



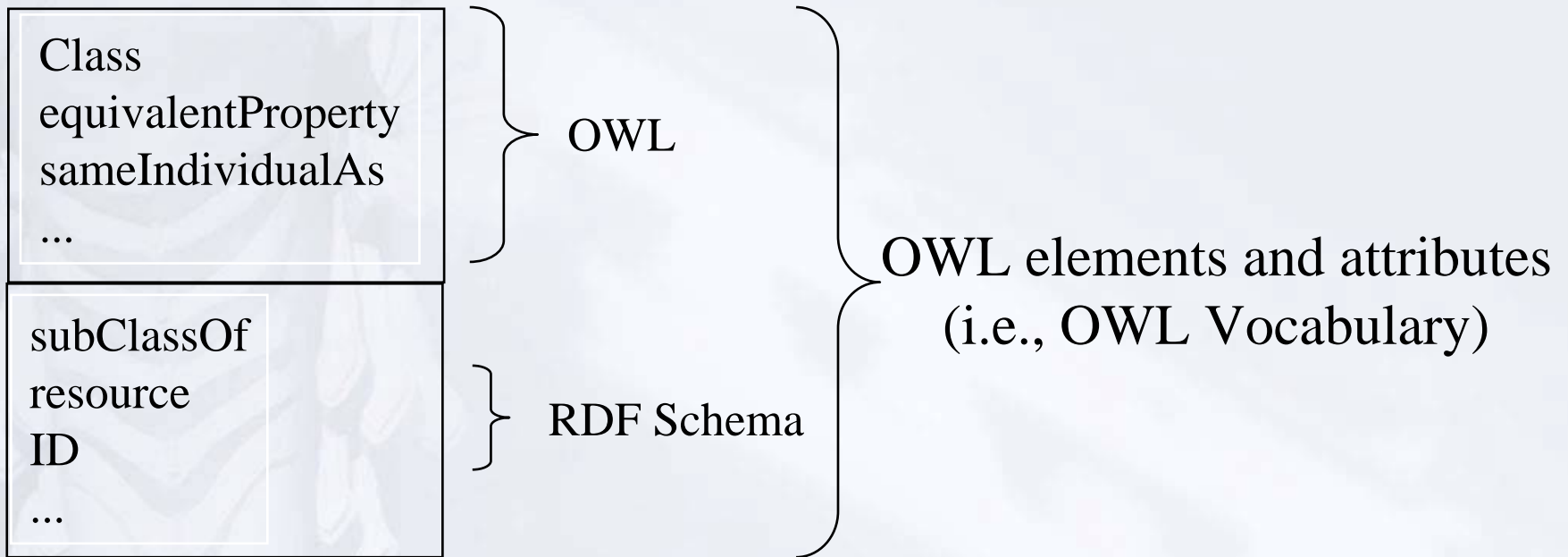
# OWL and Protégé

Hoang Huu Hanh  
IFS – TU Wien

- Finding issues
  - Roles of OWL and Protégé 2000
  - Ontology-based search and selection
  - A Formal Approach for Querying the Semantic Web
- Future works
  - Description Logics

# OWL - What's OWL?

- OWL is a set of XML elements and attributes, with standardized meaning, that are used to define terms and their relationships
- OWL extends RDF Schema:



- OWL provides an agreed-upon vocabulary for expressing semantics

## Samples of OWL Vocabulary:

**subClassOf:** this OWL element is used to assert that one class of items is a subset of another class of items.

Example: SLR is a subClassOf Camera.

**equivalentProperty:** this OWL element is used to assert that one property is equivalent to another.

Example: aperture is an equivalentProperty to f-stop.

**sameIndividualAs:** this OWL element is used to assert that one instance is the same as another instance.

Example: Vienna University of Technology is the sameIndividualAs TU-Wien.

