MDA Standards for Ontology Development

Dragan Gašević[1], Vladan Devedžić[2], Dragan Djurić[2]

[1]Simon Fraser University Surrey, Canada
[2]University of Belgrade, Serbia and Montenegro
dgasevic@acm.org, devedzic@fon.bg.ac.yu, dragandj@gmail.com
http://goodoldai.org.yu

Outline

Outline

Foreword

About presenters…

Foreword

- Dragan Gašević's profile
  - Interests
    - Semantic Web, AI+SE, education
  - LORE Lab, SFU, Canada
    http://lore.iat.sfu.ca
  - GOOD OLD AI
    http://goodoldai.org.yu
  - Teaching
  - Projects
  - Publications

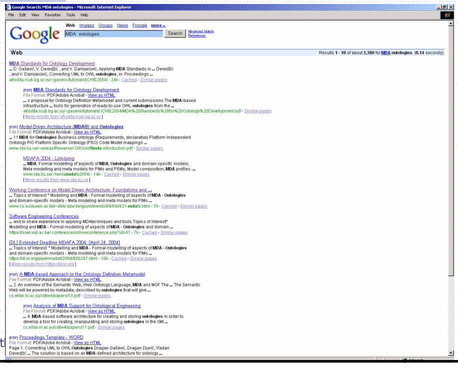
Foreword

- Dragan Djurić's profile
  - Interests
    - Semantic Web, AI+SE
  - GOOD OLD AI
    http://goodoldai.org.yu
  - Projects
  - Publications

Foreword

- Vladan Devedžić's profile
  - Interests
    - Semantic Web, AI+SE, education
  - GOOD OLD AI http://goodoldai.org.yu
  - Teaching
  - Projects
  - Publications

Google: MDA + ontologies



Foreword

- D. Gašević, D. Đurić, V. Devedžić, ***Model Driven Architecture and Ontology Development***, Springer, 2006. (Forthcoming)
- D. Gašević, D. Đurić, V. Devedžić, "Adopting Software Engineering Trends in AI," ***IEEE Intelligent Systems***, 2006 (forthcoming).

Foreword

- Courtesy to OMG's members
  - Evan Wallace
  - Elisa Kendall
  - Bob Colomb
  - Anna Gerber
  - Dan Chang
  - Lewis Hart
  - …

http://ontology.omg.org

Introduction

Introduction

- Synergy of different research fields (Devedžić, 2002)
  - multidisciplinary approach
    - software patterns, ontologies, databases, …
  - recent technologies
    - eXtensible Markup languages
    - Semantic Web and ontologies
    - Model Driven Architecture and related standards
      - UML, MOF, XMI
  - developing in parallel, but by different communities!

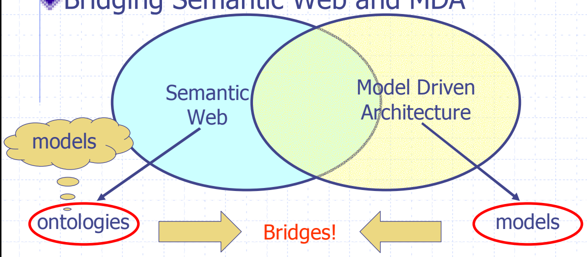
Introduction

A technological space is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. Although some technological spaces are difficult to define, they can be easily recognized (e.g. MDA, ontologies). In order to get a synergy of different technological spaces we should create bridges between them.

Kurtev et al, 2002

Introduction

- Bridging Semantic Web and MDA



Differences

- Semantic Web and MDA
  - models
    - models are abstractions
    - they eliminate or simplify concepts and relationships
  - ontologies
    - intended for knowledge representation
    - "if something is known, it should be able to be recorded in a machine-interpretable manner"
    - everything is worth capturing

Similarities

- Semantic Web and MDA
  - UML models
    - classes, attributes, relations, inheritance, …
  - ontologies
    - classes, properties, relations, inheritance, …
  - semantically different, but equivalent!
  - metadata sharing – XML!

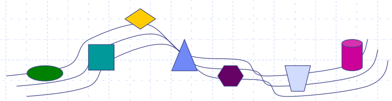
Semantic Web and Ontologies

What is an ontology?

- Classic definitions (Gruber, 1993), (Guarino, 1994)
  - a specification of a conceptualization
  - an explicit specification of some topic
  - a formal and declarative representation of some subject area

What is an ontology?

- Other important definitions (Swartout and Tate, 1999)
  - the basic structure or *armature* around which a knowledge base can be built



What is an ontology?

- Other important definitions (Hendler, 2001)
  - a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic – linguistically view
  - this definition is currently predominant in The Semantic Web community

What is an ontology?

- What does an ontology provide?
  - the vocabulary (or names) for referring to the terms in that subject area
  - the logical statements that describe:
    - what the terms are
    - how they are related to each other
    - how they can or cannot be related to each other

What is an ontology?

- What does an ontology provide?
  - rules for combining terms and relations to define extensions to the vocabulary
  - semantics independent of reader and context
  - a common understanding of topics that can be communicated between users and applications

What is an ontology?

- What is the purpose of ontologies?
  - knowledge sharing and reuse
  - description of the concepts and relationships that can exist for an IA or a community of IAs
    - the description is like a formal specification of a program

Examples of Ontologies

- The Frame ontology (Gruber, 1993)
  - partial vocabulary of the Frame ontology

    class relation (?relation)
    class function (?function)
    class class (?class)
    relation instance-of (?individual ?class)
    function all-instances (?class) :-> ?set-of-instances
    function one-of (@instances) :-> ?class

Examples of Ontologies

- The Frame ontology (Gruber, 1993)
  - partial vocabulary of the Frame ontology

    relation subclass-of (?child-class ?parent-class)

    relation superclass-of (?parent-class ?child-class)

    relation subrelation-of (?child-relation ?parent-relation)

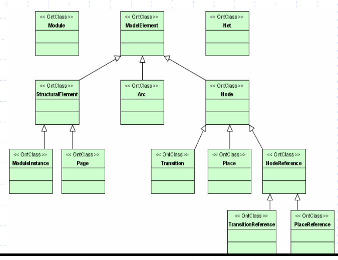
    relation direct-instance-of (?individual ?class)

    relation direct-subclass-of (?child-class ?parent-class)

Examples of Ontologies

- Petri net ontology (Gašević, 2004)
  - Ontology UML Profile representation

(excerpt)



Examples of Ontologies

- Petri net ontology (Gašević, 2004)
  - RDFS representation
  - http://afrodita.rcub.bg.ac.yu/~gasevic/projects/PNO

(excerpt)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
    <!ENTITY a 'http://protege.stanford.edu/system#'>
    <!ENTITY PN_Ontology 'http://protege.stanford.edu/PN_Ontology#'>
    <!ENTITY rdfs 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'>
]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:a="&a;" xmlns:PN_Ontology="&PN_Ontology;" xmlns:rdfs="&rdfs;">
    <rdfs:Class rdf:about="&PN_Ontology;Absolute Position" rdfs:label="Absolute Position">
        <rdfs:subClassOf rdf:resource="&PN_Ontology;Position"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="&PN_Ontology;Arc" rdfs:label="Arc">
        <rdfs:subClassOf rdf:resource="&PN_Ontology;ModelElement"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="&PN_Ontology;Feature" rdfs:label="Feature">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="&PN_Ontology;GraphicalFeature" rdfs:label="GraphicalFeature">
        <rdfs:subClassOf rdf:resource="&PN_Ontology;Feature"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="&PN_Ontology;GraphicalInformation" rdfs:label="GraphicalInformation">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
    </rdfs:Class>
    <rdf:Property rdf:about="&PN_Ontology;ID" a:maxCardinality="1" a:minCardinality="1" rdfs:label="ID">
        <rdfs:domain rdf:resource="&PN_Ontology;ModelElement"/>
        <rdfs:domain rdf:resource="&PN_Ontology;Net"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </rdf:Property>
```

Why Ontologies?

- The role of ontologies in the architecture of The Semantic Web
  - to enable intelligent services
    - information brokers
    - search agents
    - information filters
    - intelligent information integration
    - knowledge management

Why Ontologies?

- The role of ontologies in the architecture of The Semantic Web
  - to establish further levels of interoperability (semantic interoperability) on the Web
    - syntactic interoperability: reusability in parsing the data
    - semantic interoperability: mappings between terms within the data, which requires content analysis

Why Ontologies?

- Ontologies merely serve to standardize and provide interpretations for Web content
- To make content machine-understandable, Web pages must contain *semantic markup*
  - descriptions which use the terminology that one or more ontologies define

Ontology structure

- The main ontology components (Firidman-Noy & Hafner, 1997)
  - conceptualization
  - taxonomy
  - relations
  - functions
  - axioms
- Similarity with an OO model (classes, class relations, and objects)

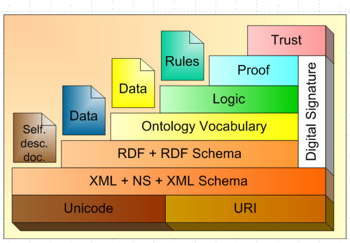
Nov 6th, 2005 ISWC2005, Galway, Ireland 31

Tools for Building Ontologies

- XML / XMLS, RDF / RDFS, and the corresponding development tools
- Ontology representation languages (The Semantic Web languages)
- Ontology-development environments (integrated graphical tools)
- Ontology-learning tools

Nov 6th, 2005 ISWC2005, Galway, Ireland 32

Semantic Web "Layer Cake"



Nov 6th, 2005 ISWC2005, Galway, Ireland 33

HTML

- Example

```
<UL>
<LI>Aho, A. V., Sethi, R., Ullman, J.D.: <EM>Compilers:
Principles, Techniques, and Tools</EM>, Addison-Wesley, 1985
</UL>
```

HTML is layout-oriented

XML

- Example

```
<?xml version="1.0"?>
<employees>
        List of persons in company:
        <person name="John">
                <phone>47782</phone>
                On leave for 2001.
        </person>
</employees>
```

an attribute within a tag

XML

- Example

```
<BOOK>
  <AUTHOR> Aho, A.V. </AUTHOR>
  <AUTHOR> Sethi, R. </AUTHOR>
  <AUTHOR> Ullman, J.D. </AUTHOR>
  <TITLE> Compilers: Principles, Techniques, and Tools </TITLE>
  <PUBLISHER> Addison-Wesley </PUBLISHER>
  <YEAR> 1985 </YEAR>
</BOOK>
```

XML is structure-oriented

XML Schema

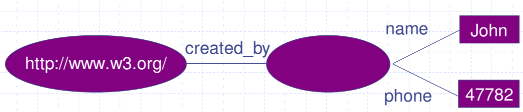
◆Example

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<xsd:element name="BOOK" type="BOOKTYPE"/>
<xsd:complexType name="BOOK_TYPE" >
   <xsd:element name="AUTHOR" type="xsd:string"
      minOccurs="1" maxOccurs="unbounded"/>
   <xsd:element name="TITLE" type="xsd:string"/>
   <xsd:element name="PUBLISHER" type="xsd:string"
      minOccurs="0" maxOccurs="1"/>
   <xsd:element name="YEAR" type="xsd:decimal"
      minOccurs="0" maxOccurs="1"/>
   <xsd:attribute name="isbn" type="xsd:string"/>
   <xsd:attribute name="nickname" type="xsd:string"/>
</xsd:complexType>
</xsd:schema>
```


RDF

◆Example

OBJECT	ATTRIBUTE	VALUE
http://www.w3.org/	created_by	#anonymous_resource1
#anonymous_resource1	name	"John"
#anonymous_resource1	phone	"47782"



RDF Schema

◆Example

```
<rdfs:Class rdf:ID="herbivore">
   <rdfs:subClassOf rdf:resource="#animal"/>
   <rdfs:subClassOf>
   ...
   </rdfs:subClassOf>
</rdfs:Class>
```

RDF and RDFS define a standard knowledge representation way

Tools for Building Ontologies

- Ontology representation languages
  - Knowledge Interchange Format (KIF)
  - Simple HTML Ontology Extensions (SHOE)
  - ISO standard for describing knowledge structures (Topic Maps)
  - DARPA Agent Markup Language (DAML)
  - Ontology Inference Layer (OIL, DAML+OIL)
  - Web Ontology Language (OWL)

Tools for Building Ontologies

- The need:
  a standardized Web ontology language
  - it is emerging at the moment
  - OIL, DAML+OIL, …
    - "OIL, a proposal for such a standard"
  - W3C recommendation – OWL
  - ISO efforts

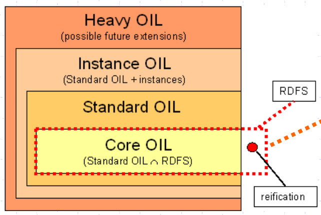
Ideally, we would like a universal shared knowledge-representation language to support the Semantic Web, but for a variety of pragmatic and technological reasons, this is unachievable in practice. Instead, we will have to live with a multitude of metadata representations.

