22 September 2008

Seminar D

Ontology Design and Interoperability

International Conference on Dublin Core and Metadata Applications

DC-2008
BERLIN

Sam Oh
Department of Library & Information
Science Sung Kyun Kwan University

DC 2008 Seminar
Ontology Design & Interoperability: Topic Maps & RDF/OWL

Professor Sam Oh, Sungkyunkwan University, Seoul, Korea

ISO TC46/SC9 & JTC1/SC34 Chairman

DCMI Board of Trustee

samoh21@gmail.com

http://home.skku.edu/~samoh/

Ontology Design & Interoperability (1): Topic Maps

What is Topic Maps?

- An International Standard for knowledge integration
- *More importantly…*
- What is Topic Maps used for?

  Organizing large bodies of information

  + Capturing organizational memory

  + Representing complex rules and processes

  + Supporting concept-based eLearning

  + Managing distributed knowledge and information

  + Aggregating information and knowledge

  + etc…

  = **Seamless Knowledge**

Perspectives on Topic Maps

- **The information management perspective**
  - A new paradigm for organizing, retrieving, and navigating information resources
- **The knowledge management perspective**
  - A knowledge representation formalism optimized for use in information management
- **The library science perspective**
  - A way to collocate all knowledge about a subject – in particular its relationship to other subjects and to information resources

The TAO of Topic Maps

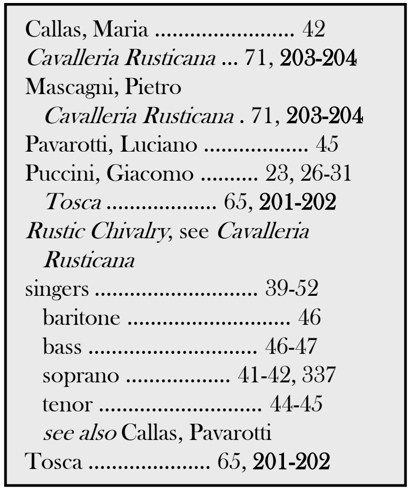
Topics

Associations

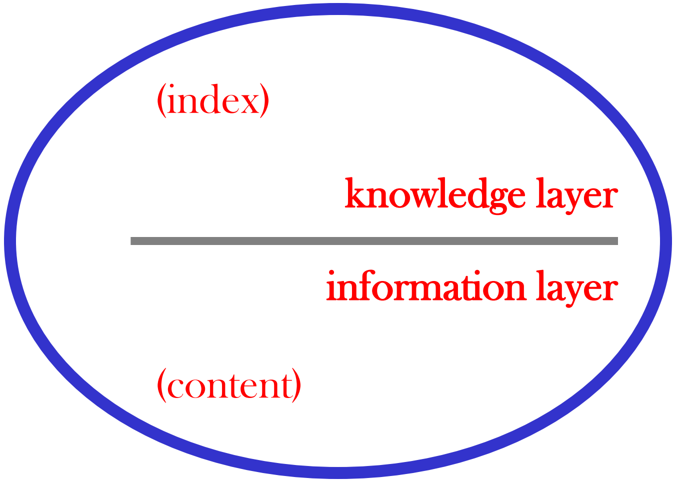
Occurrences

The core Topic Maps model

- The core concepts of Topic Maps are based on those of the back-of-book index
- The same basic concepts have been extended and generalized for use with digital information
- Envisage a 2-layer data model consisting of
  - a set of information resources (below), and
  - a "knowledge map" (above)
- This is like the division of a book into content and index

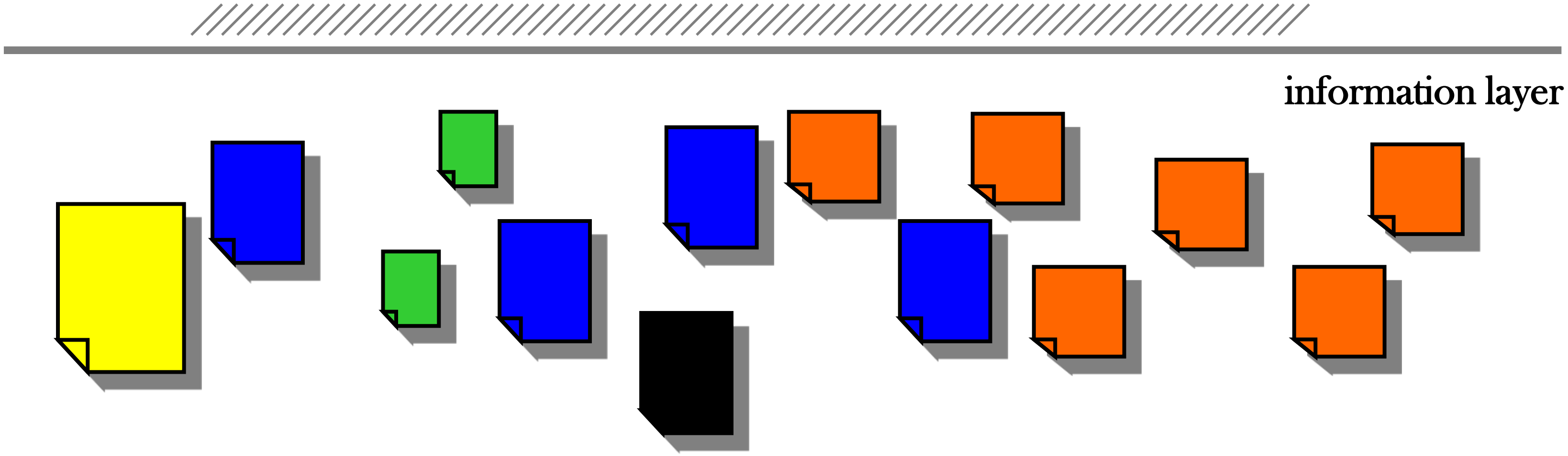


Callas, Maria	42
Cavalleria Rusticana ...	71, **203-204**
Mascagni, Pietro	
Cavalleria Rusticana .	71, **203-204**
Pavarotti, Luciano	45
Puccini, Giacomo	23, 26-31
Tosca	65, **201-202**
Rustic Chivalry, see *Cavalleria Rusticana*	
singers	39-52
baritone	46
bass	46-47
soprano	41-42, 337
tenor	44-45
see also Callas, Pavarotti	
Tosca	65, **201-202**



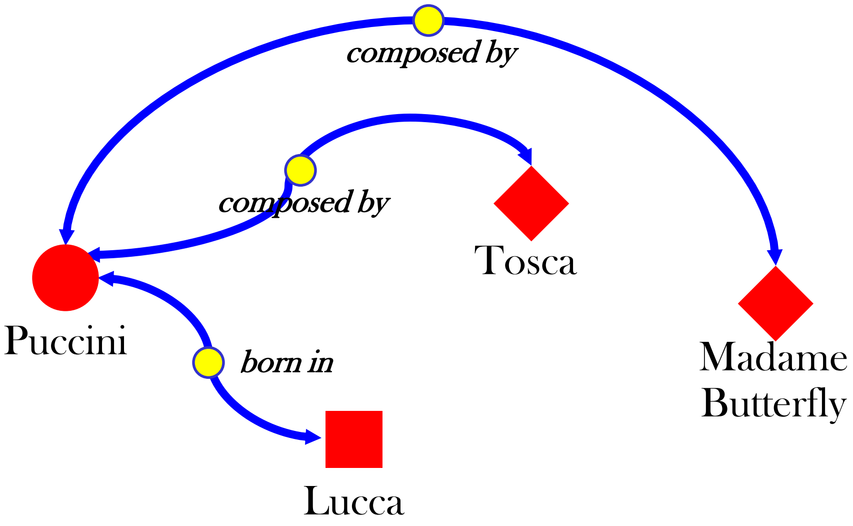
(1) The information layer

- The lower layer contains the content
  - usually digital, but need not be
  - can be in any format or notation
  - can be text, graphics, video, audio, etc.
- This is like the content of the book to which the back-of-book index belongs



(2) The knowledge layer

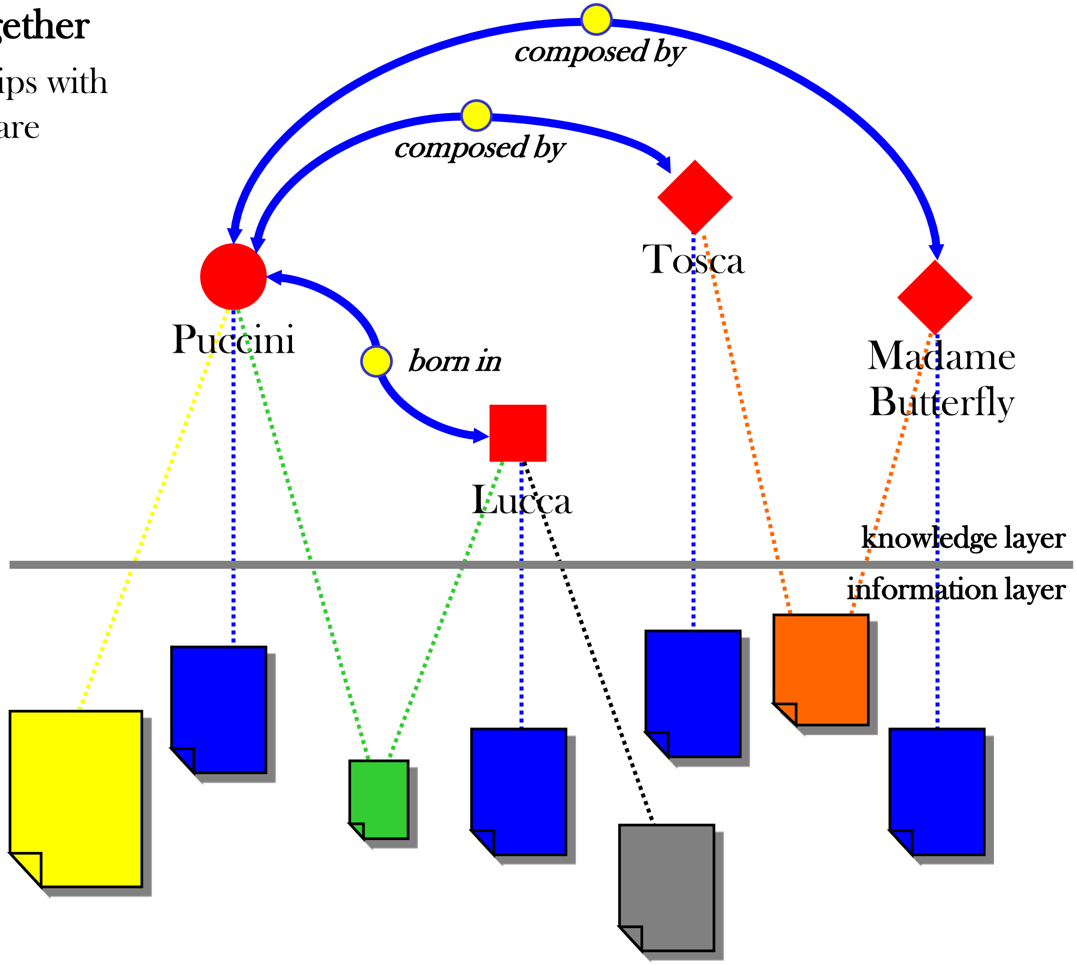
- **The upper layer consists of topics and associations**
  - Topics represent the subjects that the information is about
    - Like the list of topics that forms a back-of-book index
  - Associations represent relationships between those subjects
    - Like "see also" relationships in a back-of-book index



knowledge layer

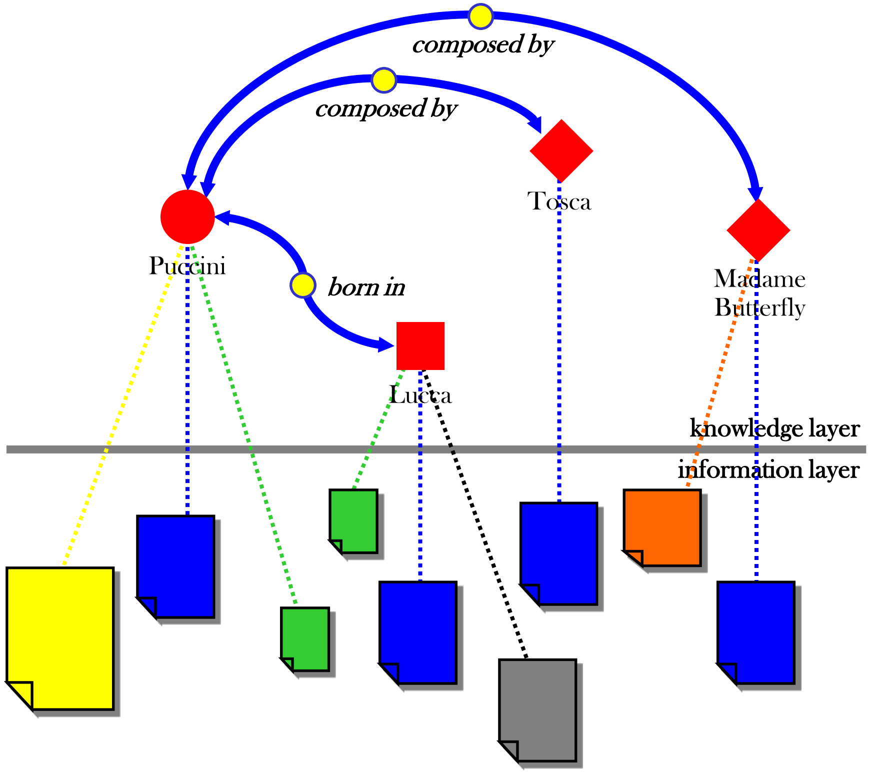
Linking the layers through occurrences

- The two layers are linked together
  - Occurrences are relationships with information resources that are pertinent to a given subject
  - The links (or locators) are like page numbers in a back-of-book index



Summary of core concepts

- A pool of information or data
  - any type or format
- A knowledge layer, consisting of:
- Topics
  - a set of knowledge topics for the domain in question
- Associations
  - expressing relationships between knowledge topics
- Occurrences
  - information that is relevant in some way to a given knowledge topic
- = The TAO of Topic Maps

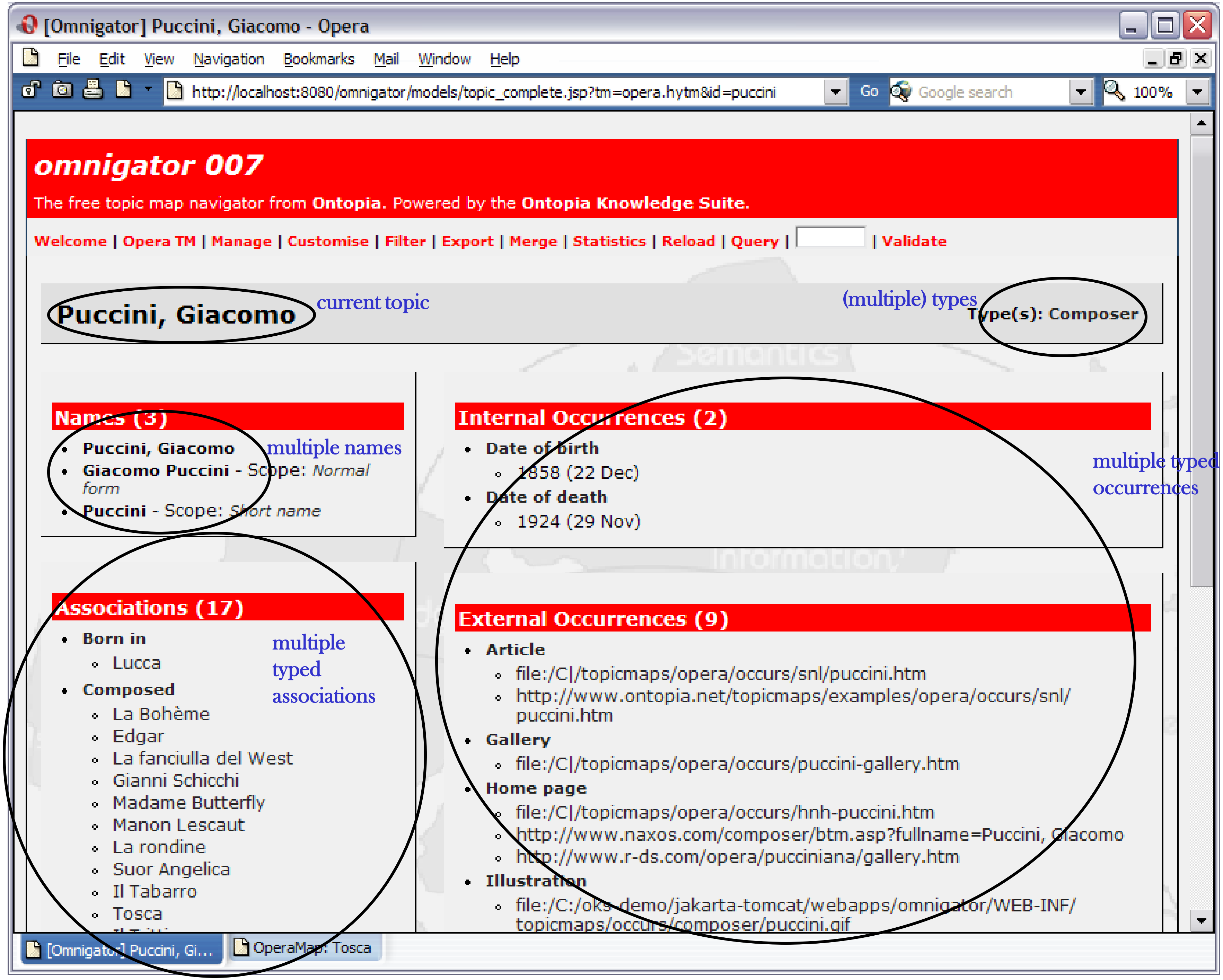


Topic Maps and ontologies

- The basic building blocks are
  - Topics: e.g. “Puccini”, “Lucca”, “Tosca”
  - Associations: e.g. “Puccini was born in Lucca”
  - Occurrences: e.g. “*http://www.opera.net/puccini/bio.html* is a biography of Puccini”
- Each of these constructs can be typed
  - Topic types: “composer”, “city”, “opera”
  - Association types: “born in”, “composed by”
  - Occurrence types: “biography”, “street map”, “synopsis”
- All such types are also topics (within the same topic map)
  - “Puccini” is a topic of type “composer” ... and “composer” is also a topic
- A topic map thus contains its own ontology
  - Our definition of “ontology”:
  - *The types and subtypes of concepts and relations that exist in some domain…*

With this simple but flexible model you can

- **Make knowledge explicit, by**
  - Identifying the *subjects* that your information is about
  - Expressing the *relationships* between those subjects
- **Bridge the domains of knowledge and information, by**
  - Describing where to find *information* about the knowledge topics
  - Linking information about a common topic across multiple repositories
- **Transcend simple categories, hierarchies, and taxonomies, by**
  - Applying rich associative structures that capture the complexity of knowledge
- **Enable implicit knowledge to be made explicit, by**
  - Providing clearly identifiable hooks for attaching implicit knowledge



[Omnigator] Puccini, Giacomo - Opera
File Edit View Navigation Bookmarks Mail Window Help
http://localhost:8080/omnigator/models/topic_complete.jsp?tm=opera.hytm&id=puccini
Go
Google search
100%
[Omnigator] Puccini, Gi...
OperaMap: Tosca

omnigator 007

The free topic map navigator from **Ontopia**. Powered by the **Ontopia Knowledge Suite**.

Welcome | Opera TM | Manage | Customise | Filter | Export | Merge | Statistics | Reload | Query | | Validate

Puccini, Giacomo

current topic

(multiple) types

Type(s): Composer

Names (3)

- **Puccini, Giacomo**
- **Giacomo Puccini** - Scope: *Normal form*
- **Puccini** - Scope: *Short name*

multiple names

Internal Occurrences (2)

- **Date of birth**
  - 1858 (22 Dec)
- **Date of death**
  - 1924 (29 Nov)

multiple typed occurrences

Associations (17)

- **Born in**
  - Lucca
- **Composed**
  - La Bohème
  - Edgar
  - La fanciulla del West
  - Gianni Schicchi
  - Madame Butterfly
  - Manon Lescaut
  - La rondine
  - Suor Angelica
  - Il Tabarro
  - Tosca

multiple typed associations

External Occurrences (9)

- **Article**
  - file:/C|/topicmaps/opera/occurs/snl/puccini.htm
  - http://www.ontopia.net/topicmaps/examples/opera/occurs/snl/puccini.htm
- **Gallery**
  - file:/C|/topicmaps/opera/occurs/puccini-gallery.htm
- **Home page**
  - file:/C|/topicmaps/opera/occurs/hnh-puccini.htm
  - http://www.naxos.com/composer/btm.asp?fullname=Puccini, Giacomo
  - http://www.r-ds.com/opera/pucciniana/gallery.htm
- **Illustration**
  - file:/C:/oks-demo/jakarta-tomcat/webapps/omnigator/WEB-INF/topicmaps/occurs/composer/puccini.gif

Advanced concepts in Topic Maps

Topics and subjects

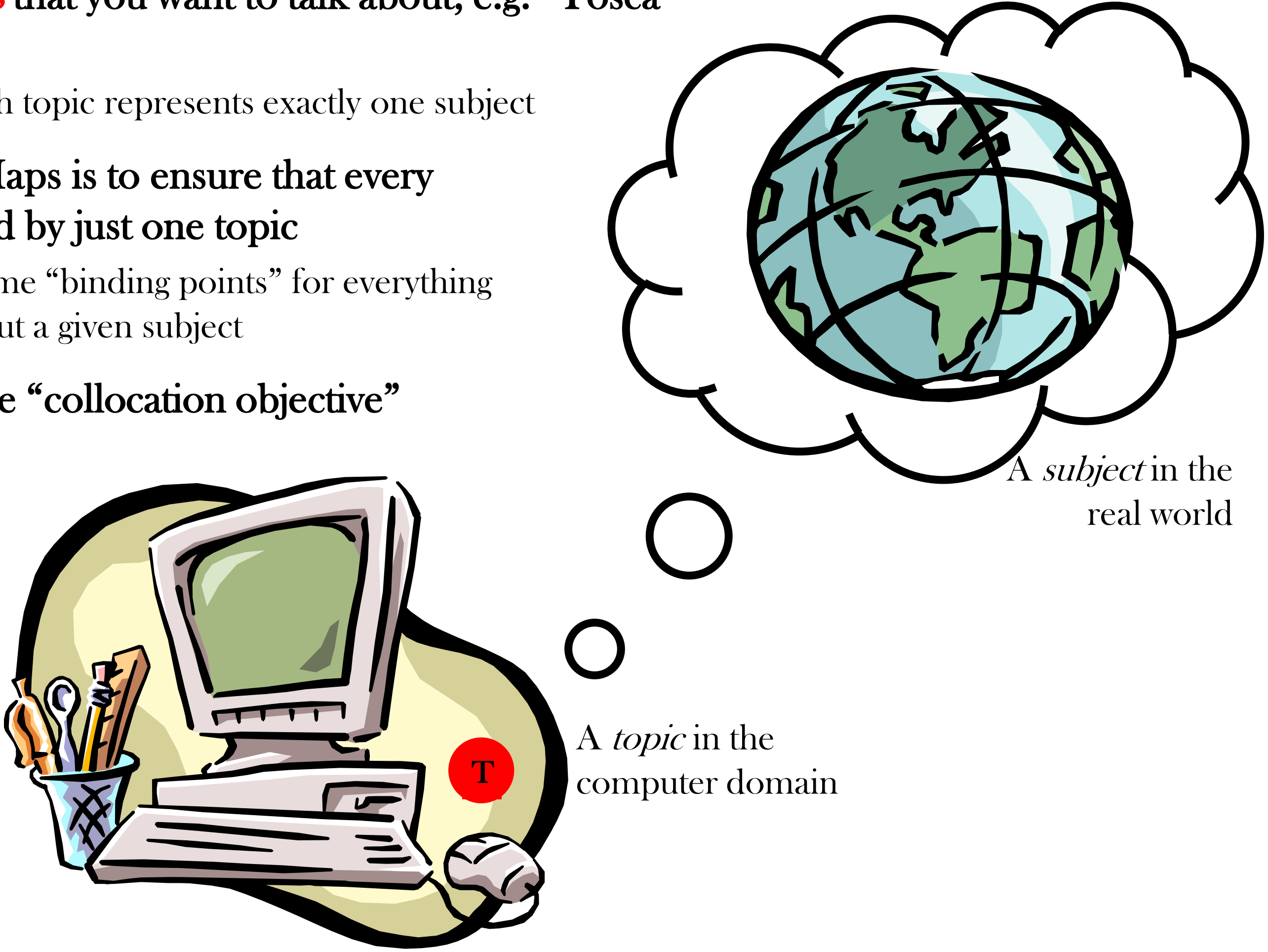
Names and identifiers

Association roles

Reification

T is for Topic

- **Topics are surrogates, or “proxies” (inside the computer) for the ineffable *subjects* that you want to talk about, e.g. “Tosca” or the “Puccini”**
  - By definition, each topic represents exactly one subject
- **The goal in Topic Maps is to ensure that every subject is represented by just one topic**
  - Topics thus become “binding points” for everything that is known about a given subject
- **This goal is called the “collocation objective”**

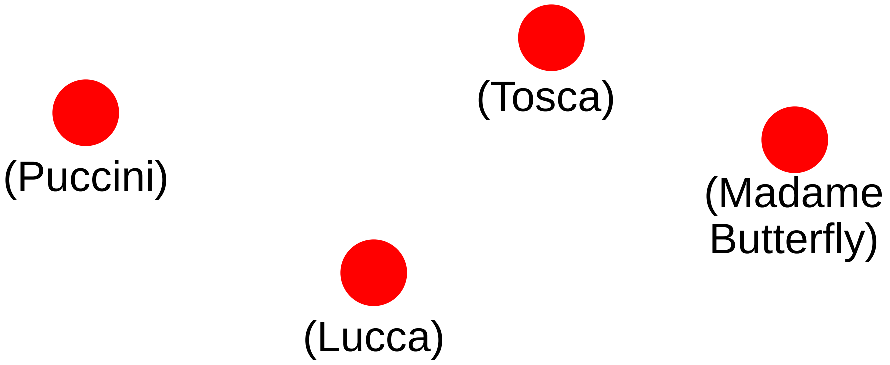


A *subject* in the real world

A *topic* in the computer domain

Topics and subjects

- **Topics represent subjects**
- **A subject can be anything you want:**
  - ISO 13250 definition:
    A subject is any "thing" whatsoever, whether or not it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever."
  - Subjects can be persons, entities, concepts, whatever – depending on the information in question
- *Some examples...*



(Tosca)

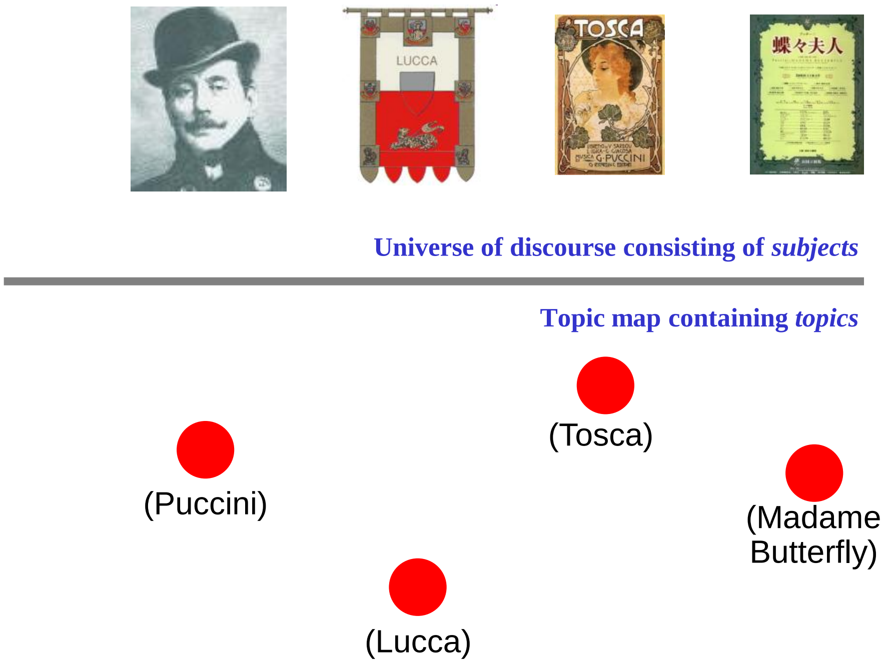
(Puccini)

(Madame Butterfly)

(Lucca)

Topic identifiers

- **Topics represent subjects**
- **But how do we know *which* subject a topic represents?**
  - And how do we know when two topics represent the same subject?
- **We cannot rely on names**
  - Synonym problem:
    - *One subject, Multiple names*
  - Homonym problem:
    - *One name, Multiple subjects*
  - Problem of multiple languages:
    - *One subject, Multiple names*
  - Names are just not reliable enough!
- **The answer is to use *identifiers***
  - Identifiers are not required, but they are an important aid in achieving the collocation objective



Universe of discourse consisting of *subjects*

Topic map containing *topics*

(Tosca)

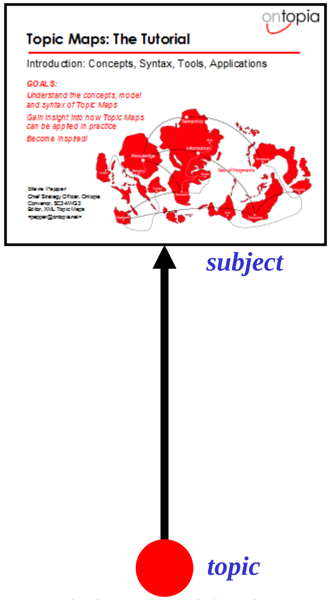
(Puccini)

(Madame Butterfly)

(Lucca)

Subject locators

- **Sometimes the subject is an information resource (like these slides)**
  - It exists somewhere within the computer system, has a location, and can therefore be “addressed”
    - For example, these slides might be located at http://www.ontopia.net/tutorials/tm-intro.ppt
  - The address of an *addressable subject* can be used to unequivocably establish the subject’s identity
  - An address (usually a URL) used to identify a subject directly is called a *subject locator*
- **But most subjects are *not* information resources**
  - Puccini, Lucca, Tosca, Madame Butterfly, love, darkness, French, …
  - These all exist outside the computer domain and cannot be addressed directly



ontopia

Topic Maps: The Tutorial

Introduction: Concepts, Syntax, Tools, Applications

GOALS:

Understand the concepts, model and syntax of Topic Maps

Gain insight into how Topic Maps can be applied in practice

Become inspired!

Steve Pepper
Chief Strategy Officer, Ontopia
Convenor, SC34/WG3
Editor, XML Topic Maps
<pepper@ontopia.net>

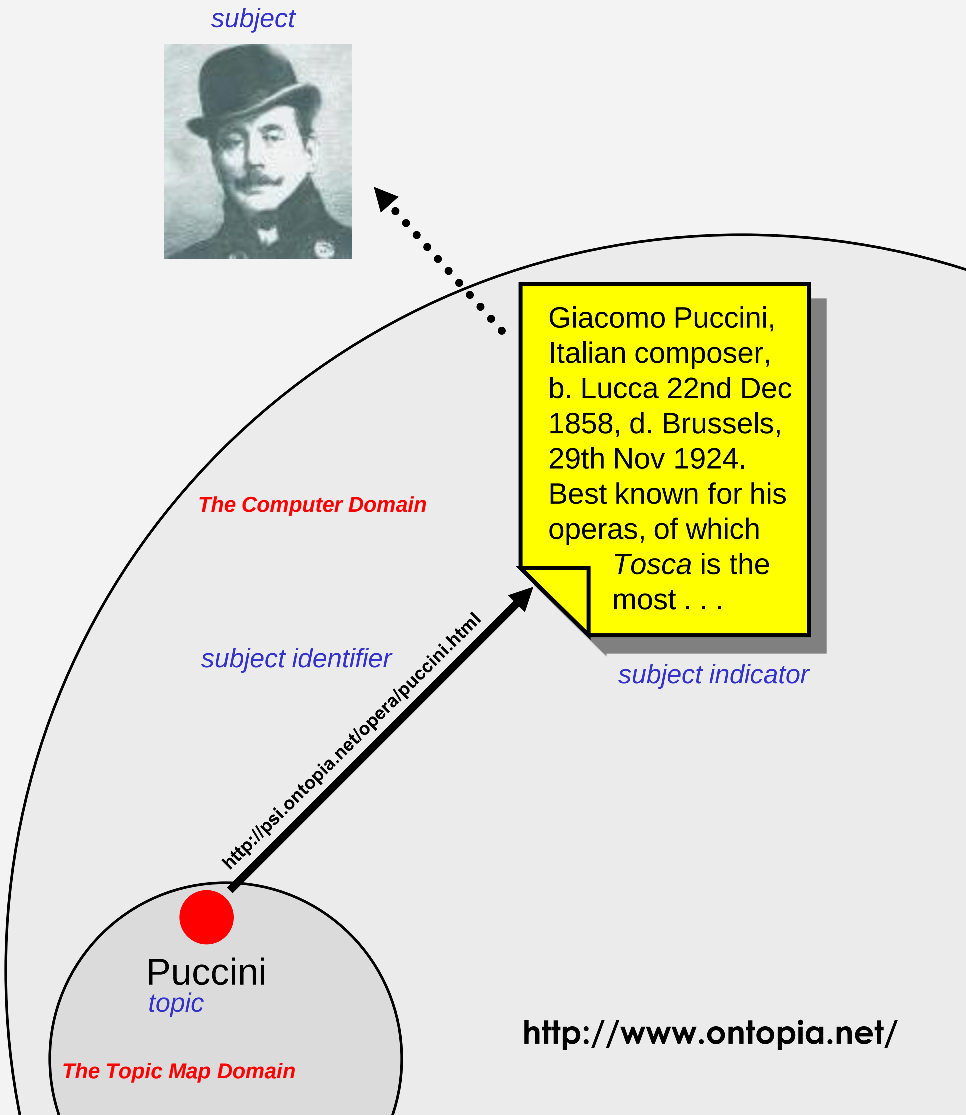
(These slides)

subject locator:
http://www.ontopia.net/tutorials/tm-intro.ppt

Life, the Universe and Everything

Subject identifiers

- **The identity of most subjects can only be established *indirectly***
  - An information resource (like a definition or a picture) can provide some kind of *indication* of the subject's identity to a human
  - Such a resource is called a *subject indicator*
  - A topic may have multiple subject indicators
- **Because it is a resource, a subject indicator has an address, even though the subject that it is indicating does not**
  - Computers can use the address of the subject indicator to establish identity
  - These are called *subject identifiers*
- **Subject indicators and subject identifiers are the two sides of the human-computer dichotomy**



subject

Giacomo Puccini, Italian composer, b. Lucca 22nd Dec 1858, d. Brussels, 29th Nov 1924. Best known for his operas, of which *Tosca* is the most . . .

subject indicator

The Computer Domain

subject identifier

http://psi.ontopia.net/opera/puccini.html

Puccini

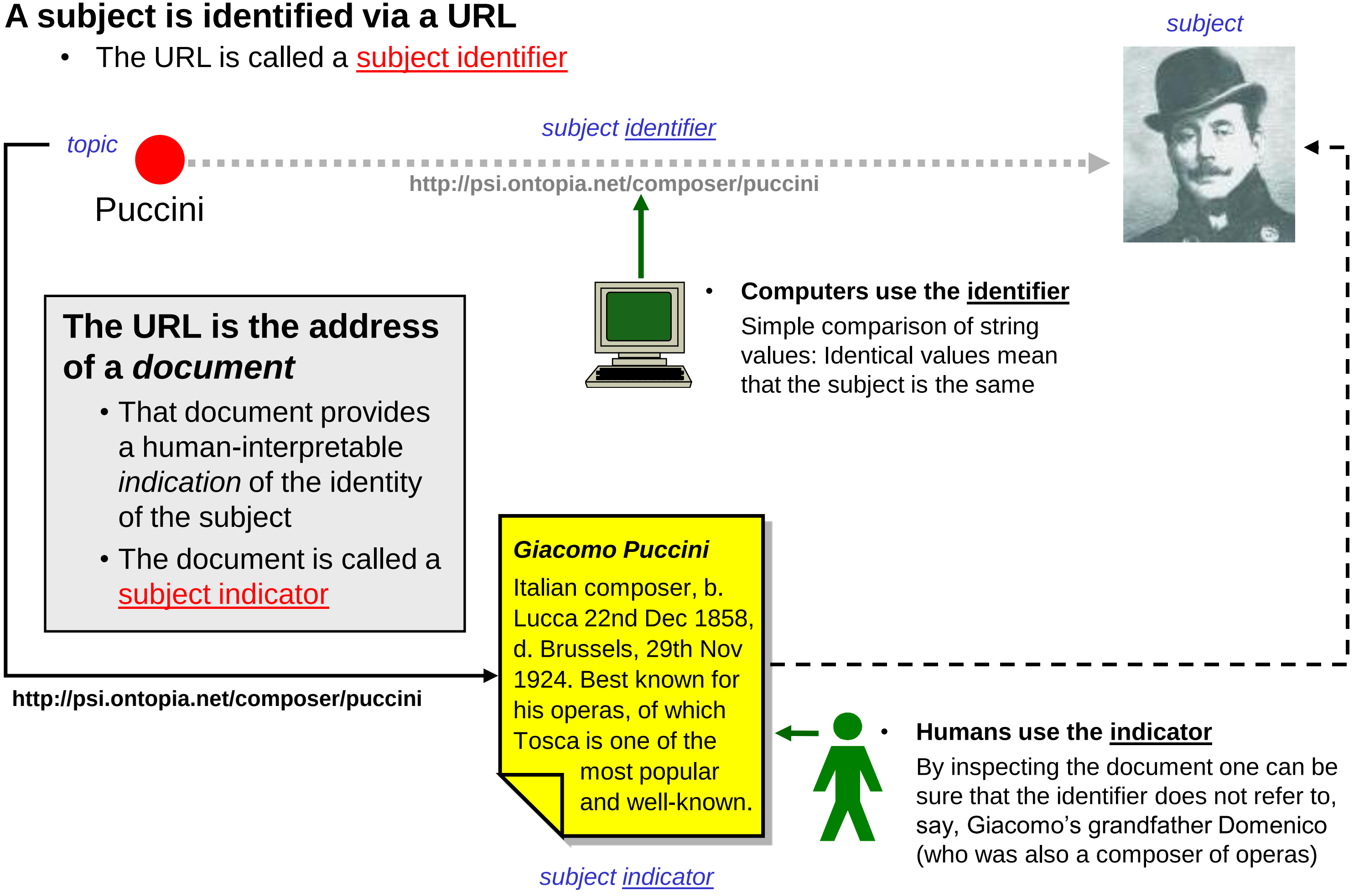
topic

The Topic Map Domain

Subject identifier and subject indicator

A subject is identified via a URL

- The URL is called a subject identifier


subject
topic
subject identifier
http://psi.ontopia.net/composer/puccini
Puccini
http://psi.ontopia.net/composer/puccini
subject indicator

The URL is the address of a *document*

- That document provides a human-interpretable *indication* of the identity of the subject
- The document is called a subject indicator

- **Computers use the identifier**
  Simple comparison of string values: Identical values mean that the subject is the same

Giacomo Puccini

Italian composer, b. Lucca 22nd Dec 1858, d. Brussels, 29th Nov 1924. Best known for his operas, of which Tosca is one of the most popular and well-known.