**Symmetric:** if  $P(x,y)$  then  $P(y,x)$

**InverseOf:** if  $P1(x,y)$  then  $P2(y,x)$

**Transitive:** if  $P(x,y)$  and  $P(y,z)$  then  $P(x,z)$

**Functional:** if  $P(x,y)$  and  $P(x,z)$  then  $y=z$

**InverseFunctional:** if  $P(x,y)$  and  $P(z,y)$  then  $x=z$

**allValuesFrom:**  $P(x,y)$  has  $y=\text{allValuesFrom}(C)$

**someValuesFrom:**  $P(x,y)$  has  $y=\text{someValuesFrom}(C)$

**hasValue:**  $P(x,y)$  and  $y=\text{hasValue}(I)$

**cardinality:**  $\text{cardinality}(P) = n$

**minCardinality:**  $\text{minCardinality}(P) = n$

**maxCardinality:**  $\text{maxCardinality}(P) = n$

**equivalentProperty:**  $P1 = P2$

**intersectionOf:**  $C = \text{intersectionOf}(C1, C2, \dots)$

**unionOf:**  $C = \text{unionOf}(C1, C2, \dots)$

**complementOf:**  $C = \text{complementOf}(C1)$

**oneOf:**  $C = \text{oneOf}(I1, I2, \dots)$

**equivalentClass:**  $C1 = C2$

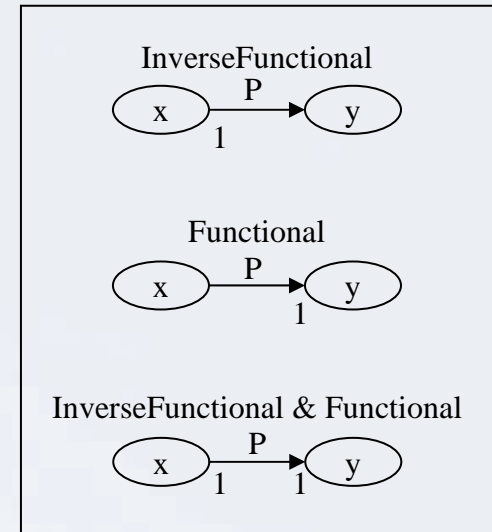
**disjointWith:**  $C1 \neq C2$

**sameIndividualAs:**  $I1 = I2$

**differentFrom:**  $I1 \neq I2$

**AllDifferent:**  $I1 \neq I2, I1 \neq I3, I2 \neq I3, \dots$

**Thing:**  $C1, C2, \dots, I1, I2, \dots, P1, P2, \dots$



## LEGEND

Properties:  $P, P1, P2, P3, \dots$

Generic classes:  $C, C1, C2, C3, \dots$

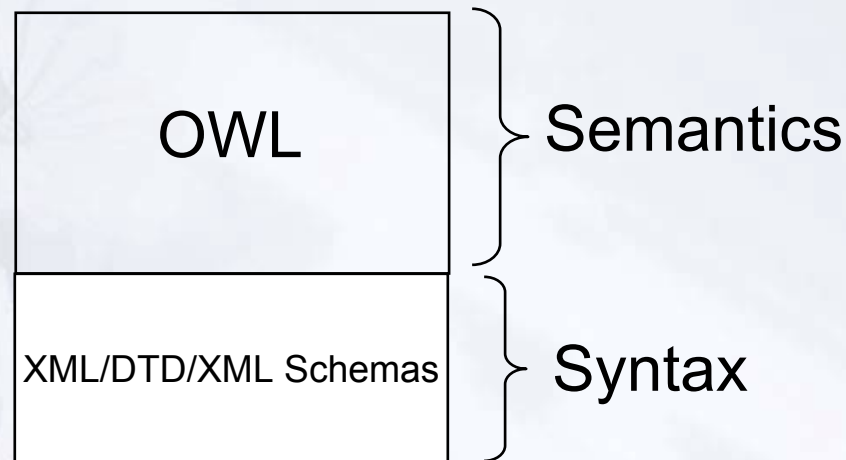
Specific classes:  $x, y, z$

Instances:  $I, I1, I2, I3, \dots$

A non-negative integer:  $n$

$P(x,y)$  is read as: "property  $P$  relates domain  $x$  to range  $y$ "

- OWL enables machines to understand data



*OWL enables machine-processable semantics!*

# OWL and Interoperability

- OWL plays an important role in data interoperability
- Two necessary conditions for interoperability:
  - Adopt a common syntax: this enables applications to *parse* the data. XML provides a common syntax, and thus is a critical first step.
  - Adopt a means for understanding the semantics: this enables applications to *use* the data. OWL provides a standard way of expressing the semantics.

- Example: consider following XML fragment

```
<SLR>  
...  
</SLR>
```

- SLR = *Single Length Reflex (camera)* or *Satellite Laser Ranging* or others...?

## What is an SLR?

## Meaning (semantics) applied on a per-application basis

```
<SLR>  
...  
</SLR>
```

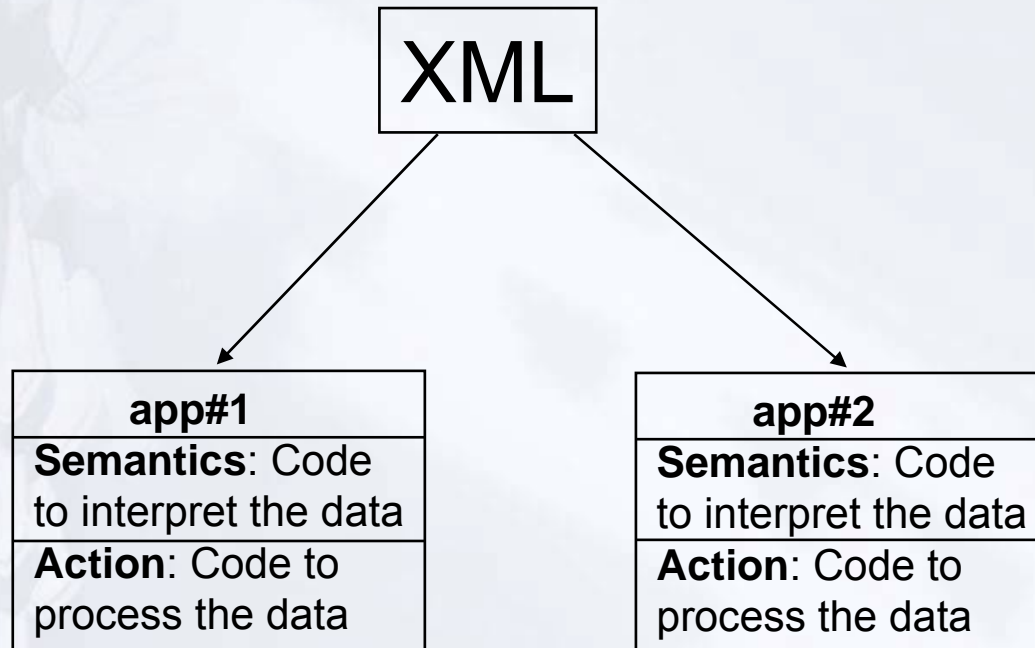
application

**Semantics:** A SLR is a type of Camera. (i.e., SLR = Single Lens Reflex)

**Actions:** These actions must be performed on the SLR data:

- check the aperture.
- check the (lens) size.
- check the cost.