S. Decker et al.

Tools for Building Ontologies

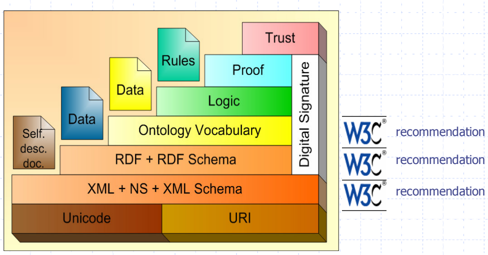
- Ontology Inference Layer (OIL)

An OIL processor will also understand RDFS



Even simple RDF Schema agents are able to process the OIL ontologies

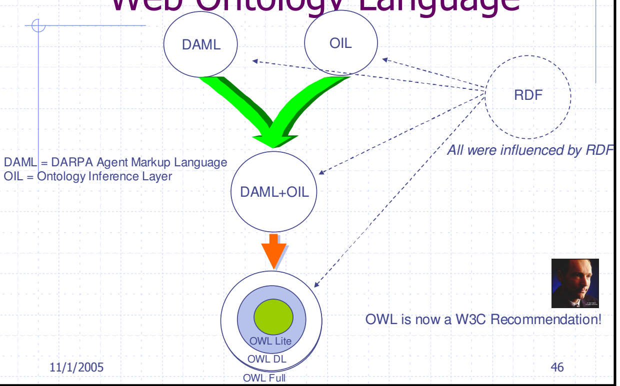
Semantic Web "Layer Cake"



Web Ontology Language

- Good starting point: Roger L. Costello, David B. Jacobs, "A Quick Introduction to OWL Web Ontology Language", presentation, The MITRE Corporation, 2003. http://www.xfront.org

Web Ontology Language

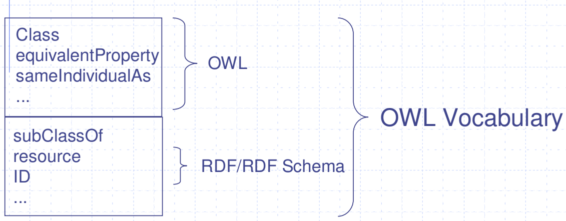


Web Ontology Language

- OWL is
  a set of XML elements and attributes, with
  well-defined meaning,
  that are used to define
  terms and their relationships

Web Ontology Language

- OWL extends RDF and RDF Schema



Introduction to OWL

- Example:
  defining two ter
  relationship
  - define the term
  - state that SLRs

SLR Single Lens Reflex (camera)
SLR Satellite Laser Ranging
SLR Self Loading Rifle
SLR Sending Loudness Rating (telecommunications)
SLR Service Level Report
SLR Service Location Register
SLR Side Looking Radar
SLR Single Line Restoral
SLR Single Linear Recording
SLR Slide Raft (aircraft door)
SLR Slush on Runway(s)
SLR Solectron
SLR Spacelift Range
SLR Sri Lanka Rupee (national currency)
SLR Statutory Liquidity Ratio
SLR Stock Level Report
SLR Stock Level Requirement
SLR Straight Leg Raise
SLR Straight Leg Raising
SLR System Level Requirement(s)

Introduction to OWL

- Example:
  defining two terms and their relationship
  - define the terms "Camera" and "SLR"
  - state that SLRs are a type of Camera

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

```
<owl:Class rdf:ID="SLR">
    <rdfs:subClassOf rdf:resource="#Camera"/>
</owl:Class>
```


Introduction to OWL

- Scenario
  - I am interested in purchasing a camera with a 75-300mm zoom lens, that has an aperture of 4.5-5.6, and a shutter speed that ranges from 1/500 sec. to 1.0 sec
  - I launch my personal "Web Bot" which crawls the Web looking for Web sites that can fulfill my request

Introduction to OWL

- Scenario
  - Assume that there exists an OWL Camera Ontology, which the Web Bot can "consult" upon it travels across the Web

Introduction to OWL

- Scenario
  - the Web Bot f
    this documen
    at a Web site

```
<PhotographyStore rdf:ID="Hunts"
                  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <store-location>Malden, MA</store-location>
  <phone>617-555-1234</phone>
  <catalog rdf:parseType="Collection">
    <SLR rdf:ID="Olympus-OM-10"
         xmlns="http://www.camera.org#">
      <lens>
        <Lens>
          <focal-length>75-300mm zoom</focal-length>
          <f-stop>4.5-5.6</f-stop>
        </Lens>
      </lens>
      <body>
        <Body>
          <shutter-speed rdf:parseType="Resource">
            <min>0.002</min>
            <max>1.0</max>
            <units>seconds</units>
          </shutter-speed>
        </Body>
      </body>
      <cost rdf:parseType="Resource">
        <rdf:value>325</rdf:value>
        <currency>USD</currency>
      </cost>
    </SLR>
  </catalog>
</PhotographyStore>
```

Is it relevant?
(SLR = Single Lens Reflex)

Introduction to OWL

- Scenario

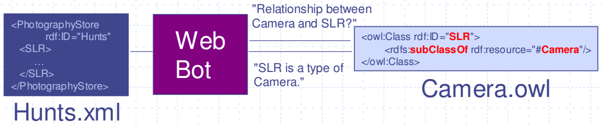
```
<Camera xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.camera.org#">
  <lens>
    <Lens>
      <size>75-300mm zoom</size>
      <aperture>4.5-5.6</aperture>
    </Lens>
  </lens>
  <body>
    <Body>
      <shutter-speed rdf:parseType="Resource">
        <min>0.002</min>
        <max>1.0</max>
        <units>seconds</units>
      </shutter-speed>
    </Body>
  </body>
</Camera>
```


```
<PhotographyStore rdf:ID="Hunts"
                  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <store-location>Malden, MA</store-location>
  <phone>617-555-1234</phone>
  <catalog rdf:parseType="Collection">
    <SLR rdf:ID="Olympus-OM-10"
         xmlns="http://www.camera.org#">
      <lens>
        <Lens>
          <focal-length>75-300mm zoom</focal-length>
          <f-stop>4.5-5.6</f-stop>
        </Lens>
      </lens>
      <body>
        <Body>
          <shutter-speed rdf:parseType="Resource">
            <min>0.002</min>
            <max>1.0</max>
            <units>seconds</units>
          </shutter-speed>
        </Body>
      </body>
      <cost rdf:parseType="Resource">
        <rdf:value>325</rdf:value>
        <currency>USD</currency>
      </cost>
    </SLR>
  </catalog>
</PhotographyStore>
```


Match?

Introduction to OWL

- Scenario
  - the Web Bot "consults" the OWL Camera Ontology



Introduction to OWL

- Scenario
  - the Web Bot "consults" the OWL Camera Ontology

```
<owl:DatatypeProperty rdf:ID="focal-length">
    <owl:equivalentProperty rdf:resource="#size"/>
    <rdfs:domain rdf:resource="#Lens"/>
    <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

"focal-length is synonymous with (lens) size.
focal-length is to be used within a Lens.
focal-length has a value that is a string."

Introduction to OWL

- Scenario
  - the Web Bot "consults" the OWL Camera Ontology

```
<owl:DatatypeProperty rdf:ID="f-stop">
    <owl:equivalentProperty rdf:resource="#aperture"/>
    <rdfs:domain rdf:resource="#Lens"/>
    <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

"f-stop is equivalent to aperture."

Introduction to OWL

- Summary: interoperability despite terminology differences!
  - achieved through the use of the OWL Camera Ontology

ge

Sum

Symmetric: if P(x, y) then P(y, x)
Transitive: if P(x,y) and P(y,z) then P(x, z)
Functional: if P(x,y) and P(x,z) then y=z
InverseOf: if P1(x,y) then P2(y,x)
InverseFunctional: if P(y,x) and P(z,x) then y=z
allValuesFrom: P(x,y) and y=allValuesFrom(C)
someValuesFrom: P(x,y) and y=someValuesFrom(C)
hasValue: P(x,y) and y=hasValue(v)
cardinality: cardinality(P) = N
minCardinality: minCardinality(P) = N
maxCardinality: maxCardinality(P) = N
equivalentProperty: P1 = P2
intersectionOf: C = intersectionOf(C1, C2, …)
unionOf: C = unionOf(C1, C2, …)
complementOf: C = complementOf(C1)
oneOf: C = one of(v1, v2, …)
equivalentClass: C1 = C2
disjointWith: C1 != C2
sameIndividualAs: I1 = I2
differentFrom: I1 != I2
AllDifferent: I1 != I2, I1 != I3, I2 != I3, …
Thing: I1, I2, …

Legend:
Properties: P, P1, P2, etc
Specific classes: x, y, z
Generic classes: C, C1, C2
Values: v, v1, v2
Instance documents: I1, I2, I3, etc.
A number: N
P(x,y): "property P relates x to y"

