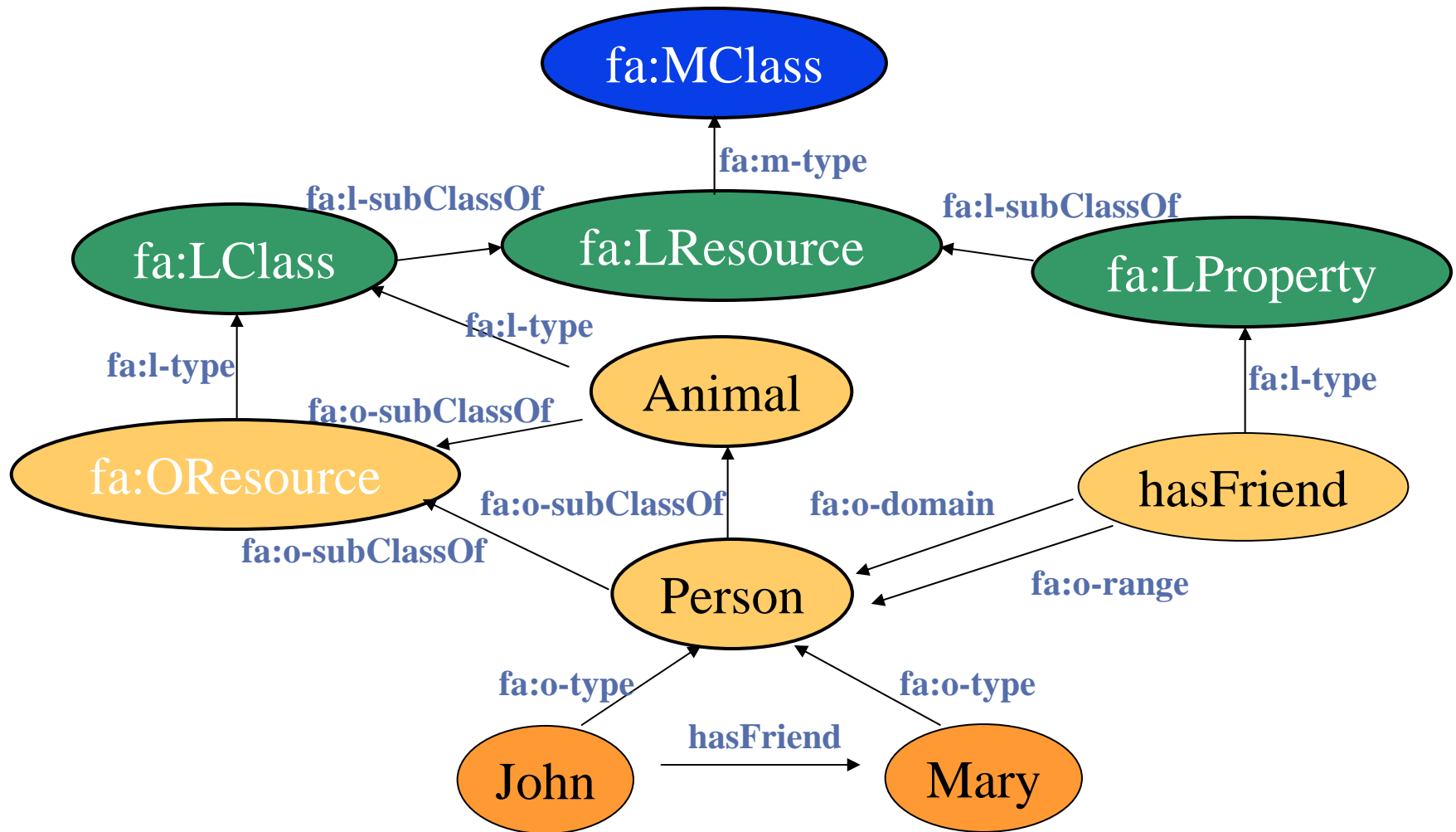
**Stratum 1 (Ontology Layer)**

**fa:OResource  
Person  
hasFriend ...**

**Stratum 0 (Instance Layer)**

**Bob, John...**

# Stratified Ontology



# Ontology Perspectives [ODM Submission]

Perspective	One Extreme	Other Extreme
Level of Authoritativeness	Least authoritative, broader, shallowly defined ontologies	Most authoritative, narrower, more deeply defined ontologies
Source of Structure	Passive (transcendent) – structure originates outside the system	Active (immanent) – structures emerges from data or application
Degree of Formality	Informal, or primarily taxonomic	Formal, having rigorously defined types, relations and theories or axioms
Model Dynamic	Read only, ontologies are static	Volatile, ontologies are fluid and changing
Instance Dynamics	Read only, resource instances are static	Volatile, resource instances change continuously
Control/Degree of Manageability	Externally focused, public (little or no control)	Internally focused, private (full control)
Application Changeability	Static (with periodic updates)	Dynamic
Coupling	Loosely-coupled	Tightly coupled
Integration Focus	Information integration	Application integration
Lifecycle Usage	Design time	Run-time

# Three Main Clusters of Applications

- Business Applications
  - characterized by having transcendent source of structure,
  - a high degree of formality
  - external control relative to nearly all users
- Analytic Applications
  - are characterized by highly changeable and flexible ontologies
  - using large collections of mostly read-only instance data
- Engineering Applications
  - are characterized by having transcendent source of structure
  - users control them primarily internally
  - are considered more authoritative
- Not all information (models) needs DL style formal semantics (e.g. for subsumption)

# Observations

- When combined with OCL, UML is capable of modeling information to the same level of precision as OWL
  - acknowledged in revised ODM submission
- So what then is an ontological model (or an ontology)
  - a model with formal, unambiguous semantics
    - does this include programming models?
  - a model represented in a OWL?
  - a model represented in a DL based languages
- description logic provides the best tradeoff between concise, unambiguous and computationally formal semantics versus usability and expressiveness
- BUT, the MDA four-level infrastructure provides a better metamodeling architecture

# Conclusion

- ODM is a valuable interim solution for allowing MDA and semantic web artifacts to interoperate
  - but is not a long term solution
- the two worlds should be integrated into a single unified information representation technology based on
  - description logic core semantics
  - MDA-like metamodeling infrastructure
- the term ontology should be reserved for standardized bodies of knowledge used for reference
  - authoritative, transcendent, formal and read-only
  - heavyweight
- the term ontology should not be used for localized, private and relatively transient bodies of knowledge