Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Inferring and Discovering Relationships using RDF Schemas

Roger L. Costello
David B. Jacobs
The MITRE Corporation
(The creation of this tutorial was sponsored by DARPA)

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Acknowledgments

- We are very grateful to the Defense Agency Research Projects Agency (DARPA) for funding the creation of this tutorial. We are especially grateful to Murray Burke (DARPA) and John Flynn (BBN) for making it all happen.
- Special thanks to Stephen Dyer for creating the labs.
- Special thanks to Jon Hanna and Frank Manola for answering our many questions.

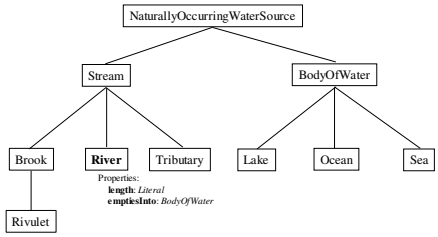
Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Purpose of RDF Schema

- The **purpose** of RDF Schema is to provide an XML vocabulary to:
  - express classes and their (subclass) relationships.
  - define properties and associate them with classes.
- The **benefit** of an RDF Schema is that it facilitates inferencing on your data, and enhanced searching.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

RDF Schema is about creating Taxonomies!



Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

What inferences can be made on this RDF/XML, given the taxonomy on the last slide?

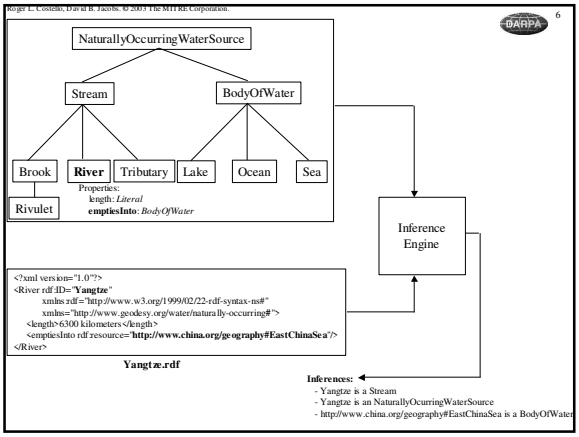
What inferences can be made with this data?

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <length>6300 kilometers</length>
   <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>
```

Yangtze.rdf

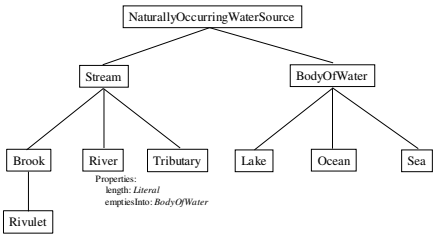
Inferences are made by examining a taxonomy that contains River.
See next slide.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.



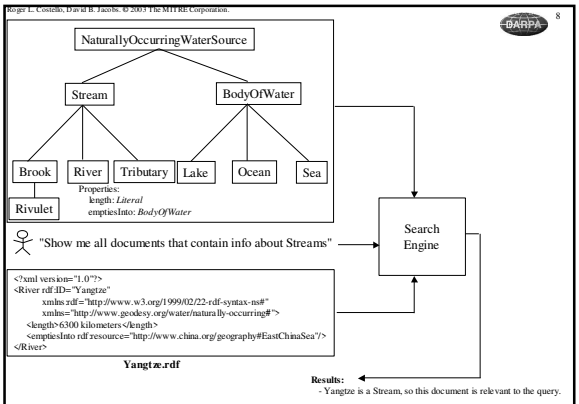
Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

How does a taxonomy facilitate searching?



The taxonomy shows that when searching for "streams", any RDF/XML that uses the class Brook, Rivulet, River, or Tributary are relevant. See next slide.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.



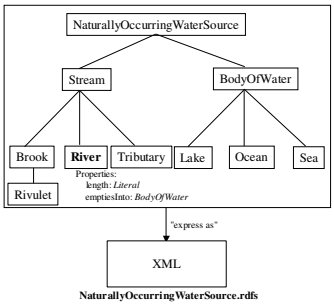
Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

You now know everything about RDF Schemas!

- RDF Schemas is all about defining taxonomies (class hierarchies).
- As we've seen, a taxonomy can be used to make inferences and to facilitate searching.
- That's all there is to RDF Schemas!
- The rest is just syntax …
  - The previous slide showed the taxonomy in a graphical form. Obviously, we need to express the taxonomy in a form that is machine-processable. RDF Schemas provides an XML vocabulary to express taxonomies.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

RDF Schema provides an XML vocabulary to express taxonomies



Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Classes/properties are defined using RDF/XML!

- RDF Schema uses the RDF/XML design pattern to define classes and properties. Recall the RDF/XML design pattern:

```
<?xml version="1.0"?>
<Class rdf:ID="resource"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="uri">
  <property rdf:resource="…"/>  [Use this syntax if the value of the property is a resource]
  <property>value</property>  [Use this syntax if the value of the property is a literal]
  ...
</Class>
```

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining a class (e.g., River)

1. All classes and properties are defined within rdf:RDF
2. Assigns a namespace to the taxonomy!
3. Defines the River class
4. Defines the Stream class
5. Since the Stream class is defined in the same document we can reference it using a fragment identifier.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

  <rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Stream">
    <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
  </rdfs:Class>

  ...

</rdf:RDF>
```

NaturallyOccurringWaterSource.rdfs (snippet)

This is read as:
"I hereby define a River Class. River is a subClassOf Stream."
"I hereby define a Stream Class. Stream is a subClassOf NaturallyOccurringWaterSource."
...

rdfs:Class

- This type is used to define a class.
- The rdf:ID provides a name for the class.
- The contents are used to indicate the members of the class.
  - The contents are ANDed together.