Web Ontology Language

- Petri net ontology (Gašević, 2004)
  - OWL representation (excerpt)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://owl.protege.stanford.edu#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Class rdf:ID="ModelElement"/>
    <owl:Class rdf:ID="StructuralElement">
        <rdfs:subClassOf rdf:resource="#ModelElement"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#element"/>
                <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:minCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Arc">
        <rdfs:subClassOf rdf:resource="#ModelElement"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#multiplicity"/>
                <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:maxCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#fromNode"/>
                                    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
...
```

Tools for Building Ontologies

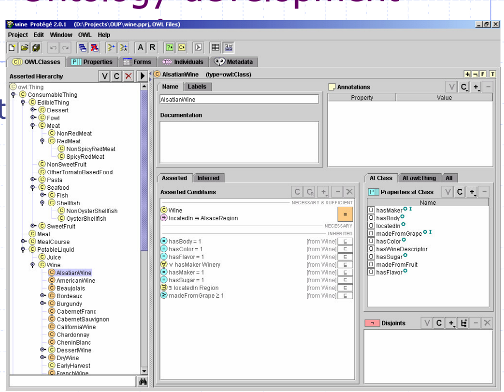
- Ontology-development tools
  - Protégé
  - OntoEdit
  - OilEd
  - Chimaera
  - ...

Ontology-development environments

- Protégé
  - Extensible platform (plug-ins)
    - Semantic Web: OWL, DAML+OIL, OIL, ...
    - Import/Export: OKBC Tab Widget, XML, TX RuleML Tab Widget, ...
    - Inference & Reasoning: Jess Tab, Algernon Tab, CLISP Tab, ...
    - Software engineering: UML Storage Backend, XMI Storage Backend, ...
    - ...

Ontology-development

- Prot



What lack ontology building tools?

- Shortcomings
  - ontologies – built on AI concepts
  - tools and languages don't use the same terminology
  - software practitioners don't know all these ontology concepts
    - they need more familiar notation and tools
    - they need a unified representation for ontologies
  - UML as a natural solution

Model Driven Architecture: An Overview

Model Driven Architecture

- Developing in parallel with Semantic Web
- Object Modeling Group effort
- The latest paradigm shift in software engineering (Bézivin, 2002)
  - from OO technology…
  - …to model technology

→ Model Driven Development (Mellor et al, 2003)

What is a model?

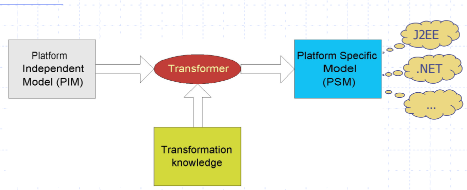
- Model – a simplified view of reality
- More formally: A model is a set of statements about some system under study
  - (Seidewitz, 2003)
- OMG is about models

Why do we need models?

We express models using concepts that are much less bound to the underlying implementation technology and are much closer to the problem domain relative to most popular programming languages.

Selic, 2003

Why do we need models?



- Use of platform independent models (PIM) as specifications
- Transform specifications into platform specific models (PSM) using tools

Model characteristics

- To be useful and effective, a model must have the following characteristics:
  - abstraction
  - understandability
  - accuracy
  - predictiveness
  - inexpensive

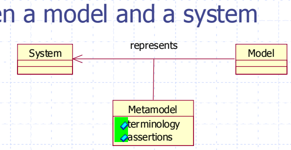
What is a metamodel?

A metamodel makes statements about what can be expressed in the valid models of a certain modeling language.

Seidewitz, 2003

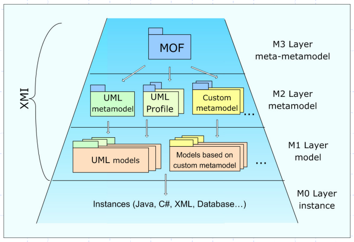
What is a metamodel?

- In fact, a metamodel:
  - is a model of a modeling language, or
  - makes statements about what can be expressed in the valid models of a certain modeling language
- The correspondence between a model and a system



Model Driven Architecture

- Common understanding of the four-layer architecture



Metametamodel

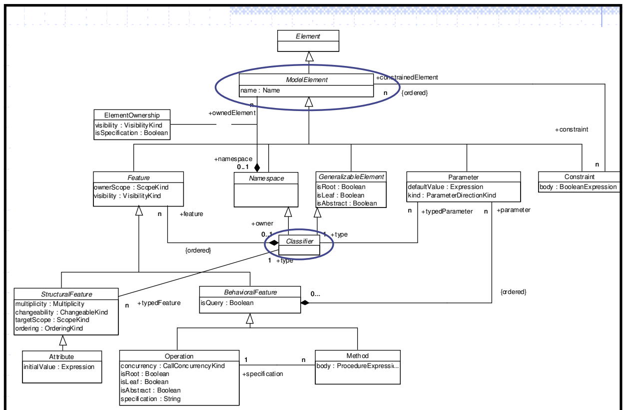
- Meta-Object Facility (MOF)
  - MDA metametamodel layer (M3)
  - self-defined language
  - *standard* metamodeling framework for model-driven and metadata-driven systems
    - modeling and development tools
    - data warehouse systems
    - metadata repositories
    - metadata = data about data, e.g., database schemas

Metametamodel

- Meta-Object Facility (MOF)
  - MOF is the MDA's basic mechanism for defining modeling languages
  - Modeling languages defined at the metamodel layer (M2):
    - Unified Modeling Language (UML)
    - Common Warehouse Metamodel (CWM)
    - Ontology Definition Metamodel (ODM)
    - …
  - MOF uses the UML graphical notation

UML Metamodel

- A part of UML metamodel

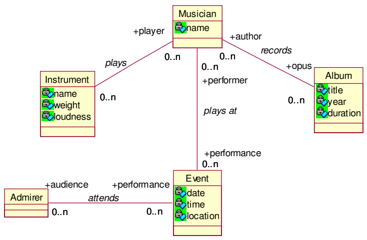


Models

- Model layer (M1)
  - the layer where we develop real-world models (or domain models)
  - in terms of UML models, that means creating classes, their relations, states, etc.

Models

- Example



The bottom-most layer

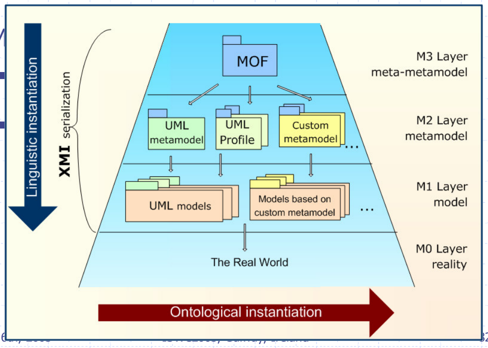
- Two approaches
  - the bottom layer is the instance layer (M0) (common understanding)
  - the instance layer contains things from our reality – concrete and abstract
    - (Atkinson & Kühne, 2003)

Orthogonal instantiation relations

- More details about the second approach
  - UML classes and and objects are instances of UML metamodel concepts
  - objects model concrete real-world things (e.g. Mike)
  - classes model abstract real-world (e.g. Person)
  - however, an object (Mike) is an instance of a class (Person)!

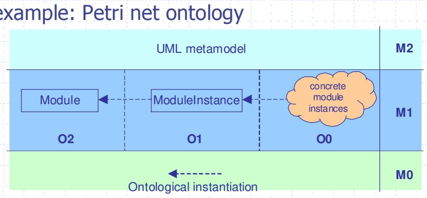
Orthogonal instantiation relations

- M



Orthogonal instantiation relations

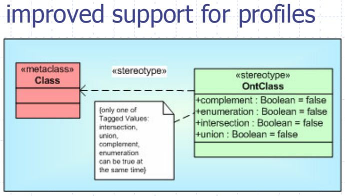
- Ontological instantiation
  - one linguistic layer (e.g. M1) may have many ontological layers
  - example: Petri net ontology



Module – "template" for deriving module instances
Module instance – a mechanism for structuring Petri nets (grouping their elements)

UML Profiles

- Extension mechanism
  - stereotypes, tagged values, and OCL constraints
  - UML2 improved support for profiles

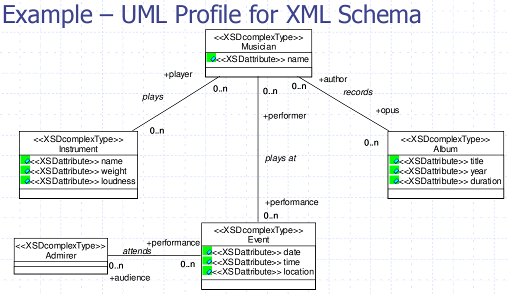


UML Profiles

- Why do we need UML Profiles?
  - we can develop a new domain language (i.e. a metamodel) using MOF
  - it is rather expensive to develop new tools
  - so, we try to use existing, well-known tools
  - most of MDA tools are oriented towards UML

UML Profiles

- Example – UML Profile for XML Schema



UML Profiles

- Why do we need UML Profiles?
  - a way to adapt UML to specific domains
  - a way to extend the UML metamodel
  - UML as a family of languages
  - some UML profiles
    - UML Profile for CORBA
    - UML Profile for Enterprise Application Integration
    - UML Profile for DTD and XML Schema definition
    - ...

MDA metadata sharing

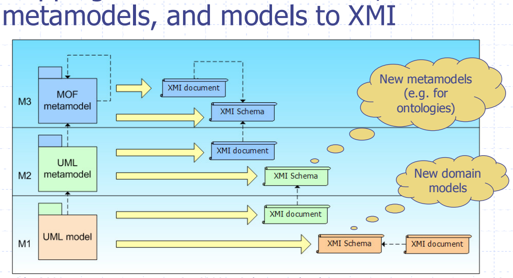
- XML Metadata Interchange (XMI)
  - a standard way of mapping objects to XML
    - XML is not object-oriented
  - uses MOF as the underlying object model
  - defines two sets of rules
    1. for serializing objects to XMI documents
    2. for generating XML Schemas from models

MDA metadata sharing

- XML Metadata Interchange (XMI)
  - may be used for serializing objects at different meta-levels
    - e.g., data, metadata, metametadata, etc.
  - important: XMI is not just for UML
    - a UML tool will usually only accept UML XMI (i.e., XMI conforming to the UML metamodel)

XML Metadata Interchange (XMI)

- Mapping MDA metametamodel, metamodels, and models to XMI



XML Metadata Interchange (XMI)

- XMI Schema production rules
  - we didn't show XMI production rules from MOF-compliant (meta)models
    - see (Grose et al, 2002) for details
  - some important documents from the OMG's site
    - XML schema for MOF metamodels
    - XMI document of the MOF metamodel
    - XMI document of the UML metamodel
    - XML schema for UML models
  - benefits and problems

UML XMI Model - Example