Meaning (semantics) applied on a per-application basis

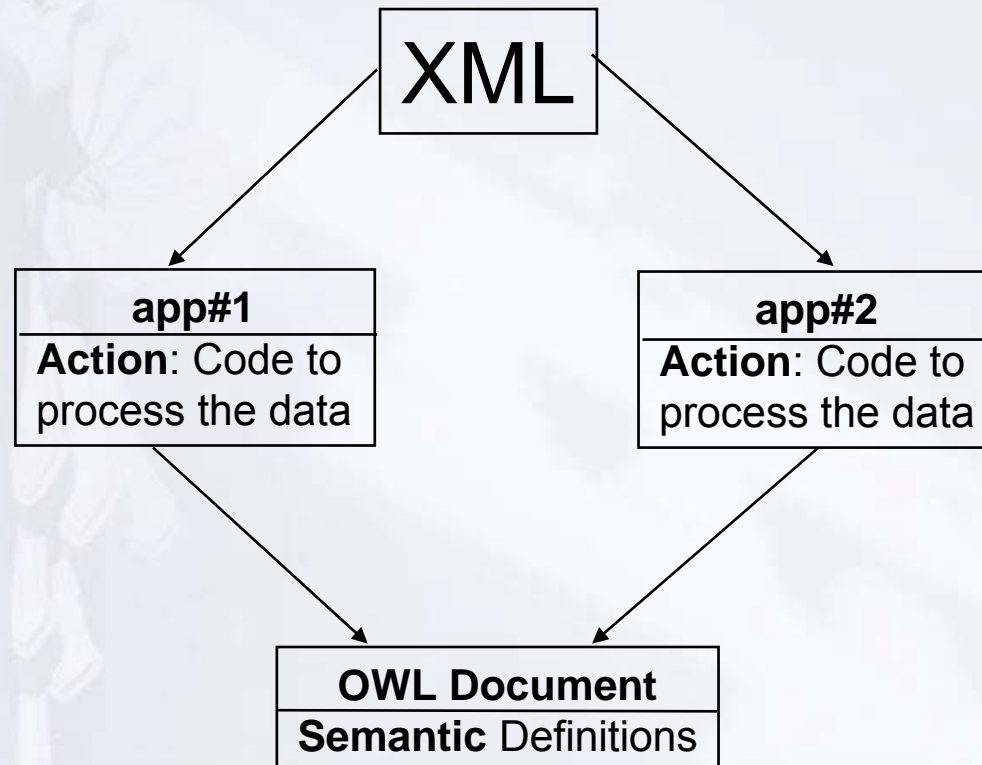


# OWL & Interoperability (cont'd)

- Problems encounter:
  - Duplicate effort
    - Each application must express the semantics
  - Variability of interpretation
    - Each application can take its own interpretation
    - Example: Mars probe distance - one application interpreted the data in inches, another application interpreted the data in centimetres.
  - No ad-hoc discovery and exploitation
    - Applications have the semantics pre-wired. Thus, when new data is encountered an application may not be able to effectively process it. This makes for brittle applications.

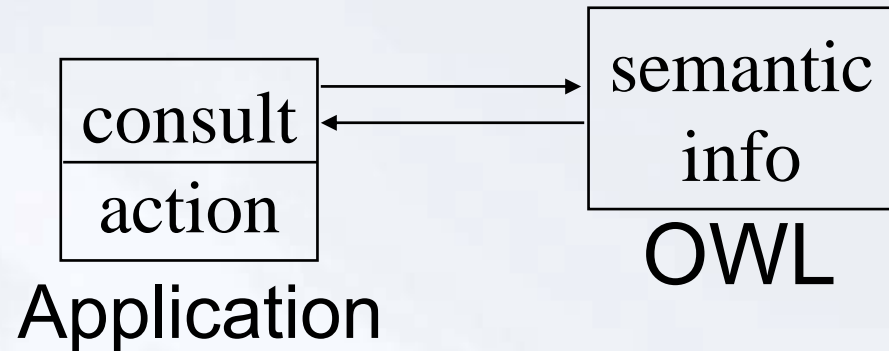
# OWL & Interoperability (cont'd)

- Better approach:
  - Extract semantic definitions from applications
  - Express semantic definitions in a standard vocabulary

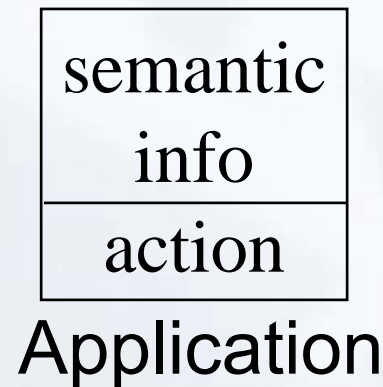


# Why not hardcode the info that is in an OWL document directly into applications?

*Why design your application like this:*



*And not like this:*



# The Answer Is...

- There are many reasons for keeping the OWL semantic info separate from the application:
  - **Leverage XML:** an OWL document is an XML document. Thus, XML tools can be used to create, edit, and validate OWL documents.
  - **Extensible:** the OWL document can grow independent of the application.
  - **Reusable:** the OWL document can be used by multiple applications.
  - **Avoid Misinterpretation:** the info in an OWL document is expressed using a well-defined, standardized vocabulary. This helps avoid misinterpretation.

## Bridging the Terminology Gap using OWL:

A key problem in achieving interoperability is to be able to recognize that two pieces of data are talking about the same thing, even though different terminology is being used.

- Example: Define the terms "Camera" and "SLR". State that SLRs are a type of Camera.

Here's how these two terms (classes) and their relationship is defined using the OWL vocabulary:

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

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

- *Scenario:*

- I am interested in purchasing a camera with a 75-300mm zoom lens size, 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.
- Assume that there exists an OWL Camera Ontology, which the web-bot can “consult” upon its travels across the web.

# Is This Document Relevant?

The Web Bot finds  
this document at a  
website:

Is it relevant?

(Note: SLR = Single Lens  
Reflex)

```
<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?

```
<PhotographyStore rdf:ID="Hunts"
                  xmlns:rdf="&rdf;">
  <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>
```

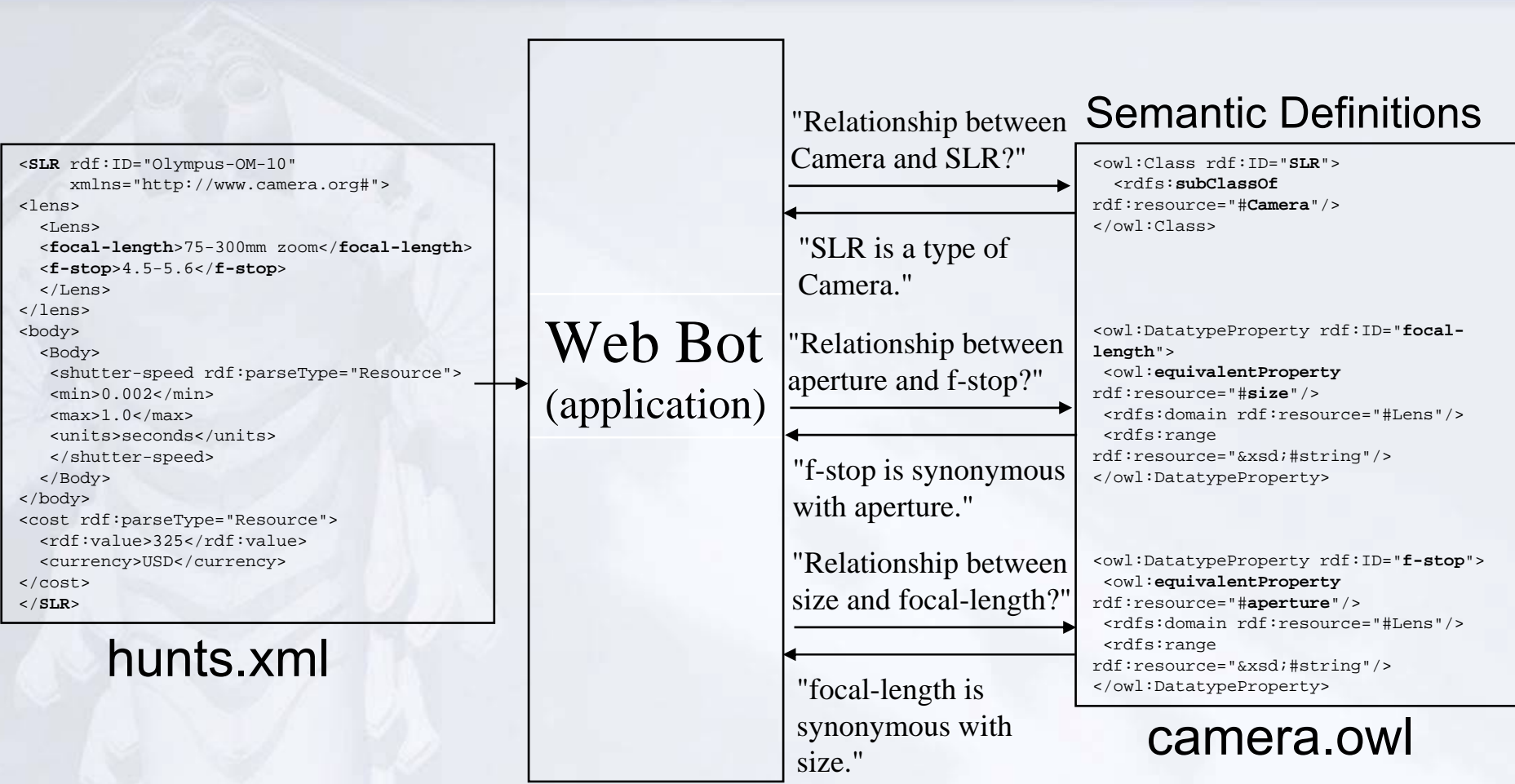
Match?

I am interested in purchasing a camera with a 75-300mm zoom lens (size) that has an aperture of 4.5-5.6, and a shutter speed that ranges from 1/500 sec. to 1.0 sec.

To determine if there is a match, these questions must be answered:

1. What's the relationship between "SLR" and "Camera"?
2. What's the relationship between "focal-length" and "size"?
3. What's the relationship between "f-stop" and "aperture"?

# XML document is a match!