Name of the class

```
<rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
</rdfs:Class>
```

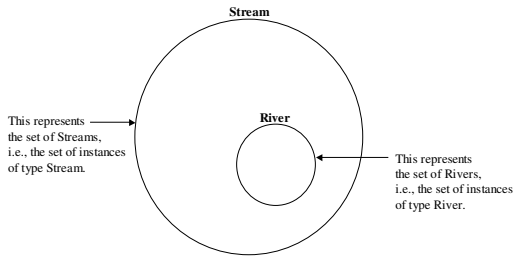
} ANDed

Equivalent!

```
<rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
</rdfs:Class>
```

```
<rdf:Description rdf:ID="River">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#Stream"/>
</rdf:Description>
```


rdfs:subClassOf

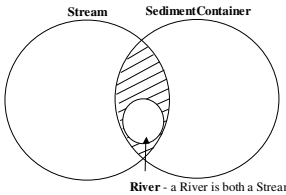


rdfs:subClassOf

- Use this property to indicate a subclass relationship between one class and another class.
- You may specify zero, one, or multiple rdfs:subClassOf properties.
  - **Zero**: if you define a class without specifying rdfs:subClassOf then you are implicitly stating that the class is a subClassOf rdfs:Resource (the root of all classes).
  - **One**: if you define a class by specifying one rdfs:subClassOf then you are indicating that the class is a subclass of that class.
  - **Multiple**: if you define a class by specifying multiple rdfs:subClassOf properties then you are indicating that the class is a subclass of each of the other classes.
    - Example: consider the River class: suppose that it has two rdfs:subClassOf properties - one that specifies Stream and a second that specifies SedimentContainer. Thus, the two rdfs:subClassOf properties indicate that a River is a Stream *and* a SedimentContainer. *That is, each instance of River is both a Stream and a SedimentContainer.*

Example of multiple rdfs:subClassOf properties

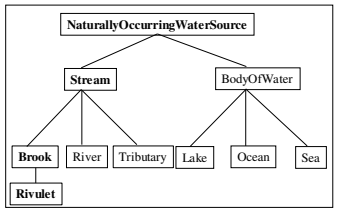
```
<rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
    <rdfs:subClassOf rdf:resource="http://www.containers.org#SedimentContainer"/>
</rdfs:Class>
```



River - a River is both a Stream and a SedimentContainer.

The conjunction (AND) of two subClassOf statements is a subset of the intersection of the classes.

rdfs:subClassOf is transitive



Consider the above **class hierarchy**. It says, for example, that:
- A Rivulet is a Brook.
- A Brook is a Stream.

Therefore, since subClassOf is transitive, a Rivulet is a Stream.
(Note that a Rivulet is also a NaturallyOccurringWaterSource.)

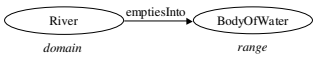
Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining a property (e.g., emptiesInto)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

  <rdf:Property rdf:ID="emptiesInto">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:range rdf:resource="#BodyOfWater"/>
  </rdf:Property>
  ...

</rdf:RDF>
```

NaturallyOccurringWaterSource.rdfs (snippet)

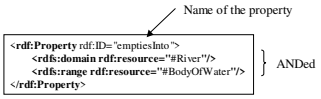
This is read as:
"I hereby define an emptiesInto Property. The domain (class) in which emptiesInto is used is River. The range (of values) for emptiesInto are instances of BodyOfWater."
That is, the emptiesInto Property relates (associates) a River to a BodyOfWater.



Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

rdf:Property

- This type is used to define a property.
- The rdf:ID provides a name for the property.
- The contents are used to indicate the usage of the property.
  - The contents are ANDed together.



Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Equivalent!

```
<rdf:Property rdf:ID="emptiesInto">
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
</rdf:Property>
```

```
<rdf:Description rdf:ID="emptiesInto">
      <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
</rdf:Description>
```

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Careful: Class and Property are in different namespaces

- Class is in the rdfs namespace.
- Property is in the rdf namespace.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

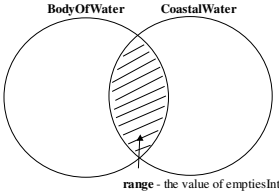
rdfs:range

- Use this property to indicate the type of values that a property will contain.
- You may specify zero, one, or multiple rdfs:range properties.
  - **Zero**: if you define a property without specifying rdfs:range then you are providing no information about the type of value that the property will contain.
  - **One**: if you define a property by specifying one rdfs:range then you are indicating that the property will contain a value whose type is that specified by rdfs:range.
  - **Multiple**: if you define a property by specifying multiple rdfs:range properties then you are indicating that the property will contain a value which belongs to every class defined by the rdfs:range properties.
    - Example: consider the property emptiesInto: suppose that it has two rdfs:range properties - one that specifies BodyOfWater and a second that specifies CoastalWater. Thus, the two rdfs:range properties indicate that emptiesInto will contain a value that is a BodyOfWater and a CoastalWater.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Example of multiple rdfs:range properties