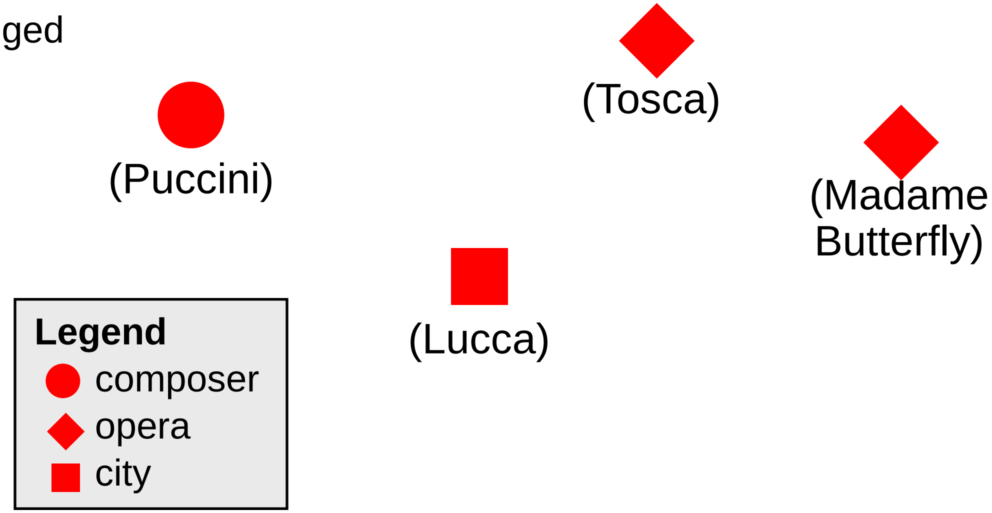
- **Humans use the indicator**
  By inspecting the document one can be sure that the identifier does not refer to, say, Giacomo's grandfather Domenico (who was also a composer of operas)

Published subjects

- **Published Subject Indicator (PSI)**
  - A subject indicator that has been made available for use across applications
- **Anyone can publish PSI sets!**
  - Adoption of PSI sets will be an evolutionary process based on trust
  - It will lead to greater and greater interoperability – between topic map applications, between topic maps and RDF, and across the Semantic Web in general
  - Publishers and users of ontologies may be among the greatest beneficiaries
- **OASIS Published Subjects TC**
  - Has published guidelines publishers of PSI sets
    - http://www.oasis-open.org/committees/tm-pubsubj/
  - Has published PSI sets for countries and languages
    - http://psi.oasis-open.org/iso/3166
    - http://psi.oasis-open.org/iso/639

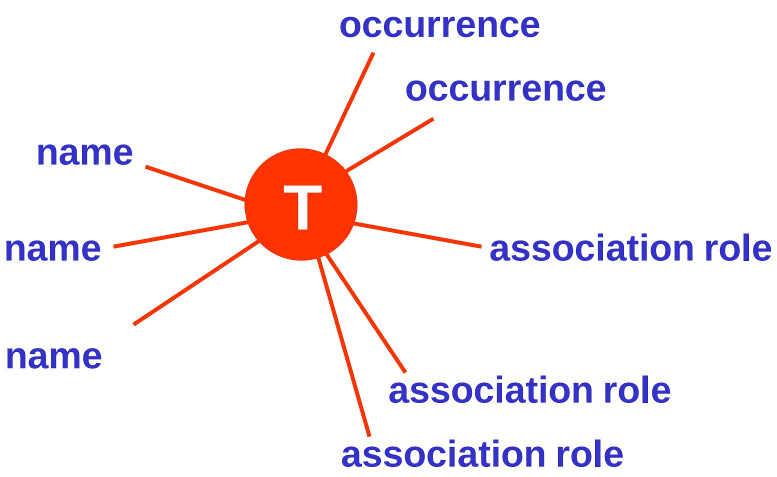
Topic types

- **A topic type represents the *class* of which the topic is an *instance*, e.g.**
  - the topic Puccini is of type “composer”
  - the topic Tosca is of type “opera”
- **Topic typing has standard type-instance (or class-instance) semantics**
  - CAUTION! This is *not* the same as the supertype-subtype (or subtypeOf) relationship...
  - You *must not* use topic typing to state that “a composer is a person”
- **A special kind of association**
  - The relationship between a topic and its type is actually just a privileged kind of association
- **Topic types are also topics**
  - You can define whatever types you need for your application



Topic characteristics

- Zero or more *names*
- Zero or more *occurrences*
- Zero or more *roles in associations*

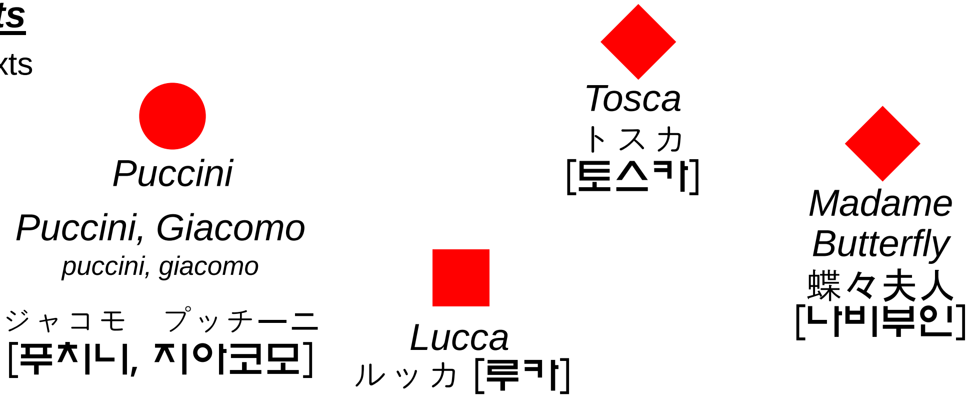


Or, as Steve Newcomb once said:

- **The truth about Topic Maps is that, at the end of the day, there are**
  - topics, which have
  - names and occurrences, and
  - play roles in associations
- ***...And that's all there is!***

Topic names

- **A topic may have multiple names (called *base names*)**
  - Applications can choose the most appropriate name for any given context
  - Typical uses are
    - Supporting multiple natural languages
    - Handling synonyms
- **Each base name may have *variants***
  - For use in specific processing contexts
  - Contexts are specified via parameters such as display, sort, etc.
  - (Display and sort are predefined; others may be defined by the topic map author)
- **In the next version of ISO 13250, names will also have *types***



Puccini

Puccini, Giacomo

puccini, giacomo

ジャコモ　プッチーニ
[푸ㅊㅣㄴㅣ, ㅈㅣ아코모]

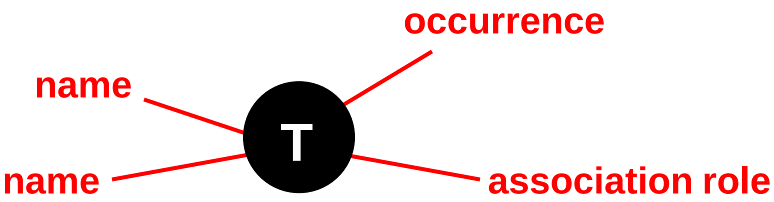
Tosca
トスカ
[토스ㅋㅏ]

Lucca
ルッカ [루ㅋㅏ]

Madame Butterfly
蝶々夫人
[ㄴㅏ비부인]

Scope

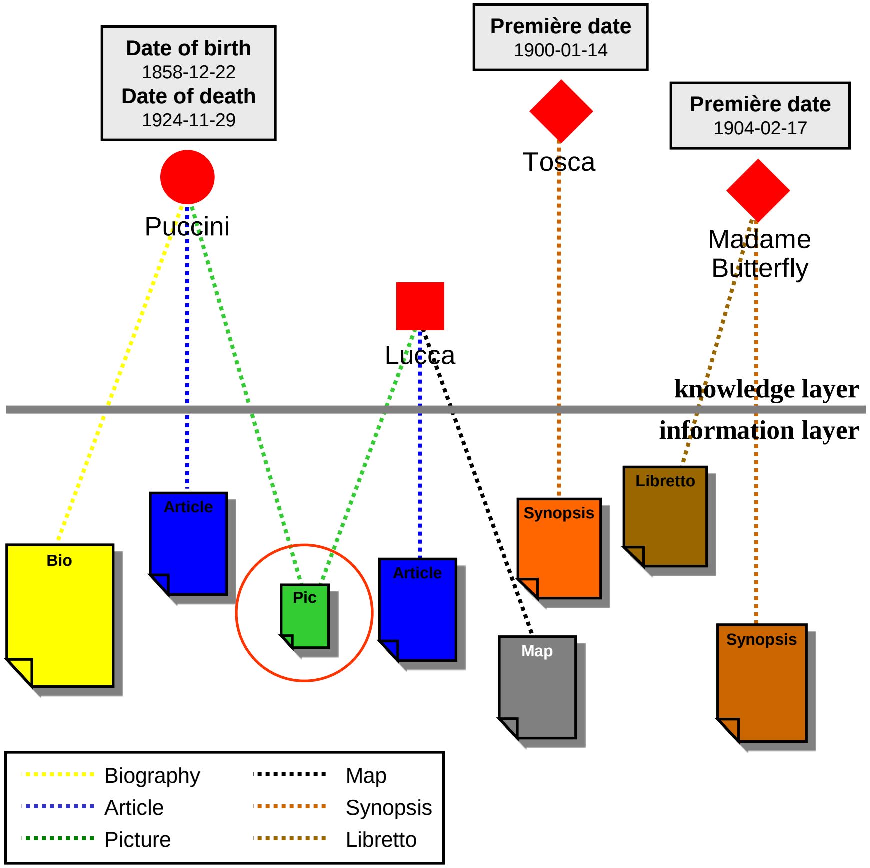
- **Topics have “characteristics”**
  - Its names and occurrences, and the roles it plays in associations with other topics
- **Every characteristic is valid within some context (scope), e.g.**
  - the name “Norge” for the topic Norway in the scope “Norwegian”
  - a certain information occurrence in the scope “technician”
  - a given association is true in the scope (according to) “Authority X”
- **Scope is expressed as a set of topics**
  - e.g. Norwegian, technician, Authority X
- **Scope can be used as the basis for filtering**
  - More on applications of scope later



Filtering by scope

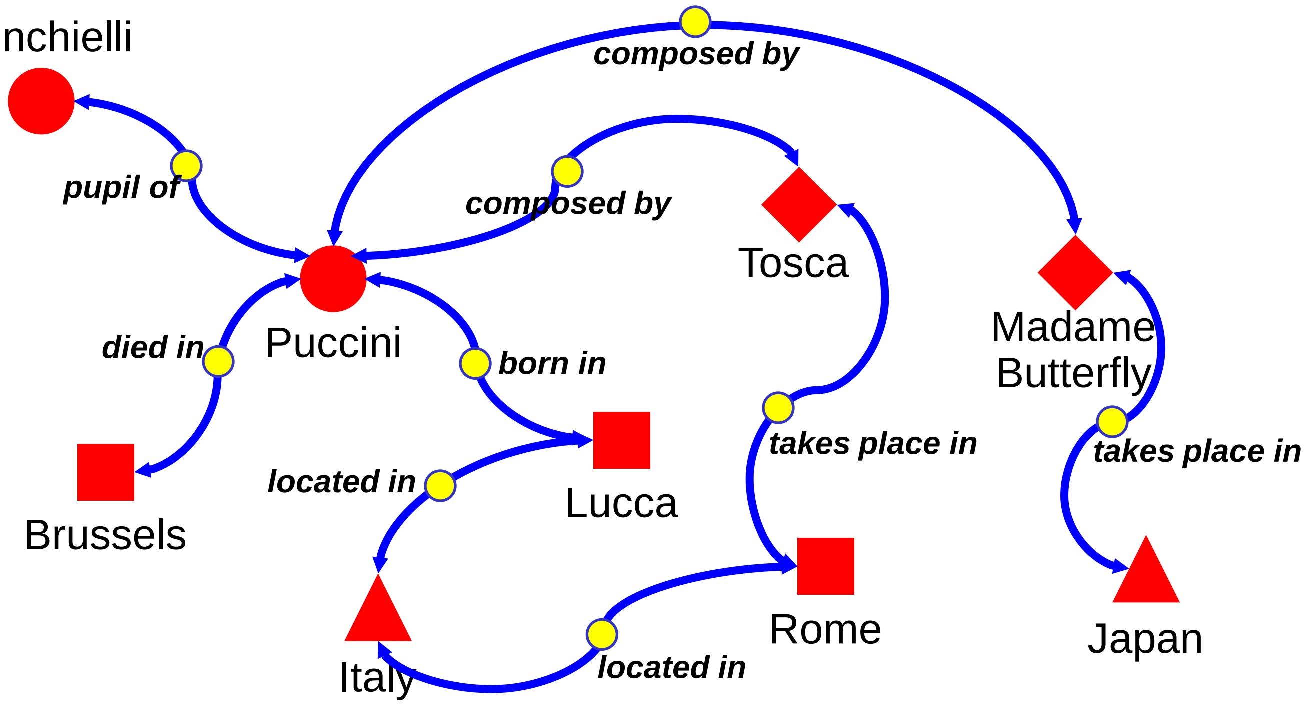
O is for Occurrence

- **Occurrences are relationships between *information resources* and subjects**
  - Resources could be documents, fields in a database, or any other kind of data
- **Occurrences can be *internal* or *external***
- **External resources are referenced by locators**
  - These correspond to page numbers in indexes
  - They usually take the form of URLs
  - They link the knowledge layer to the information layer
- **Internal occurrences are used to represent properties**
  - The data is stored inside the topic map
- **Occurrences can be typed… occurrence types are also topics**
  - You can define whatever types you need for your application



A is for Association

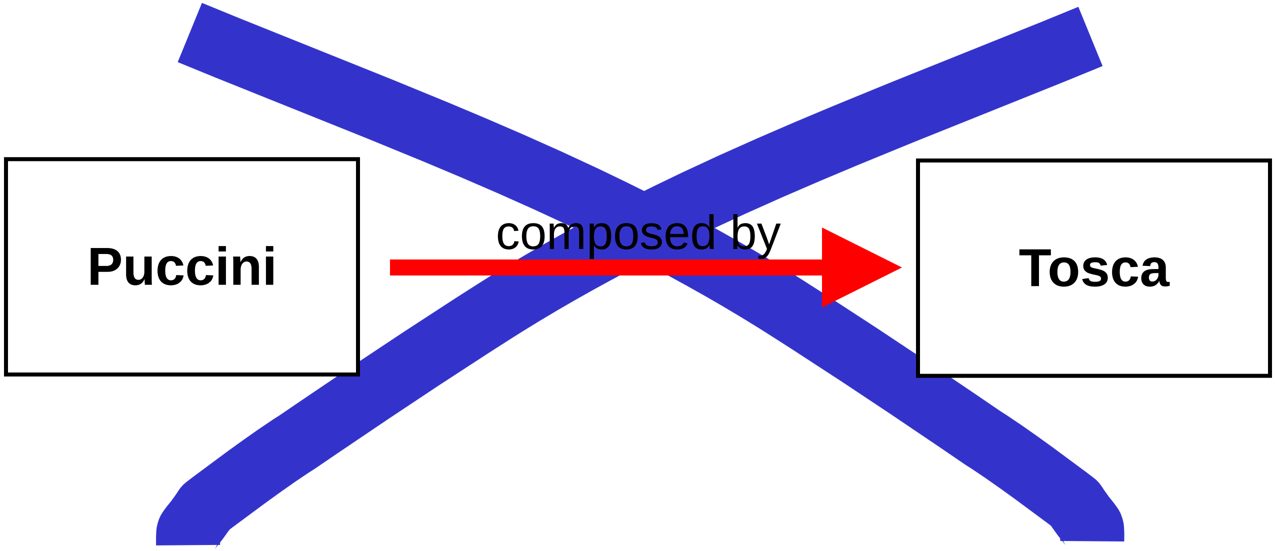
- **Associations express *relationships* between subjects, e.g. "Puccini" *was born in* "Lucca"**
- **Together, topics and associations constitute a semantic network, or "knowledge map"**
- **Associations can be typed…**
- **Association types are also topics**
  - You can define whatever types you need for your application



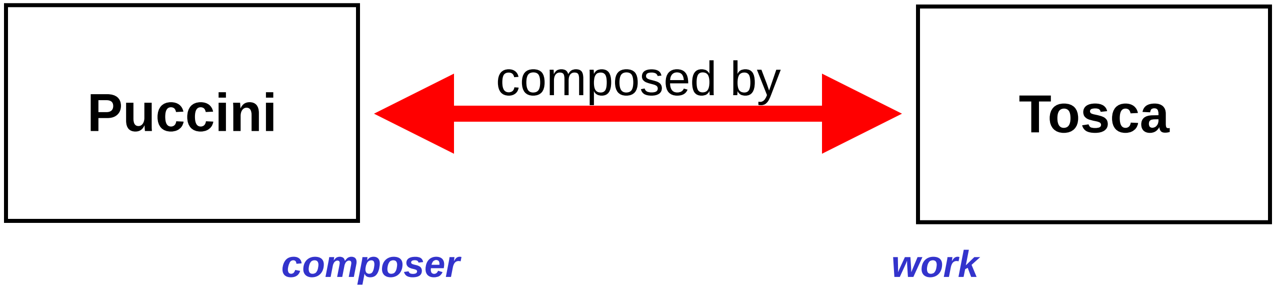
The arity of associations

- **Associations do not have to be binary**
  - i.e., there does not have to be exactly two role playing topics
- **In topic maps, associations can involve one, two or more topics**
  - Unary associations are not common
    - Can be useful for representing properties that have boolean values
      - e.g., the property of being “unfinished”
  - Binary associations are the most common
    - Often correspond to verb (subject, object) constructs
  - Ternary associations are quite common
    - Often correspond to verb(subject, object, indirect object) constructs
  - N-ary associations (n > 3) less common but sometimes useful
- **Rule of thumb:**
  - Stick to binary associations wherever possible

Associations have no direction

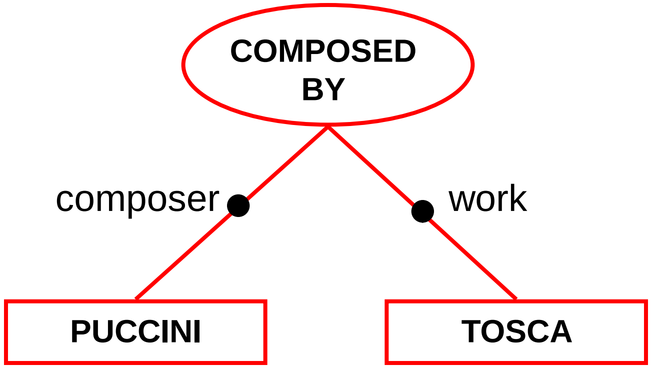


Instead associations have *roles*

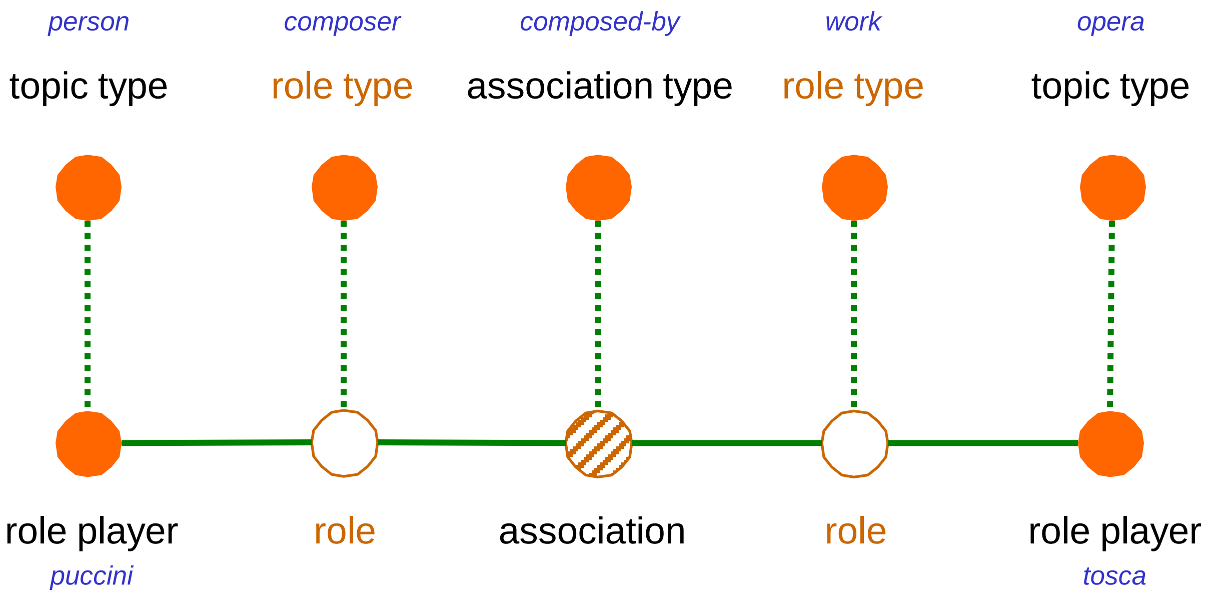


Association roles

- **Associations are inherently multidirectional**
  - Associations assert relationships between subjects
  - The statement *Puccini composed Tosca* automatically implies the statement *Tosca was composed by Puccini*
- **The impression of directionality is caused by the use of natural language as a serialization syntax**
- **Instead of *directionality* topic maps use *association roles:***
  - The topic Puccini plays the role of *composer* in the *composed by* relationship with Tosca playing the role of *work*



Roles, role players and role types



person *composer* *composed-by* *work* *opera*

topic type role type association type role type topic type

role player role association role role player

puccini *tosca*

N.B.
role == *association role* and
role type == *association role type*

Reification

- **From latin “re” = “thing”**
  - i.e. “thingification”
- **In Topic Maps, for “thing” read “topic”**
  - So reification is about turning something into a topic
  - Specifically it is about turning topic map constructs that are not already topics (i.e., names, occurrences, associations, association roles, and topic maps) into topics
- **This allows assertions to be made about**
  - roles, relationships, names, and topic maps
- **Useful for**
  - Adding metadata (especially to the topic map itself)
  - Providing further levels of detail in the map

Five cool things to do with a topic map

Querying

Constraining

Filtering

Visualizing

Merging

Querying topic maps

- **Topic Maps is based on a formal data model**
  - More on the Topic Maps Data Model later
- **This means that topic maps can be queried, like databases**
- **ISO/IEC 18048 Topic Maps Query Language (TMQL)**
  - Intended to simplify application development
  - Used to extract information and modify TMs
  - Currently being developed by WG3

Constraining topic maps

- **ISO 13250 provides no way to constrain topic maps**
- **ISO/IEC 19756 Topic Map Constraint Language (TMCL)**
  - Used to define constraints on topic maps
  - Will interoperate with OWL (Web Ontology Language)
  - Currently being developed by WG3
- **Examples of constraints:**
  - “All persons must be born somewhere”
  - “A person may have died somewhere”
  - “All persons must have a date of birth occurrence, which must contain a date”
  - “Email occurrences are unique”

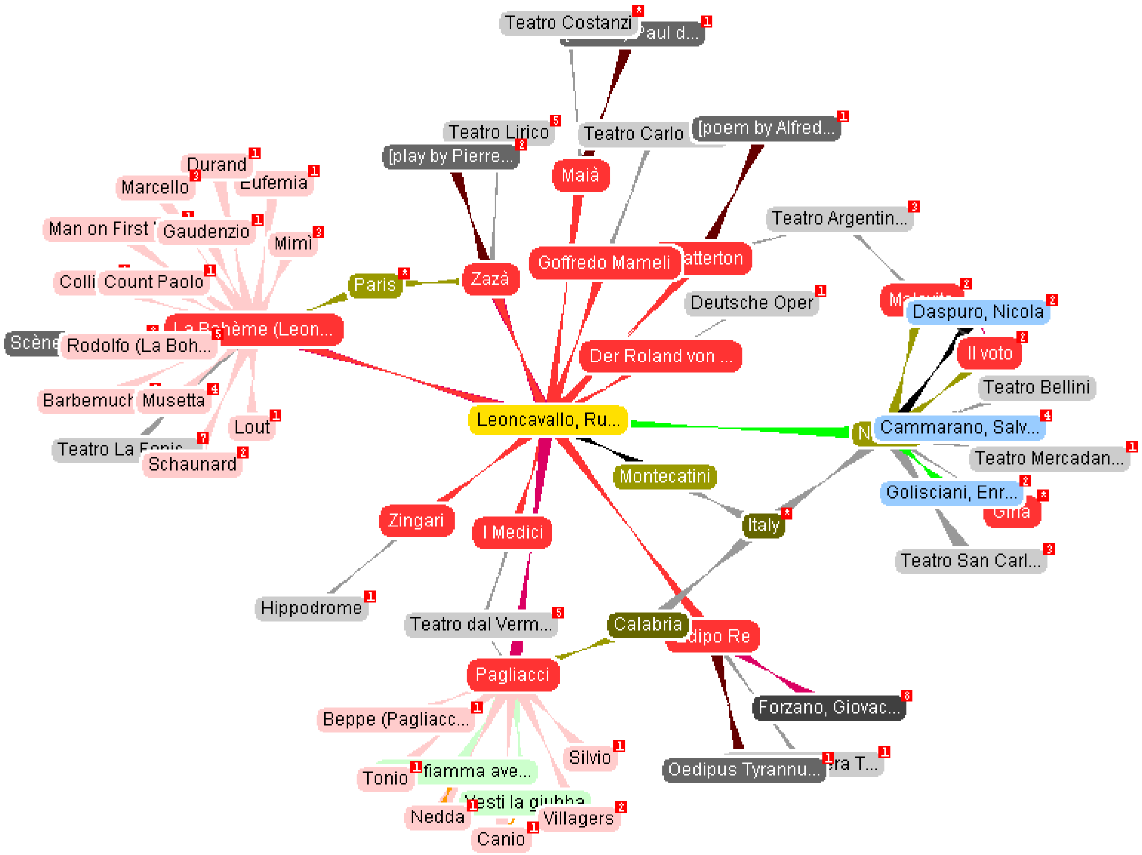
Filtering, scoping and personalizing topic maps

- **Multiple world views**
  - Reality is ambiguous and knowledge has a subjective dimension
  - Scope allows the expression of multiple perspectives in a single Topic Map
- **Contextual knowledge**
  - Some knowledge is only valid in a certain context, and not valid otherwise
  - Scope enables the expression of contextual validity
- **Keeping track of provenance**
  - When the source of knowledge is as important as the knowledge itself:
  - Scope allows retention of knowledge about the source of knowledge
- **Personalized knowledge**
  - Different users have different knowledge requirements
  - Scope permits personalization based on personal references, skill levels, security clearance, etc.

Visualizing topic maps

- The network or graph structure of a topic map can be visualized for humans

- This provides another “view” on information that can lead to new insights

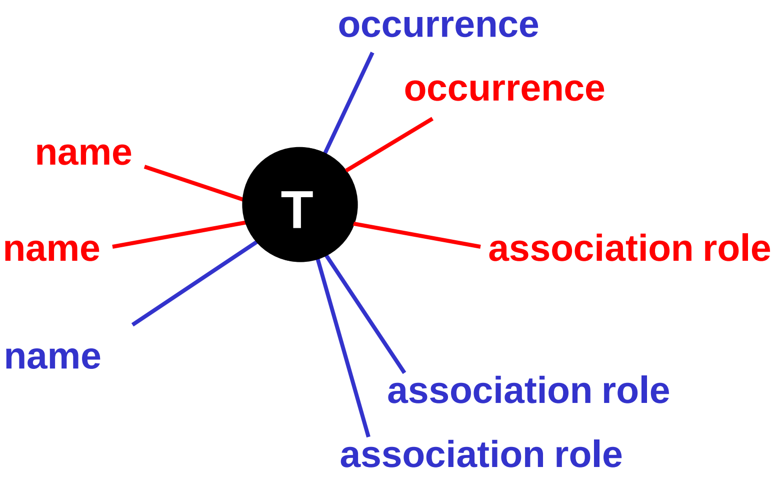


Merging topic maps

- **Topic Maps can be merged automatically**
  - You can always and in any situation take any two arbitrary topic maps and merge them to a single topic map
  - This cannot be done with databases or XML documents
- **The merge capability enables many advanced applications**
  - Information integration across repositories
  - Knowledge sharing across organizations
  - Spontaneous knowledge aggregation
  - Distributed knowledge management
  - Reuse of knowledge across applications
- **The concept that makes merging possible is subject identity**
  - This is the reason for using subject identifiers

Principles of merging in Topic Maps

- **In Topic Maps, every topic represents some subject**
- **The collocation objective requires exactly one topic per subject**
  - When two topic maps are merged, topics that represent the same subject should be merged to a single topic
  - When two topics are merged, the resulting topic has the union of the characteristics of the two original topics



...and the resulting topic has the union of the original characteristics

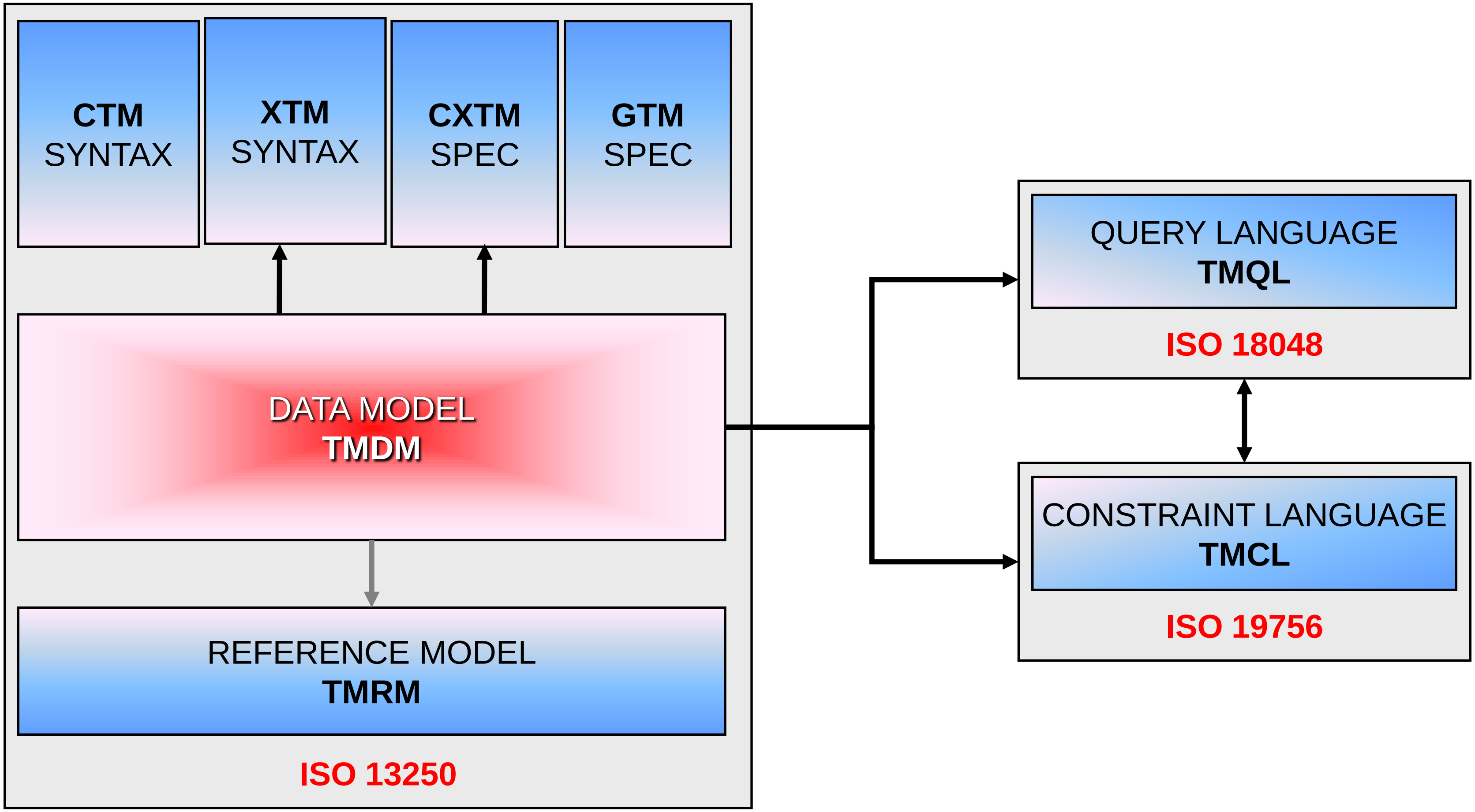
Applications of merging

- **Reuse of knowledge**
  - Store discrete fragments of knowledge as topic maps
  - Merge into other applications as needed
    - e.g. company organization chart, geographical information, etc.
- **Information integration**
  - Generate a topic map for each repository
  - Merge them to provide a unified view of the whole
    - e.g. CRM, DMS, ERP systems
- **Distributed knowledge management**
  - Capture knowledge in departmental topic maps
  - Merge them for the corporate view
    - No need to centralize KM in order to make knowledge sharable
- **“Seamless knowledge”**
  - Connect portals seamlessly across and between organizations
  - No need to enforce a common vocabulary

How Topic Maps improves access to information

- **Intuitive navigational interfaces for humans**
  - The topic/association layer mirrors the way people think
- **Powerful semantic queries for applications**
  - A formal underlying data structure
- **Customized views based on individual requirements**
  - Personalization based on scope
- **Information aggregation across systems and organizations**
  - Topic Maps can be merged automatically…

Roadmap to the TM standards



CTM SYNTAX

XTM SYNTAX

CXTM SPEC

GTM SPEC

DATA MODEL **TMDM**

REFERENCE MODEL **TMRM**

ISO 13250

QUERY LANGUAGE **TMQL**

ISO 18048

CONSTRAINT LANGUAGE **TMCL**

ISO 19756

Creating Topic Maps

Sources of Topic Maps data

Automatic generation of topic maps

Manual editing of topic maps

Legacy data handling

- **“Legacy Data” is any data that has not been topic mapped…**
- **The key decision is the type of conversion to use:**
  - One time extraction of topic map
  - Repeated batch extraction process
  - Wrapper around a legacy system
- **Each approach has advantages and disadvantages**

Aspects of topic map creation

- **Identifying typing topics**
  - i.e., the topics that define classes (or types) of topics, associations and occurrences
  - e.g., “person”, “company”, “employedBy”, “partnerOf”, “web site”, “bio”
- **Identifying individual topics**
  - e.g., “Ontopia”, “Innodata-Isogen”, “Pepper”, “Kimber”
- **Identifying individual associations**
  - e.g., “Pepper is employed by Ontopia”
- **Identifying individual occurrences**
  - e.g., a bio of Pepper, a photo of Kimber, an org. chart of Innodata-Isogen
- **These tasks are significantly different in their complexity and the frequency with which they must be performed**
- **Many can be performed automatically**

Sources of topic map data

- Structured knowledge
- Document metadata
- Structured document content
- Unstructured document content
- Information systems
- Knowledge in people’s heads

Structured knowledge

- **Ontologies and classification systems**
  - Published subjects, OWL, LCSH, DDC
  - Sources of topic types and association types for various domains
- **Database schemas**
  - Entities map to topic types
  - Relations map to association types
- **DTDs and XML schemas**
  - Some element and attribute types map directly to topic types
  - Association types may be inferred from content models
- **Metadata schemas**
  - Mostly provide characteristics for an addressable subject

Document metadata

- **Properties stored with the file**
  - Word or PowerPoint properties, HTML Dublin Core, PDF-RDF
- **Properties stored externally**
  - RDF, MPEG21, DMS metadata
- **Access is file-format or system specific, but metadata may fall into common categories**
- **Typical metadata values include:**
  - Author, Subject, Keyword
- **Metadata values represent**
  - topics associated with the topic which represents the document, or
  - topics for which this document may be some form of typed occurrence
  - associations between individual values representing topics

Leveraging existing metadata

```
<item rdf:about="http://www.oreillynet.com/.../metadata.html">
  <title>Distributed Metadata</title>
  <link>http://www.oreillynet.com/.../metadata.html</link>
  <dc:description>This article addresses...</dc:description>
  <dc:subject>metadata, rdf, peer-to-peer</dc:subject>
  <dc:creator>Dan Brickley and Rael Dornfest</dc:creator>
  <dc:publisher>O'Reilly & Associates</dc:publisher>
  <dc:date>2000-10-29T00:34:00+00:00</dc:date>
  <dc:type>article</dc:type>
  <dc:language>en-us</dc:language>
  <dc:format>text/html</dc:format>
  <dc:rights>Copyright 2000, O'Reilly
  & Associates, Inc.</dc:rights>
  ...
</item>
```

RDF statements about an addressable subject:

- Topic name and identity
- Potential associations
- Potential occurrences

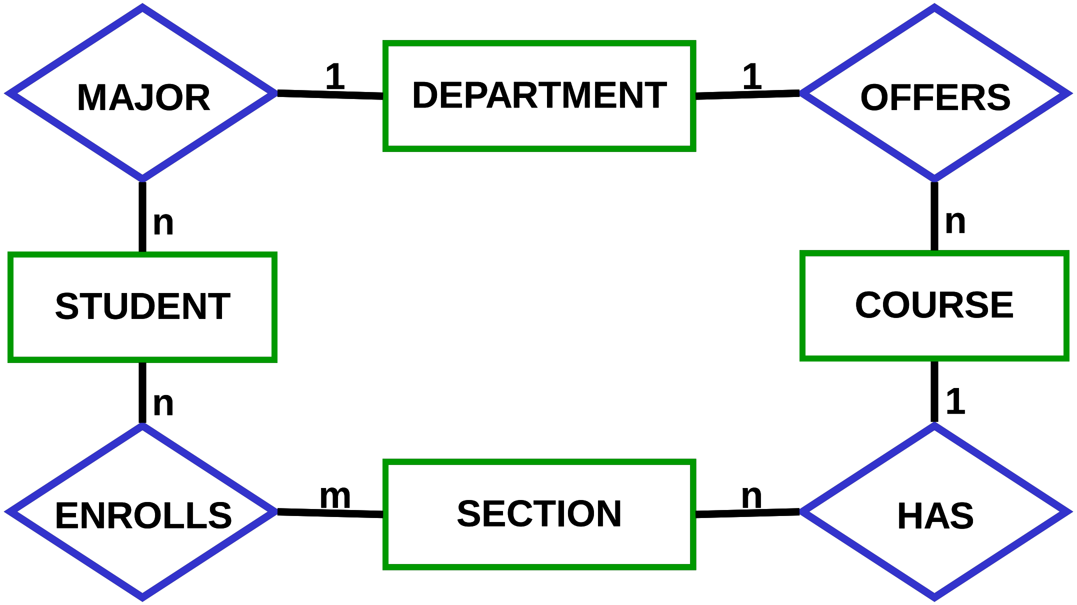
Dublin Core metadata in RDF format
(from R. Dornfest and D. Brickley, *The Power of Metadata*)