```
<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>
```

hunts.xml

Web Bot  
(application)

"Relationship between Camera and SLR?"  
 "SLR is a type of Camera."  
 "Relationship between aperture and f-stop?"  
 "f-stop is synonymous with aperture."  
 "Relationship between size and focal-length?"  
 "focal-length is synonymous with size."

### Semantic Definitions

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

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

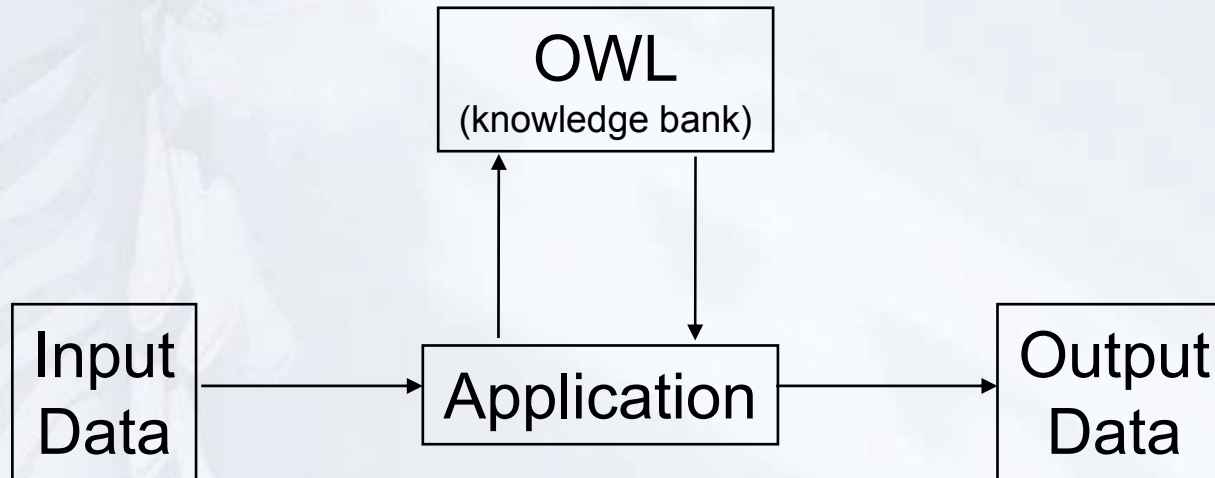
<owl:DatatypeProperty rdf:ID="f-stop">
  <owl:equivalentProperty
    rdf:resource="#aperture"/>
  <rdfs:domain rdf:resource="#Lens"/>
  <rdfs:range
    rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
```