```
<rdf:Property rdf:ID="emptiesInto">
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
      <rdfs:range rdf:resource="http://www.geodesy.org/coast#CoastalWater"/>
</rdf:Property>
```



range - the value of emptiesInto is a BodyOfWater and a CoastalWater.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

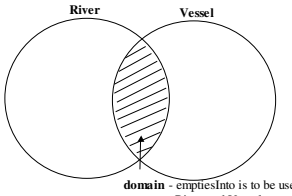
rdfs:domain

- Use this property to indicate the classes that a property will be used with.
- You may specify zero, one, or multiple rdfs:domain properties.
  - **Zero**: if you define a property without specifying rdfs:domain then you are providing no information about the class that the property will be used with, i.e., the property can be used with any class.
  - **One**: if you define a property by specifying one rdfs:domain then you are indicating that the property will be used with the class specified by rdfs:domain.
  - **Multiple**: if you define a property by specifying multiple rdfs:domain properties then you are indicating that the property will be used with a class which belongs to every class defined by the rdfs:domain properties.
    - Example: consider the property emptiesInto: suppose that it has two rdfs:domain properties - one that specifies River and a second that specifies Vessel. Thus, the two rdfs:domain properties indicate that emptiesInto will be used with a class that is a River and a Vessel.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Example of multiple rdfs:domain properties

```
<rdf:Property rdf:ID="emptiesInto">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:domain rdf:resource="http://www.containers.org#Vessel"/>
    <rdfs:range rdf:resource="#BodyOfWater"/>
</rdf:Property>
```



domain - emptiesInto is to be used in instances that are of type River and Vessel.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Note that properties are defined **separately** from classes

- With most Object-Oriented languages when a class is defined the properties (attributes) are simultaneously defined.
  - For example, "I hereby define a Rectangle class, and its attributes are length and width."
- With RDF Schema things are different. You define a class (and indicate its relationships to other classes). Separately, you define properties and then associate them with a class!
  - For the above example you would define the Rectangle class (and indicate that it is a subclass of GeometricObject). Separately, you then define a length property, indicate its range of value, and then indicate that length may be used with the Rectangle class. *(Thus, if you have an untyped Resource with a length property you can infer the Resource is a Rectangle.)* Likewise for the width property.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Advantage of separately defining classes and properties

- As we have seen, the RDF Schema approach is to define a class, and then separately define properties and state that they are to be used with the class.
- The advantage of this approach is that anyone, anywhere, anytime can create a property and state that it is usable with the class!

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

The XML Representation of the taxonomy

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

   <rdfs:Class rdf:ID="River">
      <rdfs:subClassOf rdf:resource="#Stream"/>
   </rdfs:Class>

   <rdfs:Class rdf:ID="Stream">
      <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
   </rdfs:Class>

   <rdf:Property rdf:ID="emptiesInto">
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
   </rdf:Property>

   <rdf:Property rdf:ID="length">
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
   </rdf:Property>

   ...

</rdf:RDF>
```

NaturallyOccurringWaterSource.rdfs (snippet)

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Literal value

```
<rdf:Property rdf:ID="length">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

A literal type is a simple, untyped string.

NaturallyOccurringWaterSource *Ontology*!

- NaturallyOccurringWaterSource.rdfs defines a set of classes and how the classes are related. It defines a set of properties and indicates the type of values they may have and what classes they may be associated with.
- That is, it defines an **ontology** for NaturallyOccurringWaterSources!

Inferring a resource's class from the properties' domain

Notice that in this RDF/XML instance the class of the resource (Yangtze) is not identified:

```
<?xml version="1.0"?>
<rdf:Description rdf:ID="Yangtze"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <length>6300 kilometers</length>
   <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</rdf:Description>
```

However, we can infer that Yangtze is a River because length and emptiesInto have a rdfs:domain of River, i.e., their domain asserts that these properties will be used in a River instance.

Design to facilitate inferencing!

With this design:

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <length>6300 kilometers</length>
   <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>
```

Yangtze.rdf

We are able to infer (using the ontology) that the value of emptiesInto is a BodyOfWater.

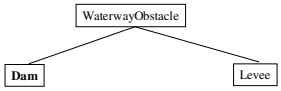
Suppose instead we had designed it as such:

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <length>6300 kilometers</length>
   <emptiesInto>East China Sea</emptiesInto>
</River>
```

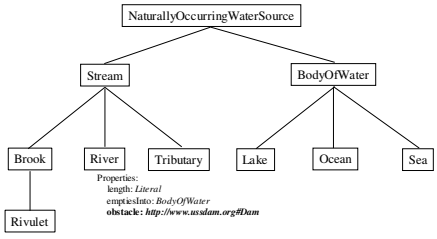
Now we can make no inferences about emptiesInto, since it just contains a literal.

Lesson Learned: to maximize the utility of your data, design to facilitate inferencing!

Example #2: WaterwayObstacle Taxonomy



NaturallyOccurringWaterSource Taxonomy (**Updated**)



What inferences can be made on this RDF/XML, given the taxonomies on the last two slides?

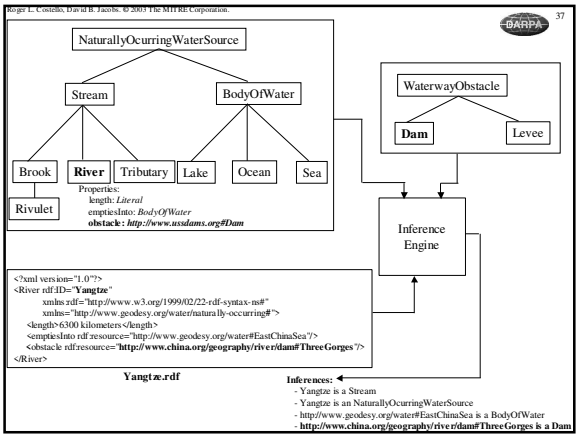
What inferences can be made with this data?

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <length>6300 kilometers</length>
   <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
   <obstacle rdf:resource="http://www.china.org/geography/river/dam#ThreeGorges"/>
</River>
```

Yangtze.rdf

Inferences are made by examining the taxonomies that contains River and Dam.
See next slide.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.



Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining the obstacle property

```
<rdf:Property rdf:ID="obstacle">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:range rdf:resource="http://www.ussdam.org#Dam"/>
</rdf:Property>
```

Read this as: "I hereby define a property called obstacle. The type of value that this property will have is of type Dam (more specifically, of type http://www.ussdam.org#Dam). This property will be used in a River class (the River class is defined locally, so we simply use a fragment identifier).

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Example #3

Create an RDF Schema for the following RDF document:

untyped resource

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#"
        xmlns:uom="http://www.nist.org#">
    <length>
        <rdf:Description>
            <rdf:value>6300</rdf:value>
            <uom:units>kilometers</uom:units>
        </rdf:Description>
    </length>
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
    <obstacle rdf:resource="http://www.china.org/geography/river/dam#ThreeGorges"/>
</River>
```

Note that the property length has a value that has no type indicated.
Note that the property units is from a different namespace (a different taxonomy).

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining length with no Type Information

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

  <rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
  </rdfs:Class>

  <rdf:Property rdf:ID="length">
    <rdfs:domain rdf:resource="#River"/>
  </rdf:Property>

  ...

</rdf:RDF>
```

No rdfs:range specified. This means that we are providing no information on the type of value that length will have.

NaturallyOccurringWaterSource.rdfs (snippet)

Disadvantage: this way of defining length yields no inferencing capability about its value.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

A resource that doesn't have a type specified may nonetheless be typed!

```
<length>
    <rdf:Description>
        <rdf:value>6300</rdf:value>
        <uom:units>kilometers</uom:units>
    </rdf:Description>
</length>
```

There is no type shown for this resource. But that doesn't mean that this resource has no type. It only means that no type has been specified in this RDF/XML instance. In the RDF Schema we can specify what its type is.

```
<rdf:Property rdf:ID="length">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:range rdf:resource="http://www.nist.org#Distance"/>
</rdf:Property>
```

Advantage: now we can infer that the contents of length is of type Distance.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Lesson Learned

This RDF Schema:

```
<rdf:Property rdf:ID="length">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:range rdf:resource="http://www.nist.org#Distance"/>
</rdf:Property>
```

Does not mandate that the RDF/XML instance specify a type, e.g.,

Best Practice

```
<length>
    <uom:Distance>
        <rdf:value>6300</rdf:value>
        <uom:units>kilometers</uom:units>
    </uom:Distance>
</length>
```

Best Practice

It is perfectly fine to keep that class information isolated to the RDF Schema, e.g.,

```
<length>
    <rdf:Description>
        <rdf:value>6300</rdf:value>
        <uom:units>kilometers</uom:units>
    </rdf:Description>
</length>
```

(However, it is better practice to expose the type information in the RDF/XML instance.)

Do Lab2

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Example #4

Create an RDF Schema for the following RDF document:

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <length rdf:datatype="http://www.nist.org#kilometer">6300</length>
   <maxWidth rdf:datatype="http://www.nist.org#meter">175</maxWidth>
   <maxDepth rdf:datatype="http://www.nist.org#meter">55</maxDepth>
</River>
```

Yangtze.rdf

The River class has three properties containing *typed literals*.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Here are the two XML Schema datatypes being referenced in the RDF

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.nist.org#">

  <simpleType name="kilometer">
    <restriction base="integer">
    </restriction>
  </simpleType>

  <simpleType name="meter">
    <restriction base="integer">
    </restriction>
  </simpleType>

</schema>
```

uom.xsd

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining properties with typed literals

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

   <rdf:Property rdf:ID="length">
     <rdfs:domain rdf:resource="#River"/>
     <rdfs:range rdf:resource="http://www.nist.org#kilometer"/>
   </rdf:Property>

   <rdfs:Datatype rdf:about="http://www.nist.org#kilometer">
     <rdfs:subClassOf rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
   </rdfs:Datatype>

   <rdf:Property rdf:ID="maxWidth">
     <rdfs:domain rdf:resource="#River"/>
     <rdfs:range rdf:resource="http://www.nist.org#meter"/>
   </rdf:Property>