```
<UML:Class xmi.id="Im19247250m106c9d71d66m24d0" name="Musician"
visibility="public" isSpecification="false" isRoot="true" isLeaf="true" isAbstract="false" isActive="false">
    <UML:Classifier.feature>
        <UML:Attribute xmi.id="Im19247250m106c9d71d66m24d1"
        name="name" visibility="private" isSpecification="false"
        ownerScope="instance" changeability="changeable"
        targetScope="instance">
            <UML:StructuralFeature.multiplicity>
                <UML:Multiplicity xmi.id="Im19247250m106c9d71d66m250c">
                    <UML:Multiplicity.range>
                        <UML:MultiplicityRange xmi.id=
                        "Im19247250m106c9d71d66m250d" lower="1" upper="1"/>
                    </UML:Multiplicity.range>
                </UML:Multiplicity>
            </UML:StructuralFeature.multiplicity>
            <!--... -->
            <UML:Attribute.initialValue>
                <UML:Expression xmi.id="Im19247250m106c9d71d66m24d2" language="java" body=""/>
            </UML:Attribute.initialValue>
            <UML:StructuralFeature.type>
                <UML:DataType xmi.idref="Im19247250m106c9d71d66m24d3"/>
            </UML:StructuralFeature.type>
        </UML:Attribute>
    </UML:Classifier.feature>
</UML:Class>
```

Java Metadata Interface (JMI)

- JMI – a Java Community Process standard
  - reflective Java API to explore any MOF model dynamically
  - a set of rules to generate Java API customized for a given MOF metamodel
    - generated interfaces inherit from the reflective interfaces
  - implementations
    - NetBeans' MetaData Repository (MDR)
    - Unisys' CMI

UML-based solutions and tools for ontology development

UML & Ontologies

- Ontologies
  - frame-based systems, description logics
  - a well-understood semantic basis
  - the main component of the Semantic Web
  - XML/RDF-based languages (e.g. OWL, DAML+OIL, OIL, …)
  - concepts are unknown to practitioners
  - lack of graphical notation

UML & Ontologies

- UML
  - object-oriented modeling paradigm
  - widely accepted modeling language
  - well-known graphical notation
  - Object Constraint Language (OCL)
  - many software tools
  - industry and research
  - XML for sharing models (XMI)

The first approach to using UML in ontology development

- Cranefield was the first to address the problem
  - the starting points:
    - UML class diagrams provide a static modeling capability that is well-suited for representing ontologies
    - UML object diagrams can be interpreted as declarative representations of knowledge
    - OCL for ontology constraints
    - advantage: using the same paradigm for modeling ontologies and knowledge

Cranefield's approach

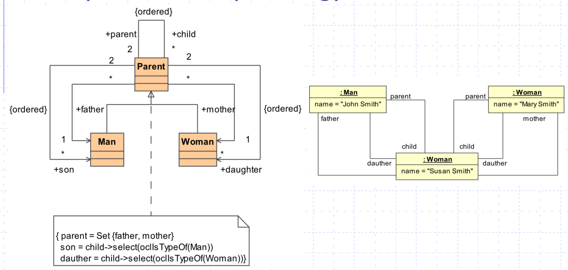
- Shortcomings of UML
  - it lacks a formal definition
  - descriptions of various elements of the language in English
  - defined by a metamodel
  - some additional constraints expressed in a semi-formal language (OCL)
  - there are initiatives that try to solve this problem (e.g., Precise UML - PUML)

Cranefield's approach

- Technology requirements
  - XMI – for sharing UML models
  - RDF – for sharing ontologies
  - UML tools that produce UML XMI
  - XSLT that transform UML XMI to:
    - a set of Java classes and interfaces corresponding to those in the ontology
    - RDF & RDF Schema

Cranefield's approach

- Example: The family ontology



Cranefield's approach

- Transformation to RDF(S)
  - XSLT implementation
    - Classes to RDFS
    - Objects to RDF
  - mapping problems:
    - UML classes have different features – attributes, associations, and association classes
    - RDFS – fields or properties
    - RDFS properties are first-class objects

Cranefield's approach

- Transformation to RDFS
  - some solutions:
    - properties in RDFS have a class prefix (but, this has a problem with class inheritance)
    - upper limit for multiplicity greater than 1 ⇒ RDFS bag
    - association ends with a UML "ordered" constraint ⇒ RDFS sequences
    - …

Cranefield's approach

- Resulting RDFS for the family ontology

(excerpt)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xml:lang="en"
    xmlns:rdfsx="http://nzdis.otago.ac.nz/2000/01/rdf-schema-extensions#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdfs:Class rdf:ID="http://nzdis.otago.ac.nz/0_1/family#Person"/>
    <rdf:Property ID="http://nzdis.otago.ac.nz/0_1/family#Person.name">
        <rdfs:domain rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Person"/>
        <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>
    <rdf:Property ID="http://nzdis.otago.ac.nz/0_1/family#Person.parent">
        <rdfs:domain rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Person"/>
        <rdfs:range rdf:resource="rdf:Bag"/>
        <rdfsx:containerElementType rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Person"/>
    </rdf:Property>
    <rdf:Property ID="http://nzdis.otago.ac.nz/0_1/family#Person.child">
        <rdfs:domain rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Person"/>
        <rdfs:range rdf:resource="rdf:Seq"/>
        <rdfsx:containerElementType rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Person"/>
    </rdf:Property>
    <rdf:Property ID="http://nzdis.otago.ac.nz/0_1/family#Person.father">
        <rdfs:domain rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Person"/>
        <rdfs:range rdf:resource="http://nzdis.otago.ac.nz/0_1/family#Man"/>
    </rdf:Property>
```


Cranefield's approach

- Some limitations of this approach
  - diagrams are just a notational feature
  - standard UML can not express advanced ontology features (e.g. restrictions)
  - one cannot conclude whether the same property was attached to more than one class

Cranefield's approach

- Some limitations of this approach
  - one cannot create a hierarchy of properties
  - the target RDFS ontology description does not have advanced restriction concepts
    - e.g., multiplicity
- However, this is a kind of agile UML-based ontology development

UML extension for ontology development

- This approach is inspired by DAML+OIL
- Theoretical discussion
  - (Backlawski et al., 2002)
  - rough classification of KR languages:
    - logical languages (e.g. KIF)
    - frame-based languages – OO database languages
    - graph-based languages – conceptual graphs (Sowa, 2000)

UML extension for ontology development

- DAML+OIL vs. UML
  - UML and DAML+OIL have many characteristic in common
    - e.g., classes and instances
  - property as a standalone concept ⇒ the main problem
  - DAML+OIL does not have any graphical notation
  - tools for DAML+OIL and experiences with it are not so plentiful as with UML
  - approaches that use stereotypes do not define semantics (e.g. DUET)

UML extension for ontology development

- DAML+OIL vs. UML
  - some further differences
    - specialization/generalization
      - UML – behavioral
      - DAML+OIL – set-theoretic

      } e.g. Square and Rectangle
      - not semantically equivalent, but mapping is consistent
    - UML is monotonic, DAML+OIL is not
    - KR languages do not have meta-level separation (e.g. instances of a class may be classes)

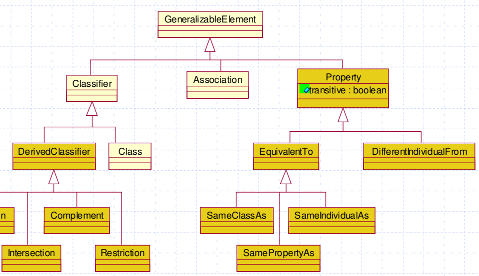
UML extension for ontology development

- DAML+OIL vs. UML
  - high-level mapping between DAML+OIL and UML

DAML+OIL Concept	Similar UML Concept
Ontology	Package
Class	Class
As Sets (union, intersection, etc.)	Not supported
Hierarchy	Class Generalization Relations
Property	Aspects of Attributes, Associations and Classes
Hierarchy	None for Attributes, limited Generalization for Associations, Class Generalization for Relations
Restriction	Constraints Association Ends, including multiplicity and roles, Implicitly as class containing the attribute
Data Types	Data Types
Instances and Values	Object Instances and Attribute Values

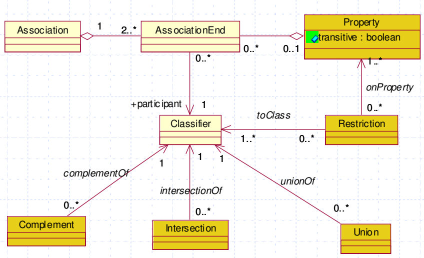
UML extension for ontology development

- Extension of the UML metamodel



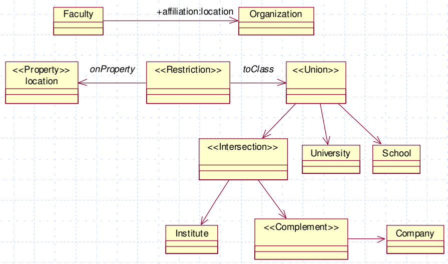
UML extension for ontology development

- Extension of the UML metamodel



UML extension for ontology development

- Example



UML extension for ontology development

- Some shortcomings
  - DAML+OIL has evolved to OWL
  - no suitable tool support
  - extended UML metamodel is an awkward solution
    - we do not need all UML concepts for ontology development
  - better solution: an independent ontology metamodel

Unified Ontology Language (UOL)

- A new MOF-defined ontology metamodel (Backlawski et al, 2002)
  - closely related to UML
  - direct mappings to/from UML
  - based on UML and DAML+OIL (but also considers OWL)
  - UOL uses the same analysis of KR languages

Unified Ontology Language (UOL)

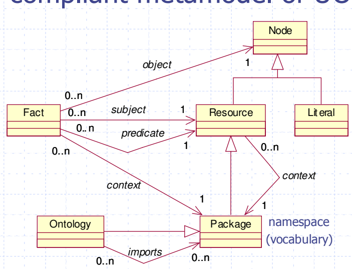
- The proposed language UOL should satisfy the following requirements:
  - it must be a MOF metalanguage
  - mappings
    - a bounded two-way mapping between core UML and core UOL
    - the two-way mapping must preserve semantic equivalence at MDA levels 0 and 1

Unified Ontology Language (UOL)

- The proposed language UOL should satisfy the following requirements:
  - core UML and core UOL must include the following notions:
    - package
    - class
    - binary association
    - generalization
    - attribute
    - multiplicity constraints

Unified Ontology Language (UOL)

- MOF-compliant metamodel of UOL



Unified Ontology Language (UOL)

- Short discussion
  - a small and high-level metamodel
  - a solid starting point for a standard MOF-based ontology language
  - we have been unable to find a practical software tool that supports this proposal and mapping to the Semantic Web languages

Transformation-Based Approach

- (Falkovych et al, 2003)
  - transforming domain models between UML and Semantic Web languages
  - focusing on the actual transformation between the two languages

Transformation-Based Approach

- Starting assumptions
  - UML and ontology languages complement one another
    - UML is designed for building models by human experts
    - OWL is designed to be used at runtime to provide guidance to intelligent processing methods
  - the translation is less than trivial, because of the differences between the two languages

Transformation-Based Approach

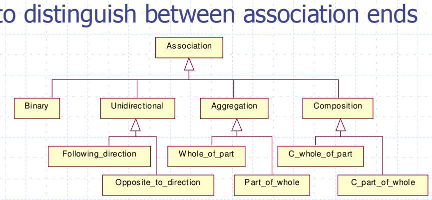
- Extraction of ontologies from UML diagrams
  - there exist large sources of ontological knowledge already available in design documents of existing applications
  - transformation of UML diagrams into DAML+OIL ontologies with preserving the semantics of UML concepts
  - reasoning about these (UML-based) ontologies
  - allows for knowledge interchange between heterogeneous environments

Transformation-Based Approach

- Property mappings
  - the work is based on the UML extension proposed by Backlawski et al.
    - especially on the "Property" problem
  - an association it is mapped into daml:ObjectProperty
  - an attribute is mapped into daml:DatatypeProperty

Transformation-Based Approach

- Taxonomy of association types
  - to preserve semantics of UML associations
  - to distinguish between association ends



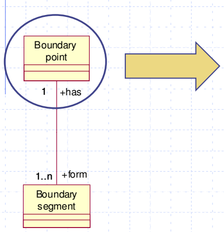
Transformation-Based Approach

- Taxonomy of association types
  - all association types can have a name or be unnamed
  - they can have role names attached to association ends
  - all associations in a particular diagram are modeled as sub-properties of corresponding association types from the taxonomy