Information systems

- **Rich repositories of metadata**
- **Access is application or technology specific (RDBMS, LDAP etc)**
- **Topic types and association types are encoded in the schema of the system**
- **A single table often equates to a topic type**
- **Each row in the table is a topic of that type**
- **Columns map to**
  - IDs, names, occurrences or associations with other topics

ID	Name	Age	Father
1	Steve	48	5
2	Hedda	18	1
3	Lisa	16	1
4	Thea	10	1

Leveraging an existing database schema



Entity-relationship diagram for a university database
(from J. Sowa, *Knowledge Representation*)

▭ Candidate topic types

◇ Candidate association types

Column names in each table provide candidate occurrence types, e.g. age, description, address

Structured document content

- **Semantic markup can be a rich source of topics and associations**
  - The content of certain elements will be the names of topics
  - Associations may be inferred from relations between elements
- **Access is standardised (few file formats)**
  - Enables the same techniques to be applied regardless of document structure
- **May enable a very detailed deconstruction of the content**

  -
    ```
    <address>
      ...
      <city>Oslo</city>
      <country>Norway</country>
    </address>
    ```

Leveraging structured content

```
<?XML version="1.0"?>
<xmldoc>
<customer>
   <accountid>AE4-Robertson</accountid>
   <name>
      <first>Eric</first>
      <mi>H</mi>
      <last>Robertson</last>
   </name>
   <title>VP Sales</title>
   <contact>
      <wphone>123-555-1212</wphone>
      <hphone>123-555-5678</hphone>
      <email>salesvp@yoyo.com</email>
   </contact>
</customer>
</xmldoc>
```

Data in XML format
(from W.R. Stanek, *Structuring Data with XML*)

Topic types:

- customer
- account

Association types:

- customer-account

Occurrence types:

- (name)
- ID?
- title
- work-phone
- home-phone
- email

Unstructured document content

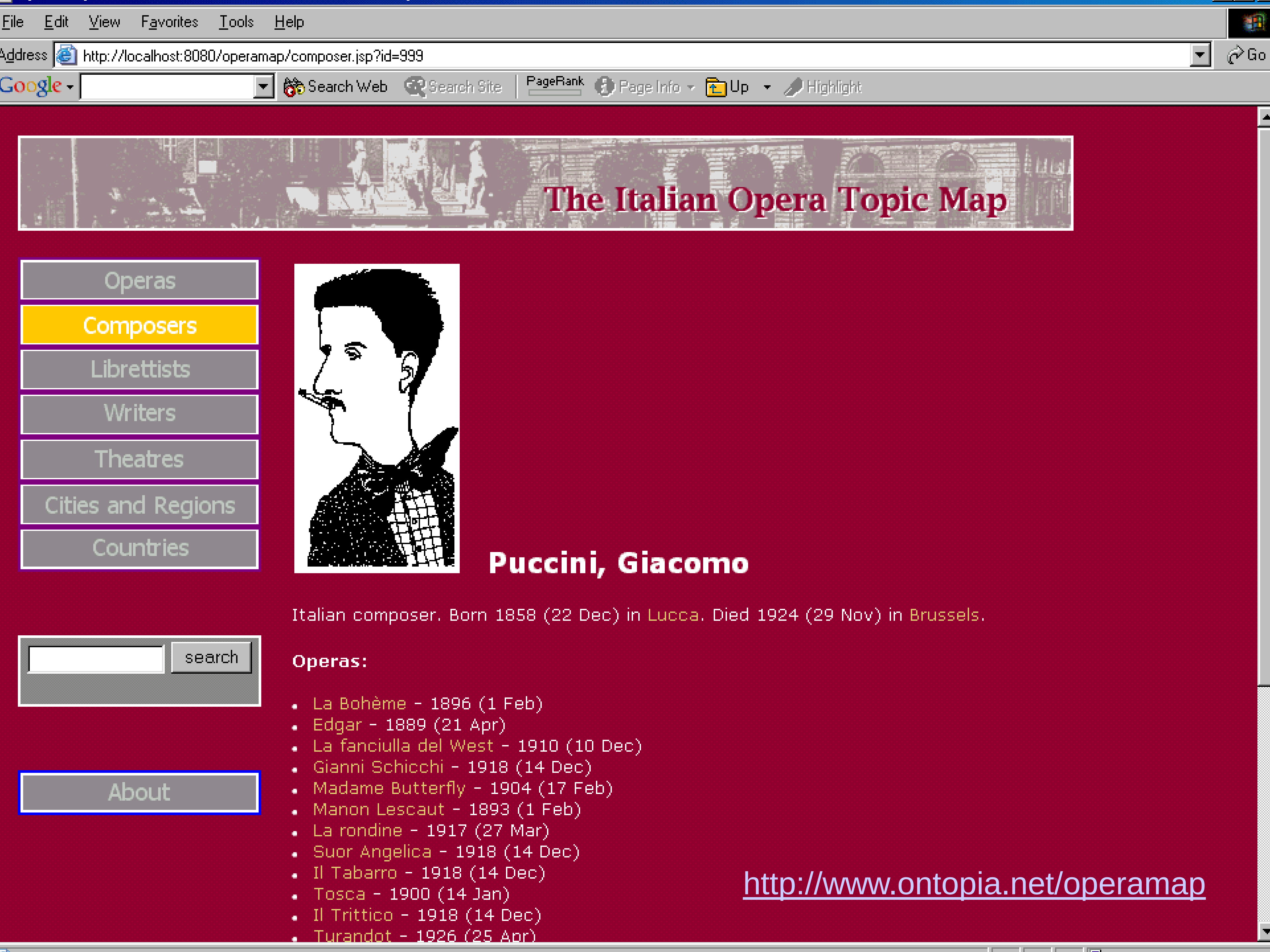
- **Extraction depends on analysis of textual content**
- **Natural Language Processing techniques may be applied**
  - Simple pattern matching
    - Identifying mentions of known topics in new data
  - Named Entity recognition
    - locating people, companies and places in the text
  - Concept extraction
    - identifying the ‘key words’ of the text
  - Taxonomic classification
    - analysis according to a human-defined taxonomy
  - Discourse analysis
    - understanding the meaning of the text
  - Bayesian inferencing
    - pattern-matching to compare new documents with those already classified
- **Most useful for attaching occurrences to pre-existing topics**

Manual topic map authoring

- **Unavoidable in some applications; unnecessary in others**
- **Most applications benefit from manual enrichment of automatically created topic maps**
- **The effort expended is normally justifiable thanks to increased reusability**
- **Generic topic map editors are not recommended**
  - The complexity of topic maps cannot be hidden in generic interfaces
- **Normally best to build a custom interface**
  - Same approach as in database applications

Topic map creation and maintenance

- **Topic map creation is not as difficult as you think**
- **There are many sources of topic map data**
- **Much of this may be leveraged very efficiently through automated processes**
- **Some of it can be mapped directly to topic maps**
- **Manual enrichment adds considerable value**
- **Most topic map creation will be a combination of autogeneration and manual enrichment**



Puccini, Giacomo

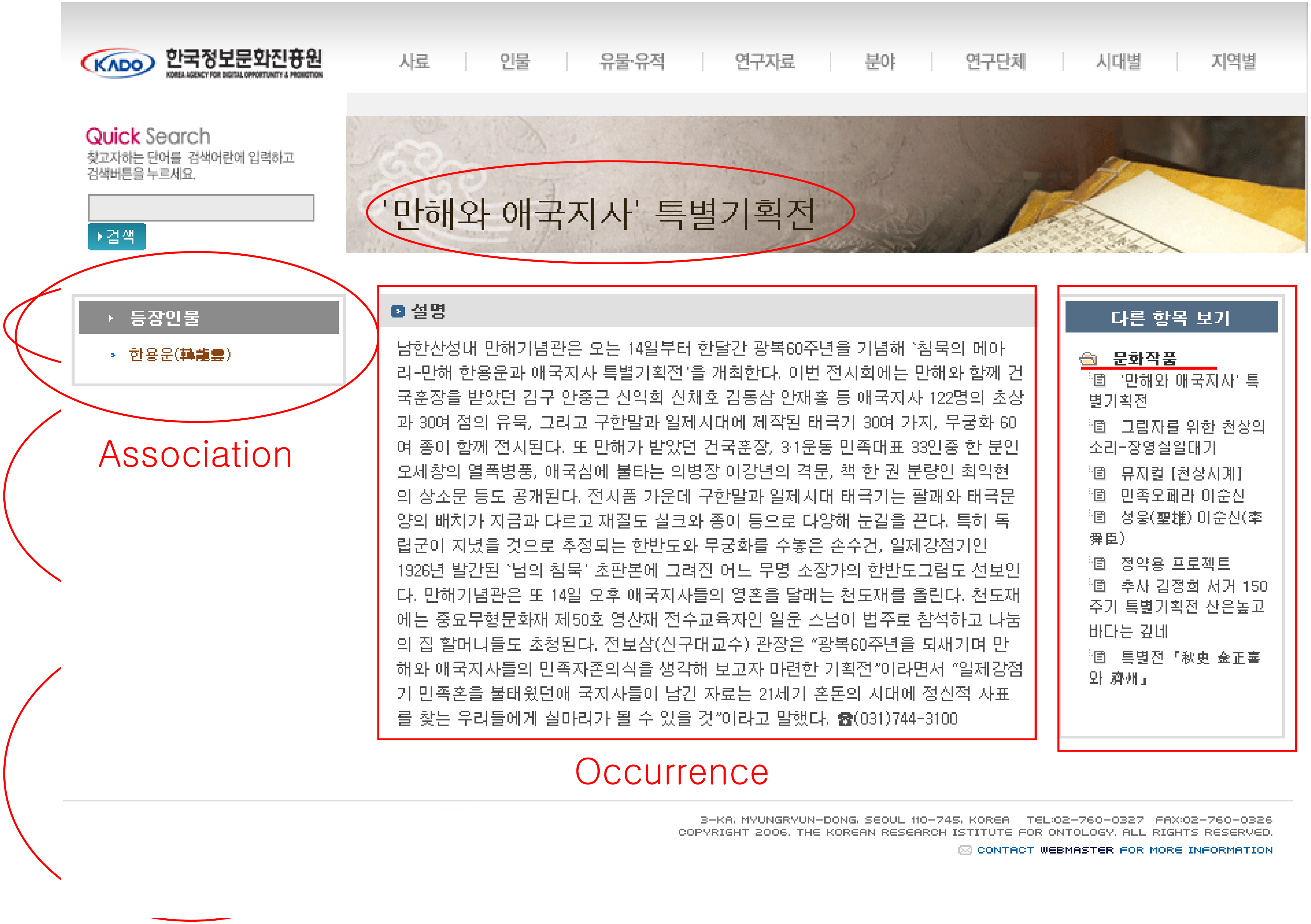
Italian composer. Born 1858 (22 Dec) in Lucca. Died 1924 (29 Nov) in Brussels.

Operas:

- La Bohème - 1896 (1 Feb)
- Edgar - 1889 (21 Apr)
- La fanciulla del West - 1910 (10 Dec)
- Gianni Schicchi - 1918 (14 Dec)
- Madame Butterfly - 1904 (17 Feb)
- Manon Lescaut - 1893 (1 Feb)
- La rondine - 1917 (27 Mar)
- Suor Angelica - 1918 (14 Dec)
- Il Tabarro - 1918 (14 Dec)
- Tosca - 1900 (14 Jan)
- Il Trittico - 1918 (14 Dec)
- Turandot - 1926 (25 Apr)

http://www.ontopia.net/operamap

Topic Map Applied to Korea History Portal


KADO 한국정보문화진흥원
KOREA AGENCY FOR DIGITAL OPPORTUNITY & PROMOTION
사료 | 인물 | 유물·유적 | 연구자료 | 분야 | 연구단체 | 시대별 | 지역별
Quick Search
찾고자하는 단어를 검색어란에 입력하고 검색버튼을 누르세요.
검색
등장인물
다른 항목 보기

'만해와 애국지사' 특별기획전

등장인물

- 한용운(韓龍雲)

설명

남한산성내 만해기념관은 오는 14일부터 한달간 광복60주년을 기념해 `침묵의 메아리-만해 한용운과 애국지사 특별기획전'을 개최한다. 이번 전시회에는 만해와 함께 건국훈장을 받았던 김구 안중근 신익희 신채호 김동삼 안재홍 등 애국지사 122명의 초상과 30여 점의 유묵, 그리고 구한말과 일제시대에 제작된 태극기 30여 가지, 무궁화 60여 종이 함께 전시된다. 또 만해가 받았던 건국훈장, 3·1운동 민족대표 33인중 한 분인 오세창의 열폭병풍, 애국심에 불타는 의병장 이강년의 격문, 책 한 권 분량인 최익현의 상소문 등도 공개된다. 전시품 가운데 구한말과 일제시대 태극기는 팔괘와 태극문양의 배치가 지금과 다르고 재질도 실크와 종이 등으로 다양해 눈길을 끈다. 특히 독립군이 지녔을 것으로 추정되는 한반도와 무궁화를 수놓은 손수건, 일제강점기인 1926년 발간된 `님의 침묵' 초판본에 그려진 어느 무명 소장가의 한반도그림도 선보인다. 만해기념관은 또 14일 오후 애국지사들의 영혼을 달래는 천도재를 올린다. 천도재에는 중요무형문화재 제50호 영산재 전수교육자인 일운 스님이 법주로 참석하고 나눔의 집 할머니들도 초청된다. 전보삼(신구대교수) 관장은 "광복60주년을 되새기며 만해와 애국지사들의 민족자존의식을 생각해 보고자 마련한 기획전"이라면서 "일제강점기 민족혼을 불태웠던애 국지사들이 남긴 자료는 21세기 혼돈의 시대에 정신적 사표를 찾는 우리들에게 실마리가 될 수 있을 것"이라고 말했다. ☎(031)744-3100

다른 항목 보기

문화작품

- '만해와 애국지사' 특별기획전
- 그림자를 위한 천상의 소리-장영실일대기
- 뮤지컬 [천상시계]
- 민족오페라 이순신
- 성웅(聖雄) 이순신(李舜臣)
- 정약용 프로젝트
- 추사 김정희 서거 150주기 특별기획전 산은높고 바다는 깊네
- 특별전 『秋史 金正喜와 濟州』

3-KA, MYUNGRYUN-DONG, SEOUL 110-745, KOREA TEL:02-760-0327 FAX:02-760-0326
COPYRIGHT 2006. THE KOREAN RESEARCH ISTITUTE FOR ONTOLOGY. ALL RIGHTS RESERVED.
CONTACT WEBMASTER FOR MORE INFORMATION

Association

Occurrence

Applications of Topic Maps

Taxonomy Management

Semantic Portals

eLearning

Business Process Modelling

Product Configuration

Information Integration

Metadata Management

Business Rules Management

IT Asset Management

Asset Management (Manufacturing)

Taxonomy management

- **Addresses the problem of managing unstructured content**
  - Organization by subject is seen as the solution – that's how users search
  - More and more companies are looking into and developing taxonomies
- **A taxonomy is a simple form of topic map**
  - Topic Maps provides subject-based organization de-luxe
- **Using Topic Maps offers many benefits:**
  - Standards-based means vendor independence and data longevity
  - Associative model allows for evolution beyond simple hierarchies
  - The taxonomy can also be used as a thesaurus, a glossary or an index
  - Identity model permits merging and reuse
- **The Dutch Tax and Customs Administration (Belastingdienst) uses the OKS as the basis of a taxonomy management system**
  - http://www.idealliance.org/papers/dx_xmle04/papers/04-01-03/04-01-03.html
- **This capability can also be added to Content Management Systems**

XML Europe2004

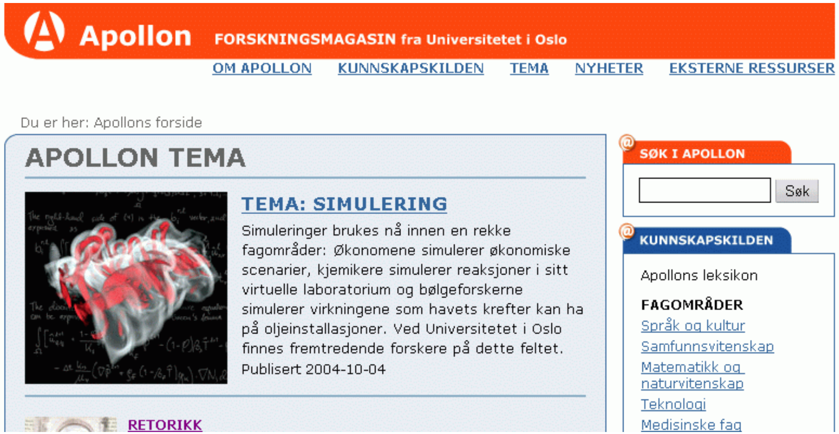
18 -21 April 2004
The Amsterdam RAI Centre, Amsterdam, The Netherlands
www.xmleurope.com

Using Topic Maps and XML to implement a closed-loop search service for the Dutch Tax and Customs Administration website

»Matthijs Breebaart, »Dutch Tax and Customs Administration

Semantic portals

- **Topic Maps as Information Architecture for web delivery applications**
  - Web sites, portals, corporate intranets, etc.
- **Site structure is defined as a topic map**
  - Each page represents a topic (subject-centric)
  - User-friendly navigation paths defined by associations
  - Topics used to classify content
  - High potential for portal connectivity using TMRAP
  - Permits evolution towards Knowledge Management solutions
- **The OKS has been used to create portals, e.g.**
  - Kulturnett.no (Norwegian public sector portal to cultural information): www.kulturnett.no
  - Apollon (University of Oslo research magazine): www.apollon.uio.no



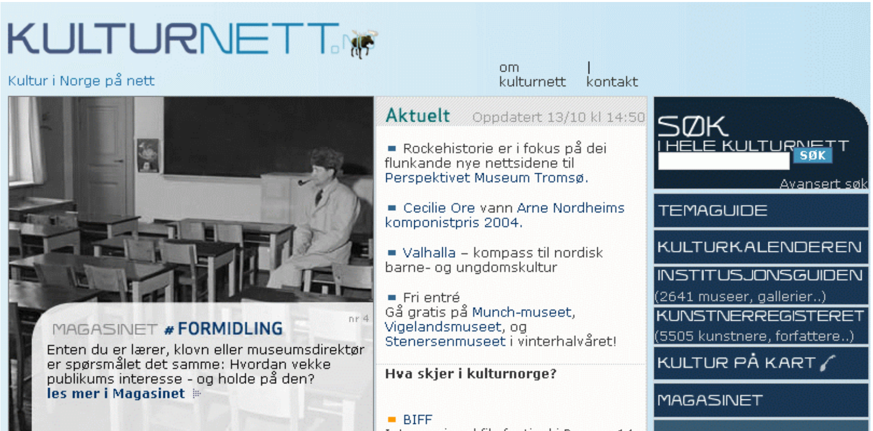
Du er her: Apollons forside

APOLLON TEMA

TEMA: SIMULERING

Simuleringer brukes nå innen en rekke fagområder: Økonomene simulerer økonomiske scenarier, kjemikere simulerer reaksjoner i sitt virtuelle laboratorium og bølgeforskerne simulerer virkningene som havets krefter kan ha på oljeinstallasjoner. Ved Universitetet i Oslo finnes fremtredende forskere på dette feltet. Publisert 2004-10-04

RETORIKK



MAGASINET # FORMIDLING nr 4

Enten du er lærer, klovn eller museumsdirektør er spørsmålet det samme: Hvordan vekke publikums interesse - og holde på den?
les mer i Magasinet

Aktuelt Oppdatert 13/10 kl 14:50

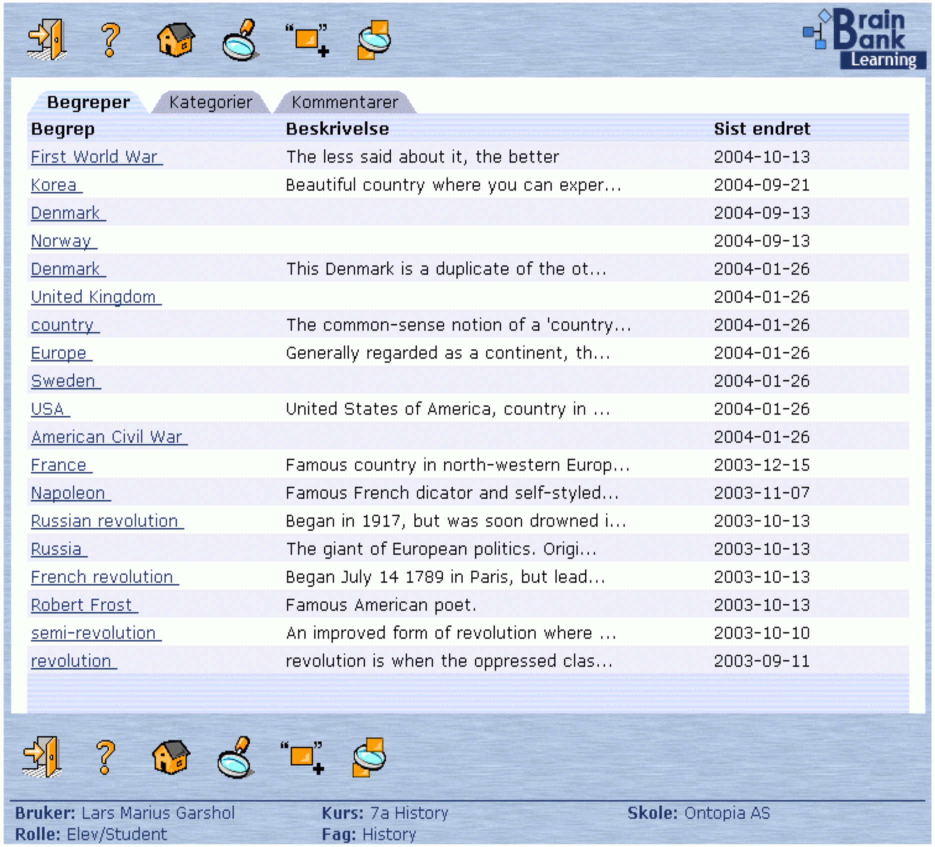
- Rockehistorie er i fokus på dei flunkande nye nettsidene til Perspektivet Museum Tromsø.
- Cecilie Ore vann Arne Nordheims komponistpris 2004.
- Valhalla – kompass til nordisk barne- og ungdomskultur
- Fri entré
  Gå gratis på Munch-museet, Vigelandsmuseet, og Stenersenmuseet i vinterhalvåret!

Hva skjer i kulturnorge?

- BIFF

E-learning

- **Topic maps are associative knowledge structures**
  - They reflect how people acquire and retain knowledge
- **BrainBank is used by students to describe what they have learned**
  - Initial users are 11-13 year olds who have no idea what a topic map is…
- **They capture the key concepts, name them, describe them, and associate them with others**
- **This helps them**
  - Capture the essence,
  - Describe what they have learned,
  - Keep track of their knowledge, and
  - Lets the teacher help them
- **BrainBank was built using the OKS**
  - An application of the Web Editor Framework
  - Demonstrates user-friendliness of TM editing

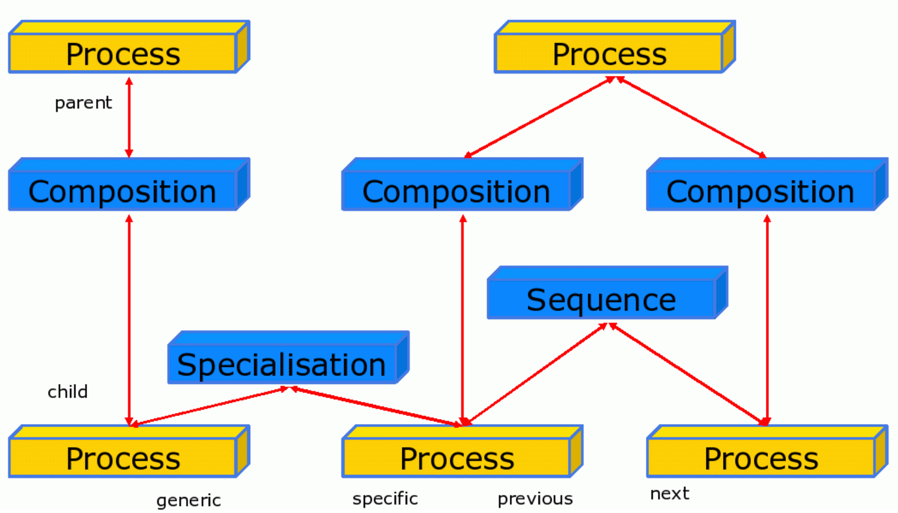


Begrep	Beskrivelse	Sist endret
First World War	The less said about it, the better	2004-10-13
Korea	Beautiful country where you can exper...	2004-09-21
Denmark		2004-09-13
Norway		2004-09-13
Denmark	This Denmark is a duplicate of the ot...	2004-01-26
United Kingdom		2004-01-26
country	The common-sense notion of a 'country...	2004-01-26
Europe	Generally regarded as a continent, th...	2004-01-26
Sweden		2004-01-26
USA	United States of America, country in ...	2004-01-26
American Civil War		2004-01-26
France	Famous country in north-western Europ...	2003-12-15
Napoleon	Famous French dicator and self-styled...	2003-11-07
Russian revolution	Began in 1917, but was soon drowned i...	2003-10-13
Russia	The giant of European politics. Origi...	2003-10-13
French revolution	Began July 14 1789 in Paris, but lead...	2003-10-13
Robert Frost	Famous American poet.	2003-10-13
semi-revolution	An improved form of revolution where ...	2003-10-10
revolution	revolution is when the oppressed clas...	2003-09-11

Bruker: Lars Marius Garshol **Kurs:** 7a History **Skole:** Ontopia AS
Rolle: Elev/Student **Fag:** History

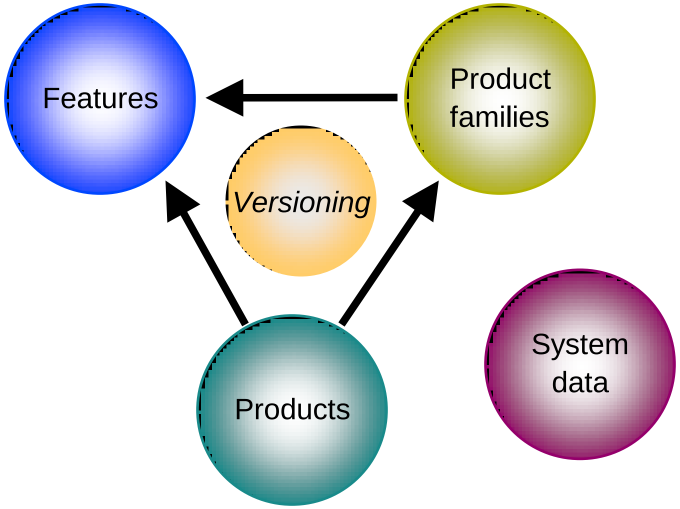
Business process modelling

- **A *multinational petrochemical company* uses the OKS for managing business process models**
  - The flexibility of the Topic Maps model allows arbitrary relationships to be captured easily
- **Processes are modelled in terms of**
  - The steps involved, their preconditions, their successors, etc
- **Processes can be related through**
  - Composition (one process is part of another),
  - Sequencing (one process is followed by another),
  - Specialization (one process is a special case of a more general process)



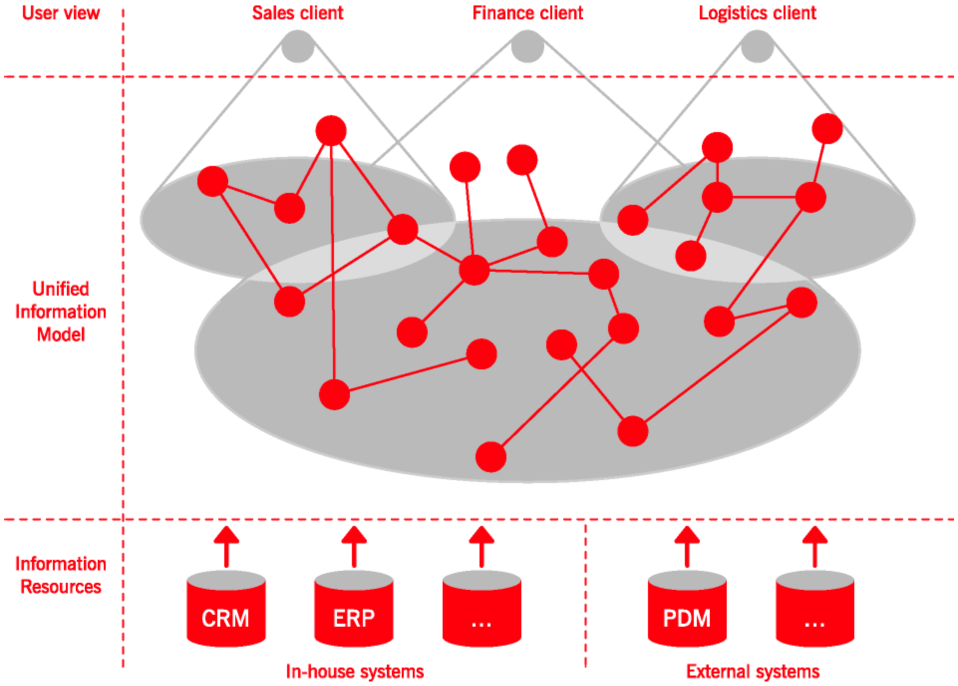
Product configuration

- **A *Scandinavian telecom company* uses the OKS to manage product configuration**
  - Products belong to families
  - Features belong to either products or product families
  - Features are grouped in feature sets
  - There are dependencies between features
  - Different features apply in different regions
  - etc.
- **The network of dependencies is already quite complex**
  - Now throw versioning into the mix!
  - Managing all this data is not easy
- **The system models dependencies in a topic map**
  - Product configuration engineers use this to configure products using a very user-friendly interface
- **The system is driven by inference rules**
  - These work on the topic map
  - Easily capture complex logic
  - Also integrates with product documentation



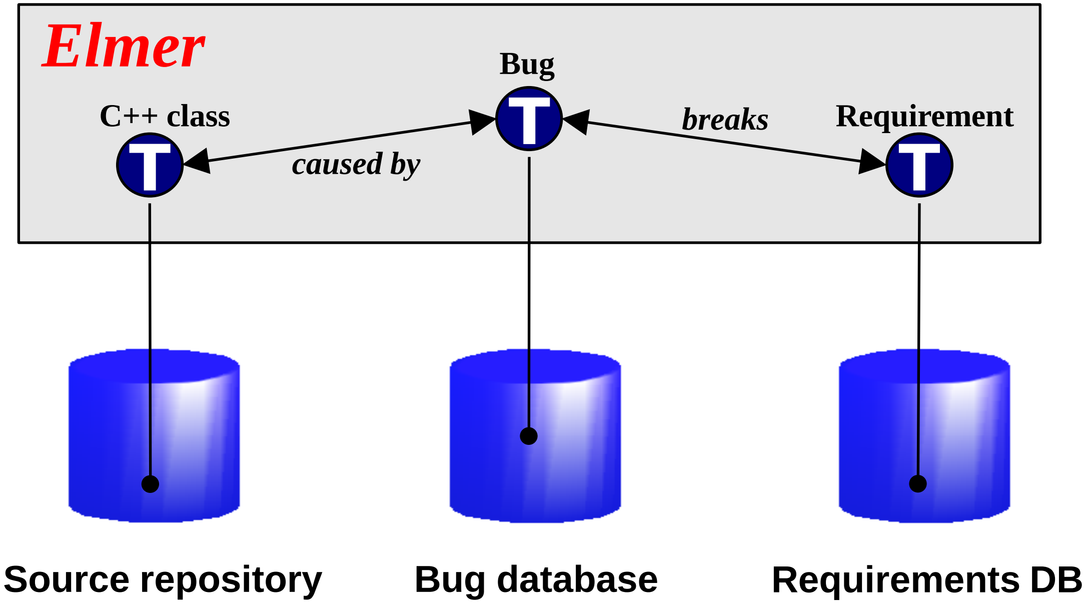
Enterprise information integration: Theory

- **Topic Maps are designed for ease of merging**
  - Generate topic maps from structured data
  - (Or create topic map views of that data)
  - Classify content according to a taxonomy topic map
  - Merge the topic maps to provide a unified view of the whole
- **Topic maps are easy to filter**
  - Create personalized views of the unified information model
  - Typing topics and scope provide built-in criteria for filtering
- **Advantages:**
  - Consolidated access to all related information
  - Does not require migration of existing content
  - Standards-based



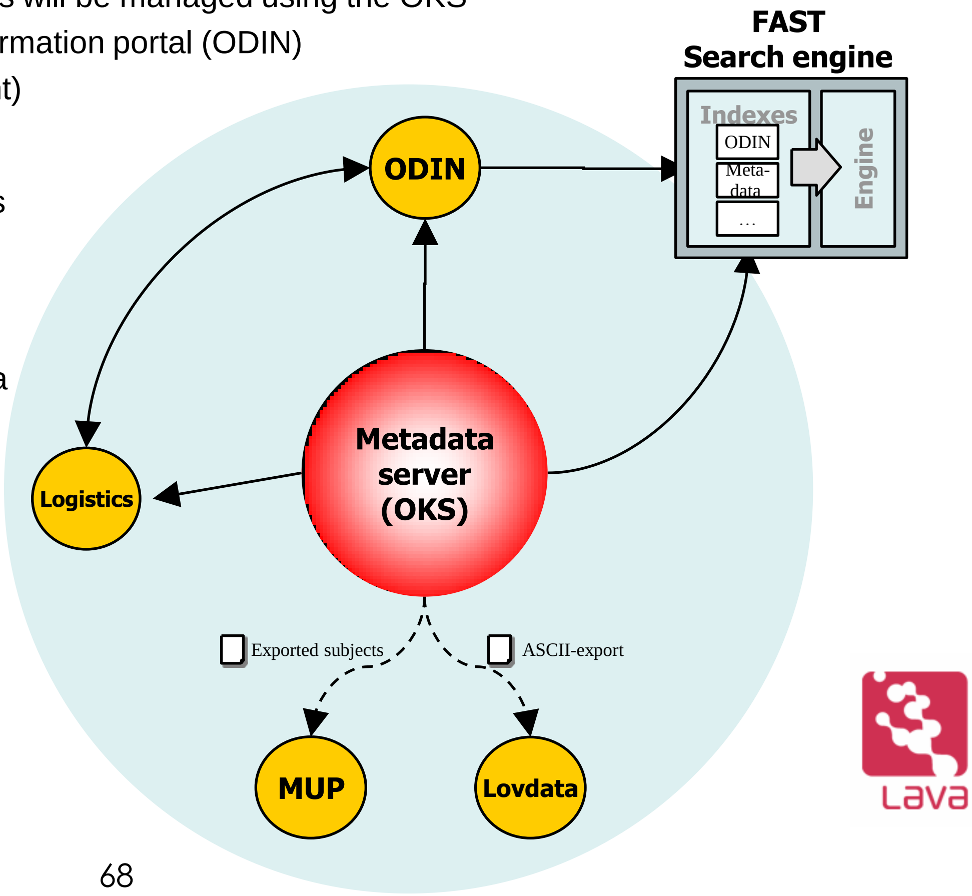
Enterprise information integration: Practice

- ***Starbase*** **is using the OKS in an internal project called Elmer**
  - This project is building an integration server for software information
- **Multiple disparate applications hold related data**
  - Building a unified topic map layer on top makes it possible to search across repositories
  - Provides data integration without changing the underlying applications
- **Access to information provided through a portal**
  - Straightforward navigation interface
  - Querying, both full-text and structured
- **Topic maps drives integration with MS Office Smarttags**
  - Terms known from Elmer are highlighted
  - (Names of topics used as a vocabulary)
  - Appear as links back into the portal



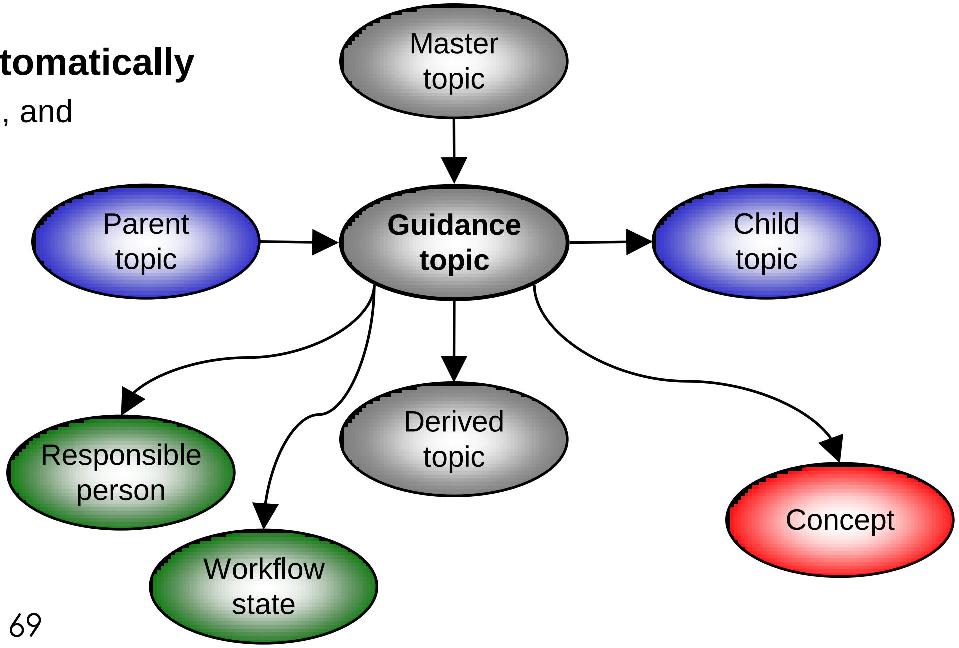
Metadata management

- **On behalf of the *Norwegian Government Administration Services* Lava Group is building a metadata server**
  - Metadata for government publications will be managed using the OKS
  - Will be used in the central public information portal (ODIN)
  - (System currently under development)
- **The system provides**
  - Authoring system used by the editors
  - Vocabulary Editor for adjusting the metadata vocabulary used
  - Metadata Export to various systems
  - Web services based on the metadata
  - Unique identifiers for documents
  - Unparallelled future flexibility



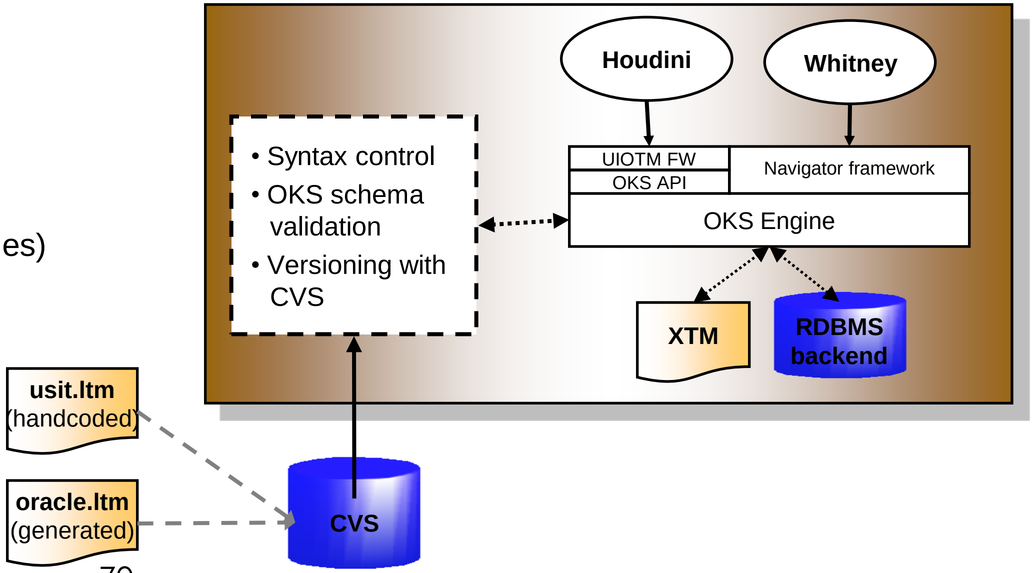
Business rules management

- **The *US Department of Energy* has used the OKS to manage guidance rules for security classification**
  - Information about the production of nuclear weapons is subject to thousands of rules
  - Rules are published in 100s of documents
  - Most documents are derived from more general documents
- **Guidance topics form a complex web of relationships that is captured in a topic map**
  - Concepts are connected to if-then-else rules
  - This constitutes a knowledge base (KB)
- **KB used with an inference engine to automatically**
  - classify information (documents, emails, ...), and
  - redact information (PDF, email, ...)
- **Benefits:**
  - Model expressive enough to capture the complexity of the rules
  - Status as ISO standard ensures stability and longevity



IT asset management

- **The University of Oslo is using the OKS to manage IT assets**
- **Servers, clusters, databases, etc are described in a TM**
- **This is used to answer questions like**
  - Service X is down, who do I call?
  - If I take Y down, what else goes?
  - If operating system Z is upgraded, what apps are affected?
- **System driven by composite topic map**
  - Partly autogenerated
  - Partly handcoded
- **Two applications provide access to the knowledge base**
  - Whitney: online
  - Houson: offline (for use in emergencies)



Houdini

Whitney

UIOTM FW

OKS API

Navigator framework

OKS Engine

- Syntax control
- OKS schema validation
- Versioning with CVS

XTM

RDBMS backend

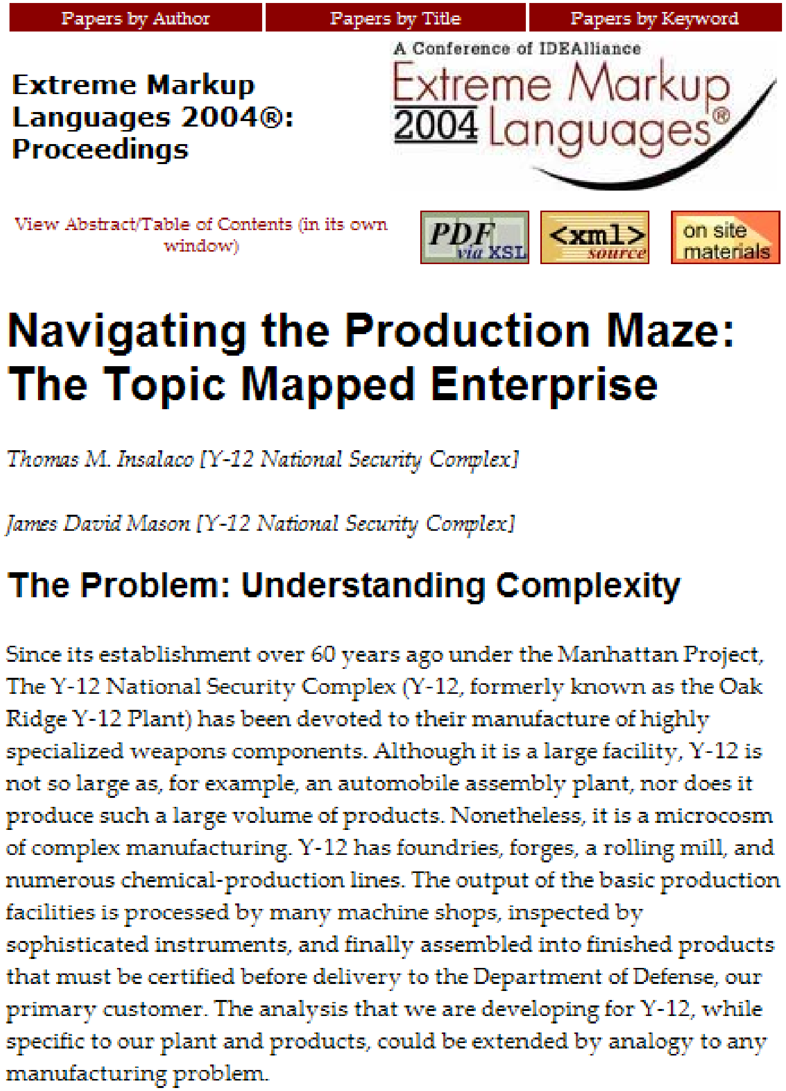
usit.ltm (handcoded)

oracle.ltm (generated)

CVS

Asset management: Manufacturing

- **The Y-12 plant at DoE is using the OKS to map its plant**
- **The purpose is to get an overview of**
  - equipment,
  - processes,
  - materials required,
  - parts already built,
  - etc.



Extreme Markup Languages 2004®: Proceedings

View Abstract/Table of Contents (in its own window)

Navigating the Production Maze: The Topic Mapped Enterprise

Thomas M. Insalaco [Y-12 National Security Complex]

James David Mason [Y-12 National Security Complex]

The Problem: Understanding Complexity

Since its establishment over 60 years ago under the Manhattan Project, The Y-12 National Security Complex (Y-12, formerly known as the Oak Ridge Y-12 Plant) has been devoted to their manufacture of highly specialized weapons components. Although it is a large facility, Y-12 is not so large as, for example, an automobile assembly plant, nor does it produce such a large volume of products. Nonetheless, it is a microcosm of complex manufacturing. Y-12 has foundries, forges, a rolling mill, and numerous chemical-production lines. The output of the basic production facilities is processed by many machine shops, inspected by sophisticated instruments, and finally assembled into finished products that must be certified before delivery to the Department of Defense, our primary customer. The analysis that we are developing for Y-12, while specific to our plant and products, could be extended by analogy to any manufacturing problem.