   <rdfs:Datatype rdf:about="http://www.nist.org#meter">
     <rdfs:subClassOf rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
   </rdfs:Datatype>
   ...
</rdf:RDF>
```

NaturallyOccurringWaterSource.rdfs (snippet)

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Indicating that rdf:resource is identifying a "datatype"

```
<rdf:Property rdf:ID="length">
     <rdfs:domain rdf:resource="#River"/>
     <rdfs:range rdf:resource="http://www.nist.org#kilometer"/>
</rdf:Property>

<rdfs:Datatype rdf:about="http://www.nist.org#kilometer">
     <rdfs:subClassOf rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</rdfs:Datatype>
```

To indicate that this is referencing a datatype, we use this

Thus, we are clearly identifying that the length property associates River with a datatype. (In all of the other properties we have seen they associated a Resource with another Resource. Here we have the case of a property associating a Resource with a datatype.)

Here's how to read this: I hereby declare that this:

http://www.nist.org#kilometer

represents a "datatype". And this datatype is a subclass of:

http://www.w3.org/2001/XMLSchema#integer

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

If the RDF Schema indicates a datatype then the RDF/XML instance **must** use rdf:datatype

```
<rdf:Property rdf:ID="length">
     <rdfs:domain rdf:resource="#River"/>
     <rdfs:range rdf:resource="http://www.nist.org#kilometer"/>
</rdf:Property>
```

This means that the property's value must be a typed literal.

```
<length rdf:datatype="http://www.nist.org#kilometer">6300</length>
```

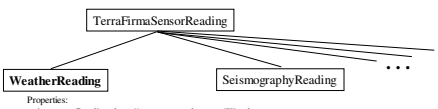
```
<rdf:Property rdf:ID="length">
     <rdfs:domain rdf:resource="#River"/>
     <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

This means that the property's value is an untyped string

```
<length>6300</length>
```

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Example #5 TerraFirmaSensorReading Taxonomy



Properties:
instrumentReading: *http://www.meteorology.org#Weather*
instrumentLocation: *http://www.geodesy.org#Location*
datetime: *http://www.w3.org/2001/XMLSchema#dateTime*

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

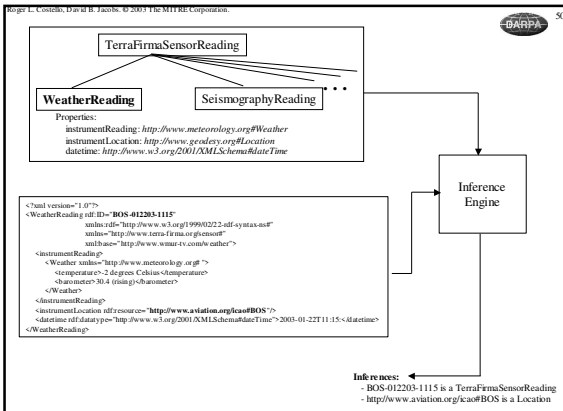
What inferences can be made on this RDF/XML, given the taxonomy on the last slide?

What inferences can be made with this data?

```
<?xml version="1.0"?>
<WeatherReading rdf:ID="BOS-012203-1115"
                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns="http://www.terra-firma.org/sensor#"
                xml:base="http://www.wmur-tv.com/weather">
   <instrumentReading>
      <Weather xmlns="http://www.meteorology.org# ">
         <temperature>-2 degrees Celsius</temperature>
         <barometer>30.4 (rising)</barometer>
      </Weather>
   </instrumentReading>
   <instrumentLocation rdf:resource="http://www.aviation.org/icao#BOS"/>
   <datetime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2003-01-22T11:15:</datetime>
</WeatherReading>
```

WMUR_TV_WeatherReading.rdf

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.



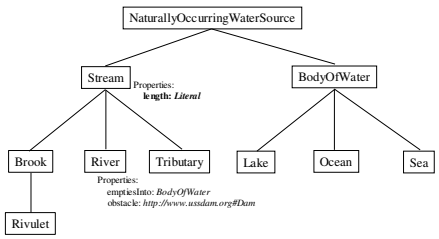
Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.


```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.terra-firma.org/sensor">

   <rdfs:Class rdf:ID="TerraFirmaSensorReading">
   </rdfs:Class>

   <rdfs:Class rdf:ID="WeatherReading">
      <rdfs:subClassOf rdf:resource="#TerraFirmaSensorReading"/>
   </rdfs:Class>

   <rdf:Property rdf:ID="instrumentReading">
      <rdfs:domain rdf:resource="#WeatherReading"/>
      <rdfs:range rdf:resource="http://www.meteorology.org#Weather"/>
   </rdf:Property>

   <rdf:Property rdf:ID="instrumentLocation">
      <rdfs:domain rdf:resource="#WeatherReading"/>
      <rdfs:range rdf:resource="http://www.geodesy.com#Location"/>
   </rdf:Property>

   <rdf:Property rdf:ID="datetime">
      <rdfs:domain rdf:resource="#Weather"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
   </rdf:Property>

   <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#dateTime">
   </rdfs:Datatype>

</rdf:RDF>
```

TerraFirmaSensorReading.rdfs

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Classes inherit properties from their ancestors



length has been defined to be a property of Stream. Therefore, all Stream subclasses inherit the length property. Note that the properties emptiesInto and obstacle are defined to be local to River.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Table showing what properties are applicable to each class

Classes →

	Stream	Brook	Rivulet	River	Tributary
length	X	X	X	X	X
emptiesInto				X	
obstacle				X	

↑ Properties

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining length and emptiesInto

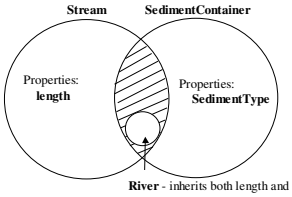
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

   <rdf:Property rdf:ID="length">
      <rdfs:domain rdf:resource="#Stream"/>
      <rdfs:range rdf:resource="http://www.nist.org#Distance"/>
   </rdf:Property>

   <rdf:Property rdf:ID="emptiesInto">
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
   </rdf:Property>
   ...