Nov 6th, 2005 ISWC2005, Galway, Ireland 124

Transformation-Based Approach

- Example: Unna

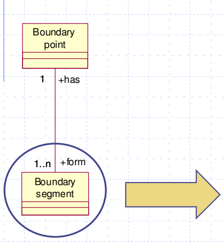



```xml
<daml:Class rdf:ID="Boundary point">
    <rdfs:label>Boundary point</rdfs:label>
    <rdfs:subClassOf>
        <daml:Restriction daml:minCardinality="1">
            <daml:onProperty rdf:resource="#has_G.4"/>
            <daml:toClass rdf:resource="#Boundary segment"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:minCardinality="1">
            <daml:onProperty rdf:resource="#_G.2"/>
            <daml:toClass rdf:resource="#Boundary segment"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="has_G.4">
    <rdfs:subPropertyOf rdf:resource="#_G.2"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="_G.2">
    <rdfs:subPropertyOf rdf:resource="#binary_unnamed"/>
</daml:ObjectProperty>
```

Nov 6th, 2005 ISWC2005, Galway, Ireland 125

Transformation-Based Approach

- Example: Unna




```xml
<daml:Class rdf:ID="Boundary segment">
    <rdfs:label>Boundary segment</rdfs:label>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#from_G.3"/>
            <daml:toClass rdf:resource="#Boundary point"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#G.2"/>
            <daml:toClass rdf:resource="#Boundary point"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="from_G.3">
    <rdfs:subPropertyOf rdf:resource="#G.2"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="G.2">
    <rdfs:subPropertyOf rdf:resource="#binary_unnamed"/>
</daml:ObjectProperty>
```

Nov 6th, 2005 ISWC2005, Galway, Ireland 126

Transformation-Based Approach

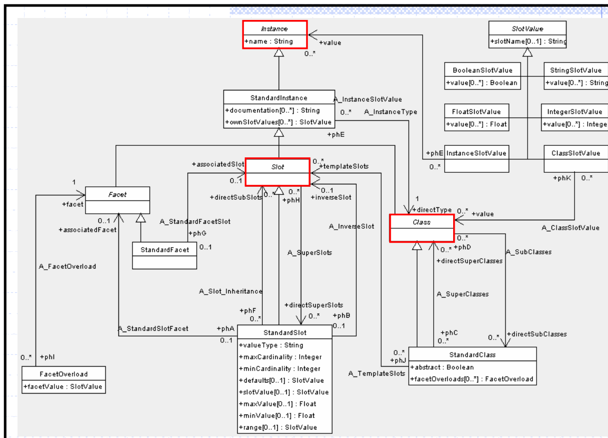
- Characteristics
  - the approach preserves the semantics of UML association in DAML+OIL
    - one can check consistency of UML models using DL-based reasoners (e.g. FaCT)
    - deriving new knowledge by using inference engines
  - XSLT implementation (from XMI to DAML+OIL) – some experiences:
    - XSLT is quite cumbersome for more complex mappings
    - very sensitive to the format of the input file

Protégé and software engineering

- Protégé
  - a leading ontology editor
  - a number of different plug-ins
    - XML, DAML+OIL, OWL (backends), and OIL (tab)
  - integrate ontology design into mainstream software projects (Knublauch, 2003)
    - Protégé objects can be used as Java objects
  - ontology languages have extended declarative power that can improve object-oriented models

Protégé and software engineering

- Protégé's metamodel
  - MOF-compatible metamodel
  - at the same metalevel as UML
  - Protégé basically implements an adapted version of the OKBC metamodel
  - improved interoperability with other systems
  - extensible and adaptable
    - e.g. a new ontology language can be added



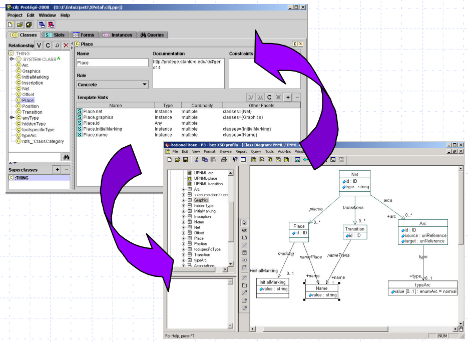
Protégé and software engineering

- Protégé's MOF-based plug-ins
  - XMI backend
    - the XMI format of the Protégé's metamodel
  - UML backend
    - the UML XMI format
  - these two backends use the NetBeans' MetaData Repository (MDR) http://mdr.netbeans.org

Protégé and software engineering

- Protégé's MOF-based plug-ins
  - MOF models can be accessed in Java using Java Metadata Interface (JMI)
  - JMI takes a MOF metamodel and generates Java interfaces
  - this can be an alternative way for creating a Protégé knowledge base

Protégé's UML backend



Protégé's UML backend

- Protégé to UML mapping
  - Classes – Classes
  - Inheritance – Inheritance
  - Primitive Slots – Attributes
  - Instance Slots – Associations
- UML to Protégé mapping
  - when the name of a new slot is already taken by a different slot: <attributeName>@<className>

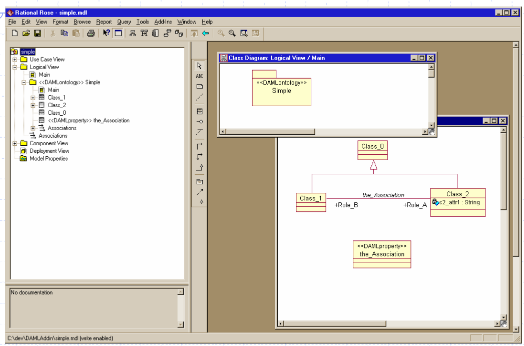
Protégé's UML backend

- UML extensions (UML Profiles) are supported
  - only unidirectional support, for export from Protégé to UML
- UML tools
  - Poisedon for UML v1.6 (recommended)
  - Rational Rose
  - MagicDraw 6.0
  - Together Control Center 6.0

DAML UML Enhanced Tool (DUET)

- DUET characteristics
  - UML visualization and authoring environment for DAML
  - core DAML concepts are being mapped into UML through a UML profile for DAML
  - capability to work with multiple ontologies simultaneously
  - implementation:
    - an add-in to Rational Rose
    - for use with ArgoUML

DUET Rational Rose Add-in



DAML UML Enhanced Tool (DUET)

- From UML to DAML
  - http://codip.grci.com/Tools/Tools.html

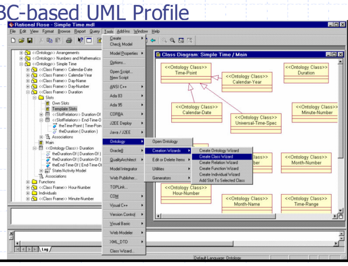
UML	DAML
<<DAMLontology>> Package	Ontology
Class	Class
<<DAMLproperty>> Class	Property
<<DAMLdatatype>> Class	Datatype
Data Type	XML Schema or DAML datatype
Associations and Roles	Object Property and Restriction
Attributes	Datatype Property
Generalization	subClassOf or subPropertyOf
Import	Import
Fully Qualified Names	Namespaces

Xpetal

- Transformation of UML models to RDF(S)
  - http://www.langdale.com.au/styler/xpetal
  - Java implementation
  - similar mapping rules as the Cranefield's XSLT and UML to RDF mappings from the OMG Data Access Facility (DAF) specification
  - only for Rational Rose models in the .mdl format
  - it is not based on standards (i.e., XMI)

Visual Ontology Modeler (VOM)

- Rational Rose plug-in
  - Sandpiper Software Inc and Knowledge System Lab (KSL)
  - OKBC-based UML Profile



Credit:
Elisa F. Kendall

Short summary of analyzed approaches

Approach	Metamodel	Model description	Transformation mechanism	Generated ontology language
Cranefield	Standard UML	UML XMI	XSLT	RDFS, Java classes
Backlawski et al	UML Profile, MOF-based ontology language	not given (XMI)	-	DAML
Falkovych at al	Standard UML	UML XMI	XSLT	DAML + OIL
Protégé	Protégé metamodel	Protégé XMI	Programmed	OWL, RDF(S), DAML+OIL, XML, UML XMI, Protégé XMI, …
	Standard UML	UML XMI		
DUET	UML Profile	Rational Rose, ArgoUML	Programmed	DAML+OIL
Xpetal	Standard UML	Rational Rose .mdl files	Programmed	RDFS

Main differences

- UML and ontology languages (Backlawski et al., 2002)
  - Monotonic and non-monotonic
  - Metalevels
  - Specialization/ generalization
  - Modularity
  - Containers and lists
  - Property
  - Class constructors
  - Cardinality constraints
  - Subproperties
  - Namespaces

UML-based ontology development

- Short summary
  - all these approaches are useful
  - none of them contains:
    - a full description of a new, MDA-based ontology language
    - a related UML Profile
    - transformations between these two languages
    - transformations to Semantic Web languages
  - solution: the OMG's initiative for Ontology Definition Metamodel (ODM)

OMG's Initiative: Ontology Definition Metamodel and Preliminary Submission

OMG's Request for Proposal (RFP)

- UML could be a means towards more rapid development of ontologies:
  - familiarity of users with UML
  - availability of UML tools
  - existence of many domain models in UML
  - similarity of those models to ontologies
  - a number of tools and investigations have demonstrated that using UML-based tools for developing ontologies can be practical

OMG's Request for Proposal (RFP)

This approach continues the Object Management Group's "gradual move to more complete semantic models". It would also create a link between the UML community and the emerging Semantic Web community, much as other metamodels and profiles have created links with the developer and middleware communities.

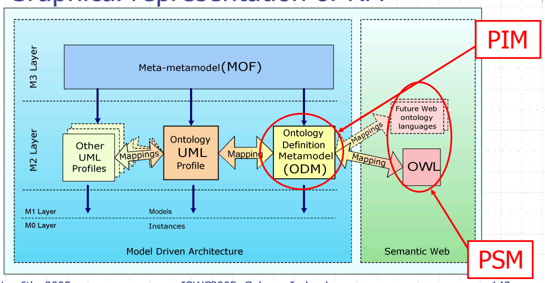
OMG Document: ad/2003-03-40

OMG's RFP: Ontology Definition Metamodel (ODM)

- Several specifications should be defined
  - a standard MOF2-compliant metamodel for Ontology Definition Metamodel (ODM) – PIM
  - a UML2 Profile to support reuse of UML notation for ontology definition
  - a mapping from ODM to the UML profile
  - a language mapping for the ODM to OWL DL

OMG's RFP: Ontology Definition Metamodel (ODM)

- Graphical representation of RFP



ODM Specification Requirements

- Mandatory Requirements
  - define ODM using MOF2 Core that represents the semantics of ontologies, including but not necessarily limited to OWL ontologies
    - depict ODM using UML
    - it must be based on the appropriate elements of 2.x versions of OMG metamodels
  - a UML2 Profile extending the UML2 metamodel for ontology definition

ODM Specification Requirements

- Mandatory Requirements
  - forward and reverse engineering of logically equivalent ontologies between environments
    - iterative development of ontologies
  - a language mapping from ODM to OWL DL
    - this mapping should be two-way and bounded
  - an XMI for MOF2.0-compliant W3C XML Schema Description (xsd) based on ODM

ODM Specification Requirements

- Optional Requirements
  - mapping multiple ontologies into a single UML model
  - round-trip engineering across language environments
  - a mapping for DAML+OIL, or other ontology languages, with the same characteristics as those described for OWL