Tomorrow is here … today

Spontaneous Knowledge Aggregation

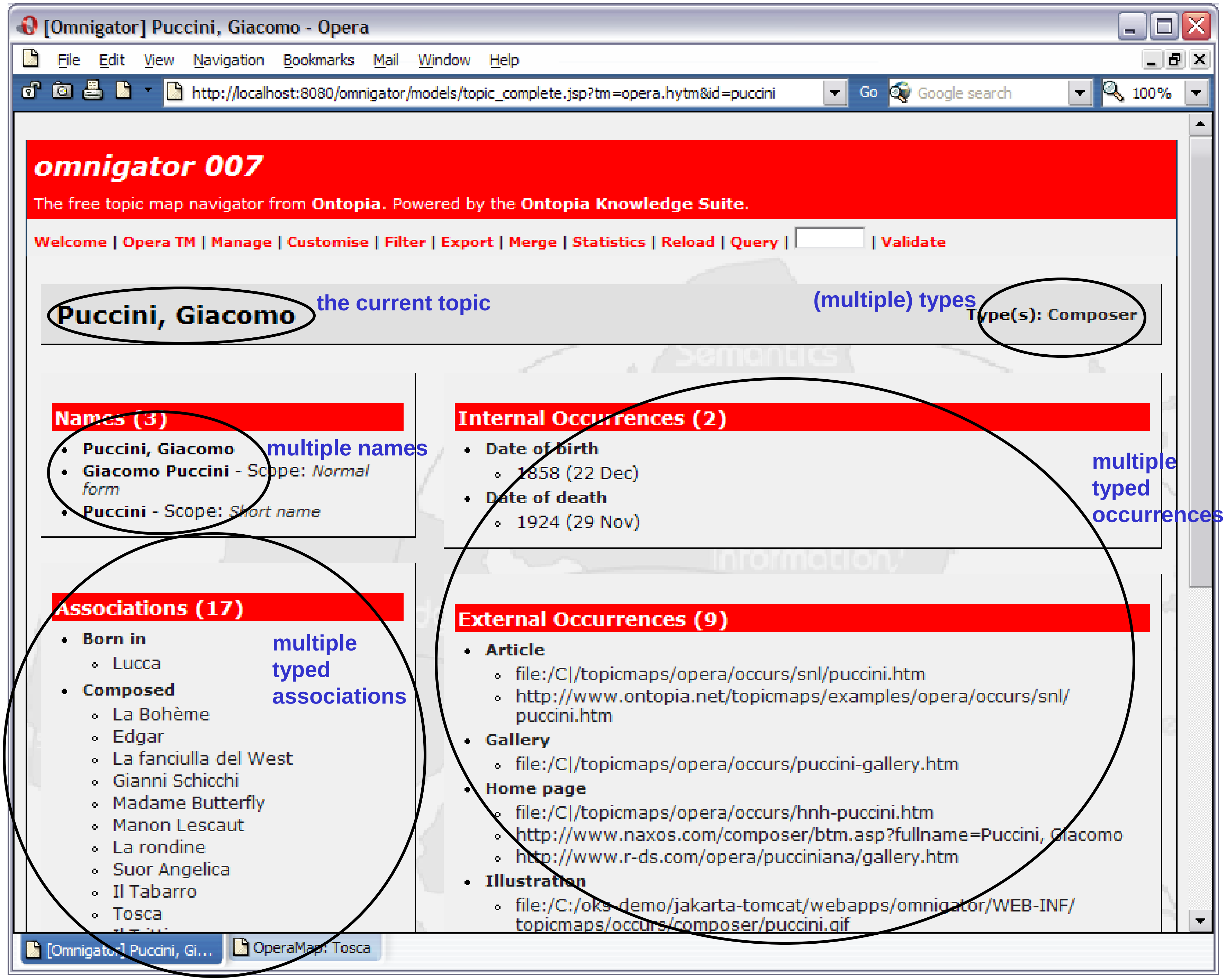
An Application of Seamless Knowledge

Automatic Portal Integration

Topic Maps Remote Access Protocol

Semantic portals

- **One of many applications of Topic Maps**
  - Topic Maps is an ideal model for portals and other forms of web-based information delivery
- **The basic concept is to have the topic map *drive* the portal**
  - *Not* just a navigational layer on top of something else
  - The *very structure* of the portal is a topic map
  - All content is organized around *topics* (“subject-centric organization”)
- **Each page represents a topic (we call this a “Topic Page”)**
  - Topics act as points of *collocation*
  - They provide a “one-stop shop” for everything that is known about a particular subject
- **Navigating the portal == Navigating the topic map**
  - Associations provide very *intuitive navigation* (“As we may think”)


[Omnigator] Puccini, Giacomo - Opera
File Edit View Navigation Bookmarks Mail Window Help
http://localhost:8080/omnigator/models/topic_complete.jsp?tm=opera.hytm&id=puccini
Go
Google search
100%
[Omnigator] Puccini, Gi...
OperaMap: Tosca

omnigator 007

The free topic map navigator from **Ontopia**. Powered by the **Ontopia Knowledge Suite**.

Welcome | Opera TM | Manage | Customise | Filter | Export | Merge | Statistics | Reload | Query | | Validate

Puccini, Giacomo

Type(s): Composer

the current topic

(multiple) types

Names (3)

- **Puccini, Giacomo**
- **Giacomo Puccini** - Scope: *Normal form*
- **Puccini** - Scope: *Short name*

multiple names

Internal Occurrences (2)

- **Date of birth**
  - 1858 (22 Dec)
- **Date of death**
  - 1924 (29 Nov)

multiple typed occurrences

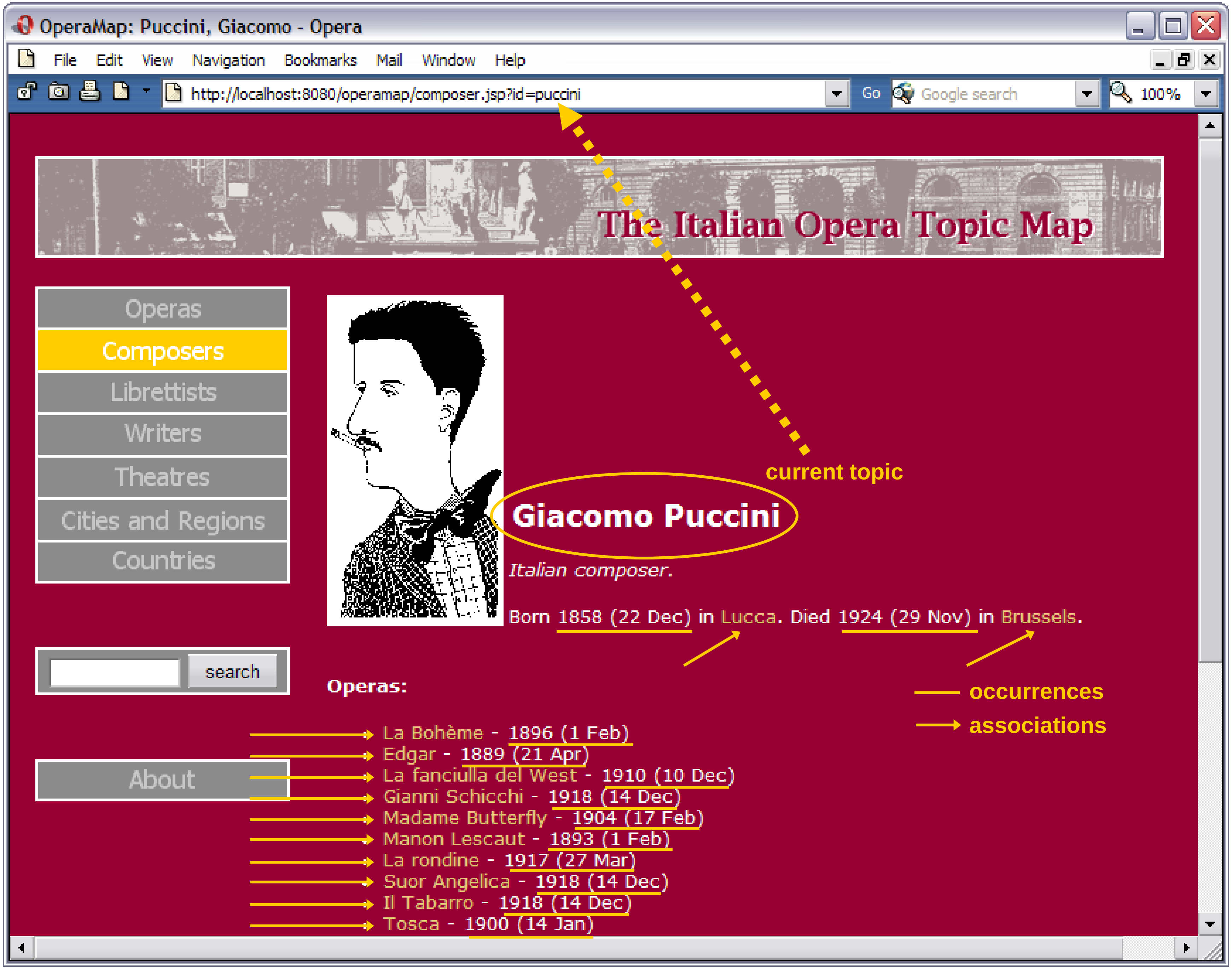
Associations (17)

- **Born in**
  - Lucca
- **Composed**
  - La Bohème
  - Edgar
  - La fanciulla del West
  - Gianni Schicchi
  - Madame Butterfly
  - Manon Lescaut
  - La rondine
  - Suor Angelica
  - Il Tabarro
  - Tosca

multiple typed associations

External Occurrences (9)

- **Article**
  - file:/C|/topicmaps/opera/occurs/snl/puccini.htm
  - http://www.ontopia.net/topicmaps/examples/opera/occurs/snl/puccini.htm
- **Gallery**
  - file:/C|/topicmaps/opera/occurs/puccini-gallery.htm
- **Home page**
  - file:/C|/topicmaps/opera/occurs/hnh-puccini.htm
  - http://www.naxos.com/composer/btm.asp?fullname=Puccini, Giacomo
  - http://www.r-ds.com/opera/pucciniana/gallery.htm
- **Illustration**
  - file:/C:/oks-demo/jakarta-tomcat/webapps/omnigator/WEB-INF/topicmaps/occurs/composer/puccini.gif



http://localhost:8080/operamap/composer.jsp?id=puccini

The Italian Opera Topic Map

current topic

Giacomo Puccini

Italian composer.

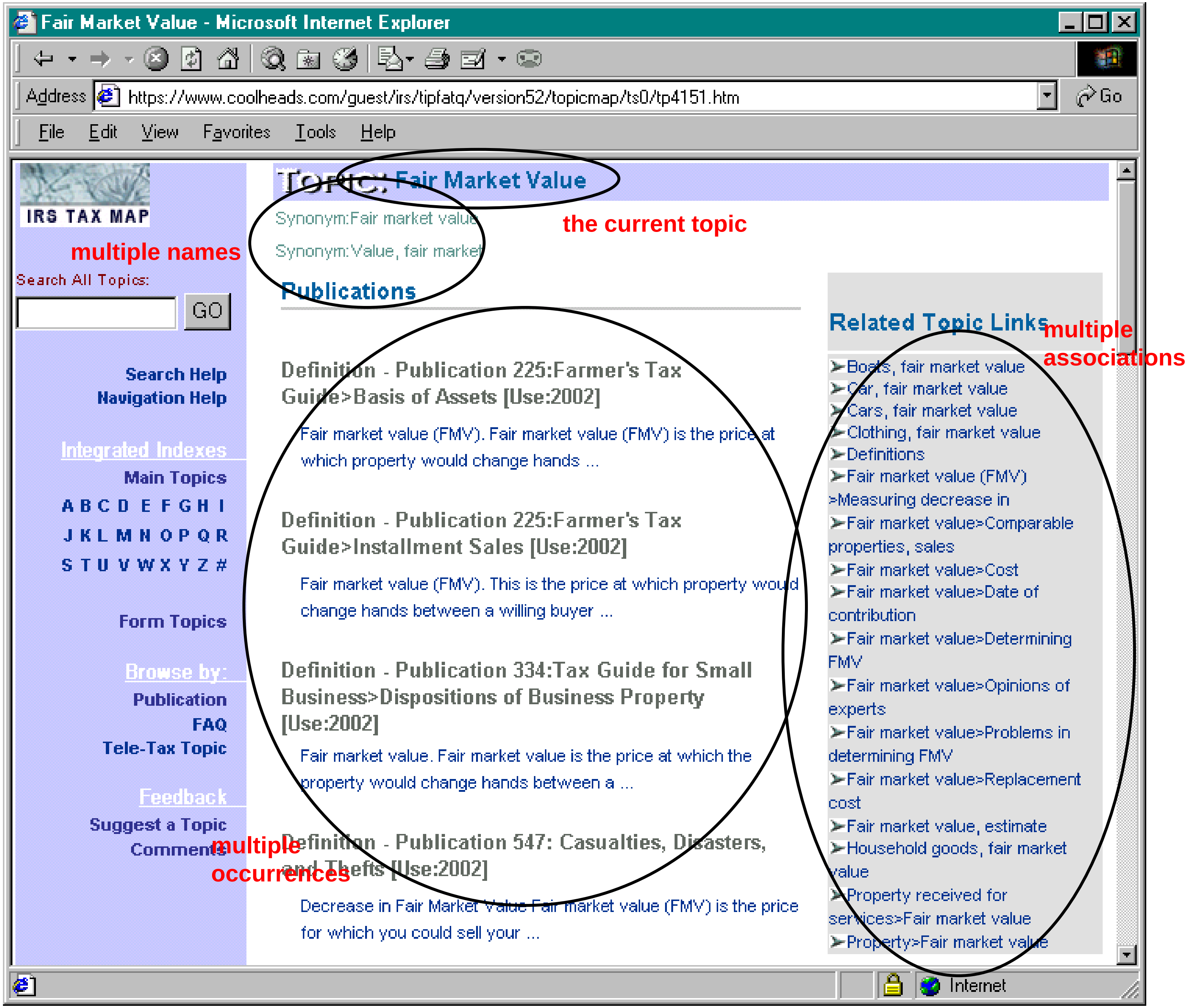
Born 1858 (22 Dec) in Lucca. Died 1924 (29 Nov) in Brussels.

Operas:

- La Bohème - 1896 (1 Feb)
- Edgar - 1889 (21 Apr)
- La fanciulla del West - 1910 (10 Dec)
- Gianni Schicchi - 1918 (14 Dec)
- Madame Butterfly - 1904 (17 Feb)
- Manon Lescaut - 1893 (1 Feb)
- La rondine - 1917 (27 Mar)
- Suor Angelica - 1918 (14 Dec)
- Il Tabarro - 1918 (14 Dec)
- Tosca - 1900 (14 Jan)

occurrences

associations



Fair Market Value - Microsoft Internet Explorer
Address
https://www.coolheads.com/guest/irs/tipfatq/version52/topicmap/ts0/tp4151.htm
Go
File
Edit
View
Favorites
Tools
Help
Internet

IRS TAX MAP

Search All Topics:

GO

Search Help
Navigation Help

Integrated Indexes
Main Topics
A B C D E F G H I
J K L M N O P Q R
S T U V W X Y Z #

Form Topics

Browse by:
Publication
FAQ
Tele-Tax Topic

Feedback
Suggest a Topic
Comments

TOPIC: Fair Market Value

Synonym:Fair market value

Synonym:Value, fair market

Publications

Definition - Publication 225:Farmer's Tax Guide>Basis of Assets [Use:2002]

Fair market value (FMV). Fair market value (FMV) is the price at which property would change hands ...

Definition - Publication 225:Farmer's Tax Guide>Installment Sales [Use:2002]

Fair market value (FMV). This is the price at which property would change hands between a willing buyer ...

Definition - Publication 334:Tax Guide for Small Business>Dispositions of Business Property [Use:2002]

Fair market value. Fair market value is the price at which the property would change hands between a ...

Definition - Publication 547: Casualties, Disasters, and Thefts [Use:2002]

Decrease in Fair Market Value Fair market value (FMV) is the price for which you could sell your ...

Related Topic Links

- Boats, fair market value
- Car, fair market value
- Cars, fair market value
- Clothing, fair market value
- Definitions
- Fair market value (FMV) >Measuring decrease in
- Fair market value>Comparable properties, sales
- Fair market value>Cost
- Fair market value>Date of contribution
- Fair market value>Determining FMV
- Fair market value>Opinions of experts
- Fair market value>Problems in determining FMV
- Fair market value>Replacement cost
- Fair market value, estimate
- Household goods, fair market value
- Property received for services>Fair market value
- Property>Fair market value

the current topic

multiple names

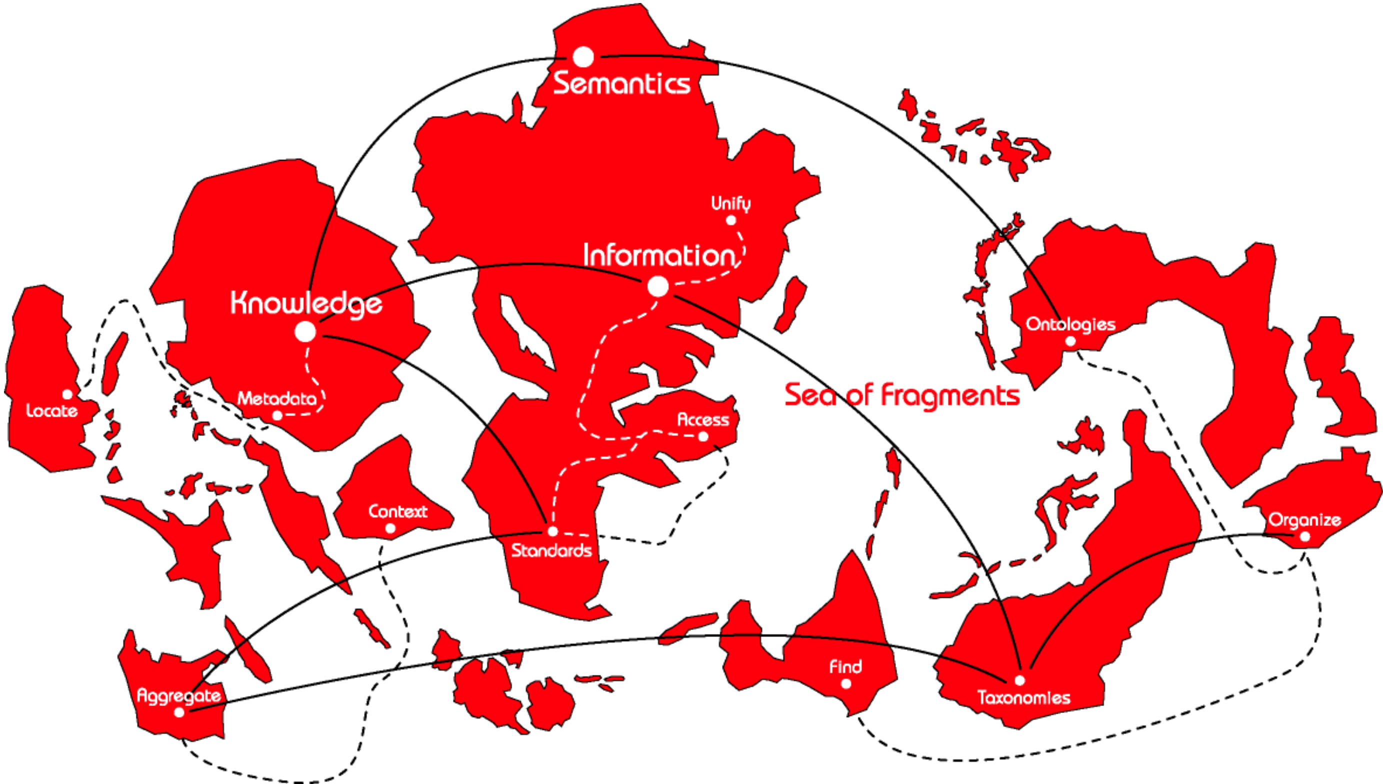
multiple associations

multiple occurrences

Topic Maps Ontology Design

Modelling your domain with Topic Maps

GOALS:
Learn how to create your own ontologies



Topic Maps terminology

- **Ontology**
  - topic types, association types, occurrence types, etc.
  - sometimes, some topic instances also belong here
- **Constraints**
  - rules governing classes of objects
- **Schema**
  - the combination of an ontology and constraints
- **Schema language**
  - a language for writing schemas
  - examples for topic maps: OSL and TMCL

Why the ontology is important

- **An ontology in Topic Maps corresponds to**
  - the set of element types and attributes in XML
  - the set of tables and columns in an RDBMS
- **That is, the ontology determines what you can *say***
- **You can't express the fact that company X owns company Y, unless you have an ownership association type**

What a Topic Maps ontology consists of

- **Topic types**
  - these are like entities or classes in other modelling formalisms
- **Association types**
  - these are types of relationships between topics
- **Occurrence types**
  - these are either
    - connections to external resources
    - properties
- **Sometimes also important instance topics**
  - these are like predefined constants

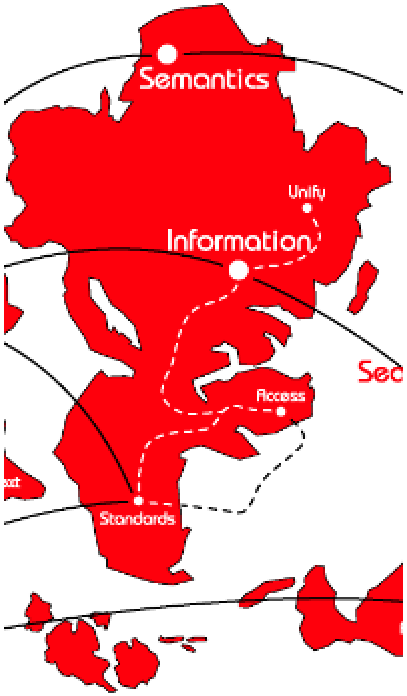
2 Fundamental rules of modelling

1. **There are many incorrect ways to model a domain, but no single correct way**
   - There are often viable alternatives
   - The best solution depends on the application
   - The result may have an element of subjectivity
   - Modelling is an aesthetic activity
2. **Ontology development is an iterative process**
   - As your ontology develops you will want to revisit earlier decisions and modify some of them
   - Fortunately, the topic map model is sufficiently flexible that this does not usually have a major impact on applications
   - Prototyping is a Good Thing

Ontology modeling vs data modeling

- **Data modeling is more pragmatic**
  - just do what the application needs
  - doesn’t necessarily assume any connection between model terms and domain terms
- **Ontology modeling is more semantic**
  - ensure that model accurately reflects meaning of terms in the domain
  - may need to twist this slightly to meet application needs

Ontology design in projects



Steps
Stakeholders
Consensus

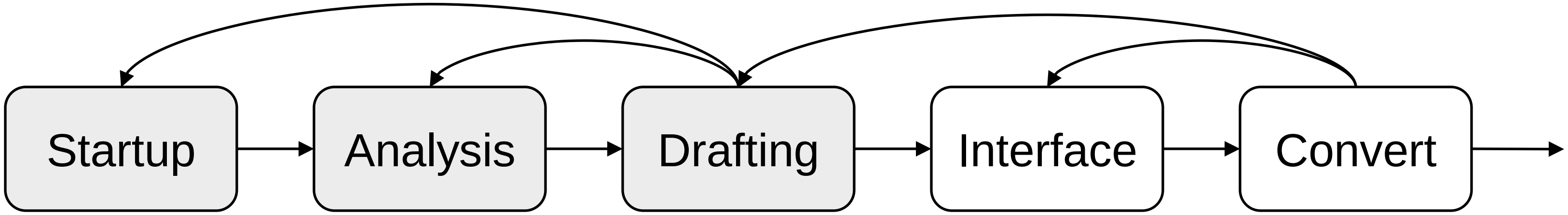
Topic Maps ontology design

- **A form of knowledge engineering or data modelling**
  - nearly always starts from a set of pre-existing data
- **Necessary skill set:**
  - domain expertise
  - understanding of the Topic Maps model
  - communication skills
  - modelling skills
- **Always involves a large subset of the project participants**

Topic Maps application life cycle

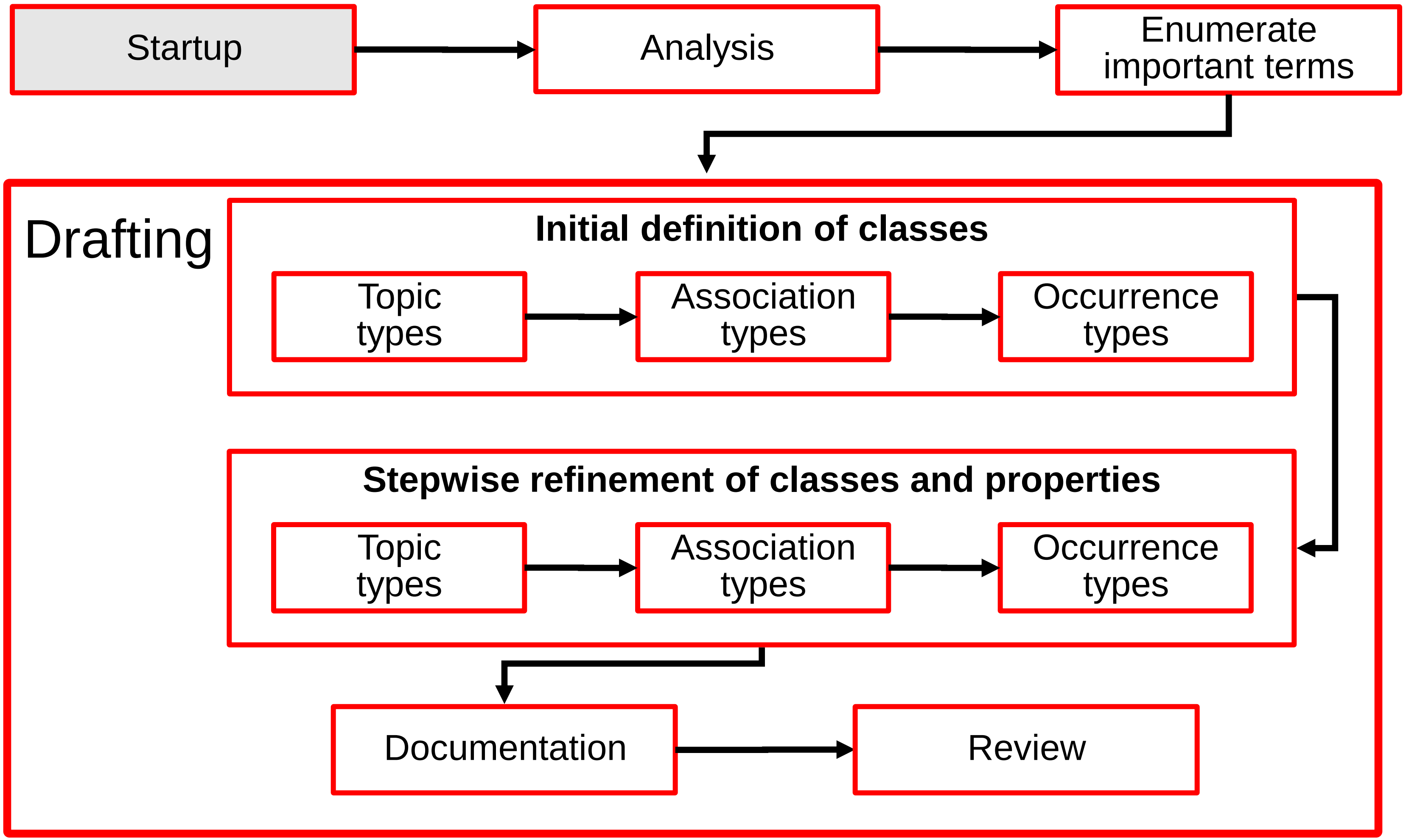
- Idea is formed
- Define requirements
- Design architecture
- Define the ontology
- Conversion and creation of content
- System implementation
- Testing (usability, correctness, stress, ...)
- Production

Ontology development process



- Startup
  - teaches modeler project goals
  - overview of data sources
- Analysis
  - modeler studies data sources
  - draft data flow for system
- Drafting
  - modeler creates draft ontology
  - format covered later
- Interface
  - workshop to draw interface
  - output is screen diagrams + new draft
- Convert
  - modeler and developers do data conversion
  - verifies that ontology matches data + updates ontology
- After this development can begin

1. Startup


Startup
Analysis
Enumerate important terms
Drafting
Initial definition of classes
Topic types
Association types
Occurrence types
Stepwise refinement of classes and properties
Topic types
Association types
Occurrence types
Documentation
Review

Key objectives

- **Understand project goals and scope**
  - know what project is trying to achieve, and what the limitations are
- **Collect initial notes towards ontology**
  - key concepts, notes on scope, granularity, and approach
- **Get overview of data sources**
  - where is the data going to come from?

Activities

- **Preparation**
  - the best way to get started is to read whatever documentation already exists
  - a phone interview or meeting with your contact person will also help
- **Workshop**
  - a common way to get started is to hold a workshop
  - the modeler is typically responsible for the workshop
  - let project start by presenting itself
  - use information gathered in preparation to ask questions to learn more
- **Interviews**
  - interviews with stakeholders and data source owners are always helpful
- **Bootstrapping**
  - competency questions
    - create a list of questions the application must be able to answer
  - brainstorming
    - let the project participants put forward every related concept they can think of

Determining domain and scope

- **Define the domain of the ontology**
  - that is, what is the ontology about? what is being modeled?
- **Define scope**
  - try to create some rules of thumb for what is appropriate to model, and what is out of scope
- **Define granularity**
  - try to create rules of thumb for how much detail should be included

Ontological commitment

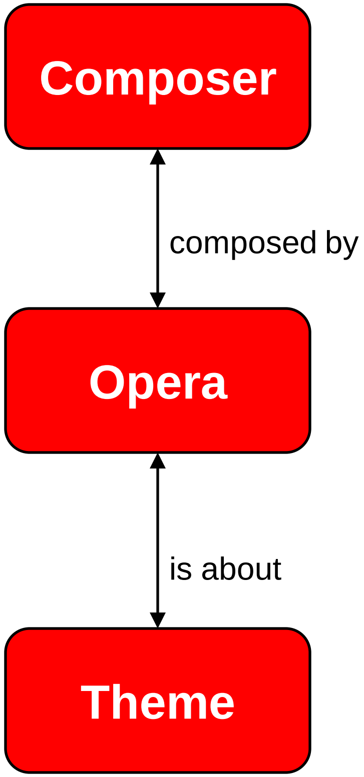
- **Or level of "precision" of the ontology**
  - how much should your ontology actually *say about the domain?*
- **Low precision**
  - "the brain" is a subcategory of "the human body"
  - says very little about anatomy, but is still useful
  - easier to develop the ontology, but fewer ways to use it
- **High precision**
  - "the brain" and "the skull" are "body part"s, "the brain" is contained in "the skull"
  - says a lot about the domain, is very precise
  - ontology harder to develop, but much more useful
- **This is a trade-off!**

Approaches to classification

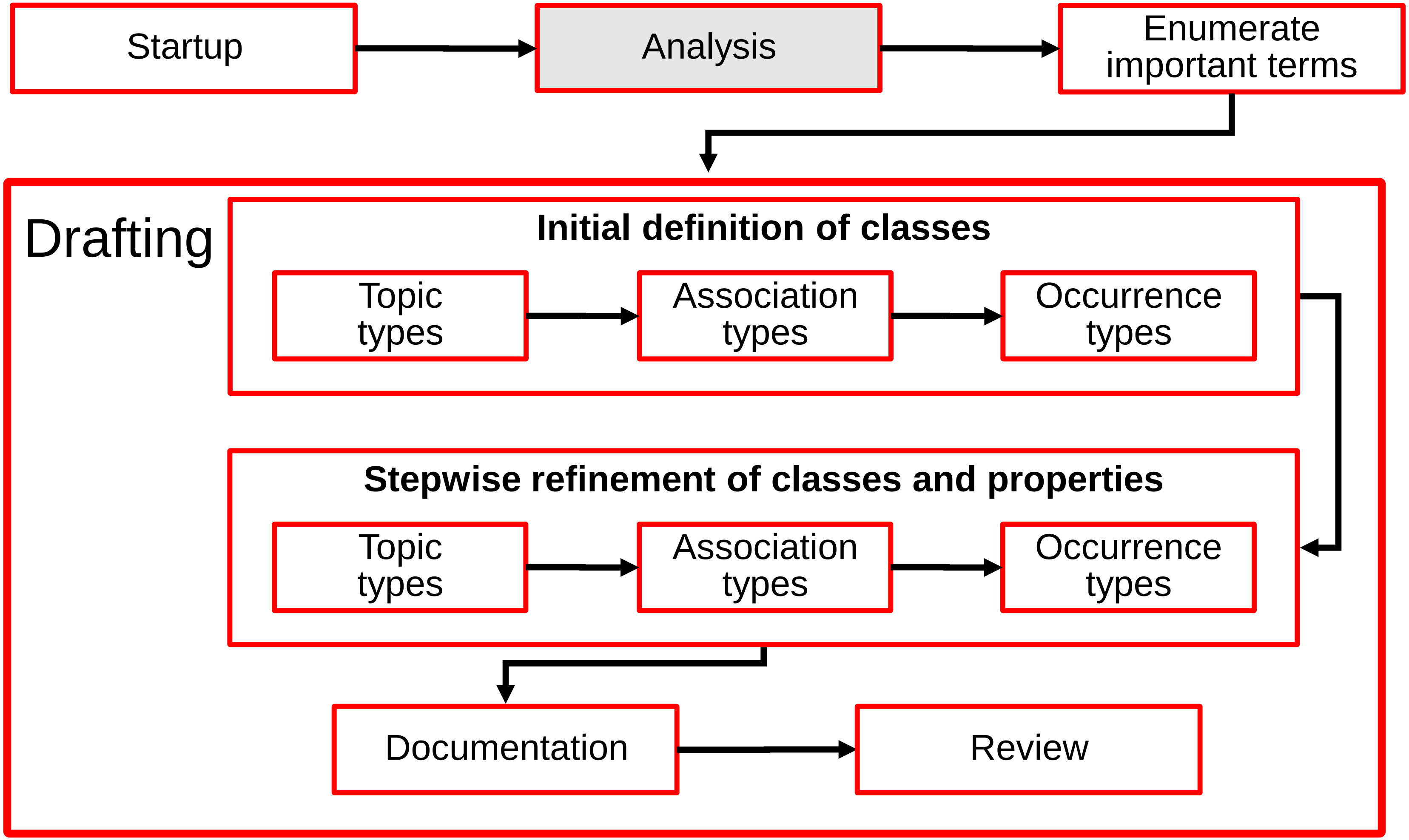
- **There are many possible approaches to classification**
  - all of them right in some contexts
- **Taxonomy**
  - just use a simple hierarchy of concepts for classification (a la Yahoo!)
- **Thesaurus**
  - taxonomy with multiple terms per concept
- **Faceted classification**
  - multiple hierarchies for different aspects of the classified subject
- **Full ontology**
  - like Italian Opera
- **Hybrid solution**
  - full ontology structure for concrete things (people, places, ...)
  - thesaurus for intangibles (general subjects)

A common solution

- **A common solution is to go for the hybrid approach**
- **Typically, this means to**
  - describe the concrete precisely
  - use a taxonomy for the more intangible aspects
- **This is a simple, cost-effective solution to the problem**



2. Analysis



Startup

Analysis

Enumerate important terms

Drafting

Initial definition of classes

Topic types

Association types

Occurrence types

Stepwise refinement of classes and properties

Topic types

Association types

Occurrence types

Documentation

Review

Key objectives

- **Understand data sources**
  - ensure that ontology can serve as target for conversion
  - generally involves quite a bit of interaction with data source owners
  - do *not* expect data source owners to cooperate!
- **Match to requirements**
  - ensure that ontology can meet requirements
- **Consider how to fill the gaps**
  - if data sources are insufficient to meet requirements, consider what to do

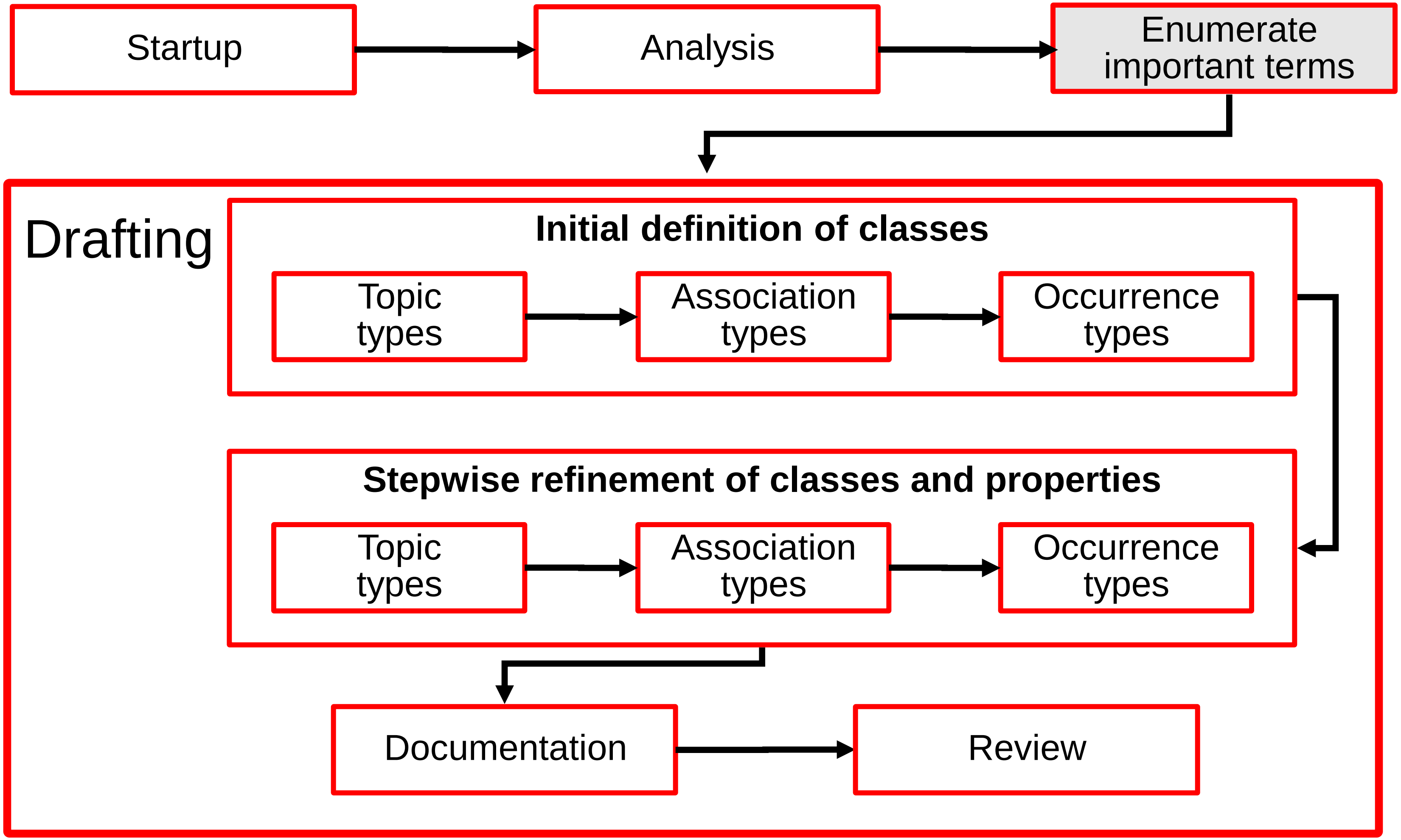
Structured sources

- **Materials to get hold of**
  - the schema for the data
  - documentation of the schema, if any
  - actual exports of the data, or samples of the data
- **Do not take "no" for an answer!**
  - data source owners are generally very reluctant to part with information
  - this often makes the process reminiscent of pulling teeth
  - keep at it! (usually neither you nor they have any choice, anyway)
- **Do not trust the schema/documentation**
  - that a field exists doesn't mean it has values
  - that the field has values doesn't mean the values take the form you expect
  - Murphy's law is a useful guide
  - looking at real data is the *only* way to make sure

Unstructured sources

- **Materials to get hold of**
  - samples of data
  - entire data set
- **Ensure that samples are representative**
  - all character encodings
  - all formats
  - all variations within the same format
  - all languages
  - ...
- **Study to see what can be done**
  - what metadata can be extracted automatically?
  - is there any way to extract keywords?
  - how are special characters represented? (expect problems here)

3. Enumerate important terms



Enumerate important terms

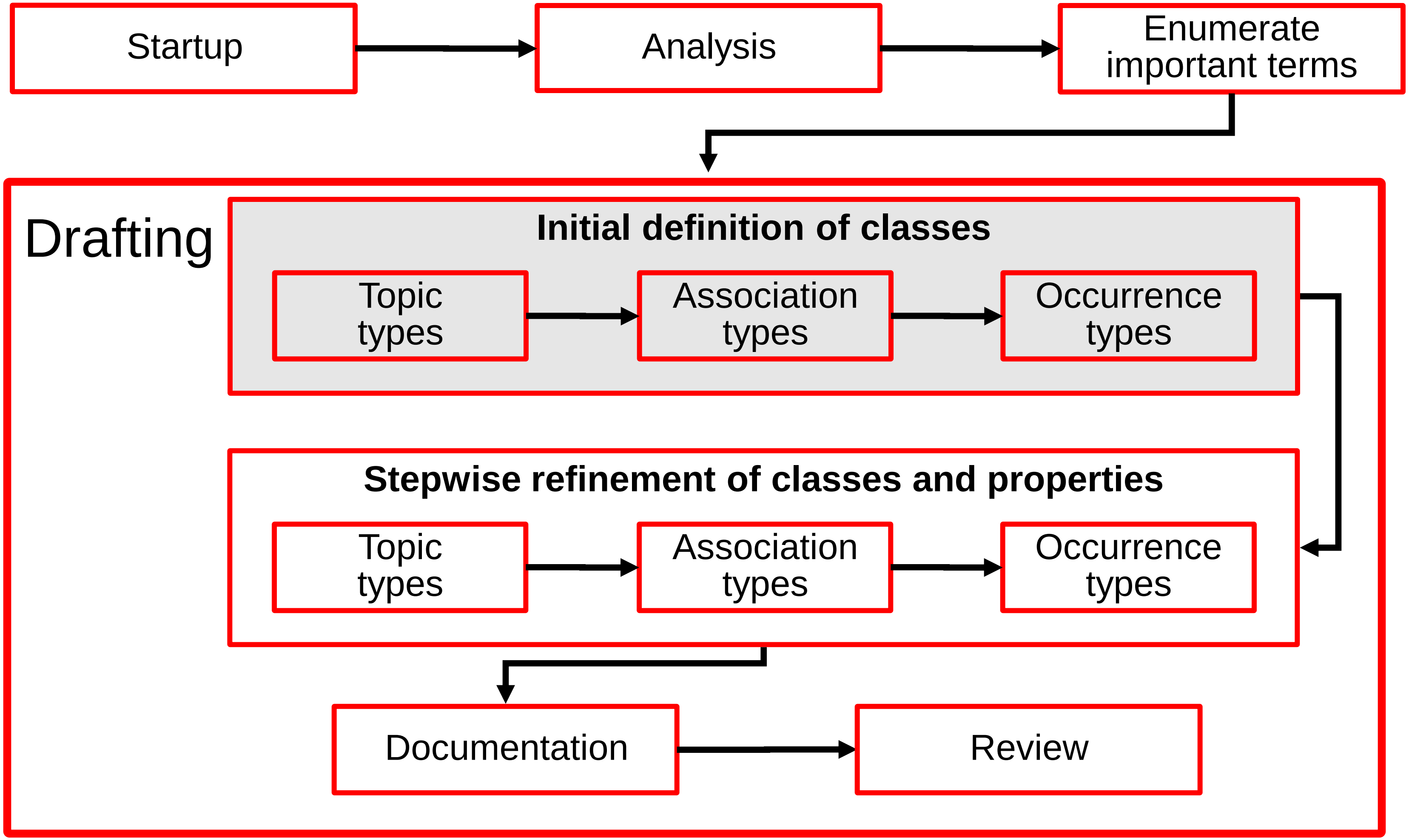
- Brainstorm *anything* you think could be relevant to the domain to be covered by the topic map
- Don't be too picky; you'll organize this later
- The result should be a ragbag of entities (individuals), classes, relationships, resources, etc.
- This is the starting point for identifying the classes that comprise the ontology
- We refine the ontology by improving our understanding of the relationships between the items in the ragbag

Your turn...

Exercise:

List 20-30 terms, concepts, entities, objects – whatever – that are pertinent to the domain to be covered by the topic map

4. Initial definition of types



Startup

Analysis

Enumerate important terms

Drafting

Initial definition of classes

Topic types

Association types

Occurrence types

Stepwise refinement of classes and properties

Topic types

Association types

Occurrence types

Documentation

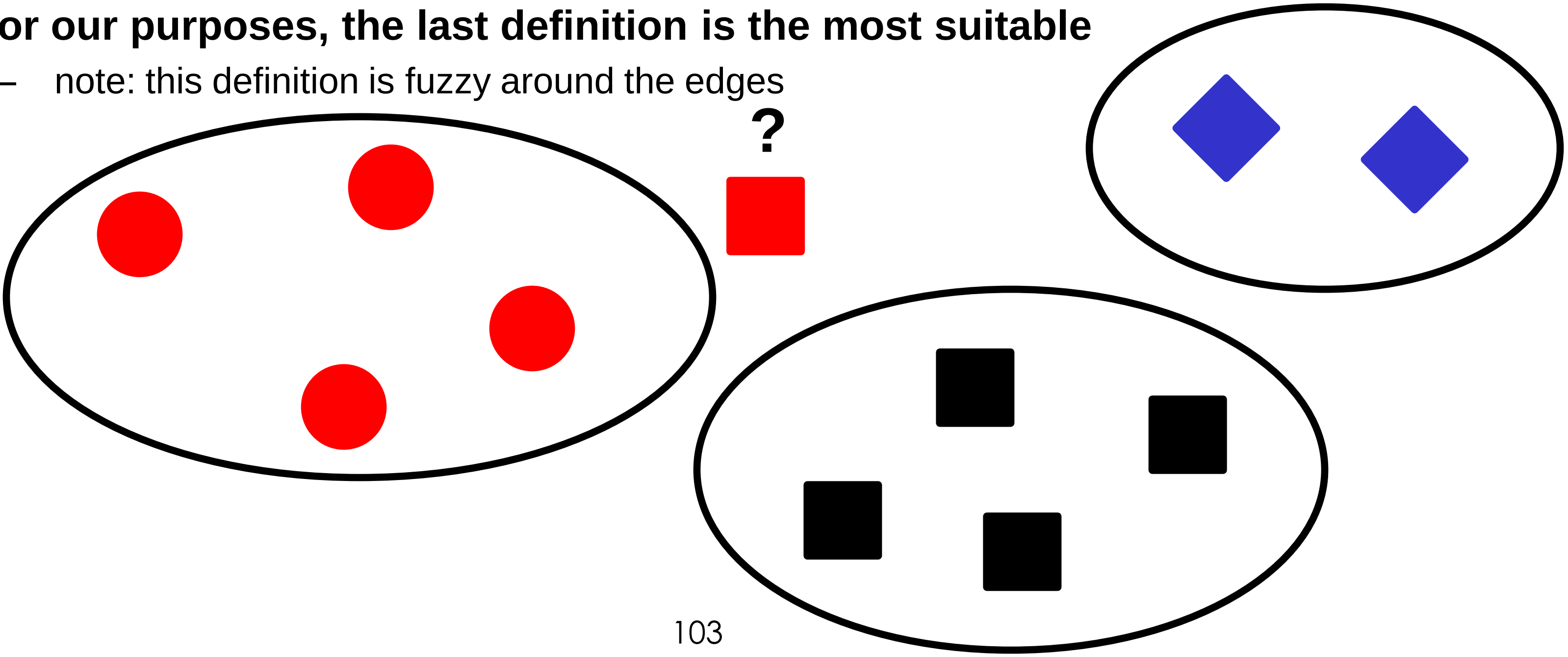
Review

Initial definition of types

1. **Defining topic types**
   - leave details for later
2. **Defining association types**
   - leave association role types until later
3. **Defining occurrence types**

What is a type?

- **The definition of *type* depends on who you ask**
- **Common answers are**
  - it's a set of things that have something in common, no matter what
  - it's a set of individuals which share at least one common property
  - it's a category, implicitly or explicitly defined, which has a set of things which can be said to be instances of that category
- **For our purposes, the last definition is the most suitable**
  - note: this definition is fuzzy around the edges



How to find the type of something

- **One way to come up with types is to look at an ordinary something that should appear in your topic map**
- **Look at it, and ask yourself “what is this?”**
- **Any answer of the form “it is a(n) _______” is on the right track**
- **You should be able to turn your answer around and find lots of other things that are also ________s, and which should go into your topic map**

Some examples

Lars Marius Garshol

USA

France

Boeing

Ontopia

Steve Newcomb

Norway

Google, Inc.

Michel Biezunski

Steve Pepper

Sorting it out

Person	Company	Country
Lars Marius Garshol	Boeing	Norway
Steve Newcomb	Ontopia	France
Michel Biezunski	Google, Inc.	USA
Steve Pepper		

So what is a type?

- **A type is a particular kind of category that has instances**
- **If it doesn't have instances, it's not a type**
  - if you wonder if "foo" is a type, check if you can think of any things which *are* "foo"s
  - if not, it's not a type
- **Example**
  - "geography" is a category, but not a type
- **Example**
  - "country" is a type, because there are things which are countries

Rules of thumb for types

- **For a good type the instances should intrinsically be instances of the type**
  - that is, they should always be instances of that type from the moment they come into existence until they stop existing
  - when you see a thing it should be immediately obvious which type it belongs to
- **Example**
  - "person" is a type, because someone either is or is not a person, and that's it