</rdf:RDF>
```

NaturallyOccurringWaterSource.rdfs (snippet)

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Properties from **all** superclasses are inherited

```
<rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
    <rdfs:subClassOf rdf:resource="http://www.containers.org#SedimentContainer"/>
</rdfs:Class>
```

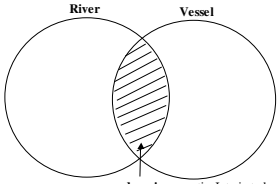


River - inherits both length and SedimentType.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining a property to be applicable to multiple classes

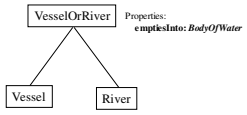
```
<rdf:Property rdf:ID="emptiesInto">
    <rdfs:domain rdf:resource="#River"/>
    <rdfs:domain rdf:resource="http://www.containers.org#Vessel"/>
    <rdfs:range rdf:resource="#BodyOfWater"/>
</rdf:Property>
```



domain - emptiesInto is to be used in instances that are of type River **and** Vessel. *How do we define emptiesInto so that it may be used with a River class **OR** a Vessel class?*

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Answer: associate emptiesInto with a superclass



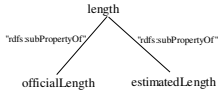
Both Vessel and River inherit the property emptiesInto. Thus, emptiesInto may be used with the Vessel class *or* the River class.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

rdfs:subPropertyOf

You can define a property to be a **specialization** of another property:

Property Hierarchy:



Notes:
1. The subproperties inherit the rdfs:range and rdfs:domain values from the parent property.
2. If a subproperty is true, then its parent property is true, e.g., if the Yangtze River has an officialLength of 6300 kilometers then it also has a length of 6300 kilometers.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Table showing what properties are applicable to each class

Classes →

	Stream	Brook	Rivulet	River	Tributary
length	X	X	X	X	X
emptiesInto				X	
obstacle				X	
estimatedLength	X	X	X	X	X
officialLength	X	X	X	X	X

↑ Properties

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Making inferences with subproperties

What inferences can we make on the estimatedLength property:

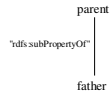
```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <estimatedLength>6300 kilometers</estimatedLength>
   <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>
```

Inference:
Since estimatedLength is a subproperty of length, we can infer that the Yangtze has a length of 6300 kilometers.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Another example of inferencing using property hierarchies

Property Hierarchy:




```
<?xml version="1.0"?>
<Person rdf:ID="Mary"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.genealogy.org#">
   <father>
      <Person rdf:about="#John"/>
   </father>
</Person>
```

"Mary has a father named John."

Inference:
Since father is a subproperty of parent, we can infer that Mary has a parent named John.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

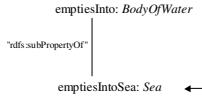
rdfs:subPropertyOf

- Use this property to create a property that specializes another property.
- You may specify zero, one, or multiple rdfs:subPropertyOf properties.
  - **Zero**: if you define a property without specifying rdfs:subPropertyOf then you are providing no relationship information to another property.
  - **One**: if you define a property by specifying one rdfs:subPropertyOf then you are indicating that the property is a subproperty of the other property.
  - **Multiple**: if you define a property by specifying multiple rdfs:subPropertyOf properties then you are indicating that the property is a subproperty of each of the other properties.
    - Example: consider the length property: suppose that it has two rdfs:subPropertyOf properties - one that specifies Distance and a second that specifies Measurement. Thus, the two rdfs:subPropertyOf properties indicate that a length is a Distance and a Measurement.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

A subproperty can narrow the range and/or domain

Property Hierarchy:



This subproperty narrows the range to Sea.

```
<rdf:Property rdf:ID="emptiesInto">
      <rdfs:domain rdf:resource="#River"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
</rdf:Property>

<rdf:Property rdf:ID="emptiesIntoSea">
      <rdfs:subPropertyOf rdf:resource="#emptiesInto"/>
      <rdfs:range rdf:resource="#Sea"/>
</rdf:Property>
```

The property emptiesInto permits a range of BodyOfWater. This property, however, narrows the range to Sea.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

rdfs:label, rdfs:comment

```
<rdf:Property rdf:ID="Creator">
   <rdfs:label xml:lang="EN">Author/Creator</rdfs:label>
   <rdfs:comment xml:lang="EN">The person or organization primarily responsible for creating the
      intellectual content of the resource. For example, authors in the case of written documents, artists,
      photographers, or illustrators in the case of visual resources.
   </rdfs:comment>
</rdf:Property>
```

rdfs:label is used to provide a human-readable version of the property/class name.

rdfs:comment is used to provide a human-readable description of the property/class.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Example #6

Create an RDF Schema for the following RDF/XML instance:

```
<?xml version="1.0"?>
<Article rdf:ID="QuickBrownFox"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.publishing.org#">
   <paragraph rdf:parseType="Literal">
      The quick brown <bold>fox</bold> jumped over the lazy dog.
      An important person once said "<quote>Now is the time for all
      good men to come to the aid of their country</quote> "
   </paragraph>
</Article>
```

QuickBrownFox.rdf

Recall that rdf:parseType="Literal" means that the contents can be any well-formed XML.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

A snippet of the Publishing Ontology

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.publishing.org">

   <rdfs:Class rdf:ID="Article">
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
   </rdfs:Class>

   <rdf:Property rdf:ID="paragraph">
      <rdfs:domain rdf:resource="#Article"/>
      <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral"/>
   </rdf:Property>

   …

</rdf:RDF>
```

The paragraph property is to be used within an Article class. The contents will be any well-formed XML.

PublishingOntology.rdfs (snippet)

Distinguish between rdfs:Literal and rdfs:XMLLiteral

- rdfs:Literal denotes a simple, untyped string.
- rdfs:XMLLiteral denotes any well-formed XML string.

Example #7

Create an RDF Schema for the following RDF/XML instance:

```
<?xml version="1.0"?>
<Catalogue rdf:ID="BookCatalogue"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns="http://www.publishing.org#"
           xmlns:dc="http://purl.org/metadata/dublin-core#"
           xml:base="http://www.bn.com">
  <item>
    <Book rdf:ID="_0-06-099325-2"
          xml:base="http://www.publishing.org/book">
      <dc:Title>Lateral Thinking</dc:Title>
      <dc:Creator>Edward de Bono</dc:Creator>
      <dc:Date>1973</dc:Date>
      <dc:Publisher>Harper &amp;amp; Row</dc:Publisher>
    </Book>
  </item>
  <item>
    <Book rdf:ID="_0-440-34319-4"
          xml:base="http://www.publishing.org/book">
      <dc:Title>Illusions: The Adventures of a Reluctant Messiah</dc:Title>
      <dc:Creator>Richard Bach</dc:Creator>
      <dc:Date>1977</dc:Date>
      <dc:Publisher>Dell Publishing Co.</dc:Publisher>
    </Book>
  </item>
  ...
</Catalogue>
```

Barnes_and_Noble_BookCatalogue.rdf

Another snippet of the Publishing Ontology

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.publishing.org">

  <rdfs:Class rdf:ID="Catalogue">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Book">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  </rdfs:Class>

  <rdf:Property rdf:ID="item">
    <rdfs:domain rdf:resource="#Catalogue"/>
    <rdfs:range rdf:resource="#Book "/>
  </rdf:Property>

  ...

</rdf:RDF>
```

PublishingOntology.rdfs

The Dublin Core properties - Title, Creator, Date, Publisher - are defined in the Dublin Core RDF Schema (see next slide).

Dublin Core RDF Schema

```
<?xml version='1.0'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
         xml:base="http://purl.org/metadata/dublin-core">

 <rdf:Description rdf:ID="Title">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Title</rdfs:label>
  <rdfs:comment>The name given to the resource, usually by the Creator or Publisher.</rdfs:comment>
 </rdf:Description>

 <rdf:Description rdf:ID="Creator">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Author/Creator</rdfs:label>
  <rdfs:comment>The person or organization primarily responsible for creating the intellectual content of the resource. For example,
  authors in the case of written documents, artists, photographers, or illustrators in the case of visual resources.</rdfs:comment>
 </rdf:Description>

 <rdf:Description ID="Date">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Date</rdfs:label>
  <rdfs:comment>A date associated with the creation or availability of the resource. Such a date is not to be confused with one belonging
  in the Coverage element, which would be associated with the resource only insofar as the intellectual content is somehow about that
  date. Recommended best practice is defined in a profile of ISO 8601 [Date and Time Formats (based on ISO8601), W3C Technical Note,
  http://www.w3.org/TR/NOTE-datetime] that includes (among others) dates of the forms YYYY and YYYY-MM-DD. In this scheme,
  for example, the date 1994-11-05 corresponds to November 5, 1994.</rdfs:comment>
 </rdf:Description>

 <rdf:Description ID="Publisher">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Publisher</rdfs:label>
  <rdfs:comment>The entity responsible for making the resource available in its present form, such as a publishing house, a
  university department, or a corporate entity.</rdfs:comment>
 </rdf:Description>
 ...
</rdf:RDF>
```


Equivalent!