camera.owl

## Semantic Definitions Separate from Application!

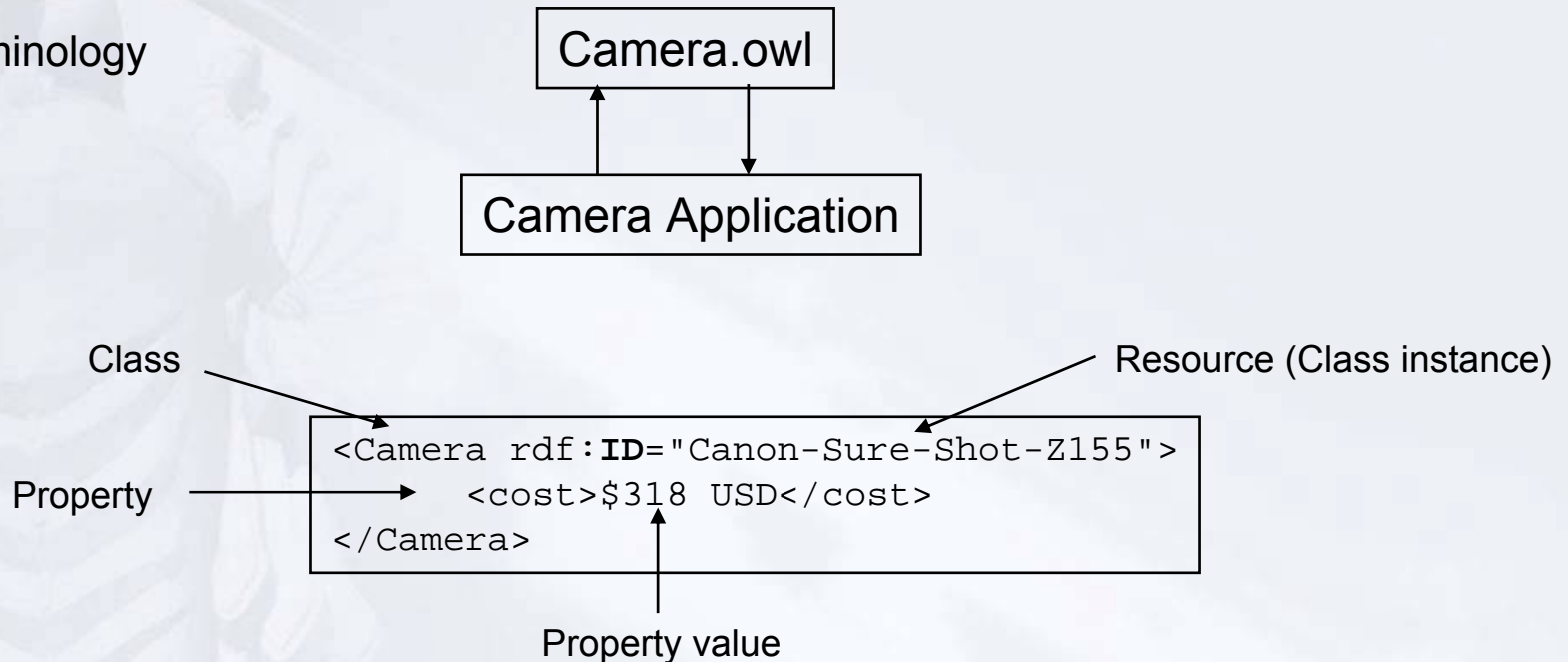
- The example demonstrated how a web-bot was able to dynamically process an XML document from a Web site, despite the fact that the XML document used terminology different than was used to express the request. This interoperability was achieved by using the OWL Camera Ontology!
- This example also demonstrated the ***architectural design principle of cleanly separating the application code*** (e.g., Web Bot) from the semantic definitions (e.g., Camera.owl).

An OWL Document is a Rich  
“Knowledge Bank”  
that your Applications can Tap Into



# Example – Camera Application

- Let's assume that the Camera application has access to an OWL Camera document:
- Terminology



This sample input data is following the RDF design pattern:

- the root element shows the Class of things being described, and the ID attribute identifies the instance of the Class.
- One or more Properties of the Class is provided.

# What Info in an OWL Document

- An OWL document contains this info:
  - **Class hierarchy info:** an OWL document defines class/subclass relationships.
  - **Synonym info:** an OWL document identifies equivalent classes and equivalent properties.
  - **Class association info:** an OWL document maps one or more classes to one or more classes, via a property (i.e., domain/range info).
  - **Property metadata info:** an OWL document contains a lot of metadata for properties.
  - **Class definition info:** an OWL document specifies the composition of classes.
- The following slides present examples to show how applications can use this info to enhance its robustness and extensibility.

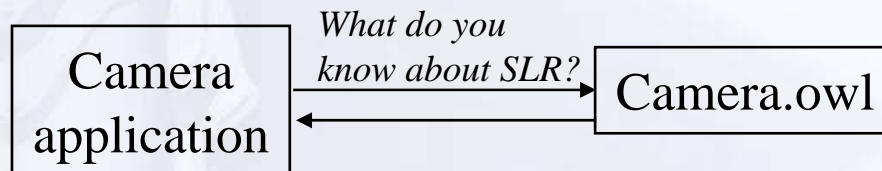
# Using Class Hierarchy Info to Enhance your Applications

Problem: design an application that filters inputs which do not contain Camera data.

Sample input:

```
<SLR rdf:ID="Olympus-OM-10">  
    ...  
</SLR>
```

This input contains Camera data and should not be filtered. If the application is designed to just look for <Camera> elements then it will incorrectly filter this input. However, by consulting the OWL Camera document (Camera.owl) it can dynamically discover that an SLR is a type of Camera:



*An SLR is a type  
of Camera.*

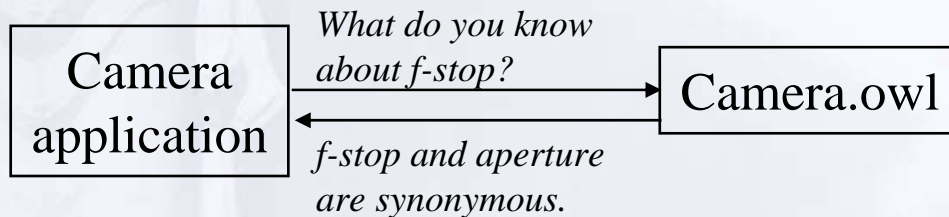
# Using Synonym Info to Enhance your Applications

Problem: design an application that filters inputs which do not contain Lens aperture data.

Sample input:

```
<Lens rdf:ID="Hasselblad_500V" >
  <f-stop>...</f-stop>
</Lens>
```

This input contains Lens aperture data and should not be filtered. If the application is designed to just look for <aperture> elements then it will incorrectly filter this input. However, by consulting Camera.owl it can dynamically discover that f-stop is synonymous with aperture:



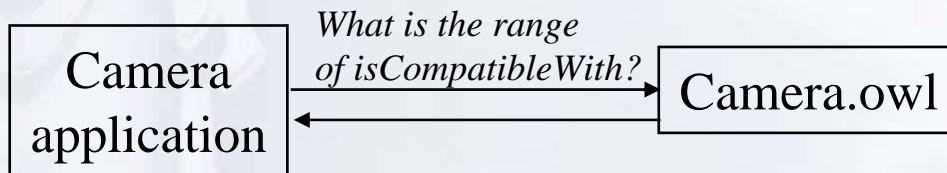
# Using Class Association Info to Enhance your Applications

Problem: design an application that filters inputs which do not contain info about a Lens that is compatible with a Canon-EOS-1 body.