  - "female" and "male" are also acceptable types
  - "lawyer" is a bad type, because lawyers are only lawyers at work, and look and act exactly like other people

Topic type or role type?

- **"Lawyer" is a bad type because lawyers are not lawyers by their very nature, but because of their relation to something else (their work)**
- **This means that "lawyer" is really a role type, not a topic type**
- **The same applies to "composer",**
  - because the definition of composer is "someone who has composed a musical work"
  - the topic type is really "person"
- **Another example would be "learning object"**
  - the e-learning community has had fierce debates about the definition of this
  - really, it is anything that's used to teach someone something
  - in other words, it's not inherent in the thing that it is a learning object, it becomes one when it is used for this purpose

What to do about role types?

- **Decide that it really deserves to be a topic type anyway**
  - because there are special rules for these topics, or
  - because these topics are especially prominent and deserve this treatment, or
  - something else
- **Alternatively,**
  - mark it as a role type, and
  - try to uncover the association type and real topic type later

Identify types of associations

- **Anything that is a relationship between topics is an association**
  - this should make it easy to find these in the ragbag
- **Consider whether you need relationships that are not in the ragbag**
  - if so, these are association types, too

Define occurrence types

- **Occurrences are a bit tricky, because they are used for two different things in Topic Maps**
  - properties like date of birth, height, number of feet, ...
  - references to external resources
  - the references are really just properties of type URI, but conceptually they are quite different
- **From here on we refer to**
  - properties, which are internal occurrences, and
  - external occurrences

Define occurrence types

- **Information resources are connected into the topic map with external occurrences**
- **Information resources are things like**
  - documents,
  - images,
  - web sites/pages,
  - ...
- **Therefore, most things which are resources in the ragbag should become external occurrences**

Occurrences versus topics

- **That something is an external resource doesn't mean it has to be an occurrence**
- **Imagine external occurrences where you find that you want to record**
  - the title,
  - the author,
  - the date of publication, etc etc
- **In this case, you want to talk about the information resource that is the occurrence**
- **The way to do this is to make a topic for the information resource**
  - the occurrence is a topic-resource relationship, anyway, so you can turn the occurrence into an association instead
  - the URL of the information resource now becomes the *subject locator of the topic*

Properties versus topics

- **In some cases properties can only have certain specific values**
- **For example, for "person", the value of "sex" can only be male/female**
- **Another example is "workflow state", which may be**
  - initiated, draft, accepted, published, deleted
- **In these cases it may be better to**
  - create topics for the values (with a topic type of their own), and
  - use an association instead of an occurrence type to connect them

What is a topic name?

- **Topic names seem intuitively unproblematic**
  - however, in some cases the distinction between name and property can be troublesome
- **The general principle is that anything you might want to display as the label, name, or title of a topic is a topic name**
  - names can have types
  - usually there is a default name, that you can just call “name”

Your turn...

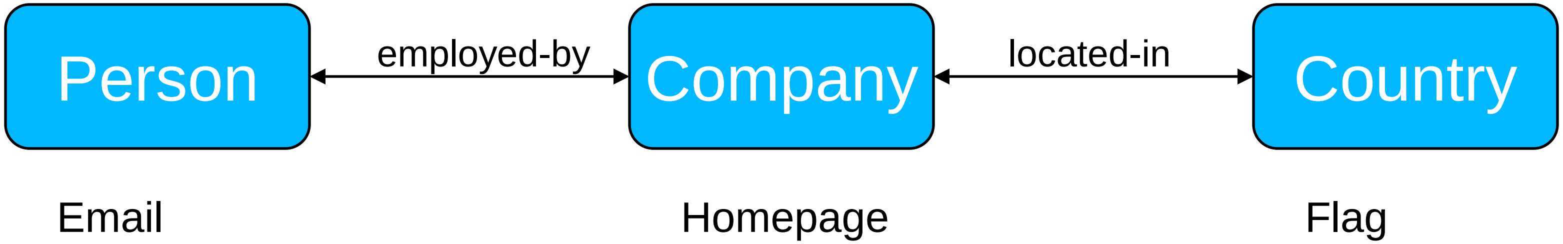
Start from the ragbag, and

- ***consider possible interpretations of each term***
- ***choose a definition for each term***
  - *you don't have to write it down, as long as you can come up with one on demand*
- ***classify each term as one of the following, based on its definition***
  - *topic type*
  - *association type*
  - *role type*
  - *instance*
  - *topic name*
  - *external occurrence type*
  - *property type*
  - *problem area*
  - *subject locator*
- ***change the name if necessary now that you know what it is***

Attaching properties to topic types

- **The next step is to “connect the dots” and create an ontology that hangs together**
- **For each association type**
  - determine what topic types it connects
  - note that the needed topic type(s) may not exist in your list yet
- **For each occurrence type**
  - determine to what topic types it applies
  - note that the needed topic type(s) may not exist in your list yet

Creating a diagram



Person

employed-by

Company

located-in

Country

Email

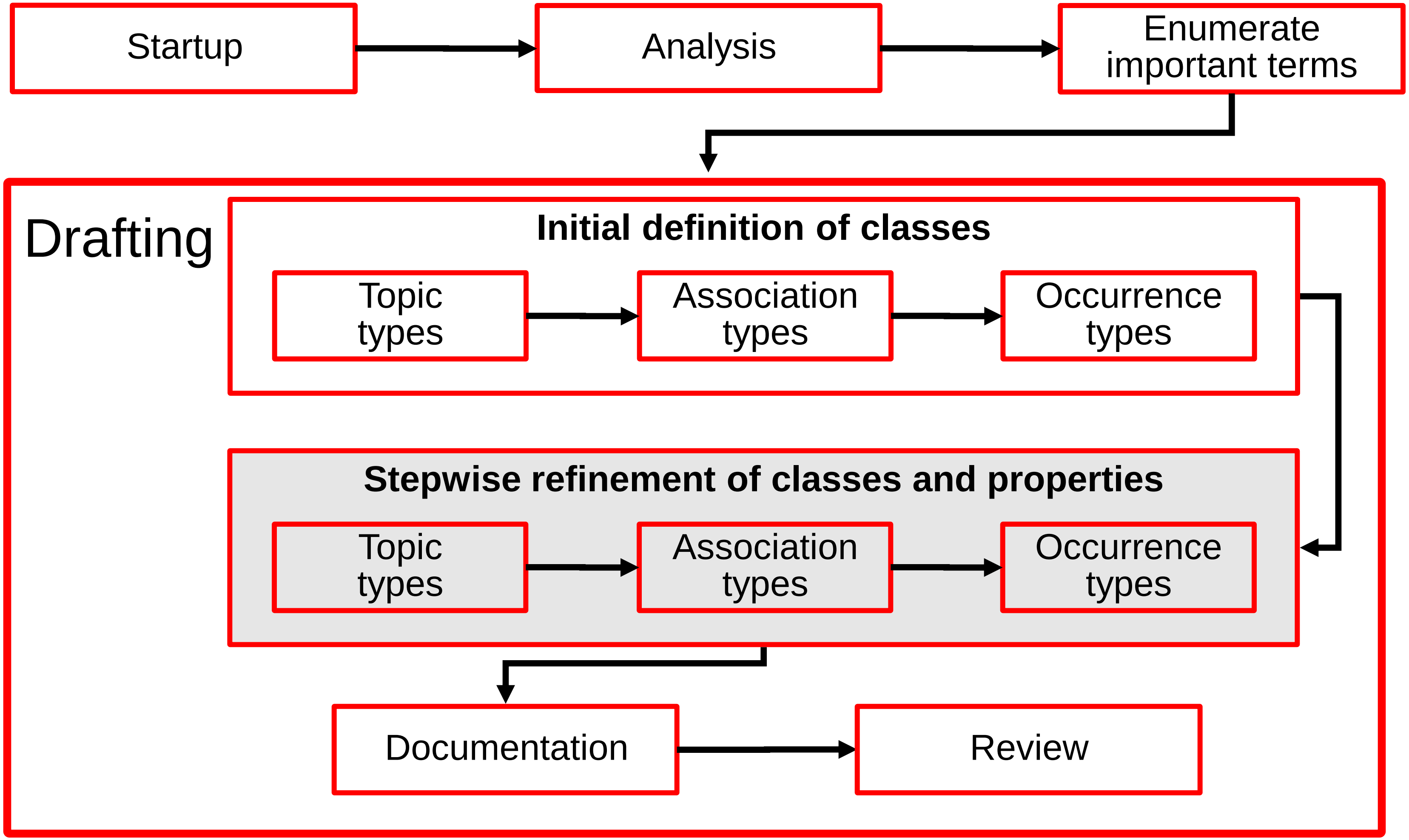
Homepage

Flag

Your turn...

- **Create a diagram for your ontology, similar to the one on the previous slide**
  - use the classification of your terms to do the diagram
  - include only ontology-level information

5. Refine classes and properties

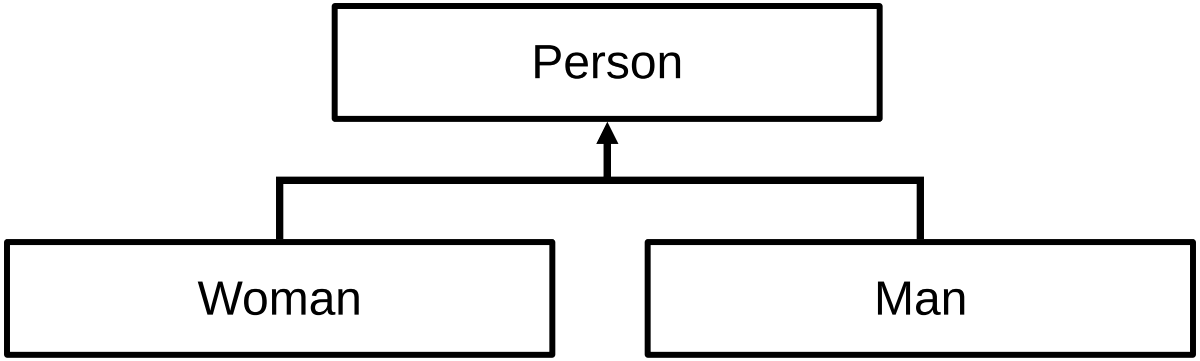


Refining classes and properties

- **Refining topic types**
- **Modelling topic names**
- **Refining association types**
- **Refining occurrence types**
- **Working out guidelines for usage**

Supertypes

- **Quite often there are relationships between the types in an ontology**
- **For example, man, woman, and person are clearly related**
- **In fact,**
  - every man is a person, and
  - every woman is a person, but
  - not every person is a man, and
  - not every person is a woman
- **This means that person is a supertype for both man and woman**
- **Capturing this explicitly in the ontology is usually a good idea**



Person

Woman

Man

How type hierarchies work

- **The superclass-subclass relationship has defined semantics**
  - therefore: make sure you use it correctly
  - if you abuse the semantics you *will get incorrect results!*
- **If A is a superclass of B, then**
  - both A and B *must be classes*
  - if C is an instance of B, it *must also be an instance of A*
  - *if D is a subclass of B, it must also be a subclass of* A

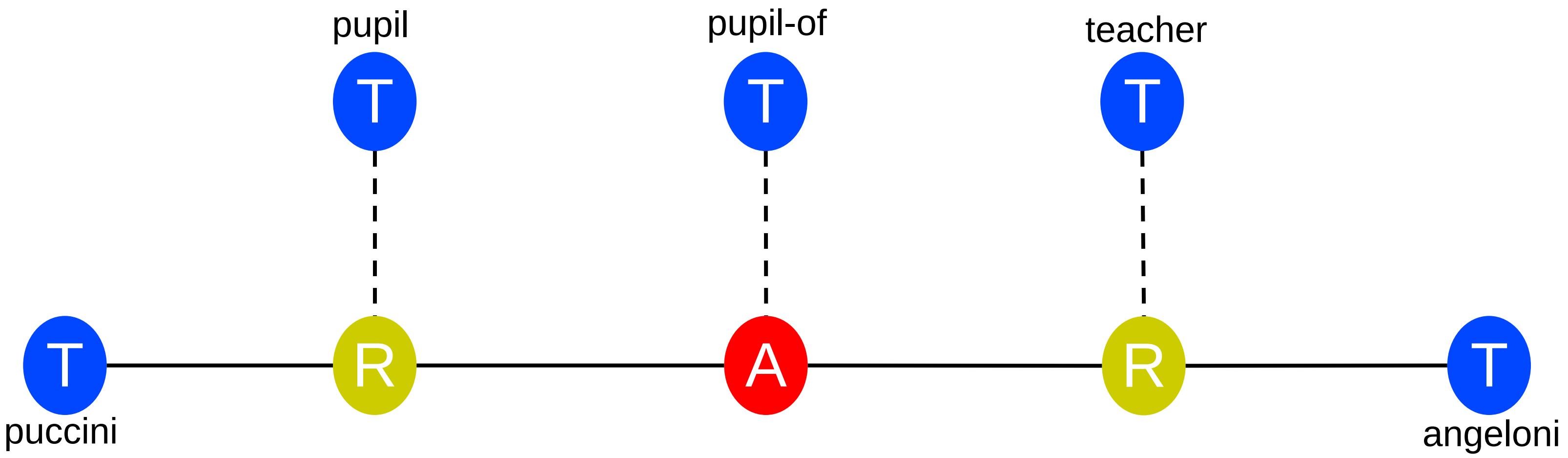
Name conflicts

- **In topic maps it's allowed for two different topics to have the same name**
- **However, in user interfaces this may be confusing**
  - imagine trying to specify the location of the Louvre and having to choose between three[1] Parises in the same drop-down list...
- **One solution: define a naming policy that avoids duplicate names**
  - usually this will look like "Name (Disambiguator)", where the disambiguator may be the type, or some characteristic
  - result: "Paris (city)" vs "Paris (Greek hero)"
  - or: "Paris (France)" vs "Paris, Ontario”

[1] France has 1 Paris, Canada 2, and the US at least 14

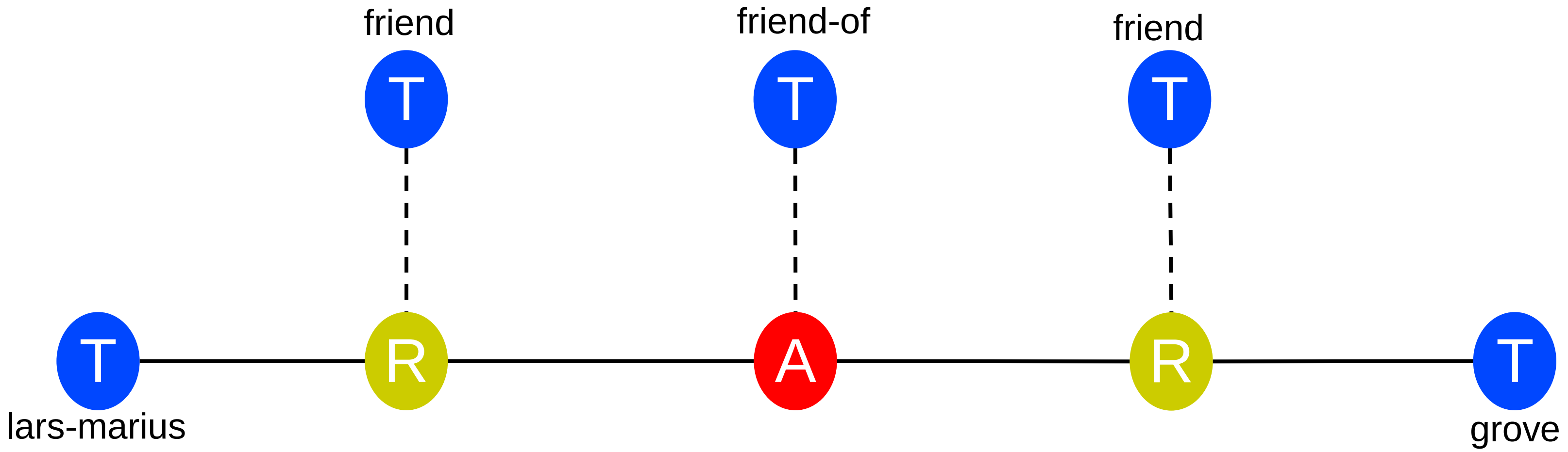
Association role types

- So far we haven’t given association role types any thought
- The purpose of these is to tell us what each topic in the association is doing
- All associations must have them



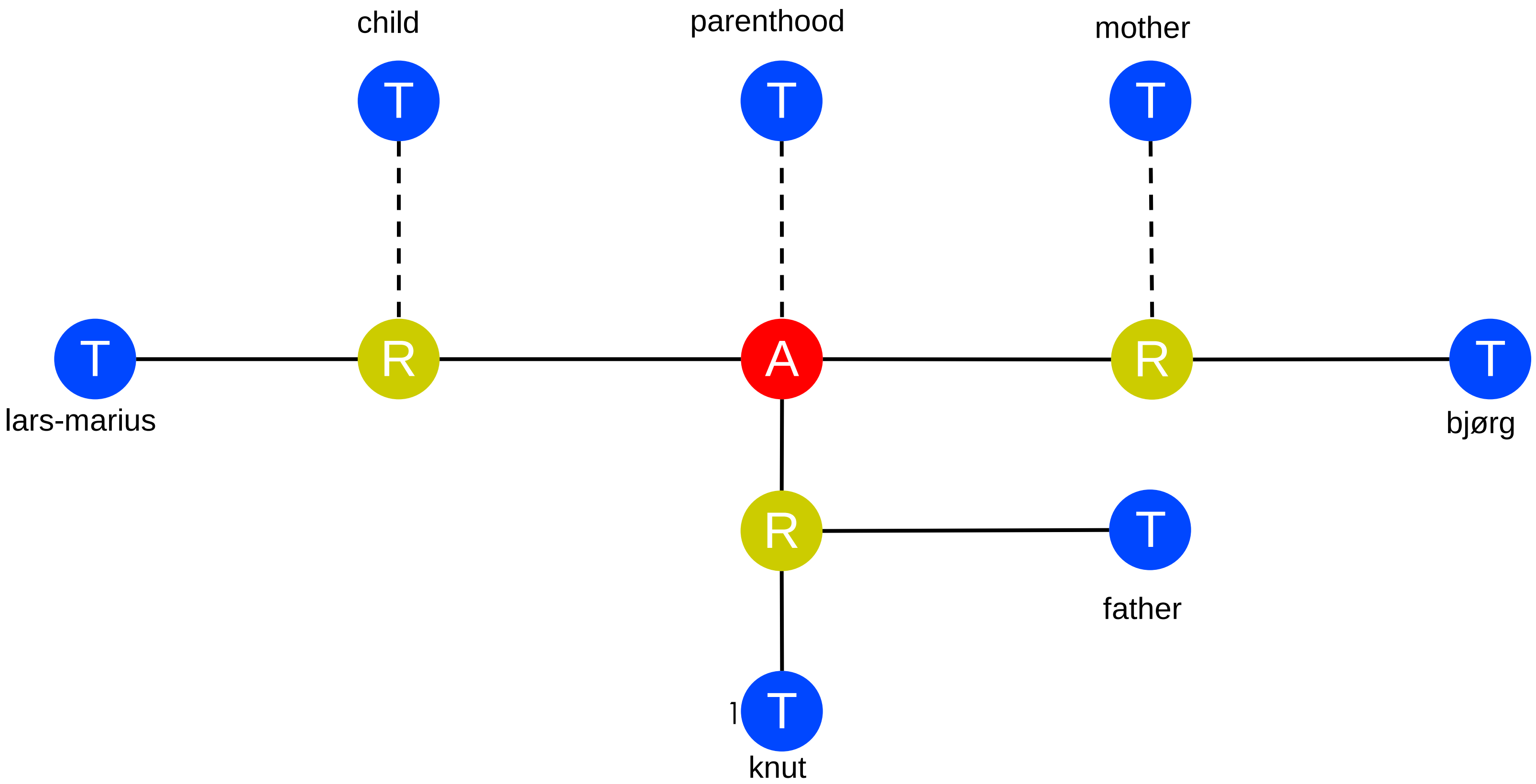
Symmetric associations

- **Some associations are the same in both directions**
  - for example, if I know you, you will also know me
- **In this case we should play the same role in the association**
  - we are doing the same thing in the association, so...



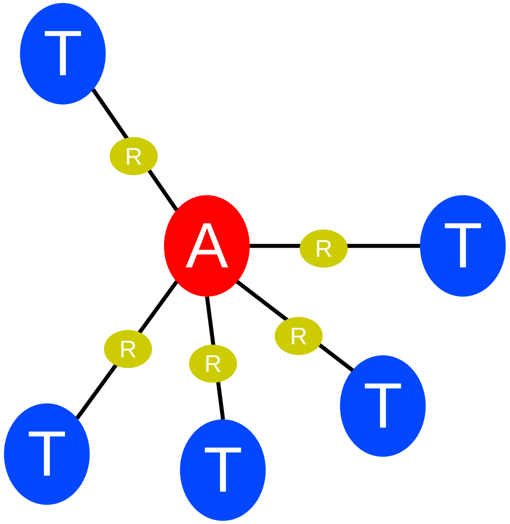
How many roles?

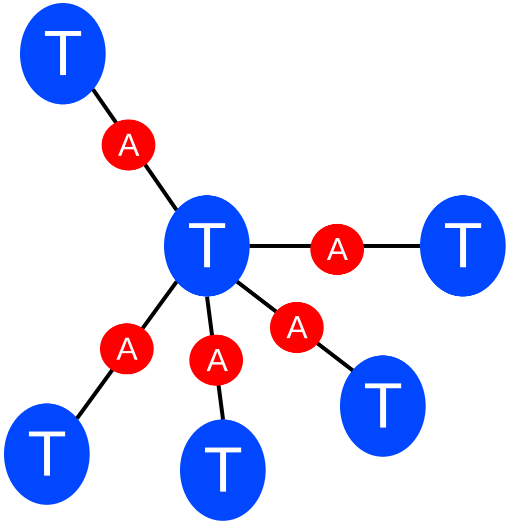
- **Associations can have any number of roles**
- **Two roles is by far the most common**
  - such associations are called *binary associations*
- **However, sometimes you need more than two roles**



Rules of thumb for association roles

- **Keep the number of roles as low as possible**
  - consider whether introducing the intermediate topic makes sense for you
- **Avoid repeating roles**
  - if one of the roles can be played multiple times in the same association this indicates that the association represents a collection/group
  - in these cases, you probably should have a topic for the collection/group





Coming up with role types

- **Many associations occur over and over again**
  - employment(employee, employer)
  - part-whole(part, whole)
  - containment(container, containee)
  - parent-child(parent, child)
  - located-in(located, location)
  - is-about/has-subject(subject, work)
- **Some role types are very broad and can be used in many association types**
  - (object, agent)

Naming association types

- **Approaches to creating intuitive labeling**
- **Nouns**
  - expressing the nature of the relationship
    e.g., “first performance”
  - compounds created from the role names
    e.g., “teacher/pupil”
- **Verbs**
  - very natural, but they imply direction (subject-verb-object)
  - consider multiple names in different scopes:
    Tosca was composed by Puccini
    Puccini composed Tosca

Refining external occurrences

- **For each type of external occurrence decide**
  - which topics can have this occurrence type,
  - how many occurrences of this type it may have, and
  - what the allowed formats of the information resources are
- **In some cases there may be no restrictions**

Refining internal occurrences

- **For each type of internal occurrence decide**
  - which topics can have this occurrence type,
  - how many occurrences of this type it may have, and
  - what the allowed syntax(es) for the strings is
- **In some cases there may be no restrictions**

Your turn...

Continue refining your ontology:

- ***consider what names your topic types may have***
  - *- number of each, scopes on each*
- ***refine the association types***
  - *- define the role types*
  - *- define how many associations of this type each topic type can have*
- ***refine the occurrence types***
  - *- how many occurrences of each type can topics of the different types have?*
  - *- external occurrences: rules for file format or location?*
  - *- internal occurrences: rules for the syntax?*

6. Documenting the ontology

- **The ontology is necessary to create a topic map, but not always sufficient**
  - if more than one person is going to use the ontology, documentation is usually necessary
  - documentation is always useful
  - documentation always requires resources to produce and maintain
- **Benefits of documentation**
  - helps ensure consistent usage of the ontology
  - can be a lifesaver in the future when nobody remembers how to use the ontology any more
  - describing the ontology is a useful way to sanity-check it
  - ensures agreement on the ontology within the team
- **Documentation can take several forms**
  - a guide document that explains the ontology
  - a set of published subject indicators
  - a schema for the ontology

Guide document

- **Basically a normal document that explains the ontology in English**
- **Should have diagrams for presenting the big picture**
  - boxes and arrows work fine for this
  - UML, ER, or ORM can also be used
- **Tables and lists of types and descriptions for these is also good**

<topic> IDs

- **The IDs of topics are important**
  - They will be the target of a large number of references
  - You'll constantly be referring to these topics in queries, schemas, web pages, scripts, etc.
- **Choose a rule-based ID-naming scheme and stick to it**
- **If you don't edit by hand IDs only matter for ontological topics**

Published subject indicators

- **This means creating URIs for all topics in the ontology**
  - *Highly recommended, since it simplifies application development*
  - *Also greatly simplifies merging in many situations*
- **It also means creating one or more web pages documenting them**
  - Example: http://psi.oasis-open.org/iso/3166/
- **No particular requirements on the form of these pages**
  - The URIs should resolve to descriptions of the topics, though
  - It's possible to make the PSIs be the guide document

Fill in the individuals

- **Doing so will make you want to say many new things**
- **Some of these will be difficult with the existing ontology**
- **This is an indication that the ontology is in need of extension**
- **Go back and consider how to work the new features in**

New kinds of relationships

- **New subjects may require new relationships to be defined**
- **Explicit transitive relationships**
  - e.g. A is in B, B is in C therefore A is in C
  - Ideally applications can make transitive deductions at runtime, but they may need to be precomputed
- **'Short-cuts' which directly bind topics more complex relationships**
  - e.g. A lives in B, B is in C; therefore A lives in C

Inconsistent usage

- **Confused use of subject/relationships**
- **Inconsistent classification is worse than no classification!**
- **Confusion may indicate that there is an unclear distinction. Options are:**
  - Clarify the distinction
  - Remove the distinction entirely
  - Create additional subject / relationship classes to enable clear partitioning
- **Documentation and education is key!**

Unused subjects / relationships

- **May be an instance of the ‘Confused use’ scenario**
- **May be a result of over-enthusiastic initial design**
- **Justification needed to remove them from the ontology**
- **Possible reasons for removal**
  - we can't afford to maintain this information
  - we can't imagine any possible use for this information
  - this information is already captured elsewhere
  - we don't understand this well enough to define it properly yet

Review personnel

- **As for original ontology design**
- **Also include the designers / maintainers of toolsets used in the implementation**

Conclusion

- **Ontology design is an art, not a science**
  - there is no single correct answer
  - practice makes perfect
- **Many of the decisions depend on external factors**
  - the use of the ontology
  - the organization that is going to use it
  - systems connected to the topic map application
  - the tools used on the application
  - need (or lack thereof) for intergrating with other topic map applications
- **When in doubt, go for beauty**

DC 2008 Seminar

Ontology Design & Interoperability (2): RDF/OWL

Professor Sam Oh, Sungkyunkwan University, Seoul, Korea

ISO TC46/SC9 & JTC1/SC34 Chairman

DCMI Board of Trustee

samoh21@gmail.com

http://home.skku.edu/~samoh/

Resource Description Framework

Why XML is Not Enough

- Main advantage of using XML is reusing the parser and document validation
- Many different possibilities to encode a domain of discourse
- Leads to difficulties when understanding of foreign documents is required

==> Next step: separate content from structure!

Introduction to RDF

- RDF (Resource Description Framework)
  - Beyond Machine readable to *Machine understandable*

- RDF unites a wide variety of stakeholders:
  - Digital librarians, content-raters, privacy advocates, B2B industries, AI...
  - Significant (but less than XML) industrial momentum, lead by W3C

- RDF consists of two parts
  - RDF Model (a set of triples)
  - RDF Syntax (different XML serialization syntaxes)

- RDF Schema for definition of Vocabularies (simple Ontologies) for RDF (and in RDF)

URIs

- Uniform Resource Identifier
  - The generic set of all names/addresses consisting of short strings that refer to resources

- URLs (Uniform Resource Locators) are a particular type of URI, used on the WWW

- URIs look like URLs, sometimes with fragment identifiers to point at specific parts of a document

  http://somedomain.com/some/path/to/file#fragment

RDF Data Model

- **Resources**
  - A resource is a thing you talk about (can reference)
  - Resources have URI's
  - RDF definitions are themselves Resources

- **Properties**
  - slots, define relationships to other resources or atomic values

- **Statements**
  - "Resource has Property with Value"
  - (Values can be resources or atomic XML data)

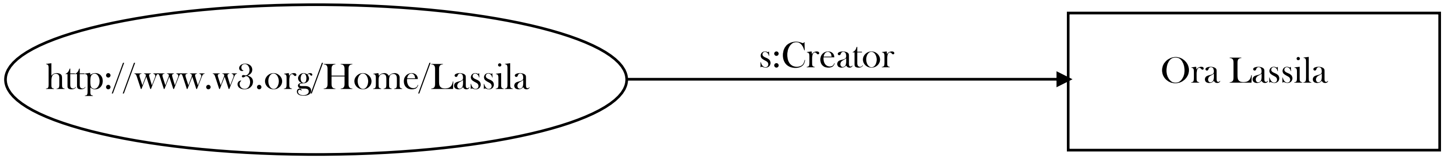
A Simple Example

- **Statement**
  - “Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila”

- **Structure**

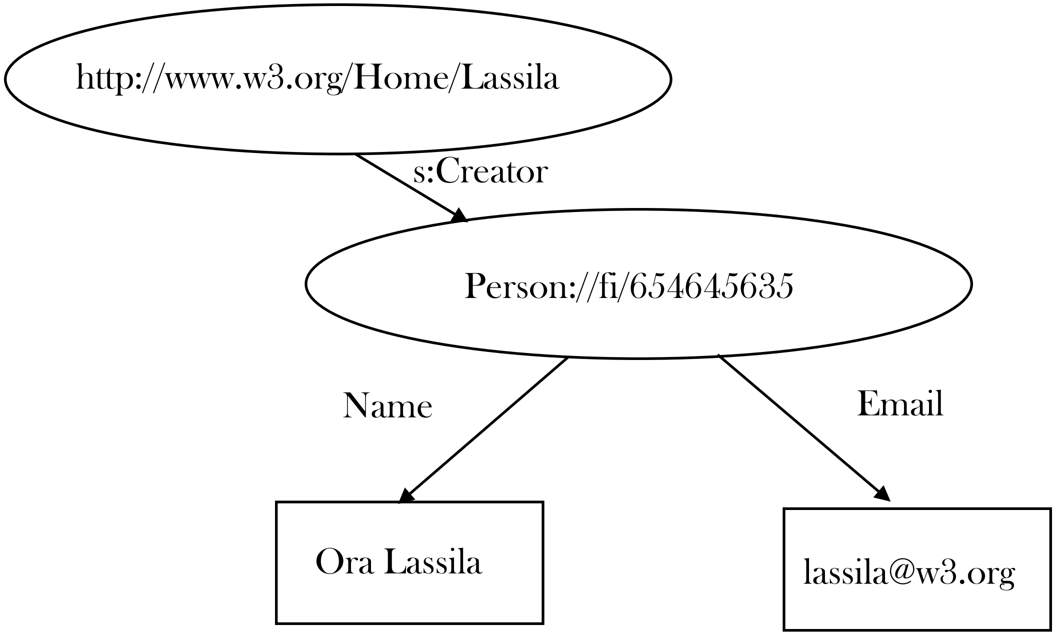
– Resource	(subject)	http://www.w3.org/Home/Lassila
– Property	(predicate)	http://www.schema.org/#Creator
– Value	(object)	"Ora Lassila”

- **Directed graph**



Another Example

- To add properties to Creator, point through an intermediate Resource.



XML Namespaces

- RDF namespace uses “rdf” prefix by convention
- RDF Schema namespace uses “rdfs” prefix by convention
- Properties are declared in other namespaces, so they are Web-unique

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:vassar=”http://www.vassar.edu/schema.rdf”
              xmlns:biblio=”http://www.library\ies.org/schema.rdf”>

<Description rdf:about=”http://www.cs.vassar.edu/~ide”>
    <biblio:author-of>Encoding Syntactic Annotation</biblio:author-of>
</Description>
<Description rdf:about=”http://www.vassar.edu”>
    <vassar:employee rdf:resource=”http://www.cs.vassar.edu/~ide”/>
</Description>
```