```
<rdf:Description rdf:ID="Title">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:label>Title</rdfs:label>
  <rdfs:comment>The name given to the resource, usually by the Creator or Publisher.</rdfs:comment>
</rdf:Description>
```

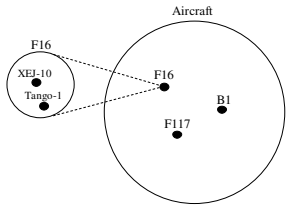
```
<rdf:Property rdf:ID="Title">
  <rdfs:label>Title</rdfs:label>
  <rdfs:comment>The name given to the resource, usually by the Creator or Publisher.</rdfs:comment>
</rdf:Property>
```


Metaclasses

- A metaclass is a class of classes. Example:
  - The instances of an Aircraft class are:
    - F16, B1, F117, etc.
  - The properties of Aircraft are:
    - wingspan, range, weight, etc.
  - Each of the instances of Aircraft are themselves classes!
    - The instances of a F16 class are specific F16 planes.
    - The properties of F16 are:
      - tailNum, deployedAt, etc.
  - The Aircraft class is a metaclass!

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Aircraft is a Metaclass



Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Defining Aircraft and its Instances

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.nato.org/aircraft">

  <rdfs:Class rdf:ID="Aircraft"/>                                   Classes

  <rdfs:Class rdf:ID="F16"/>                                        Classes
  ...

  <rdf:Property rdf:ID="wingspan">                                  Properties of Aircraft
    <rdfs:domain rdf:resource="#Aircraft"/>                         Properties of Aircraft
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>  Properties of Aircraft
  </rdf:Property>                                                   Properties of Aircraft
  ...

  <rdf:Property rdf:ID="deployedAt">                                Properties of F16
    <rdfs:domain rdf:resource="#F16"/>                              Properties of F16
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>  Properties of F16
  </rdf:Property>                                                   Properties of F16
  ...

</rdf:RDF>
```

Aircraft.rdfs

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Sample Instances

```
<?xml version="1.0"?>
<Aircraft rdf:about="http://www.nato.org/aircraft#F16"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.nato.org/aircraft#">
  <wingspan>10 meters</wingspan>
  ...
</Aircraft>
```

F16 is being used as an instance!

F16.rdf

F16 is being used as a class!

```
<?xml version="1.0"?>
<F16 rdf:ID="XEJ-10"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns="http://www.nato.org/aircraft#">
  <deployedAt>Persian Gulf</deployedAt>
  ...
</F16>
```

XEJ-10.rdf

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Equivalent!

```
<?xml version="1.0"?>
<Aircraft rdf:about="http://www.nato.org/aircraft#F16"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.nato.org/aircraft#">
  <wingspan>10 meters</wingspan>
  ...
</Aircraft>
```

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://www.nato.org/aircraft#F16"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.nato.org/aircraft#">
  <rdf:type rdf:resource="http://www.nato.org/aircraft#Aircraft"/>
  <wingspan>10 meters</wingspan>
  ...
</rdf:Description>
```

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Merge Descriptions

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://www.nato.org/aircraft#F16"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.nato.org/aircraft#">
  <rdf:type rdf:resource="http://www.nato.org/aircraft#Aircraft"/>
  <wingspan>10 meters</wingspan>
  ...
</rdf:Description>
```

```
<rdfs:Class rdf:ID="F16"/>
```

```
<?xml version="1.0"?>
<rdfs:Class rdf:ID="F16"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns="http://www.nato.org/aircraft#">
  <rdf:type rdf:resource="http://www.nato.org/aircraft#Aircraft"/>
  <wingspan>10 meters</wingspan>
  ...
</rdfs:Class>
```

"The F16 is an Aircraft, with a wingspan of 10 meters, …"

This clearly shows F16 being used as both a class and as an instance!

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

RDF Schemas: simple, yet powerful

- Let's summarize what we have learned:
  - Use RDF Schema to define:
    - a class hierarchy (a taxonomy),
    - properties
      - associate them with a class (use rdfs:domain)
      - indicate the range of values (use rdfs:range)
  - Once an RDF Schema is defined then it can be used to infer additional facts about data:
    - a class is an instance of all superclasses
    - a property is a specialization of its superproperty

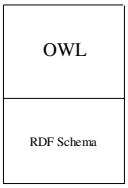
Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

Desire more expressiveness

- Although you can express a lot with RDF Schemas it is lacking in some desirable expressiveness:
  - Two classes, same concept - people use different words to represent the same thing. It would be very useful to be able to state "this class is equivalent to this second class".
    - One person may create an ontology with a class called "Airplane". Another person may create an ontology with a class called "Plane". It would be useful to be able to indicate that the two classes are equivalent.
  - Cardinality constraints - oftentimes it is useful to indicate the allowable number of occurrences of a property
    - Example. We would like to be able to express that a River has only "one" officialLength property.
    - Example. We would like to be able to express that an Ocean has one maxDepth.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

RDF Schemas: Building Block to More Expressive Ontology Languages



OWL = Web Ontology Language
- see the OWL Tutorial at:
http://www.xfront.com/owl/

RDF Schema was designed to be extended. The ontology languages all use RDF Schema's basic notions of Class, Property, domain, and range.

Roger L. Costello, David B. Jacobs. © 2003 The MITRE Corporation.

RDF Schema vs XML Schema

- XML Schemas is all about **syntax**.
- RDF Schema is all about **semantics**.
- An XML Schema tool is intended to **validate** that an XML instance conforms to the syntax specified by the XML Schema.
- An RDF Schema tool is intended to **provide additional facts** to supplement the facts in RDF/XML instances.
- XML Schemas is **prescriptive** - an XML Schema prescribes what an element may contain, and the order the child elements may occur.
- RDF Schemas is **descriptive** - an RDF Schema simply describes classes and properties.