ODM Specification Requirements

- Optional Requirements
  - a profile, a metamodel, and mappings needed to support OWL Full
    - RDF and RDFS would also be supported as a consequence
  - design ODM to allow for specification of ontology metadata to describe context and scope of the development and intended use of a corresponding ontology

ODM Initial Submissions

- Submitters
  - International Business Machines (IBM)
  - Distributed Systems Technology Centre (DSTC)
  - Gentleware & AT&T (sponsor)
  - Sandpiper Software, Inc. & Knowledge Systems Laboratory (KSL), Stanford University

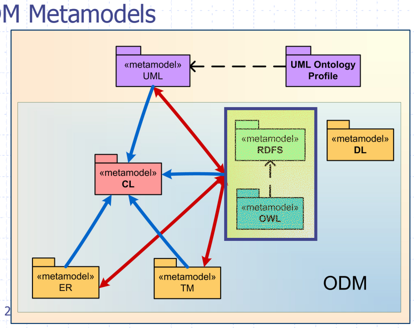
Summary ODM Proposals

- Metamodels

Proposal	Ontology language-independent	MOF-compliant metamodel	UML Extension	Explicit UML Ontology Profile	OWL-dependent metamodel
IBM	–	+	–	+	+
DSTC	+	+	+	–	–
Gentleware	–/+	+	+	+	–/+
Sandpiper & KSL	+	+	–	–	–

OMG ODM Current Proposal

- ODM Metamodels

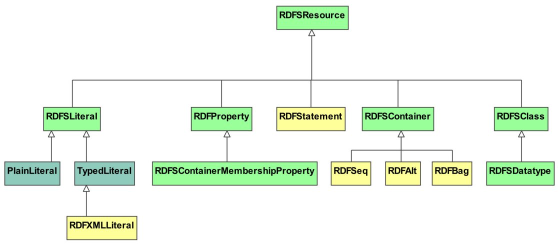


OMG ODM Current Proposal

- Some ODM issues w.r.t. MOF2
  - ODM proposal was tailored w.r.t. theECore metamodel
  - The name of any element within the same namespace must be unique (e.g. the *contains* association)
  - Many associations were left unnamed in ODM joint submission – MOF2 incompatible
    - if an association does not have direct RDFS counterpart, verbs should be used for names

OMG ODM Current Proposal – RDFS Metamodel

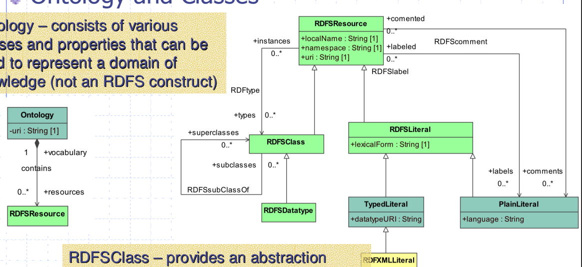
- RDFSResource – All things described by RDF are called resources



OMG ODM Current Proposal – RDFS Metamodel

- Ontology and Classes

Ontology – consists of various classes and properties that can be used to represent a domain of knowledge (not an RDFS construct)



RDFSClass – provides an abstraction mechanism for grouping similar resources

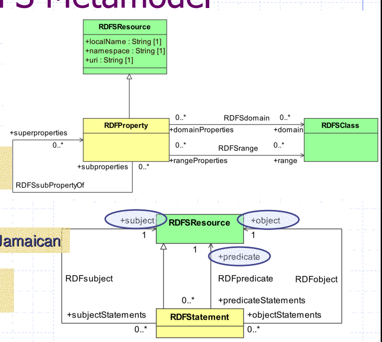
OMG ODM Current Proposal – RDFS Metamodel

- Property and Statement

Property – Relates Resources to Resources or Literals

Bob Marley was Jamaican

Statement – Conects concrete Resources

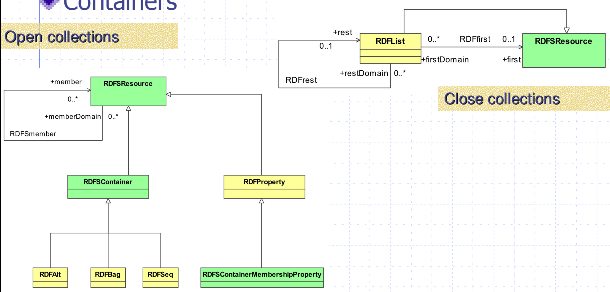


OMG ODM Current Proposal – RDFS Metamodel

- Containers

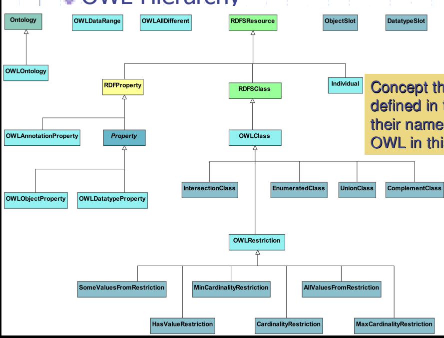
Open collections

Close collections



OMG ODM Current Proposal – OWL Metamodel

- OWL Hierarchy

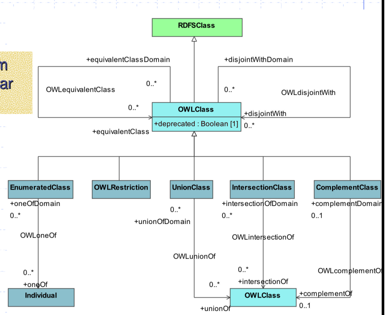


Concept that are not explicitly defined in the OWL language – their names are not prefixed with OWL in this metamodel

OMG ODM Current Proposal – OWL Metamodel

- Classes

Class – an abstraction mechanism for grouping Individuals with similar characteristics



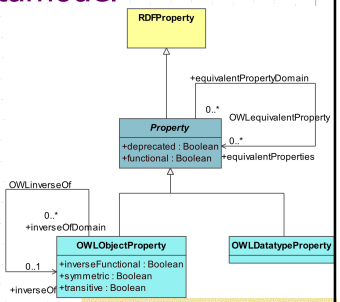
OMG ODM Current Proposal – OWL Metamodel

- Properties

Property – Relates Individual to another Individual or DataValue (not an OWL construct)

OWLObjectProperty – Relates Individual to another Individual

OWLFunctionalProperty
OWLTransitiveProperty
OWLSymmetricProperty
OWLInverseFunctionalProperty



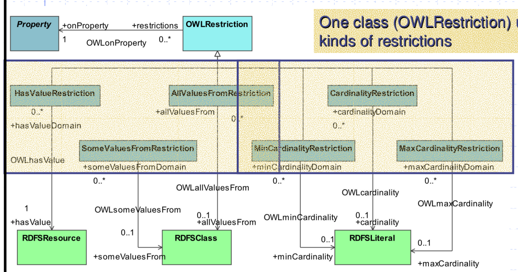
OWLDatatypeProperty – Relates Individual to another Datavalue

OMG ODM Current Proposal – OWL Metamodel

- OWL Restriction

OWLRestriction – An anonymous Class of all Individuals that satisfy certain Property restriction

One class (OWLRestriction) used for all kinds of restrictions

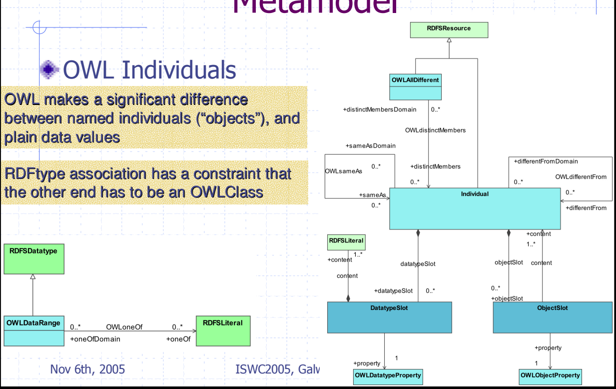


OMG ODM Current Proposal – OWL Metamodel

- OWL Individuals

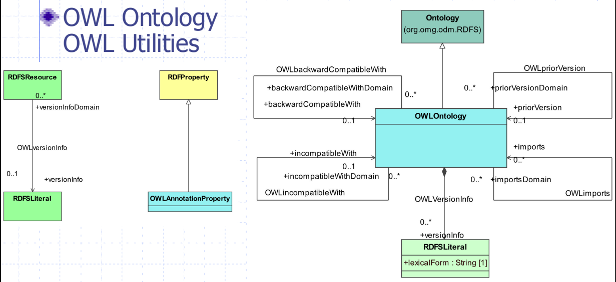
OWL makes a significant difference between named individuals (“objects”), and plain data values

RDFtype association has a constraint that the other end has to be an OWLClass



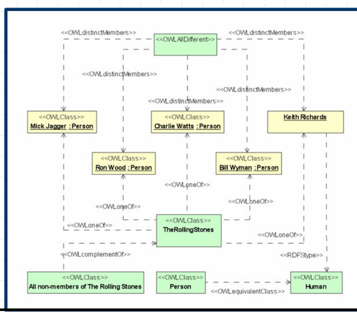
OMG ODM Current Proposal – OWL Metamodel

- OWL Ontology
  OWL Utilities



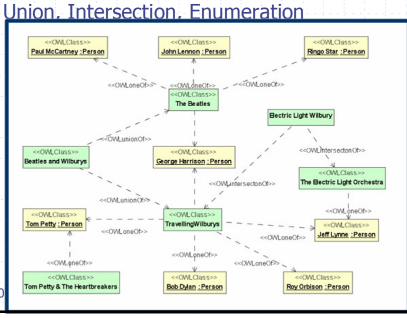
OMG OUP Current Proposal

- Ontology UML Profile (OUP) – OWL Classes



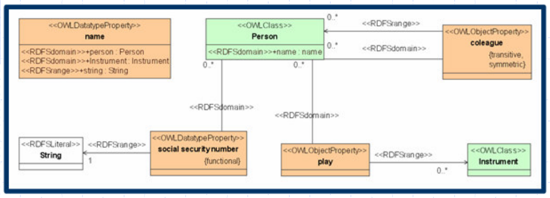
OMG OUP Current Proposal

- Ontology UML Profile (OUP) – OWL Classes
  - Union, Intersection, Enumeration



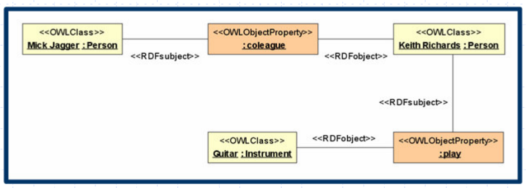
OMG OUP Current Proposal

- Ontology UML Profile (OUP) – OWL Properties



OMG OUP Current Proposal

- Ontology UML Profile (OUP) – OWL Statement



ODM Schedule from July 2004

- March 28th 2003 – RFP Released
- August 18th – Initial Submissions
- S
- J

And, where are we now?