OWL Web Ontology Language

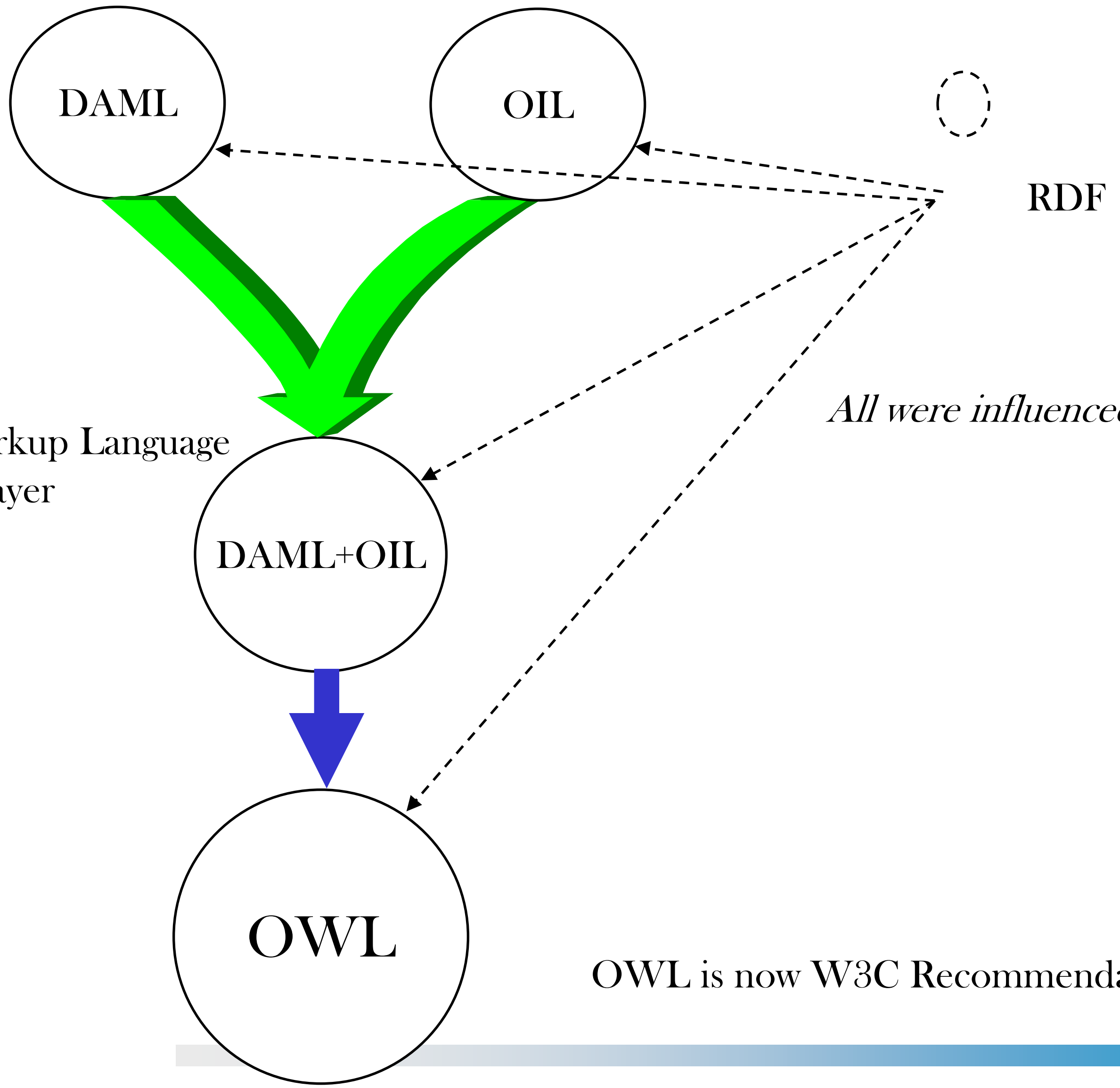
Roger L. Costello

David B. Jacobs

The MITRE Corporation

(The creation of this tutorial was sponsored by DARPA)

Origins of OWL



All were influenced by RDF

DAML = DARPA Agent Markup Language
OIL = Ontology Inference Layer

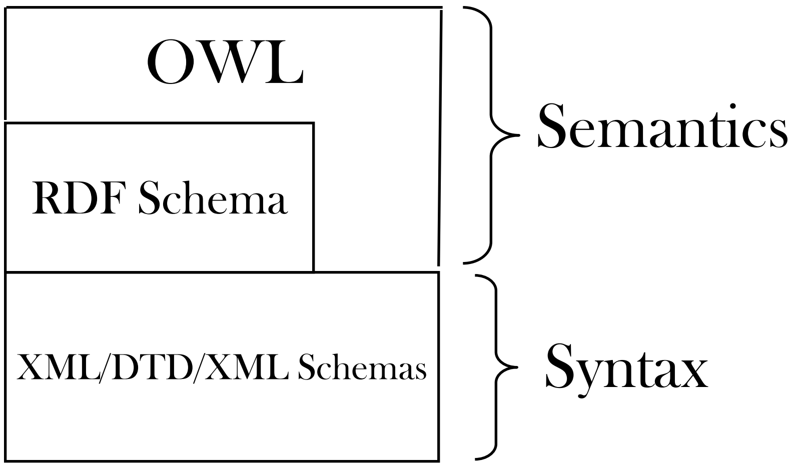
OWL is now W3C Recommendation!

Purpose of OWL

- The **purpose** of OWL is identical to RDF Schemas - to provide an XML vocabulary to define classes, properties and their relationships.
  - RDF Schema enables you to express very rudimentary relationships and has limited inferencing capability.
  - OWL enables you to express much richer relationships, thus yielding a much enhanced inferencing capability.

- The **benefit** of OWL is that it facilitates a much greater degree of inferencing than you get with RDF Schemas.

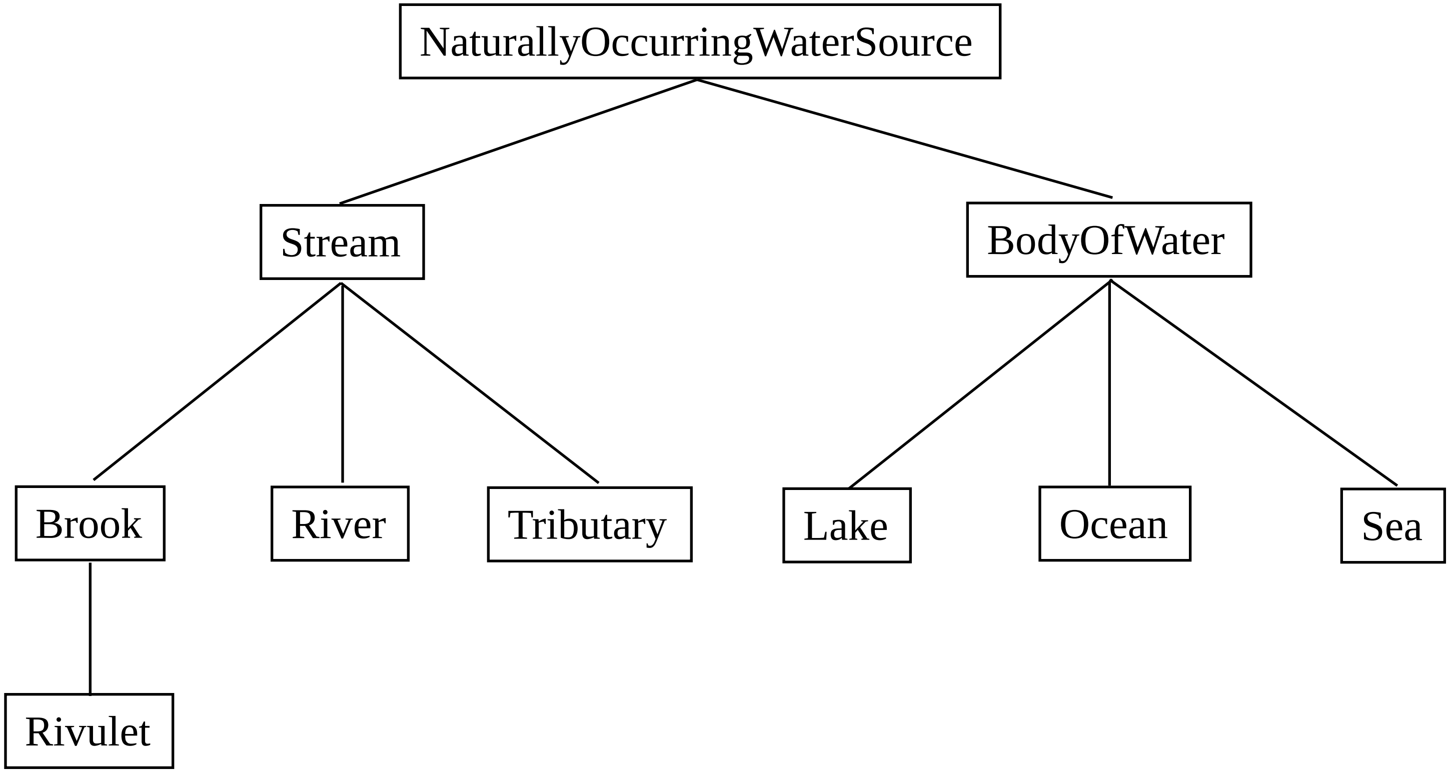
OWL and RDF Schema enables machine-processable semantics



OWL = RDF Schema + more

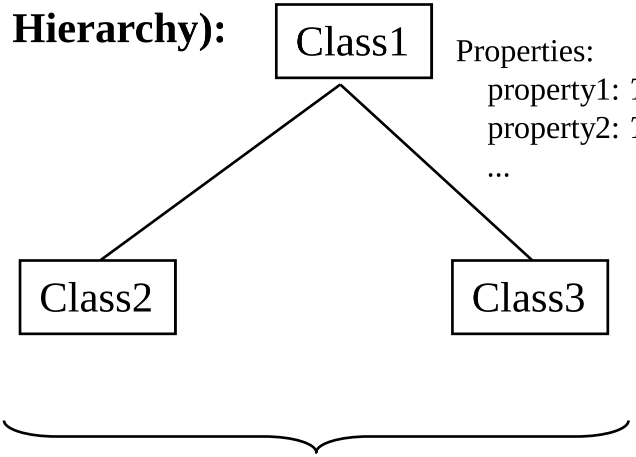
- Note: all of the elements/attributes provided by RDF and RDF Schema can be used when creating an OWL document.

We will use the below "water taxonomy" to explain OWL



Notations used

Taxonomy (Class Hierarchy):



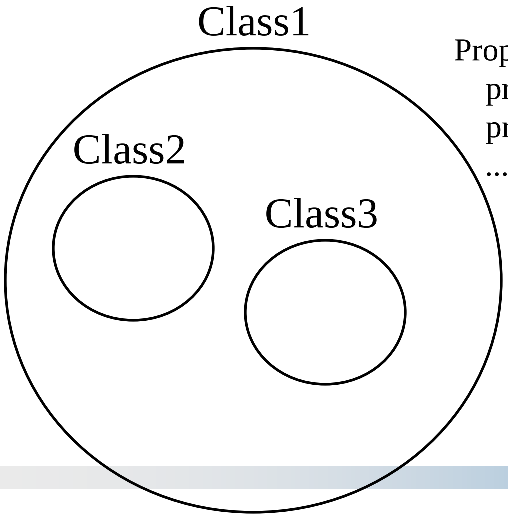
Properties:
property1: *Type1*
property2: *Type2*
...

These are the properties of Class1. The name of the property is shown (e.g., property1), and its range is shown in italics (e.g., *Type1*).

Class2 and Class3 are subclasses of Class1.

Venn Diagram:

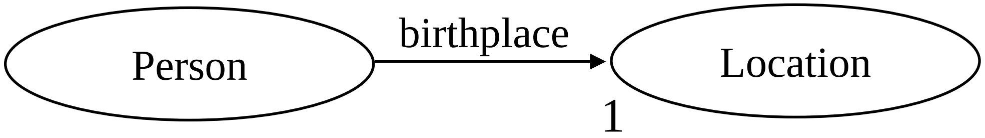
An alternate notation to the above class hierarchy is to use a Venn diagram, as shown here.



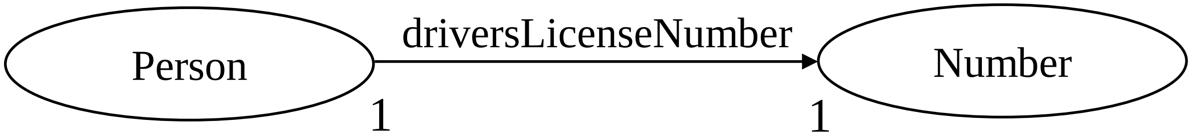
Properties:
property1: *Type1*
property2: *Type2*
...

Notations used…

This notation is used to indicate that a person has only one birthplace location:



This notation is used to indicate that a person has only one driver's license number. Further, a driver's license number is associated with only one person:



Two Minute Preview of OWL

- Before getting into the details of OWL let's spend a couple of minutes examining three examples which demonstrate some of the capabilities of OWL.

Example 1: The Robber and the Speeder

DNA samples from a robbery identified John Walker Lindh as the suspect. Here is the police report on the robbery:

```
<Robbery rdf:ID="report-2003-03-17-XTf4">
    <description>...</description>
    <suspect>
        <Person rdf:about="http://www.person.org#John_Walker_Lindh"/>
    </suspect>
</Robbery>
```

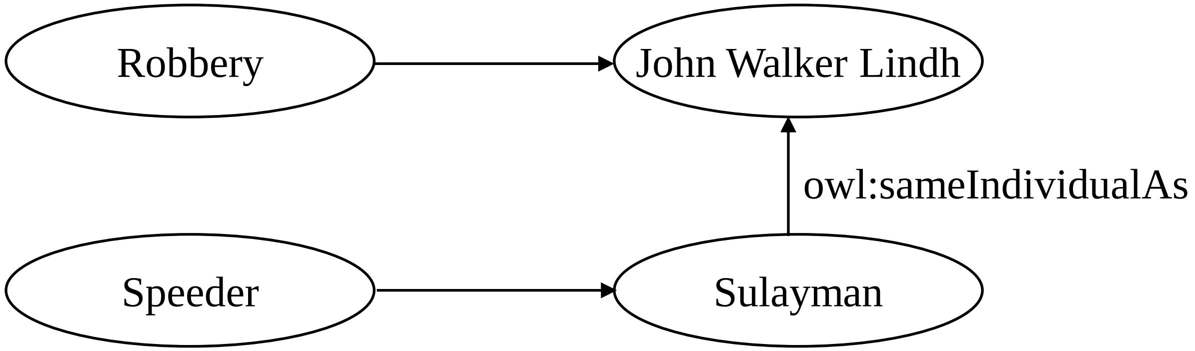
Later in the day a state trooper gives a person a ticket for speeding. The driver's license showed the name Sulayman. Here is the state trooper's report on the speeder:

```
<Speeder rdf:ID="report-2003-03-17-QWRP">
    <description>...</description>
    <driver>
        <Person rdf:about="http://www.person.org#Sulayman"/>
    </driver>
</Speeder>
```

Any Relationship between the Robber and the Speeder?

The Central Intelligence Agency (CIA) has a file on Sulayman:

```
<Person rdf:about="http://www.person.org#Sulayman">
   <owl:sameIndividualAs rdf:resource="http://www.person.org#John_Walker_Lindh"/>
</Person>
```



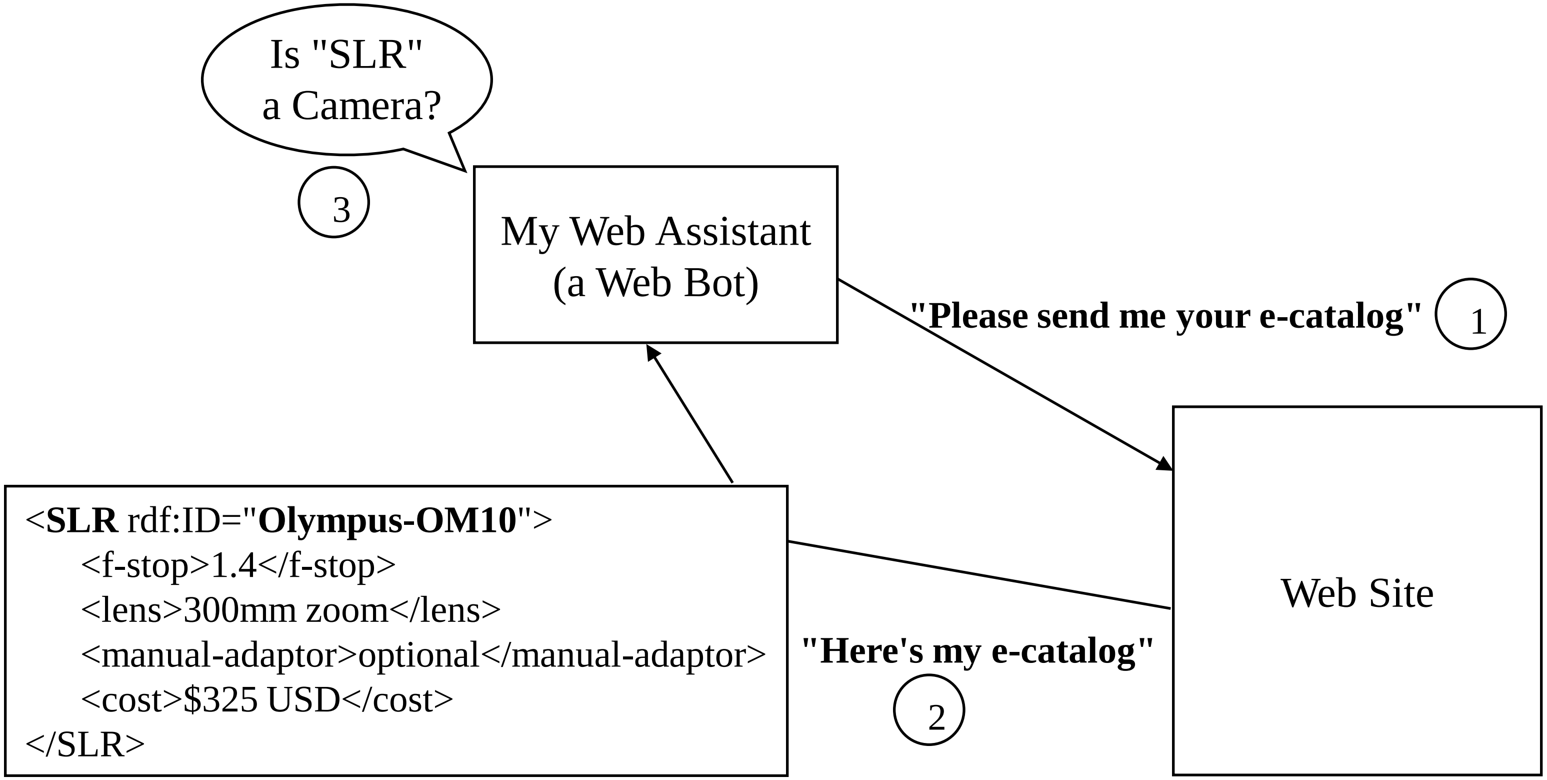
The local police, state troopers, and CIA share their information, thus enabling the following inference to be made:

Inference: The Robber and the Speeder are one and the same!

Lesson Learned

- OWL provides a property (owl:sameIndividualAs) for indicating that two resources (e.g., two people) are the same.

Example 2: Using a Web Bot to Purchase a Camera



Is "SLR" a Camera?

3

My Web Assistant (a Web Bot)

"Please send me your e-catalog"

1

Web Site

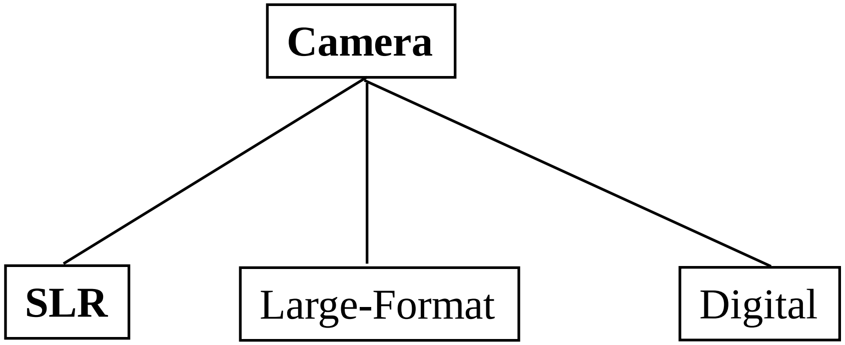
"Here's my e-catalog"

2

```
<SLR rdf:ID="Olympus-OM10">
     <f-stop>1.4</f-stop>
     <lens>300mm zoom</lens>
     <manual-adaptor>optional</manual-adaptor>
     <cost>$325 USD</cost>
</SLR>
```

* A Web Bot is a software program which crawls the Web looking for information.

Camera OWL Ontology



Camera

SLR

Large-Format

Digital

My Web Assistant program consults the Camera OWL Ontology. The Ontology shows how SLR is classified. The Ontology shows that SLR is a type (subclass) of Camera. Thus, my Web Assistant Bot dynamically realizes that:

Inference: The Olympus-OM10 SLR is a Camera!

Lesson Learned

- OWL provides elements to construct taxonomies (called class hierarchies). The taxonomies can be used to dynamically discover relationships!

Example 3: The Birthplace of King Kamahameha is …

Upon scanning the Web, three documents were found which contain information about King Kamahameha:

1

```
<Person rdf:about="http://www.person.org#King_Kamahameha">
   <birthplace rdf:about="http://www.states.org#Hawaii"/>
</Person>
```

2

```
<Person rdf:about="http://www.person.org#King_Kamahameha">
   <birthplace rdf:resource="http://www.history.org#Sandwich_Islands"/>
</Person>
```

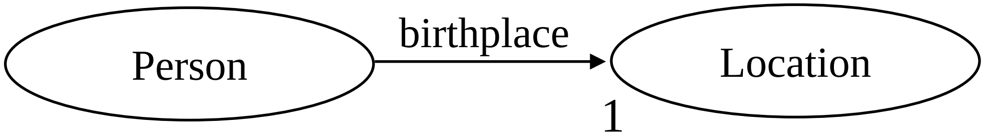
3

```
<Person rdf:about="http://www.person.org#King_Kamahameha">
   <birthplace rdf:resource="http://www.tourism.org#Aloha_State"/>
</Person>
```

Question: What is the birthplace of King Kamahameha?

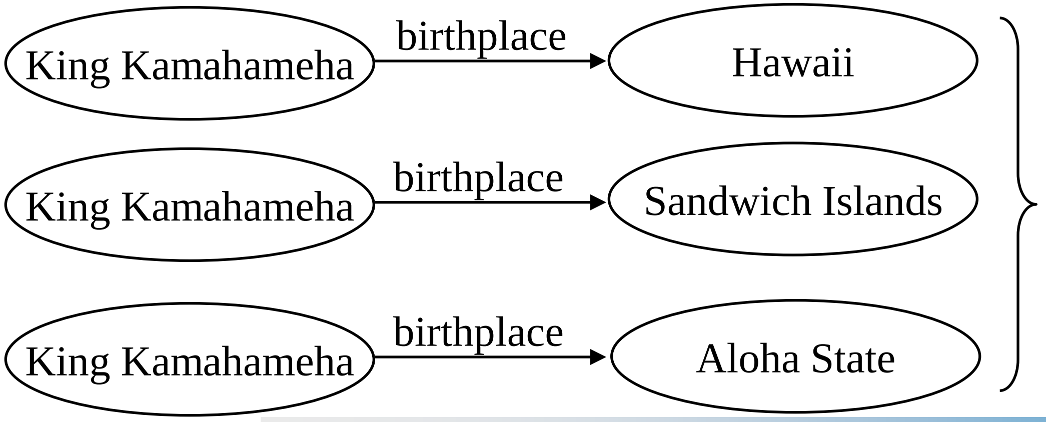
Answer: all three!

The Person OWL Ontology indicates that a Person has only one birthplace location:



Thus, the Person OWL Ontology enables this inference to be made:

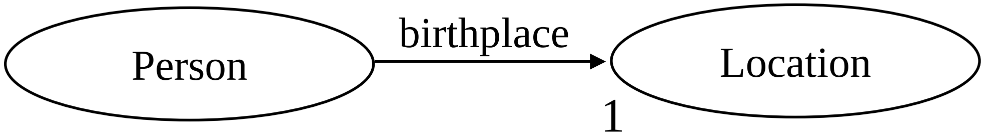
Inference: Hawaii, Sandwich Islands, and Aloha State all represent the same location!



They all represent the same location!

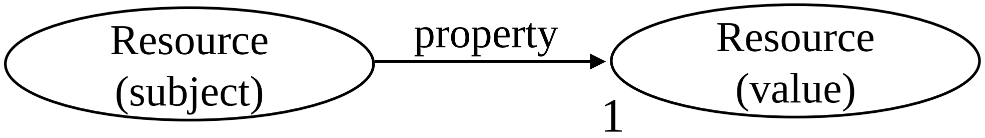
Lesson Learned

In the example we saw that the Person Ontology defined this relationship:



This is read as: "A person has exactly one birthplace location."

This example is a specific instance of a general capability in OWL to specify that a subject Resource has exactly one value:



We saw in the example that such information can be used to make inferences.