Sample input:

```
<Lens rdf:ID="Sigma_24mm_f2.8" >
    ...
</Lens>
```

This input contains info about a Lens that is compatible with the Canon-EOS-1 body. If the application is designed to just look for Canon lenses then it will incorrectly filter this input. However, by consulting Camera.owl it can dynamically discover the compatible lenses: (assume a property, *isCompatibleWith*, which maps Canon-EOS-1 body to compatible lenses)



*Sigma\_24mm\_f2.8,*  
*Canon\_50mm\_f1.8, ...*

# Using Property Metadata to Enhance your Applications

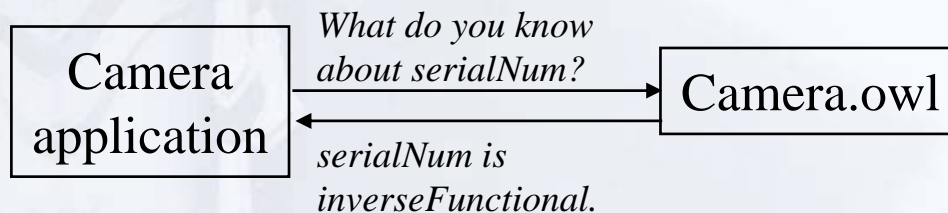
Problem: design an application that filters inputs which do not contain info about the Canon-Sure-Shot-Z155 (serialNum="ABCD").

Sample input:

```
<rdf:Description>
  <serialNum>ABCD</serialNum>
</rdf:Description>
```

This input contains Canon-Sure-Shot-Z155 data and should not be filtered.

If the application is designed to just look for `<Camera rdf:ID="Canon-Sure-Shot-Z155">` then it will incorrectly filter this input. However, by consulting Camera.owl it can dynamically discover that the serialNum uniquely identifies Canon-Sure-Shot-Z155:



# Metadata Properties

- OWL provides lots of metadata for properties:  
`SymmetricProperty, FunctionalProperty, TransitiveProperty, inverseOf, InverseFunctionalProperty, allValuesFrom, someValuesFrom, hasValue, cardinality, minCardinality, maxCardinality.`
- The ways that your application takes advantage of this metadata is limited only by your imagination!

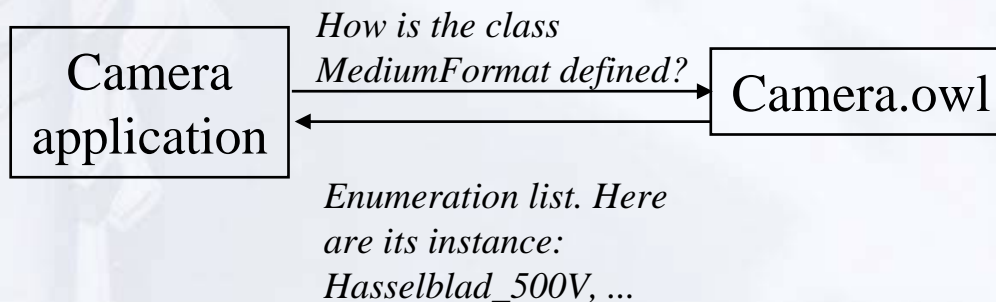
# Using Class Definition Info to Enhance your Applications

Problem: design an application that filters inputs which do not contain info about MediumFormat cameras.

Sample input:

```
<rdf:Description rdf:ID="Hasselblad_500V" >
    ...
</rdf:Description>
```

The MediumFormat class is defined in Camera.owl as an enumeration list. By consulting Camera.owl the application can retrieve the list, and thus dynamically determine that this input describes a MediumFormat camera:



# Is it required that there be just one ontology for a domain?

Issue: How many versions of an ontology are allowed?

Example: How many versions of a Camera ontology are allowed on the Web?

There are two cases:

1. Only a single Camera ontology is allowed - there is universal agreement on a Camera ontology.

Comment: in a highly distributed, autonomous environment such as the Web, this is not realistic.

2. Every application has its own Camera ontology (ontology-per-application)

Comment: the interoperability will be broken down.

Here is the middle ground: The number of Camera ontologies is greater than one, but less than one-per-application.

## The Evolution of a "Web of Camera Ontologies"

Consider this scenario: A community comes together to create a Camera ontology. Within that community there is perfect interoperability.

Independently, another community comes together to create another version of the Camera ontology. Within that community there is perfect interoperability, but across the two communities there is zero interoperability.

# Is it required that there be just one ontology for a domain?

- Someone in the first community recognizes that a class in that community's Camera ontology is equivalent to a class in the other community's Camera ontology, so this equivalence is added to both ontologies (using `owl:equivalentClass`). Therefore, a bridge has been created between the two Camera ontologies, and now there can be interoperability between the two communities.
- Ultimately, all the different Camera ontologies get linked together, and there is global interoperability.

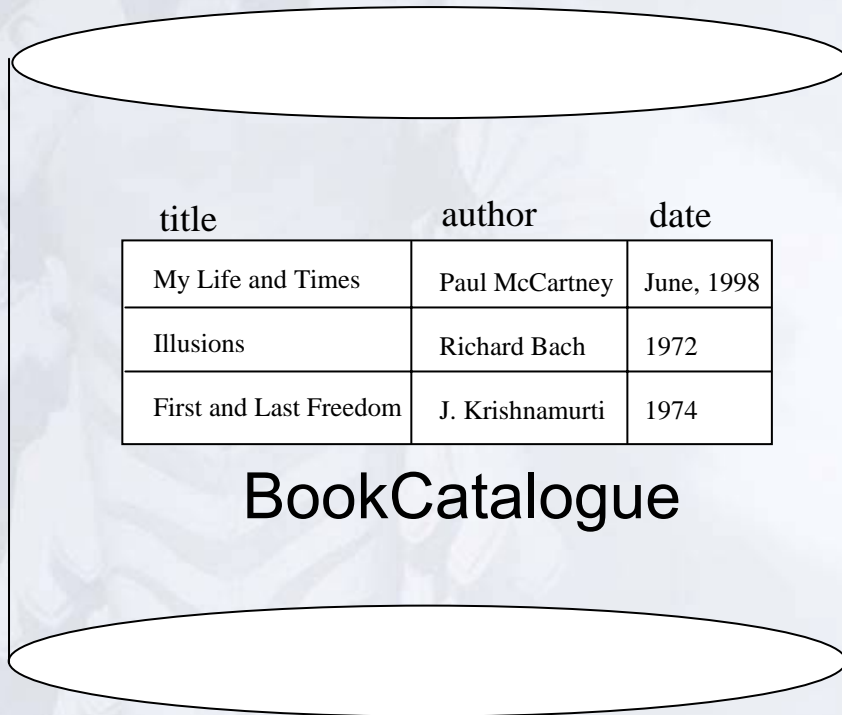
Now going the other direction (from macro to micro) ...

- Within a community there may be subcommunities which refine the community's Camera ontology. Provided the subcommunity's ontology is linked to the community's ontology then there will be perfect interoperability.
- Therefore, there can be diversity in ontologies both at a micro level as well as at a macro level.