Well, standardization is a long process ☺…

- October 11th – Revised Submissions deadline
- November ~2nd – Revised Presentations
- February 2005 – ADTF & PTC votes to Recommend
- April 2005 – Board of Directors votes to Adopt

Modeling Spaces

Practical challenges

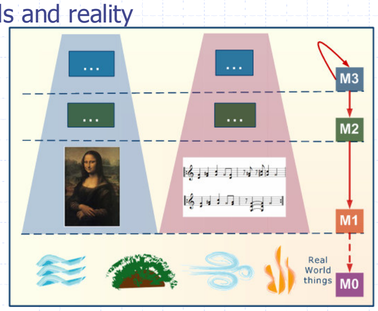
- Questions that could be confusing:
  - Should we assume the code we write as a model or not?
  - What are models and metamodels and why do we need them?
  - How do ontologies corresponds to models?
  - What means transforming a UML model into a programming language or XML?
  - …
- Modeling spaces

Modeling spaces

- Essentials
  - Model is a simplified abstraction of reality
    - something can be taken as a model if it is an abstraction of things from the real world, but it is simultaneously a thing from the real world (viewpoint!)
    - models can be defined using metamodeling concepts formally or implicitly

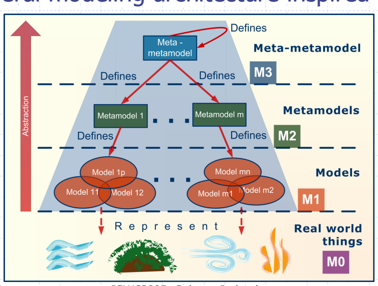
Modeling spaces

- Models and reality



Modeling spaces

- A general modeling architecture inspired by MDA

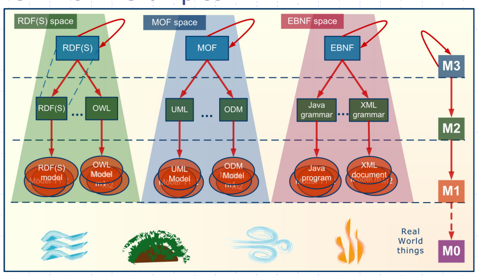


Modeling spaces

- Definition
  - A modeling space (MS) is a modeling architecture defined by a particular metametamodel
  - Metamodels defined by a metametamodel and models defined by those metamodels represent the real world from one point of view, i.e. from the point of view of that MS
  - Metamodels are a means (languages) for defining models, i.e. a general viewpoint

Modeling spaces

- Well-known examples

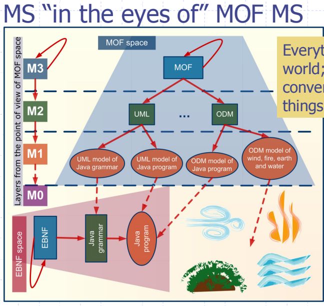


Modeling spaces

- Perspectives or dualities
  - If we model the real world in a certain MS (MOF MS), we use some models (UML)
  - If we model the same reality in another MS (RDF(S)), we describe it with different kinds of models (ontology)
    - highlight other characteristics when abstracting from reality
  - The models from the first MS can be a part of reality that we model using the models from the second MS

Modeling spaces

- EBNF MS "in the eyes of" MOF MS



Everything is in the real world; we just use a convention to put some things in layers

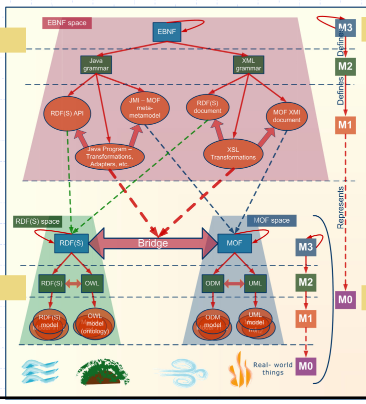
MOF and RDF(S) Modeling Spaces

- Two types of modeling spaces
  - *Orthogonal spaces*
    - one MS is represented in another MS
    - in round-trip engineering to facilitate different stages of modeling some system
    - Java and Java UML Profile
  - *Parallel spaces*
    - one MS models the same set of real-world things as another MS, but in another way
    - RDF(S) ontologies and MOF-based models

MOF and RDF(S) Modeling Spaces

Concrete MS

Orthogonal MS



Conceptual MS

Parallel MS

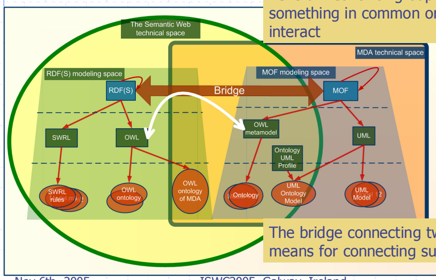
Semantic Web and MDA Technical Spaces

- Technical Spaces
  - A working context with a set of additional concepts, body of knowledge, tools, required skills, and possibilities
  - Inspiration for modeling spaces
  - A technical space (TS) is a working context that includes various related MSs, and often it is built around one MS
    - MDA MS – EBNF MS (JMI and XMI), MOF MS

Semantic Web and MDA Technical Spaces

- Overlapping TS through different MS

TS is a means for grouping MSs that have something in common or simply need to interact

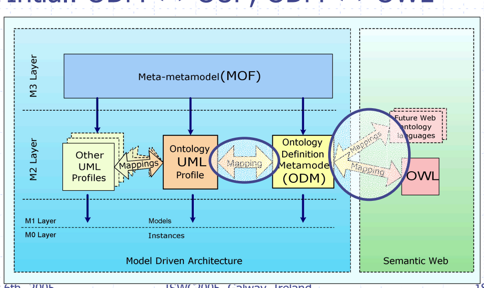


The bridge connecting two MSs is also a means for connecting surrounding TSs

Mappings of MDA-based languages and ontologies

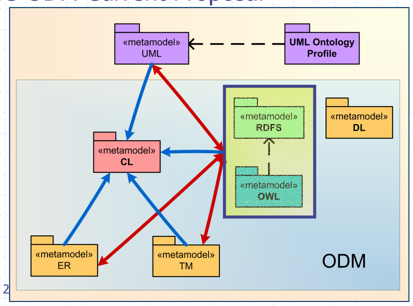
Mappings

- Intial: ODM<->OUP, ODM<->OWL



Mappings

- OMG ODM Current Proposal

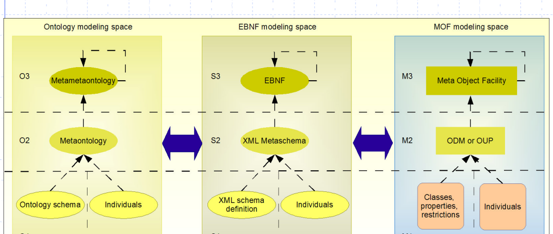


Mappings

- Important facts for mappings
  - modeling and technological spaces (Djurić et al, 2006)(Kurtev et al, 2002)
    - ontologies, MDA, and XML
  - epistemological equivalence between moldeling space layers
  - implementation techniques
    - XSLT, MOF2 QVT, RDQL, Java

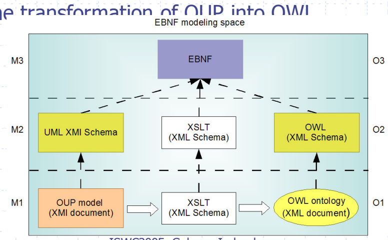
Epistemological equivalences between modeling layers

- Metamodel-driven model transformations



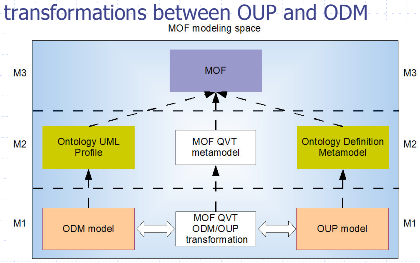
An example of a mapping

- A transformation in the EBNF modeling space
  - (XML technical space)
  - the transformation of OUP into OWL



An example of a mapping

- Transformations in the MOF modeling space
  - transformations between OUP and ODM



Transformation techniques

- Implementation technologies for the transformations

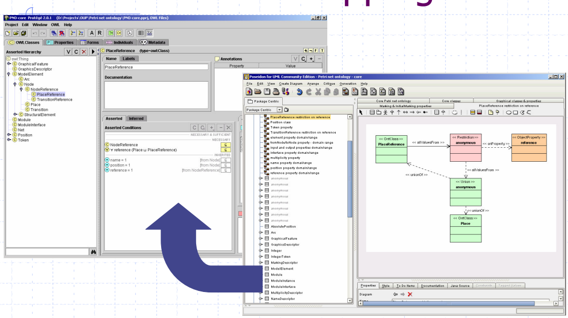
Target language (columns) / Source language (rows)

Source language \ Target language	ODM	OUP	OWL
ODM	–	EBNF MS: XSLT; MOF MS: QVT	EBNF MS: XSLT
OUP	EBNF MS: XSLT; MOF MS: QVT	–	EBNF MS: XSLT
OWL	EBNF MS: Programmed*, XSLT	EBNF MS: Programmed*, XSLT	–

What is implemented?

- Implemented mappings
  - Ontology UML Profile to OWL
    - XSLT – UML XMI to OWL XML
    - acts as an extension of UML tools
    - tested on Poseidon for UML CE
    - sharing between Poseidon for UML and Protégé

OUP to OWL Mapping: XSLT



What is implemented?

- Implemented mappings
  - ODM to OWL – XSLT
    - XSLT – ODM XMI to OWL XML
    - export option of the AIR ontology editor
  - …

MDA-Based Infrastructure for Ontology Development

MDA-Based Ontology Editor

- Approaching AI and software engineering
  - different AI knowledge representation techniques
    - frame systems, logics, rules, fuzzy systems, neural networks, Petri nets, …
  - OBOA – Object-oriented
  - AIR – Model Driven Architecture
    - http://goodoldai.org.yu

OBOA / GET-BITS

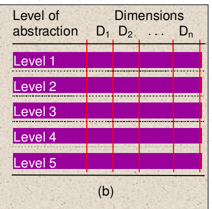
- OBOA - OBject-Oriented Abstraction (Devedžić et al., 1996-present)
  - a model of intelligent systems
  - a framework for developing intelligent systems
  - a set of software components for building intelligent systems
  - a layered reference architecture and ontology of intelligent systems

OBOA / GET-BITS

- OBOA - OBject-Oriented Abstraction (Devedžić et al., 1996-present)

Level of abstraction	Objective	Semantics
Level 1	Integration	Multiple agents or systems
Level 2	System	Single agent or system
Level 3	Blocks	System building blocks
Level 4	Units	Units of blocks
Level 5	Primitives	Parts of units

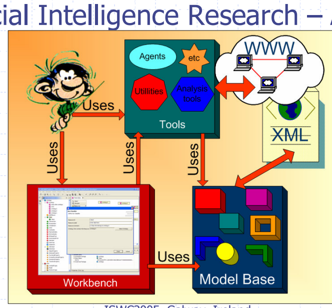
(a)



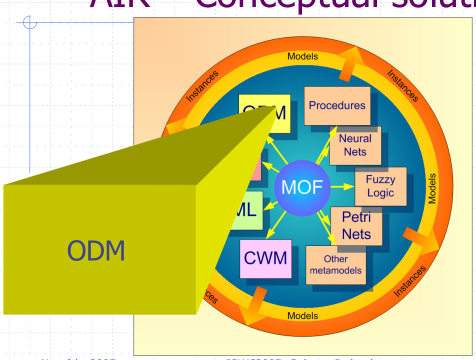
(b)