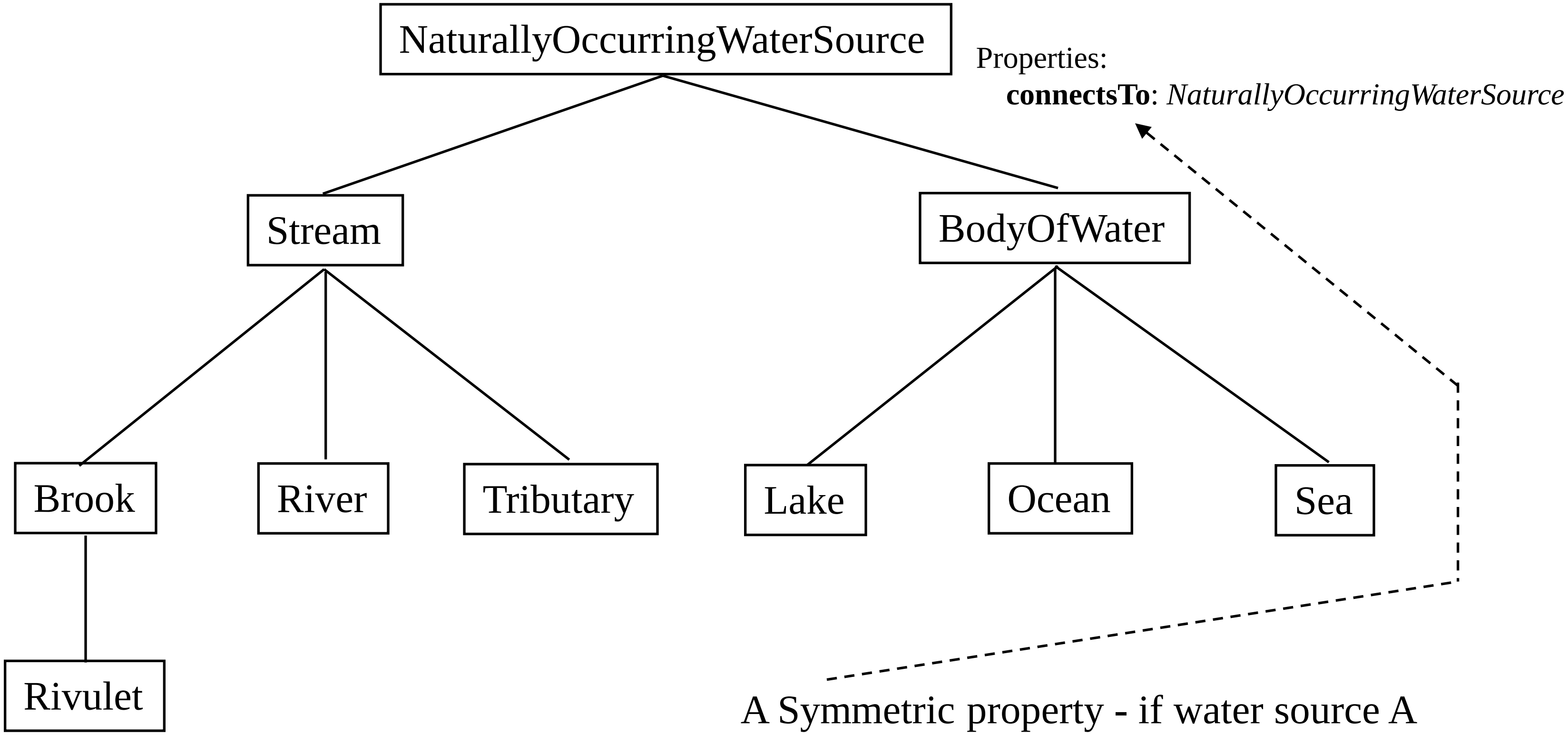
OWL Terminology: properties that relate a resource to exactly one other resource are said to have a *cardinality=1*.

Using OWL to Define Properties

Defining Property Characteristics

- RDF Schema provides three ways to characterize a property:
  - range: use this to indicate the range of values for a property.
  - domain: use this to associate a property with a class.
  - subPropertyOf: use this to specialize a property.
- Note: OWL documents also use rdfs:range, rdfs:domain, and rdfs:subPropertyOf.
- On the following slides we show the additional ways that OWL provides to characterize properties.
  - We will see that these additional property characteristics enable greater inferencing.

Symmetric Properties



Properties:
connectsTo: *NaturallyOccurringWaterSource*

A Symmetric property - if water source A connectsTo water source B then water source B connects to water source A.

Symmetric Property

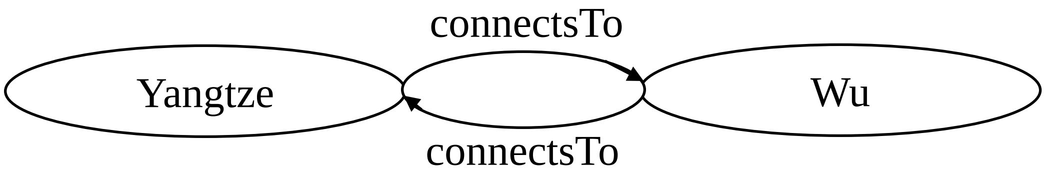
Assume that connectsTo has been *defined, in an OWL document, to be a Symmetric property*:

```
<?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#">
    <connectsTo>
        <River rdf:about="http://www.china.org/rivers#Wu"/>
    </connectsTo>
</River>
```

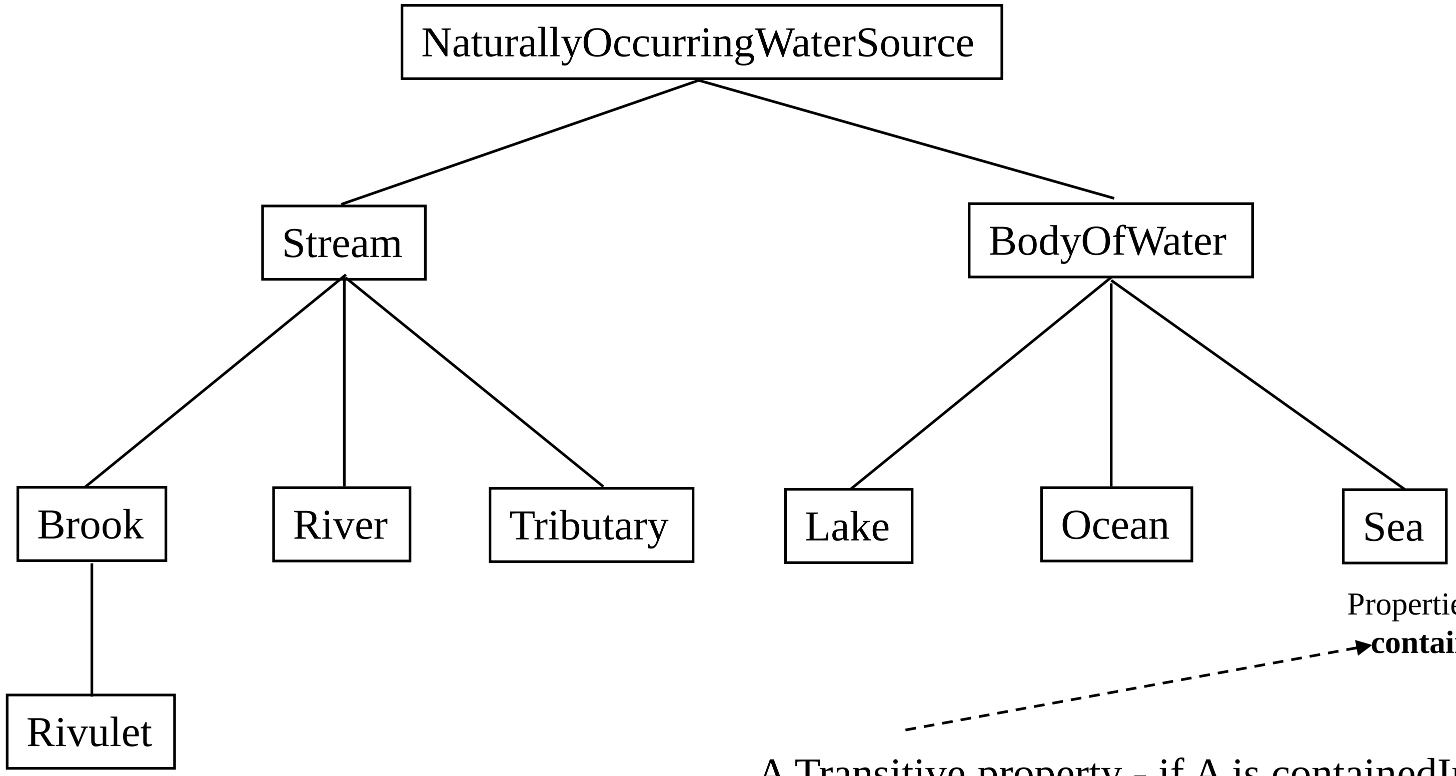
Yangtze.rdf

Since connectsTo has been defined to be a Symmetric property we can infer that:

The Wu River connectsTo the Yangtze River.



Transitive Properties



NaturallyOccurringWaterSource

Stream

BodyOfWater

Brook

River

Tributary

Lake

Ocean

Sea

Rivulet

Properties:
containedIn: *BodyOfWater*

A Transitive property - if A is containedIn B, and B is containedIn C then A is containedIn C.

Transitive Property

Suppose that you retrieve these two documents from two different Web sites. One describes the EastChinaSea and the other describes the ChinaSea:

```
<?xml version="1.0"?>
<Sea rdf:ID="EastChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <containedIn>
         <Sea rdf:about="http://www.china.gov#ChinaSea"/>
    </containedIn>
</Sea>
```

EastChinaSea.rdf

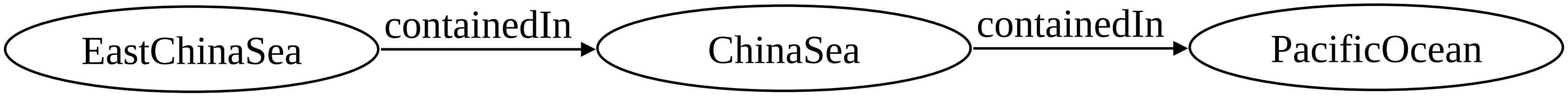
```
<?xml version="1.0"?>
<Sea rdf:about="http://www.china.gov#ChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <containedIn>
         <Ocean rdf:about="http://www.geodesy.org#PacificOcean"/>
    </containedIn>
</Sea>
```

ChinaSea.rdf

If containedIn is defined to be a Transitive property then we can infer that:

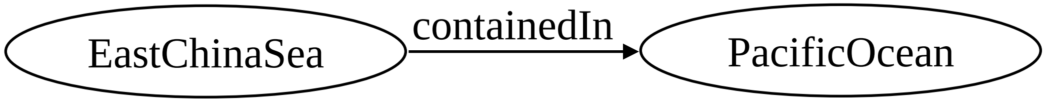
The EastChinaSea is containedIn the PacificOcean.

Transitive Property



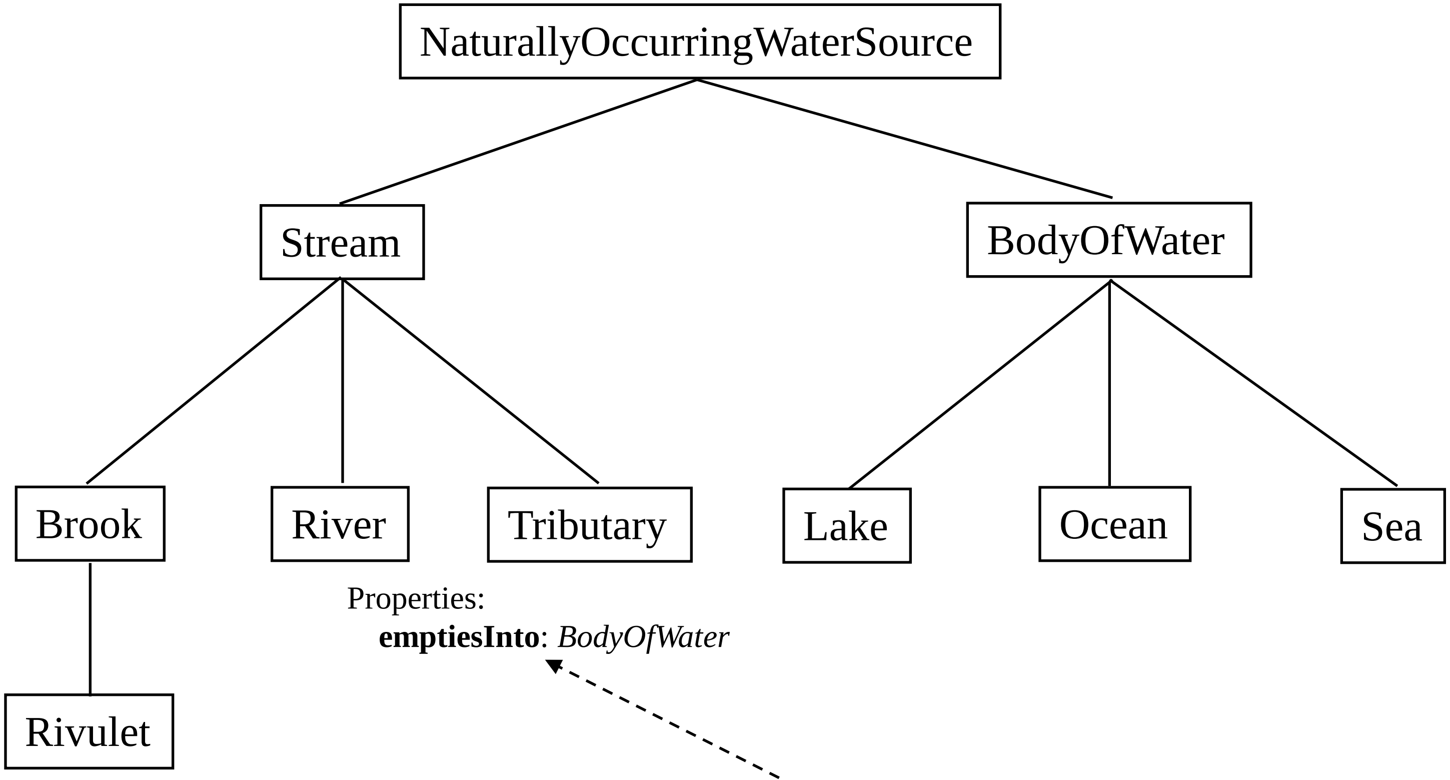
EastChinaSea containedIn ChinaSea containedIn PacificOcean

If containedIn is defined to be Transitive, we can infer that:



EastChinaSea containedIn PacificOcean

Functional Properties



Properties:
emptiesInto: *BodyOfWater*

A Functional property - for each instance there is at most one value for the property.

Functional Property

Suppose that there are two independent documents describing the Yangtze River:

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

Yangtze-doc1.rdf

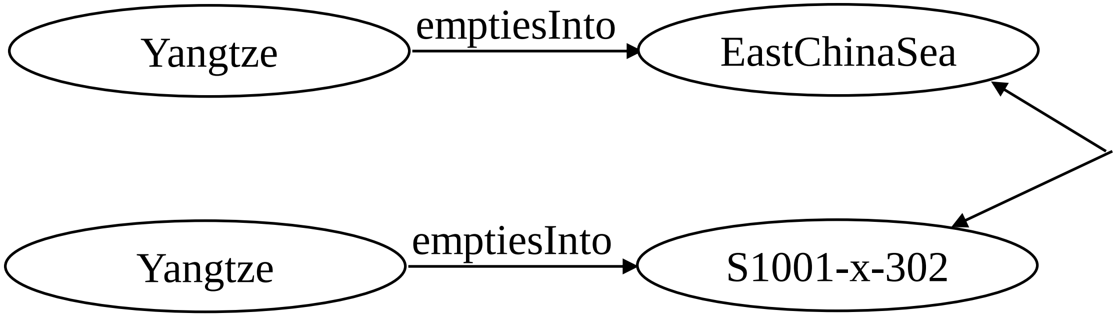
```
<?xml version="1.0"?>
<River rdf:about="http://www.china.org/rivers#Yangtze"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <emptiesInto rdf:resource="http://www.national-geographic.org#S1001-x-302"/>
</River>
```

Yangtze-doc2.rdf

If emptiesInto is defined to be functional then we can infer that:

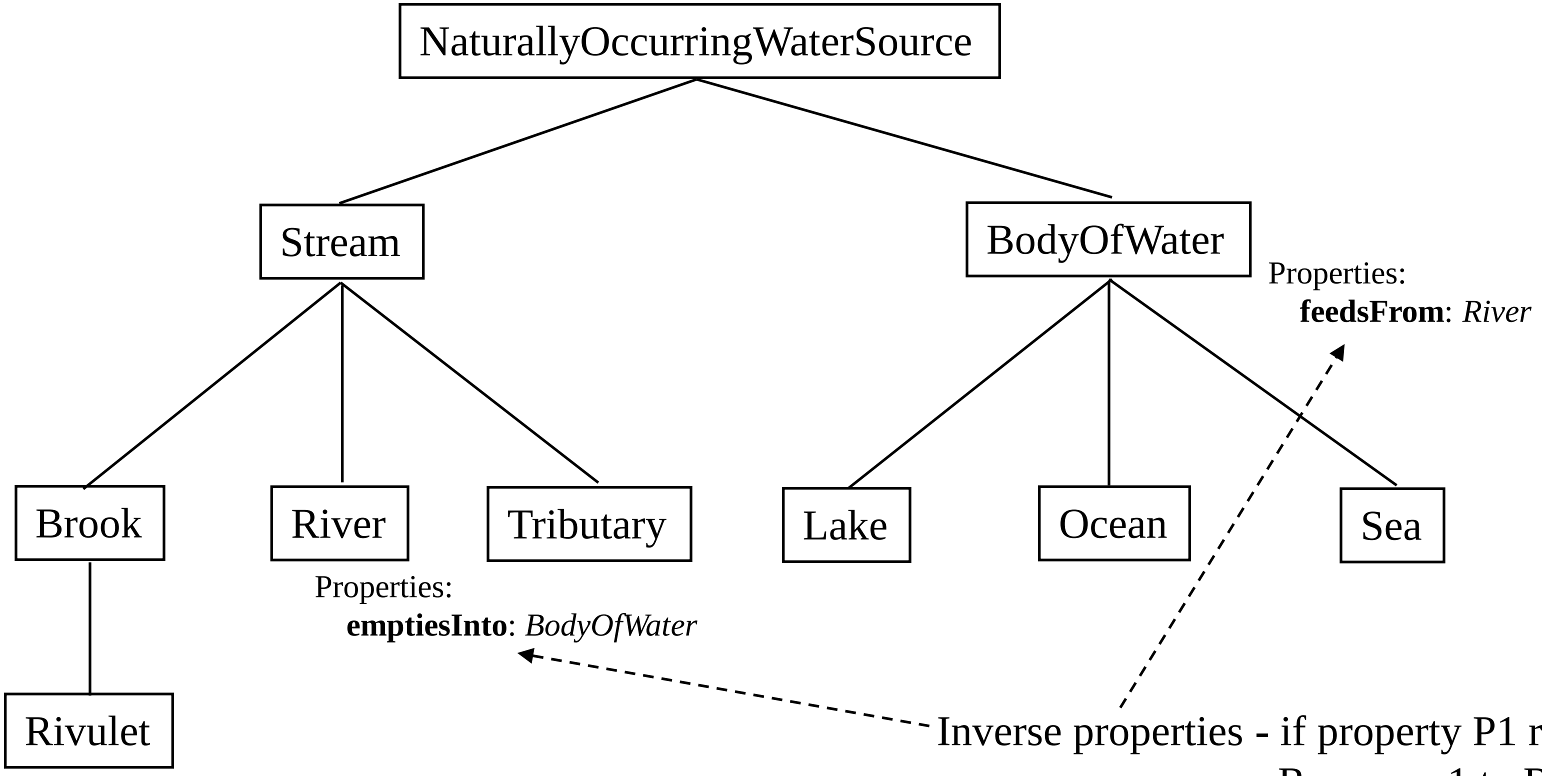
http://www.china.org/geography#**EastChinaSea** = http://www.national-geographic.org#**S1001-x-302**

Functional Property (cont.)



If emptiesInto has been defined to be Functional then we can infer that these two values must refer to the same thing.

Inverse Properties



NaturallyOccurringWaterSource

Stream

BodyOfWater

Properties:
feedsFrom: *River*

Brook

River

Tributary

Lake

Ocean

Sea

Properties:
emptiesInto: *BodyOfWater*

Rivulet

Inverse properties - if property P1 relates Resource 1 to Resource 2, then its Inverse property relates Resource 2 to Resource 1.

Inverse Properties

Consider this 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#">
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>
```

Yangtze.rdf

The above states that:

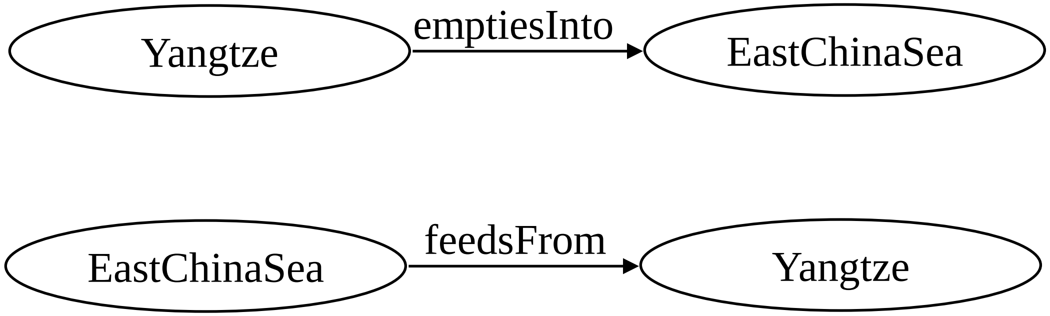
The Yangtze emptiesInto the EastChinaSea.

If emptiesInto and feedsFrom are defined to be Inverse properties then we can infer that:

The EastChinaSea feedsFrom the Yangtze.

emptiesInto <---> feedsFrom (Inverse Properties)

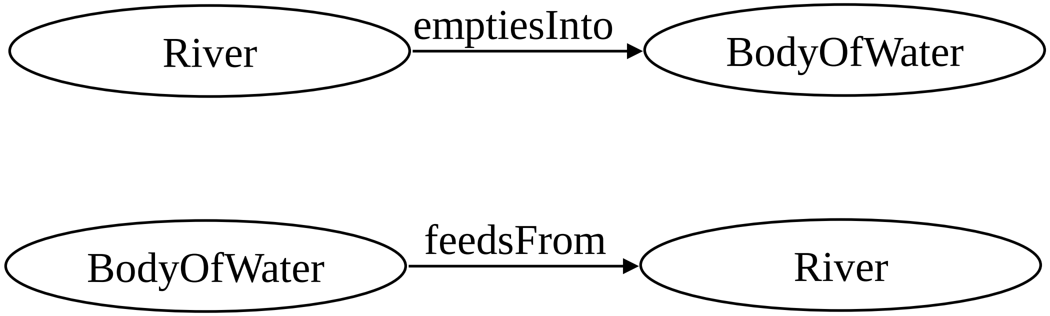
A specific instance:



Yangtze emptiesInto EastChinaSea

EastChinaSea feedsFrom Yangtze

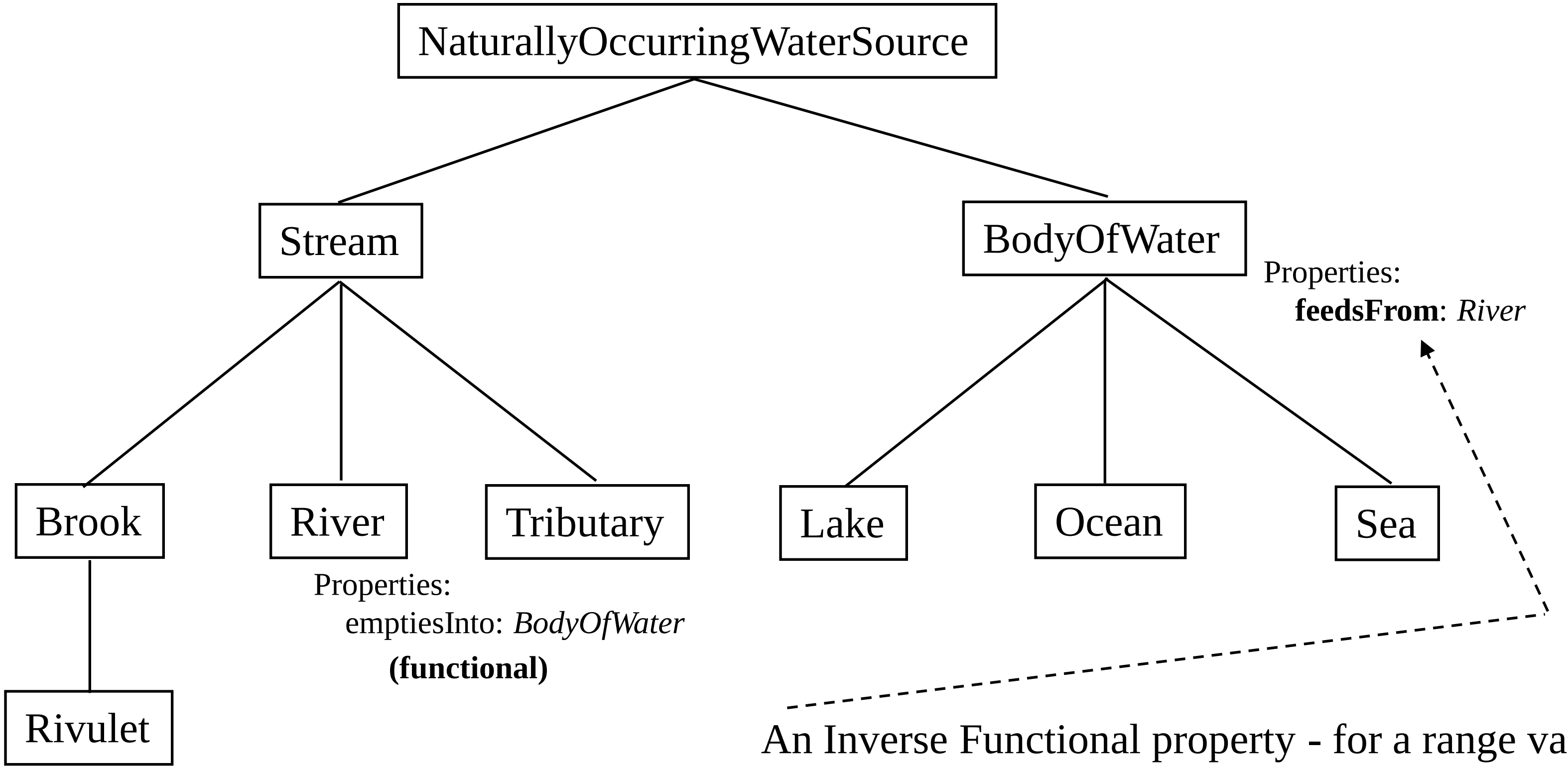
The general case:



River emptiesInto BodyOfWater

BodyOfWater feedsFrom River

Inverse Functional Properties



An Inverse Functional property - for a range value the domain is unique.

Inverse Functional Property

These two independent documents discuss "feeding from" the Yangtze:

```
<?xml version="1.0"?>
<Sea  rdf:ID="EastChinaSea"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <feedsFrom>
        <River rdf:about="http://www.china.org/rivers#Yangtze"/>
    </feedsFrom>
</Sea>
```

EastChinaSea.rdf

```
<?xml version="1.0"?>
<Sea rdf:ID="S1001-x-302"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <feedsFrom>
        <River rdf:about="http://www.china.org/rivers#Yangtze"/>
    </feedsFrom>
</Sea>
```

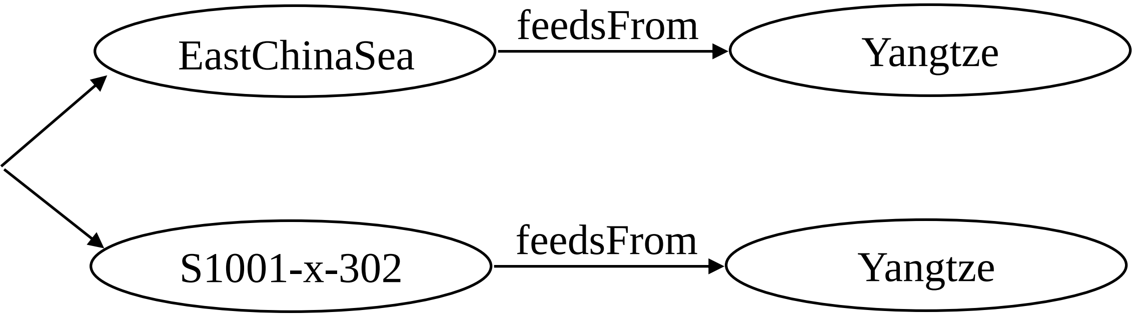
S1001-x-302.rdf

Inverse Functional Property (cont.)

If feedsFrom has been defined to be InverseFunctional then we can infer that:

EastChinaSea = S1001-x-302.

If feedsFrom has been defined to be Inverse Functional then we can infer that these two Resources must refer to the same thing.



Confused about the difference between a Functional property and an Inverse Functional property

Consider the birthdate property in this document:

```
<Person rdf:ID="JohnDoe">
    <birthdate>March 24, 1971</birthdate>
</Person>
```

A Person has a single birthdate. Therefore, birthdate is a Functional Property.

Question: Is birthdate an Inverse Functional Property? Answer: No. If birthdate was an Inverse Functional property then only one person could have a birthdate. There are many people that have the same birthdate. Thus, birthdate is not an Inverse Functional property.

Example of an Inverse Functional Property

Consider the email property in this document:

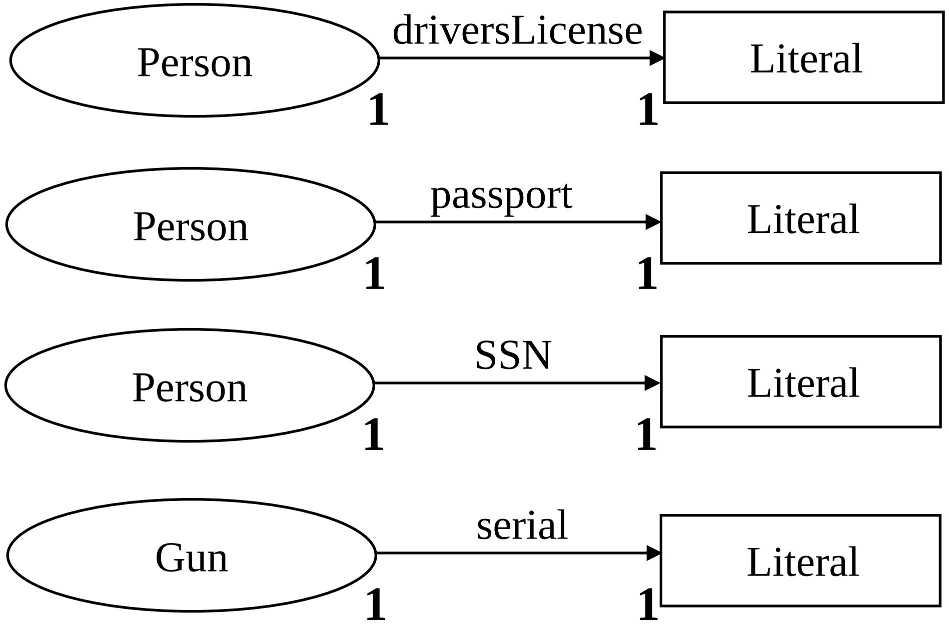
```
<Person rdf:ID="SallyJane">
    <email>sally-jane@yahoo.com</email>
</Person>
```

An email address applies to only one person. Therefore, email is an Inverse Functional Property.

Question: Is email a Functional Property? Answer: No. If email was a Functional Property then a person could have only one email. Many people have multiple email addresses. Thus, email is not a Functional Property.

Examples of properties that are both Functional and Inverse Functional

Most ID-like properties are both Functional and Inverse Functional, e.g., driver's license, passport number, SSN, serial number, etc.



Time for Syntax!

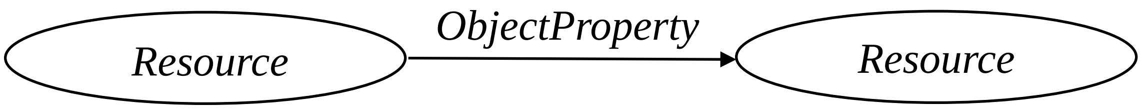
- On the previous slides we have seen the different ways that OWL provides to characterize properties.
- Now let's look at the OWL syntax for expressing these property characteristics.

Defining Properties in OWL

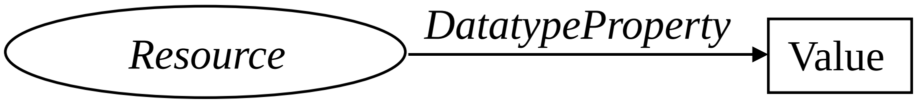
- Recall that with RDF Schema the rdf:Property was used for both:
  - relating a Resource to another Resource
    - Example: The emptiesInto property relates a River to a BodyOfWater.
  - relating a Resource to an rdfs:Literal or a datatype
    - Example: The length property relates a River to a xsd:nonNegativeInteger.
- OWL decided that these are two classes of properties, and thus each should have its own class:
  - owl:ObjectProperty is used to relate a Resource to another Resource
  - owl:DatatypeProperty is used to relate a Resource to an rdfs:Literal or an XML Schema built-in datatype

ObjectProperty vs. DatatypeProperty

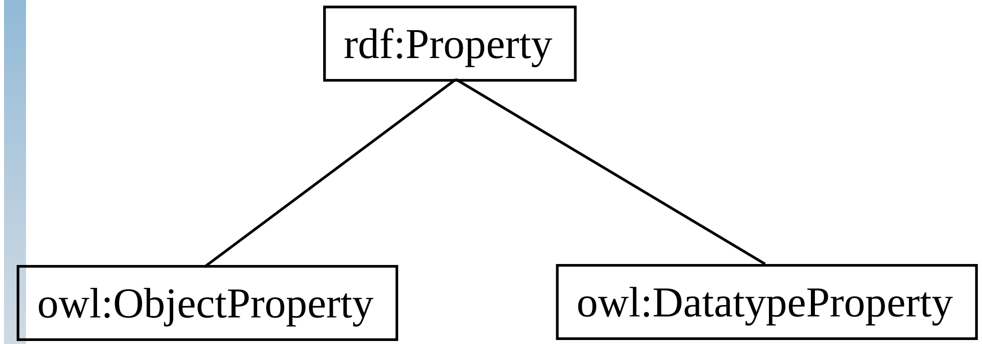
An ObjectProperty relates one Resource to another Resource:



A DatatypeProperty relates a Resource to a Literal or an XML Schema datatype:



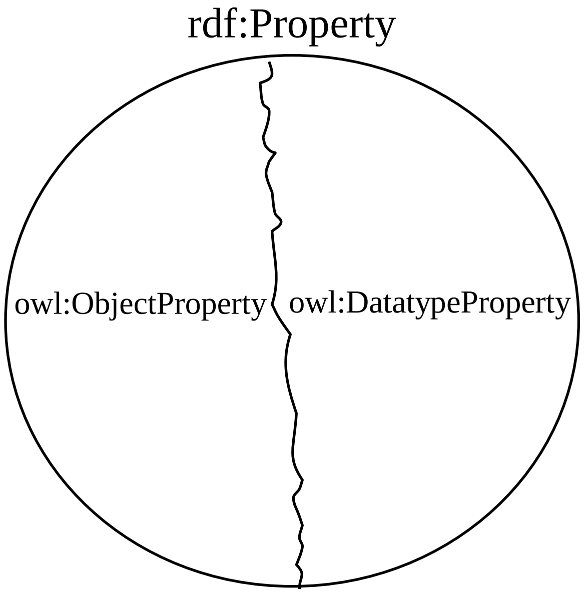
owl:ObjectProperty and owl:DatatypeProperty are subclasses of rdf:Property



rdf:Property

owl:ObjectProperty

owl:DatatypeProperty



rdf:Property

owl:ObjectProperty

owl:DatatypeProperty

Defining Properties in OWL vs. RDF Schema

RDFS

```
<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/2001/XMLSchema#nonNegativeInteger"/>
</rdf:Property>
```

OWL

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

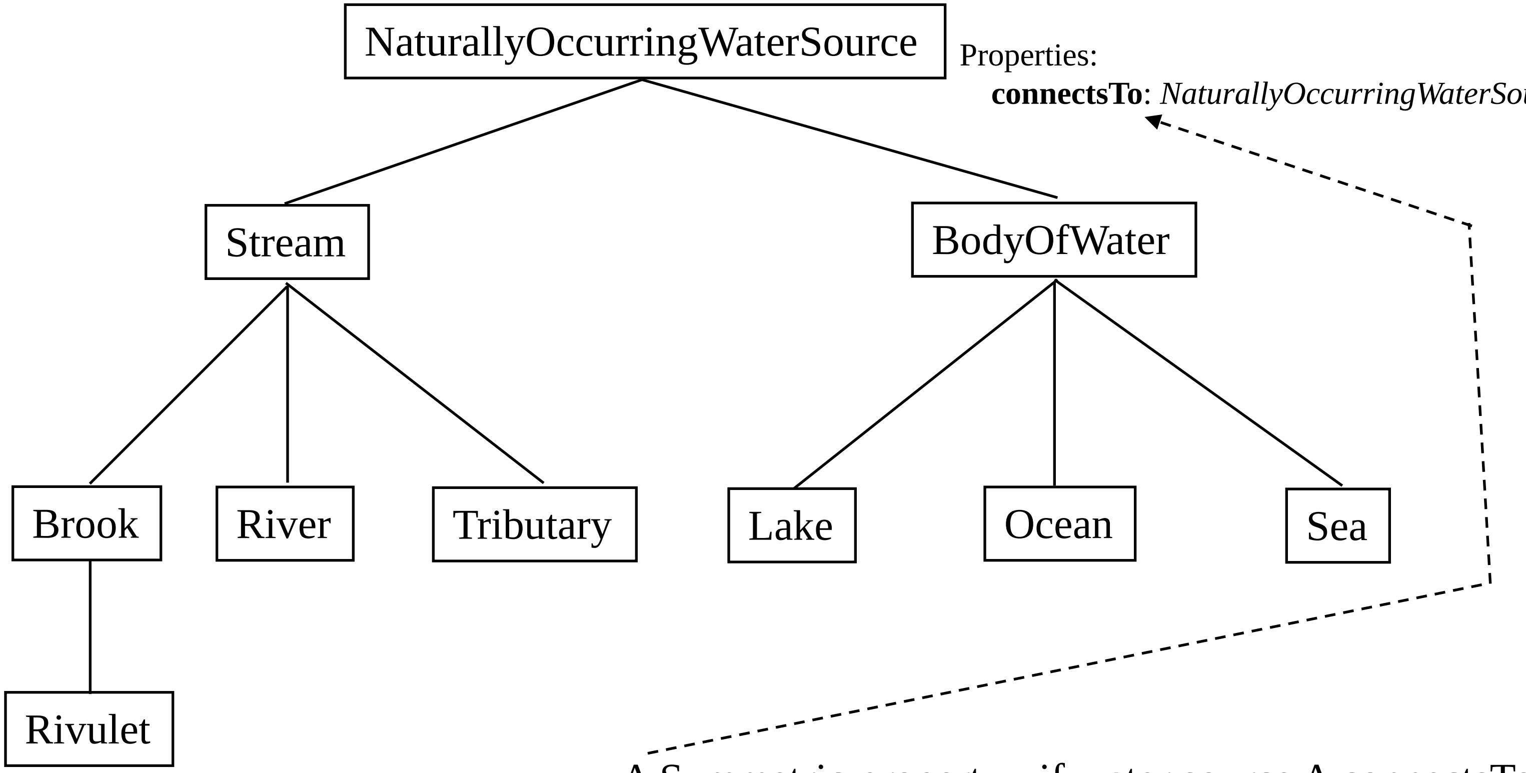
```
<owl:DatatypeProperty rdf:ID="length">
        <rdfs:domain rdf:resource="#River"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>
```

The OWL Namespace

```
<?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#"
         xmlns:owl="http://www.w3.org/2002/07/owl#">

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

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Defining Symmetric Properties



Properties:
connectsTo: *NaturallyOccurringWaterSource*

A Symmetric property - if water source A connectsTo water source B then water source B connects to water source A.