# Upper-, Mid-level, Lower-Ontologies

- An upper-ontology defines very broad, universal Classes and properties
  - Example: Cyc Upper Ontology  
<http://www.opencyc.org> (6,000 concepts)
- A mid-level ontology is an upper ontology for a specific domain
- A lower-ontology is an ontology for a specific domain, with specific Classes and properties.
- Example: Suppose that we create a mid-level ontology.  
We can merge into an upper-level ontology by defining our ontologies root class as a `subClassOf` a class in the upper-ontology.

# Database = Ontology + Instances



```

<owl:Class rdf:ID="BookCatalogue"/>

<owl:DatatypeProperty rdf:ID="title">
  <rdfs:domain rdf:resource="#BookCatalogue"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="author">
  <rdfs:domain rdf:resource="#BookCatalogue"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="date">
  <rdfs:domain rdf:resource="#BookCatalogue"/>
  <rdfs:range rdf:resource="&xsd:date"/>
</owl:DatatypeProperty>

```

```

<?xml version="1.0"?>
<BookCatalogue>
  <title>My Life and Times</title>
  <author>Paul McCartney</author>
  <date>June, 1998</date>
</BookCatalogue>

```

# OWL vs. Database

- Advantages of using OWL to define an Ontology:
  - Extensible: much easier to add new properties. Contrast with a database - adding a new column may break a lot of applications (see example on next slide)
  - Portable: much easier to move an OWL document than to move a database.
- Advantages of using a Database to define an Ontology:
  - Mature: the database technology has been around a long time and is very mature.

# Protégé

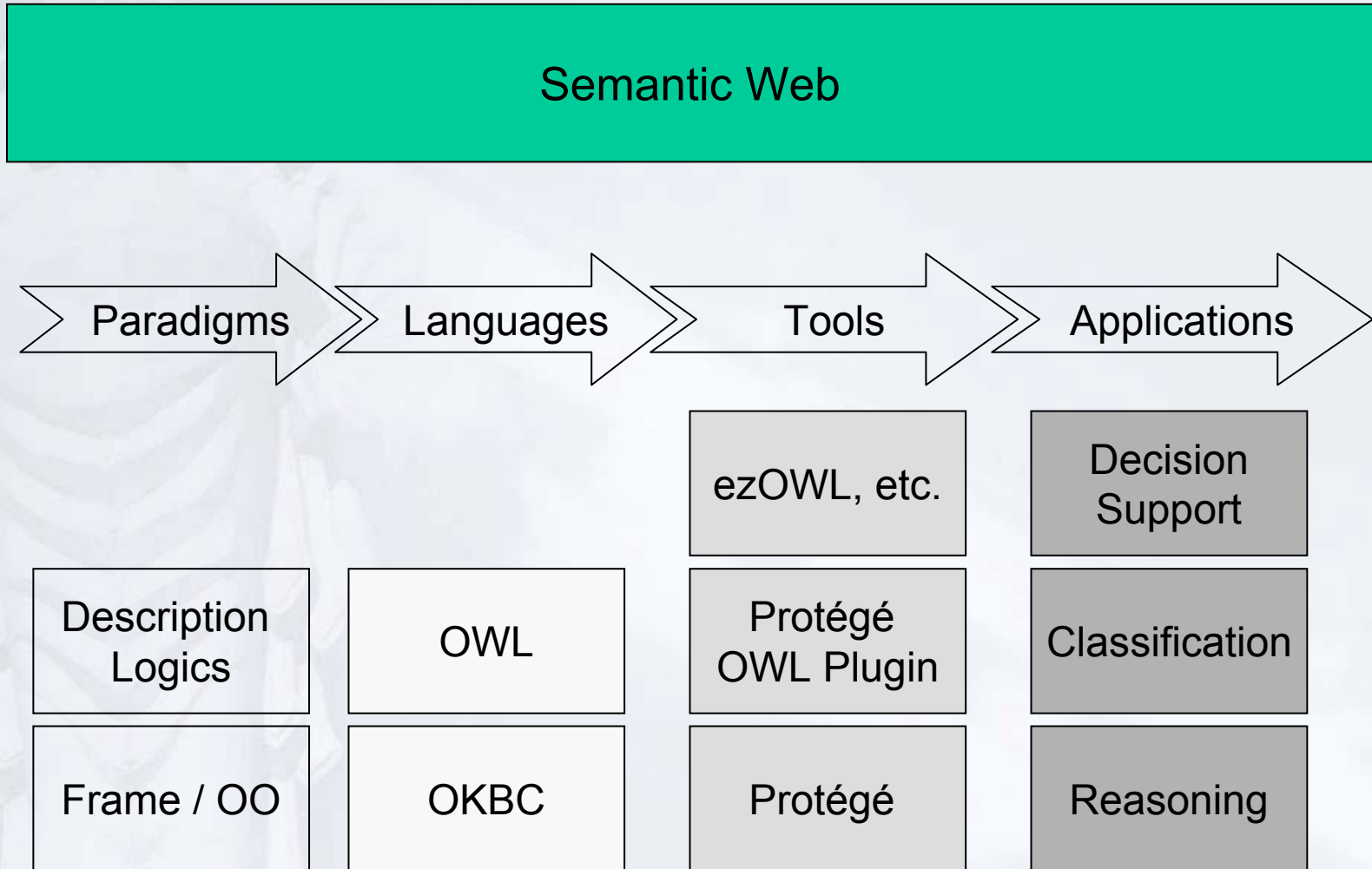


# Protégé Features

- Protégé is an extensible and customizable toolset for constructing ontologies and for developing applications that use these ontologies
- Outstanding features:
  - Automatic generation of graphical-user interface, based on user-defined models, for acquiring domain instances
  - Extensible knowledge model and architecture
  - Possible embedding of standalone applications in Protégé knowledge engineering environment and vice versa
  - Scalability to very large knowledge bases.