AIR

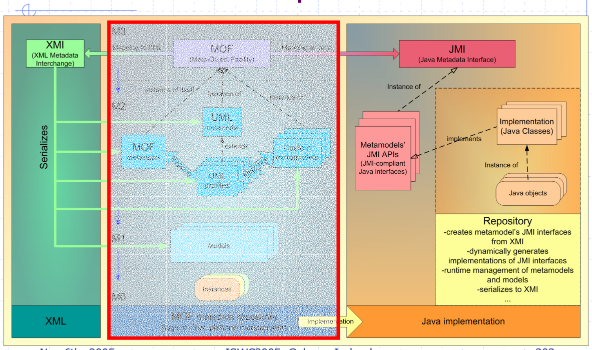
- Artificial Intelligence Research – AIR



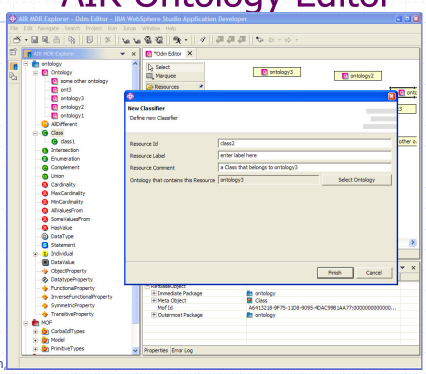
AIR – Conceptual solution



AIR – Conceptual solution



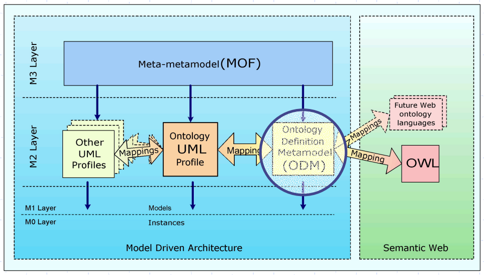
AIR Ontology Editor



AIR Ontology Editor

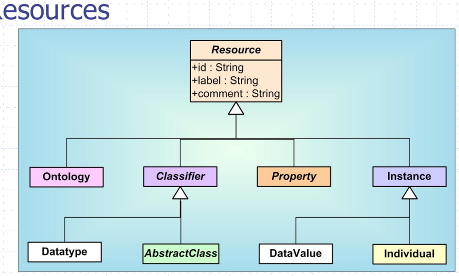
- AIR ontology plug-in
  - Ontology editor – an AIR plug-in
    - JMI-compliant repository
    - NetBeans' MDR
  - Based on GOOD OLD AI ODM and OUP
    - http://goodoldai.org.yu
  - XSLT for exporting into OWL

MDA-Based Ontology Infrastructure

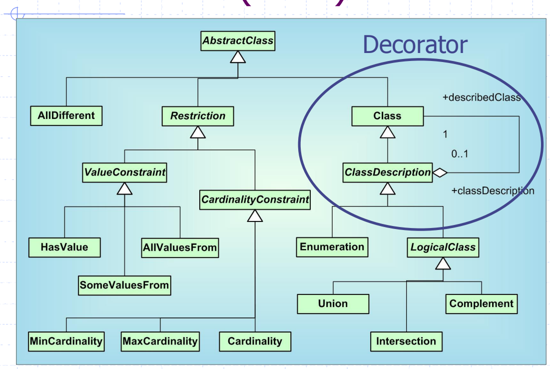


Ontology Definition Metamodel (ODM)

- Resources

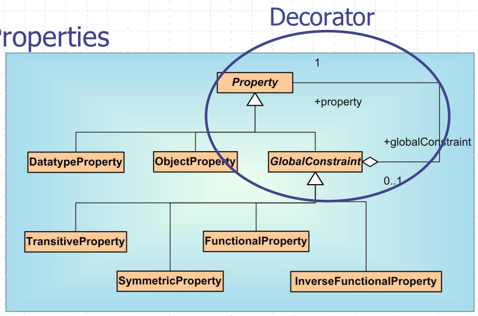


Ontology Definition Metamodel (ODM)



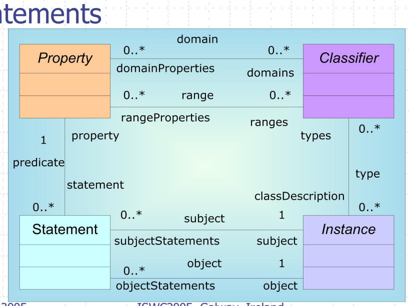
Ontology Definition Metamodel (ODM)

- Properties

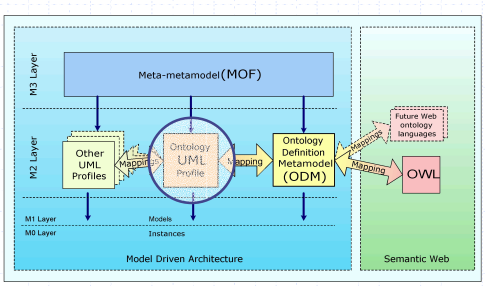


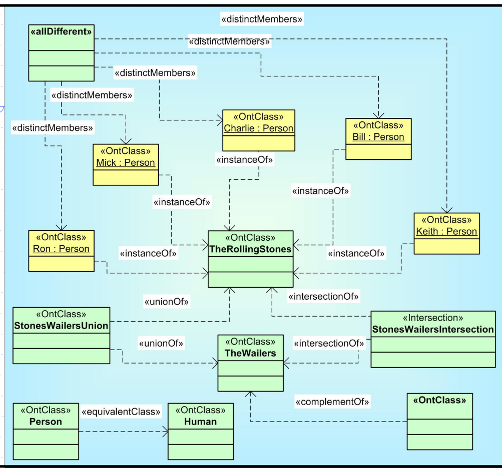
Ontology Definition Metamodel (ODM)

- Statements



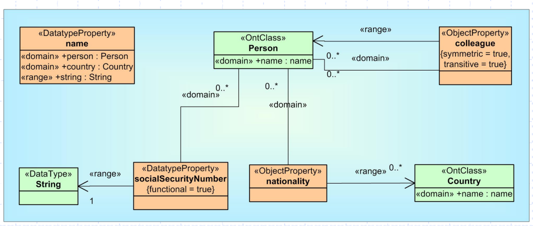
Ontology UML Profile





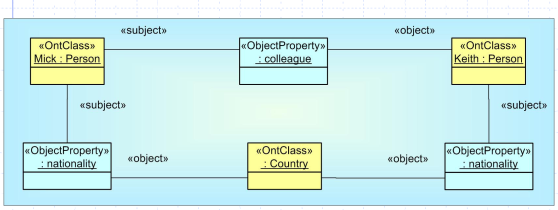
Ontology UML Profile

- Properties



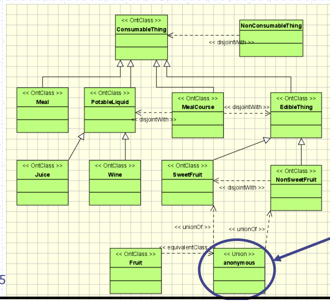
Ontology UML Profile

- Statements



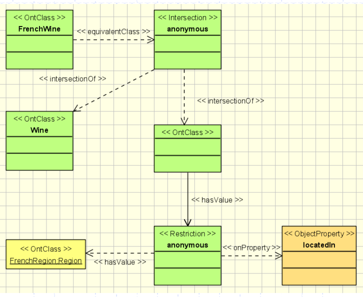
The Wine ontology

- Well-known example

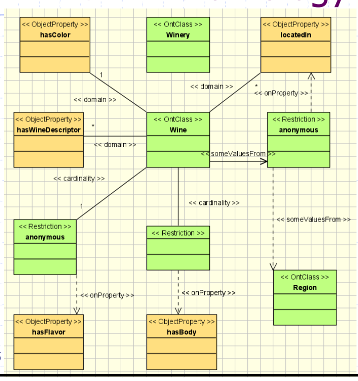


Tagged value
odm.anonymous

The Wine ontology

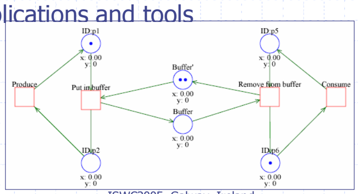


The Wine ontology



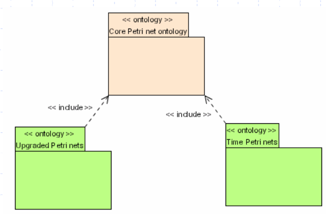
The Petri net ontology

- Petri nets – a formal tool
  - modeling, simulation, and analysis
  - directed bipartite graph
  - applications and tools



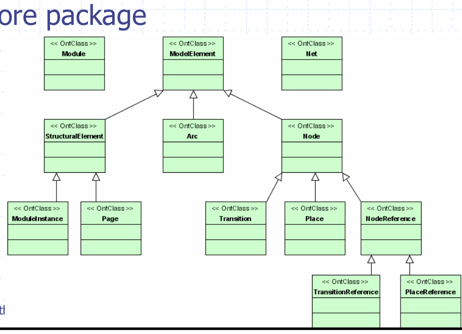
The Petri net ontology

- Modular organization (example)



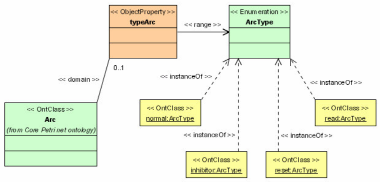
The Petri net ontology

- Core package



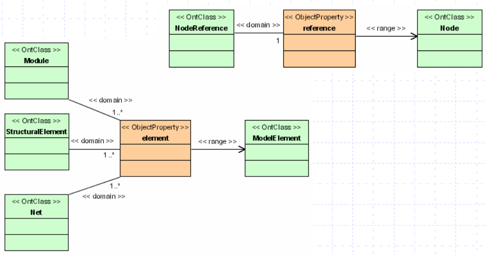
The Petri net ontology

- Core package - enumeration



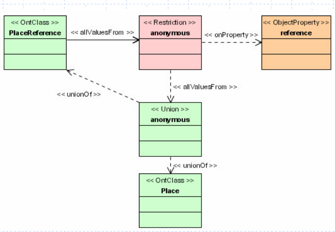
The Petri net ontology

- Core package - properties



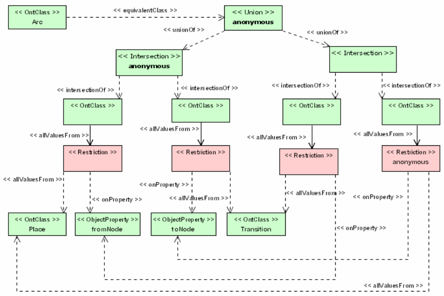
The Petri net ontology

- Core package - restrictions



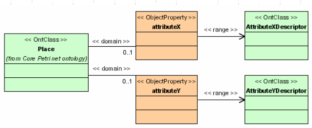
The Upgraded Petri net ontology

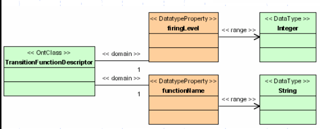
- Extension package – new restrictions



The Upgraded Petri net ontology

- Extension package – new classes and properties





MDA Standards for Ontology Development

Dragan Gašević[1], Vladan Devedžić[2], Dragan Djurić[2]

[1]Simon Fraser University Surrey, Canada

[2]University of Belgrade, Serbia and Montenegro

dgasevic@acm.org, devedzic@fon.bg.ac.yu, dragandj@gmail.com

http://goodoldai.org.yu