# Storage Formats

- User edit and view ontologies in a manner that insulates them from the ultimate storage format
- Ontologies may be read in from, written out to, and inter-converted between a large number of formats:
  - Relational database (ODBC)
  - UML
  - XML/XML Schema
  - RDF
  - Topic Maps
  - DAML+OIL
  - OWL



- Extension of Protégé to allow editing OWL ontologies
- Project started April 2003, based on ideas from previous projects (OilTab, RDF and DAML+OIL backends)
- Currently in: <http://protege.stanford.edu/plugins/owl>
- Features:
  - Loading and saving OWL files
  - Graphical editors for class expressions
  - Access to description logics inference components such as classifiers (RACER)
  - Powerful platform for hooking in custom-tailored components.

diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

Asserted Hierarchy V C X

- owl:Thing
  - Date
  - DaysOfWeek
  - DaysOfWeek-1
  - Events
  - Time
  - owl:Class<sup>D</sup>
  - owl:Property
    - owl:DatatypeProperty
    - owl:ObjectProperty<sup>D</sup>

**Date (type=owl:Class)** + - F T

Name Labels Annotations V C + -

Date

Documentation

Annotations

Property	Value

Asserted Inferred

Asserted Conditions C C<sub>R</sub> + - X

NECESSARY & SUFFICIENT

NECESSARY

- owl:Thing E
- Day ≤ 31 E
- Month ≤ 12 E
- Year ≥ 1930 E

At Class At owl:Thing All

Properties at V C + -

Name

- Year
- Day
- Month
- DOW

Disjoin V C + E<sup>+</sup> - X

# Protégé/Properties (Slots)



diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

Properties

- Content
- Day
- DOW
- EDate
- ETime
- Hour
- Minute
- Month
- protege:valuesComputer
- Year

DOW (type=owl:ObjectProperty)

Name: DOW

Labels:

Equivalent Properties:

Annotations:

Documentation:

Domain defined

Domain: Date

Range

Range: Instance

Classes: DaysOfWeek

Allows multiple values

Inverse Functional

Inverse:

Symmetric

Transitive

diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

**Classes**

- owl:Thing
  - Date (3)
  - DaysOfWeek (7)
  - DaysOfWeek-1
  - Events (14)
  - Time (11)
  - owl:Class D (10)
  - owl:Property

**Display Slot**

S:NAME

**Direct Instances**

- diary\_event\_01
- diary\_event\_02
- diary\_event\_03
- diary\_event\_04
- diary\_event\_05
- diary\_event\_06
- diary\_event\_07
- diary\_event\_08
- diary\_event\_09
- diary\_event\_10
- diary\_event\_11
- diary\_event\_12
- diary\_event\_13
- diary\_event\_14

**diary\_event\_02 (type=Events)**

Name Labels SameAs DifferentFrom

diary\_event\_02

**Documentation**

**Content**

Reading the paper

**EDate**

diary\_date\_2004.01.26

**ETime**

diary\_time\_1100

diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

**Forms**

- owl:Thing C
- Date M C
- DaysOfWeek M
- DaysOfWeek-1 M
- Events C
- Time M C
- owl:Class M C
- rdf:Property M C

**Display Slot** S:NAME

**Selected Widget Type** IntegerListWidget

Name Labels SameAs DifferentFrom

Documentation

Day V C - Month V C - Year V C -

DOW V C + -

diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

**Default Namespace**

**Namespace Prefixes**

Prefix	Namespace	Imported
rdfs *	http://www.w3.org/2000/01/rdf-schema#	<input type="checkbox"/>
rdf *	http://www.w3.org/1999/02/22-rdf-syntax-ns#	<input type="checkbox"/>
owl *	http://www.w3.org/2002/07/owl#	<input type="checkbox"/>
rss	http://purl.org/rss/1.0/	<input type="checkbox"/>
vcard	http://www.w3.org/2001/vcard-rdf/3.0#	<input type="checkbox"/>
dc	http://purl.org/dc/elements/1.1/	<input type="checkbox"/>
protege	http://protege.stanford.edu/plugins/owl/protege#	<input checked="" type="checkbox"/>
jms	http://jena.hpl.hp.com/2003/08/jms#	<input type="checkbox"/>

**Annotations**

Property	Value
owl:versionInfo	0.2
rdfs:seeAlso	Schedule Ontology

**Comment**

**owl:AllDifferent**  
 Sets of "all different" Individuals

**Distinct Members of Selected Set**

diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

**Asserted Hierarchy**

- owl:Thing
  - Date
  - DaysOfWeek
  - DaysOfWeek-1
  - Events
  - Time
  - owl:Class **D**
  - owl:Property

**Properties**

- Content
- Day
- DOW
- EDate
- ETime
- Hour
- Minute
- Month

[Ontology Diagram : 100%] [#Classes:5] [#Properties:10]

```

classDiagram
    class Date {
        protege:valuesComputer
        Year
        minCardinality 1930
        Day
        maxCardinality 31
        Month
        maxCardinality 12
        DOW
    }
    class DaysOfWeek {
        protege:valuesComputer
    }
    class Time {
        protege:valuesComputer
        Minute
        maxCardinality 59
        Hour
        maxCardinality 23
    }
    class Events {
        protege:valuesComputer
        Content
        ETime
        EDate
    }
    Date --> DaysOfWeek : DOW
    Events --> Date : EDate
    Events --> Time : ETime
    
```

**Date**

- protege:valuesComputer
- Year
- minCardinality 1930
- Day
- maxCardinality 31
- Month
- maxCardinality 12
- DOW

**DaysOfWeek**

- protege:valuesComputer

**Time**

- protege:valuesComputer
- Minute
- maxCardinality 59
- Hour
- maxCardinality 23

**Events**

- protege:valuesComputer
- Content
- ETime
- EDate

Grid  Comment  Inheritance  Properties  Inherited Properties  Property-Ref  Restriction

diary Protégé 2.0 (C:\Program Files\Protege\_2.0\projects\diary\diary.pprj, OWL Files)

Project Edit Window OWL Help

OWLClasses Properties Forms Individuals Metadata Queries ezOWL

Query

Class: Events Slot: Content

More Fewer Clear

Query Name: find reading events

Query Library: find reading events

diary\_event\_04 (type=Events)

Name Labels SameAs DifferentFrom

diary\_event\_04

Documentation

Content: back to work, reading book

EDate: diary\_date\_2004.01.26

ETime: diary\_time\_1430

Search Results (6)

- diary\_event\_02 (Events)
- diary\_event\_06 (Events)
- diary\_event\_01 (Events)
- diary\_event\_09 (Events)
- diary\_event\_04 (Events)
- diary\_event\_13 (Events)

The background of the slide is a faded, light-colored image of a classical statue. The statue appears to be a figure with a large, ornate, multi-tiered chest or breastplate, possibly a deity or a personification of a virtue. The figure is standing and looking slightly to the left. The overall tone is light and ethereal.

**Thank you very much**