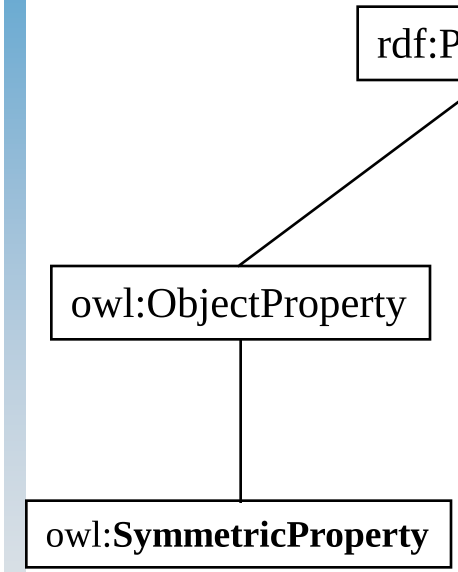
Syntax for indicating that a property is Symmetric

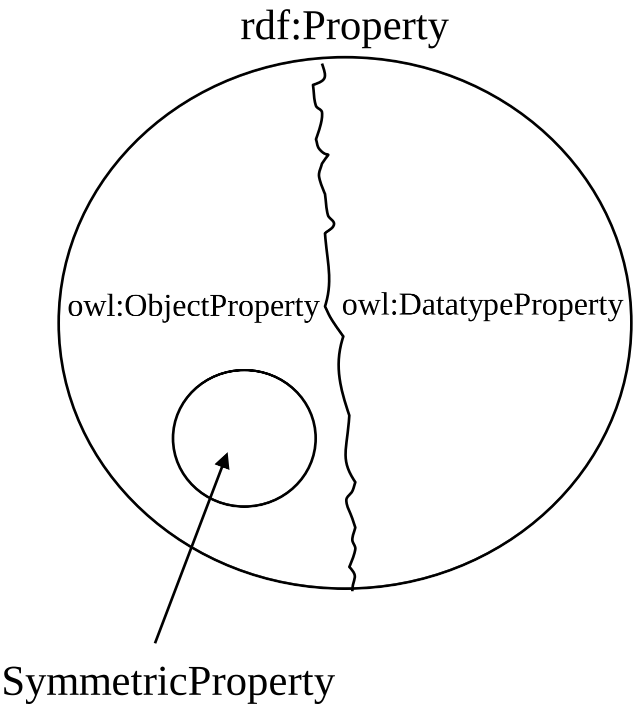
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
              xmlns:owl="http://www.w3.org/2002/07/owl#"
              xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:ObjectProperty rdf:ID="connectsTo">
         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
         <rdfs:domain rdf:resource="#NaturallyOccurringWaterSource"/>
         <rdfs:range rdf:resource="#NaturallyOccurringWaterSource"/>
    </owl:ObjectProperty>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Read this as: "connectsTo is an ObjectProperty. Specifically, it is a Symmetric Object Property."

owl:SymmetricProperty is a subclass of owl:ObjectProperty





Consequently, the range of a SymmetricProperty can only be a Resource, i.e., the range cannot be a Literal or a datatype.

Equivalent!

Read this as: "connectsTo is an ObjectProperty. Specifically, it is a Symmetric Object Property."

1

```
<owl:ObjectProperty rdf:ID="connectsTo">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
        <rdfs:domain rdf:resource="#NaturallyOccurringWaterSource"/>
        <rdfs:range rdf:resource="#NaturallyOccurringWaterSource"/>
</owl:ObjectProperty>
```

Read this as: "connectsTo is a SymmetricProperty."

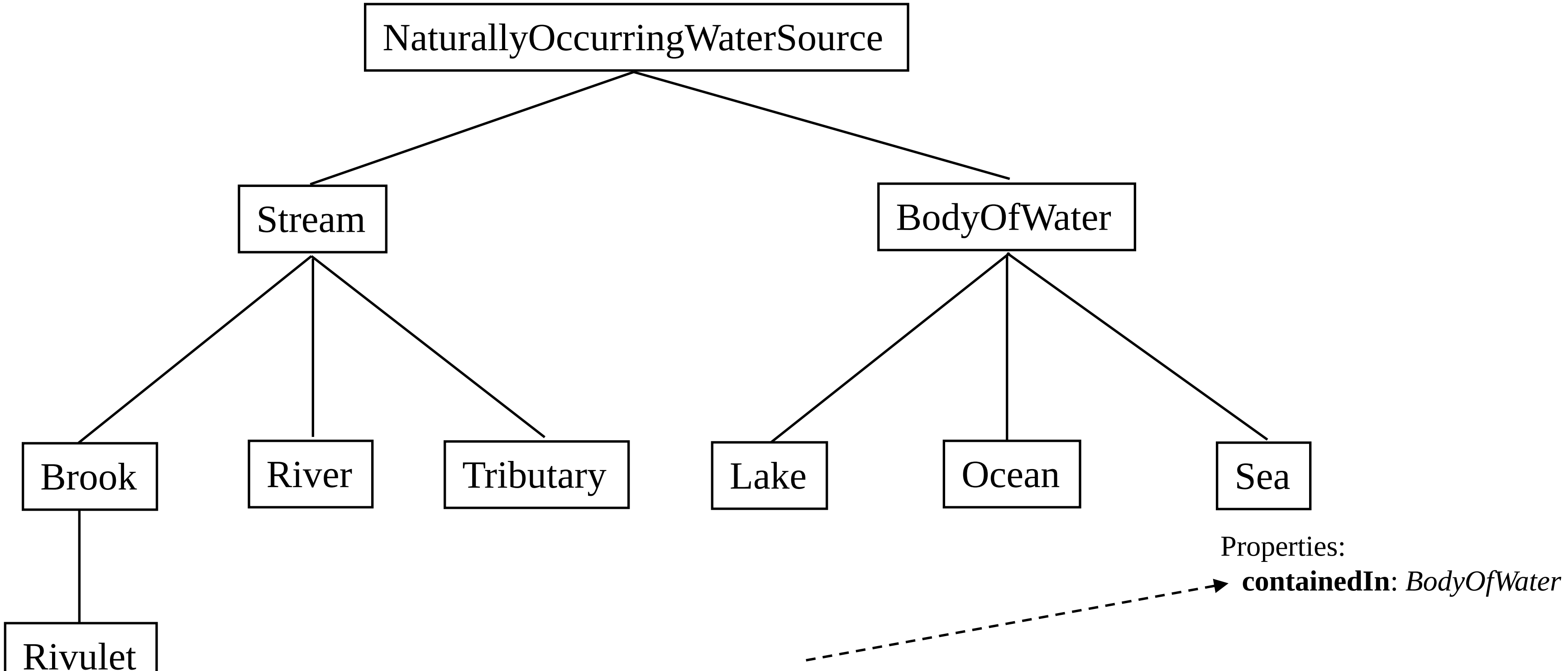
2

```
<owl:SymmetricProperty rdf:ID="connectsTo">
        <rdfs:domain rdf:resource="#NaturallyOccurringWaterSource"/>
        <rdfs:range rdf:resource="#NaturallyOccurringWaterSource"/>
</owl:SymmetricProperty>
```

Question: Why would you ever use the first form? The second form seems a lot more straightforward. Right?

Answer: In this example, you are correct, the second form is more straightforward. However, you will see in a moment that we can define a property to be of several types, e.g., Symmetric **and** Functional. In that case it may be more straightforward to use the first form (and use multiple rdf:type elements).

Defining Transitive Properties



NaturallyOccurringWaterSource

Stream

BodyOfWater

Brook

River

Tributary

Lake

Ocean

Sea

Rivulet

Properties:
containedIn: *BodyOfWater*

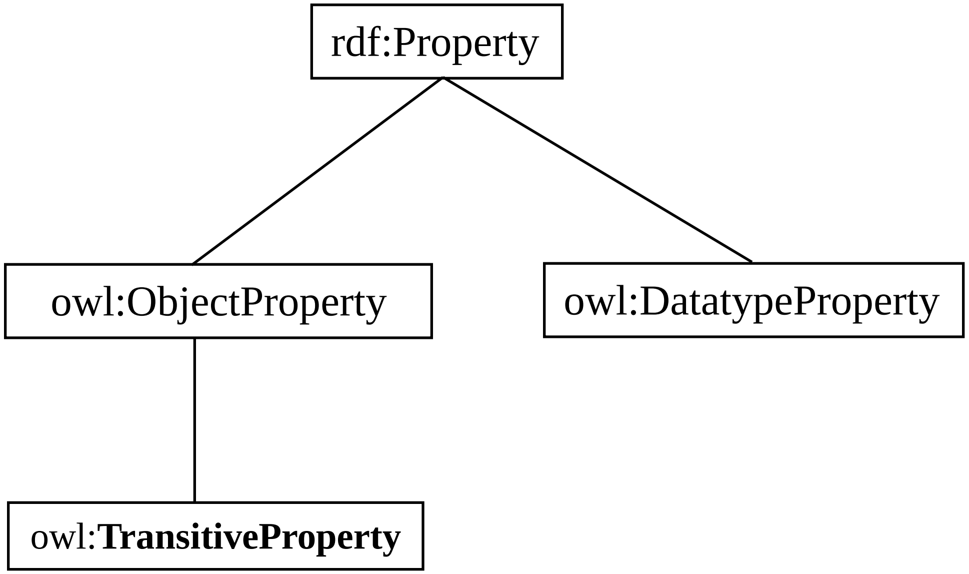
A Transitive property - if A is containedIn B, and B is containedIn C then A is containedIn C.

Syntax for indicating that a property is Transitive

```
<?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#"
              xmlns:owl="http://www.w3.org/2002/07/owl#"
              xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:ObjectProperty rdf:ID="containedIn">
         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
         <rdfs:domain rdf:resource="#Sea"/>
         <rdfs:range rdf:resource="#BodyOfWater"/>
    </owl:ObjectProperty>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

owl:TransitiveProperty is a subclass of owl:ObjectProperty

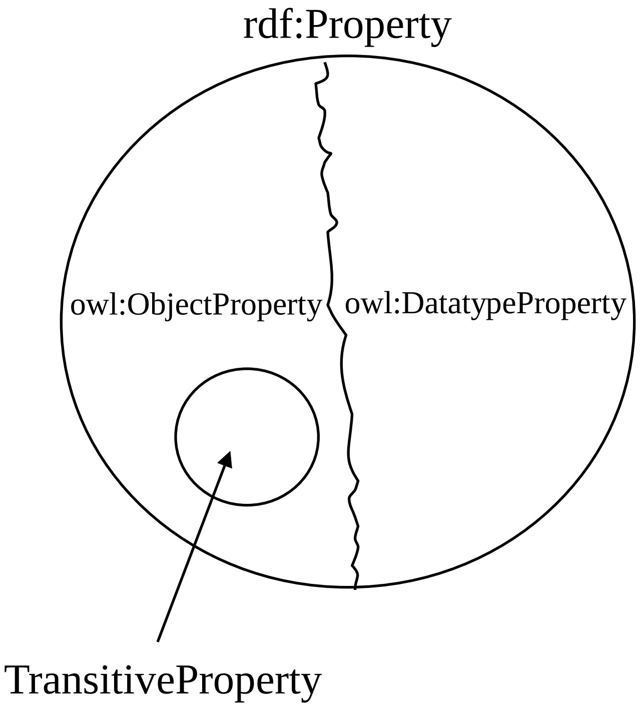


rdf:Property

owl:ObjectProperty

owl:DatatypeProperty

owl:**TransitiveProperty**



rdf:Property

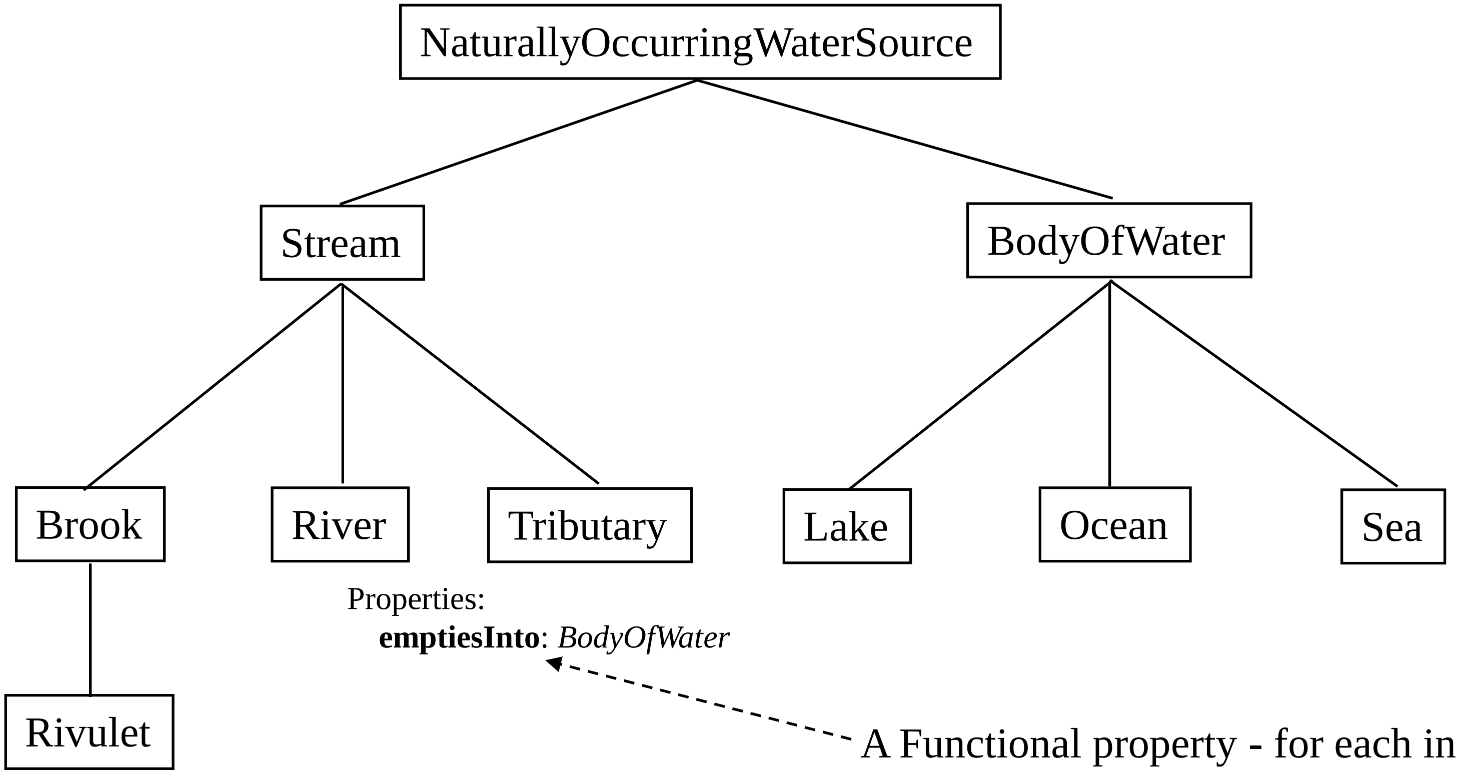
owl:ObjectProperty

owl:DatatypeProperty

TransitiveProperty

Consequently, the range of a TransitiveProperty can only be a Resource, i.e., the range cannot be a Literal or a datatype.

Defining Functional Properties



NaturallyOccurringWaterSource

Stream

BodyOfWater

Brook

River

Tributary

Lake

Ocean

Sea

Rivulet

Properties:
emptiesInto: *BodyOfWater*

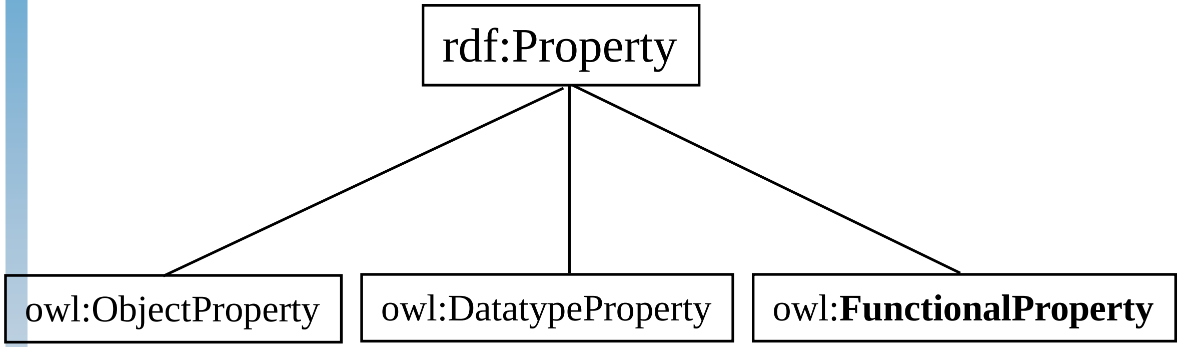
A Functional property - for each instance there is at most one value for the property.

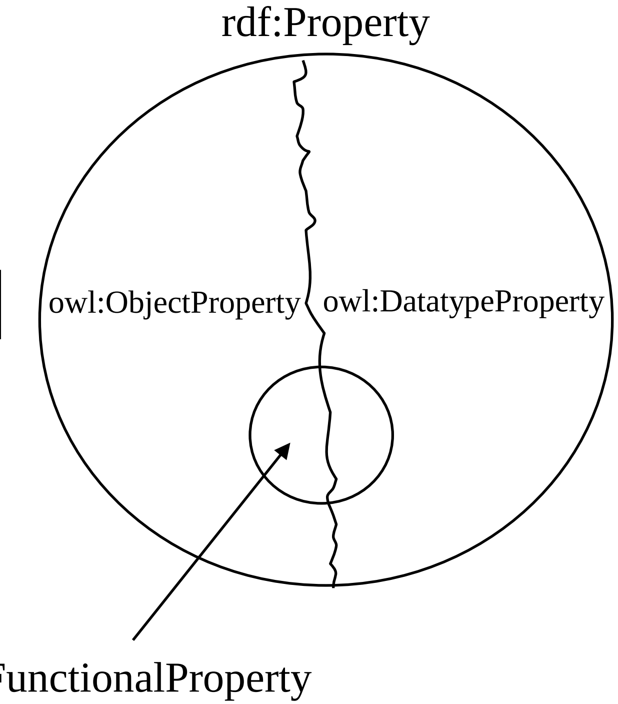
Syntax for indicating that a property is Functional

```
<?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#"
               xmlns:owl="http://www.w3.org/2002/07/owl#"
               xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:ObjectProperty rdf:ID="emptiesInto">
         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
         <rdfs:domain rdf:resource="#River"/>
         <rdfs:range rdf:resource="BodyOfWater"/>
    </owl:ObjectProperty>


   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

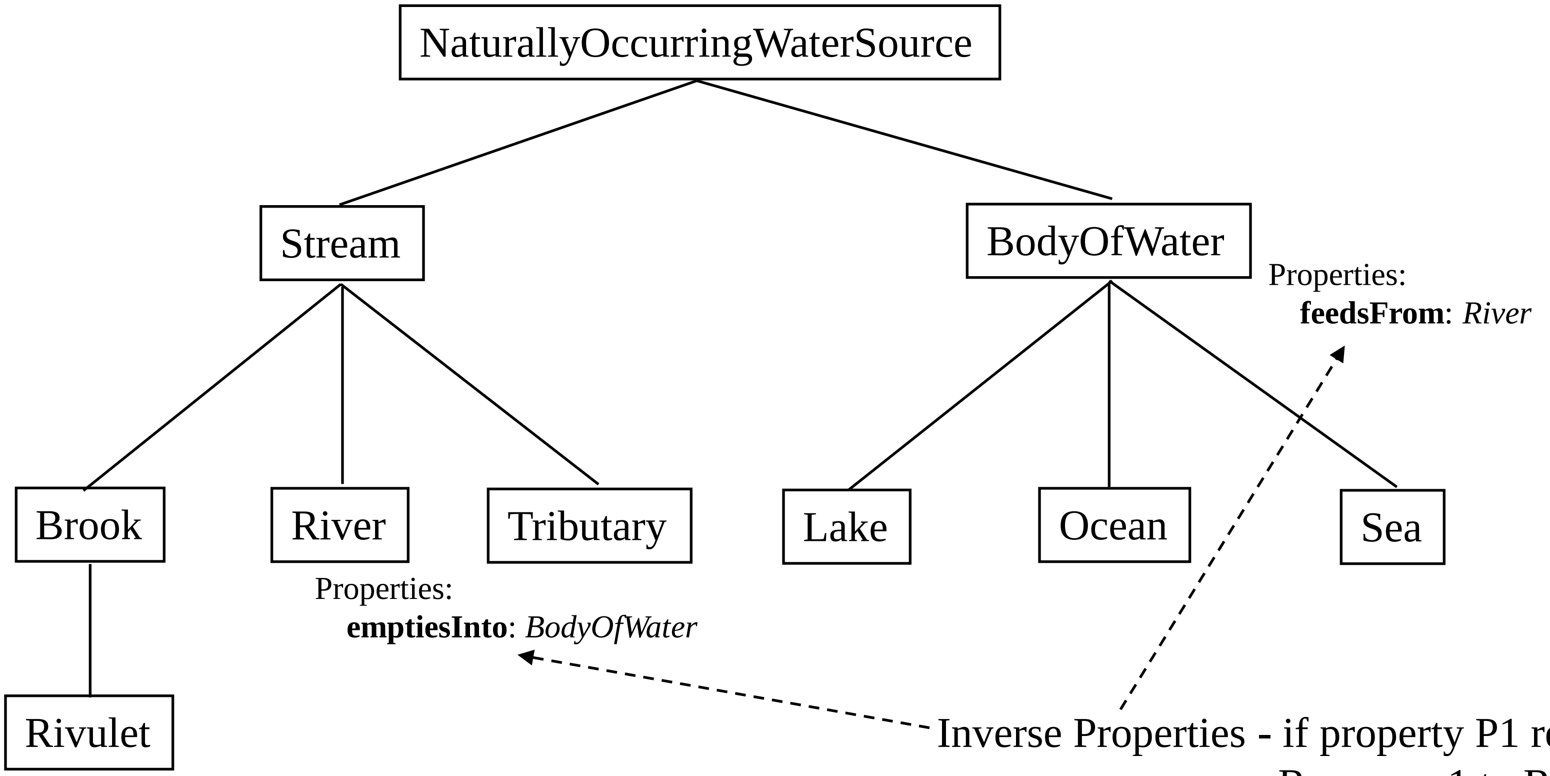
owl:FunctionalProperty is a subclass of rdf:Property





Consequently, the range of a FunctionalProperty can be either a Resource or a Literal or a datatype.

Defining Inverse Properties



NaturallyOccurringWaterSource

Stream

BodyOfWater

Properties:
feedsFrom: *River*

Brook

River

Tributary

Lake

Ocean

Sea

Rivulet

Properties:
emptiesInto: *BodyOfWater*

Inverse Properties - if property P1 relates Resource 1 to Resource 2, then its inverse property relates Resource 2 to Resource 1.

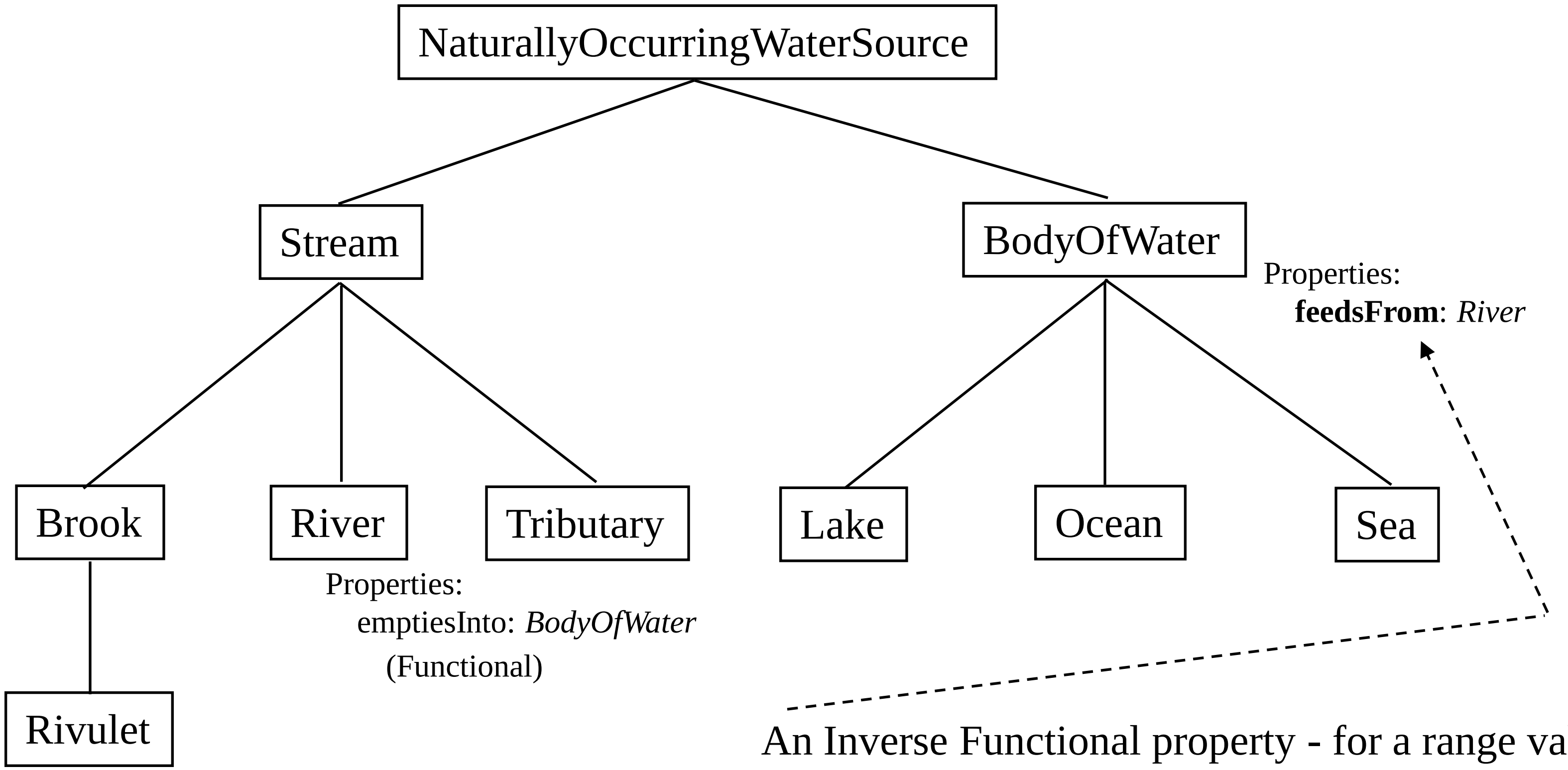
Syntax for indicating that a property is the inverse of another property

```
<?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#"
              xmlns:owl="http://www.w3.org/2002/07/owl#"
              xml:base="http://www.geodesy.org/water/naturally-occurring">

  <owl:ObjectProperty rdf:ID="emptiesInto">
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
     <rdfs:domain rdf:resource="#River"/>
     <rdfs:range rdf:resource="#BodyOfWater"/>
  </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="feedsFrom">
     <owl:inverseOf rdf:resource="#emptiesInto"/>
     <rdfs:domain rdf:resource="#BodyOfWater"/>
     <rdfs:range rdf:resource="#River"/>
  </owl:ObjectProperty>

  ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Notice that the values for domain and range are flipped from that in emptiesInto.

Defining Inverse Functional Properties



NaturallyOccurringWaterSource

Stream

BodyOfWater

Properties:
feedsFrom: *River*

Brook

River

Tributary

Lake

Ocean

Sea

Properties:
emptiesInto: *BodyOfWater*
(Functional)

Rivulet

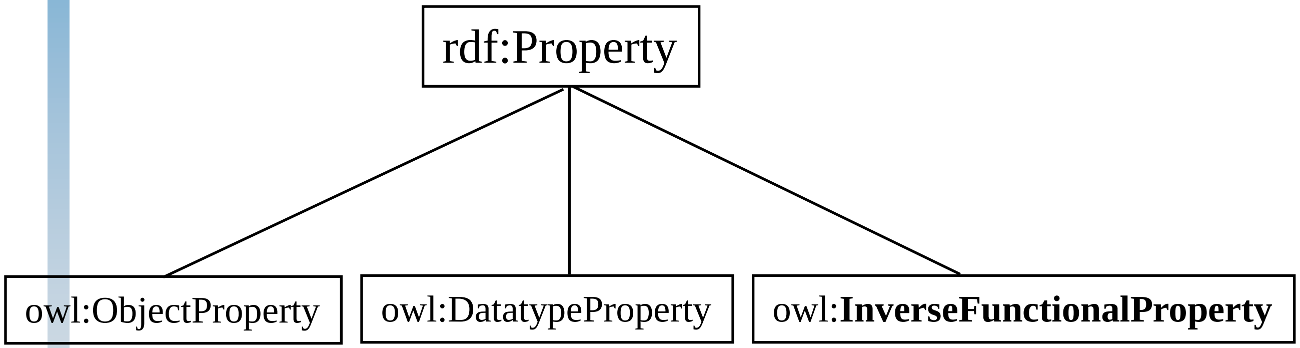
An Inverse Functional property - for a range value the domain is unique.

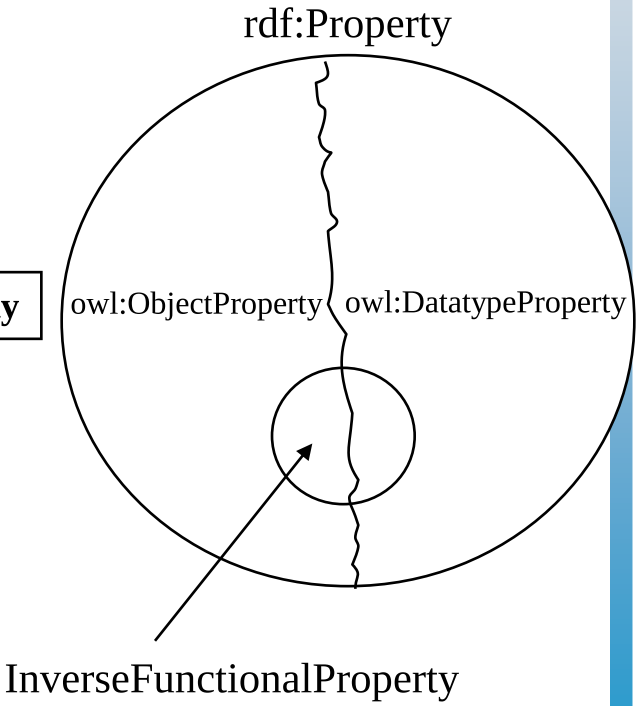
Syntax for indicating that a property is Inverse Functional

```
<?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#"
              xmlns:owl="http://www.w3.org/2002/07/owl#"
              xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:ObjectProperty rdf:ID="emptiesInto">
         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
         <rdfs:domain rdf:resource="#River"/>
         <rdfs:range rdf:resource="#BodyOfWater"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:ID="feedsFrom">
         <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
         <owl:inverseOf rdf:resource="#emptiesInto"/>
         <rdfs:domain rdf:resource="#BodyOfWater"/>
         <rdfs:range rdf:resource="#River"/>
    </owl:ObjectProperty>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

owl:InverseFunctionalProperty is a subclass of rdf:Property



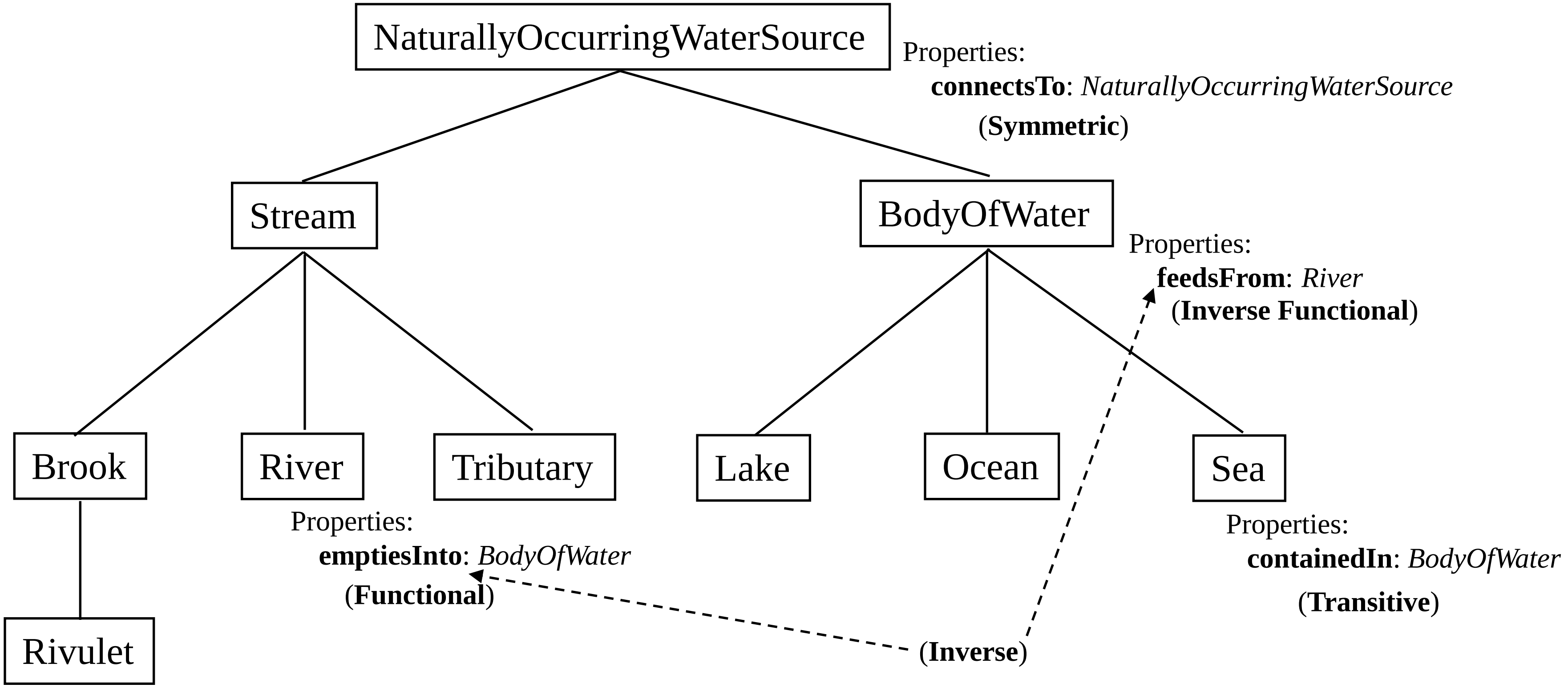


Consequently, the range of an InverseFunctionalProperty can be either a Resource or a Literal or a datatype.

Summary of the different ways to characterize properties

- In the preceding slides we have seen the different ways of characterizing properties. We saw that a property may be defined to be:
  - A Symmetric property.
  - A Transitive property.
  - A Functional property.
  - The Inverse of another property.
  - An Inverse Functional property.

Summary of Properties for the Water Taxonomy



NaturallyOccurringWaterSource

Properties:
connectsTo: *NaturallyOccurringWaterSource*
(**Symmetric**)

Stream

BodyOfWater

Properties:
feedsFrom: *River*
(**Inverse Functional**)

Brook

River

Tributary

Lake

Ocean

Sea

Rivulet

Properties:
emptiesInto: *BodyOfWater*
(**Functional**)

(**Inverse**)

Properties:
containedIn: *BodyOfWater*
(**Transitive**)

Inferences we can make now that we have characterized the properties

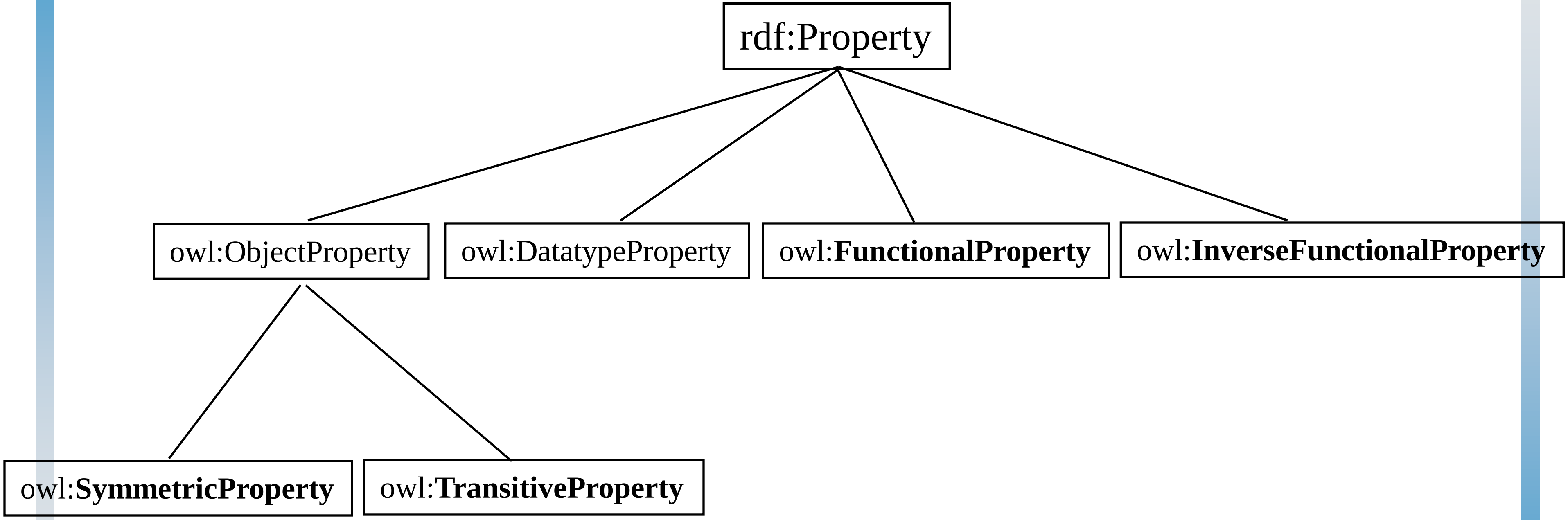
```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>
    <connectsTo rdf:resource="http://www.china.org/rivers#Wu"/>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

Yangtze.rdf

We can infer that:

1. The EastChinaSea feedsFrom the Yangtze. (Since emptiesInto is the inverse of feedsFrom)
2. The Wu connectsTo the Yangtze. (Since connectsTo is symmetric)
3. The EastChinaSea is a BodyOfWater. (Since the range of emptiesInto is a BodyOfWater.
4. The Wu is a NaturallyOccurringWaterSource. (Since the range of connectsTo is NaturallyOccurringWaterSource)

Hierarchy of the property classes


rdf:Property
owl:ObjectProperty
owl:DatatypeProperty
owl:**FunctionalProperty**
owl:**InverseFunctionalProperty**
owl:**SymmetricProperty**
owl:**TransitiveProperty**

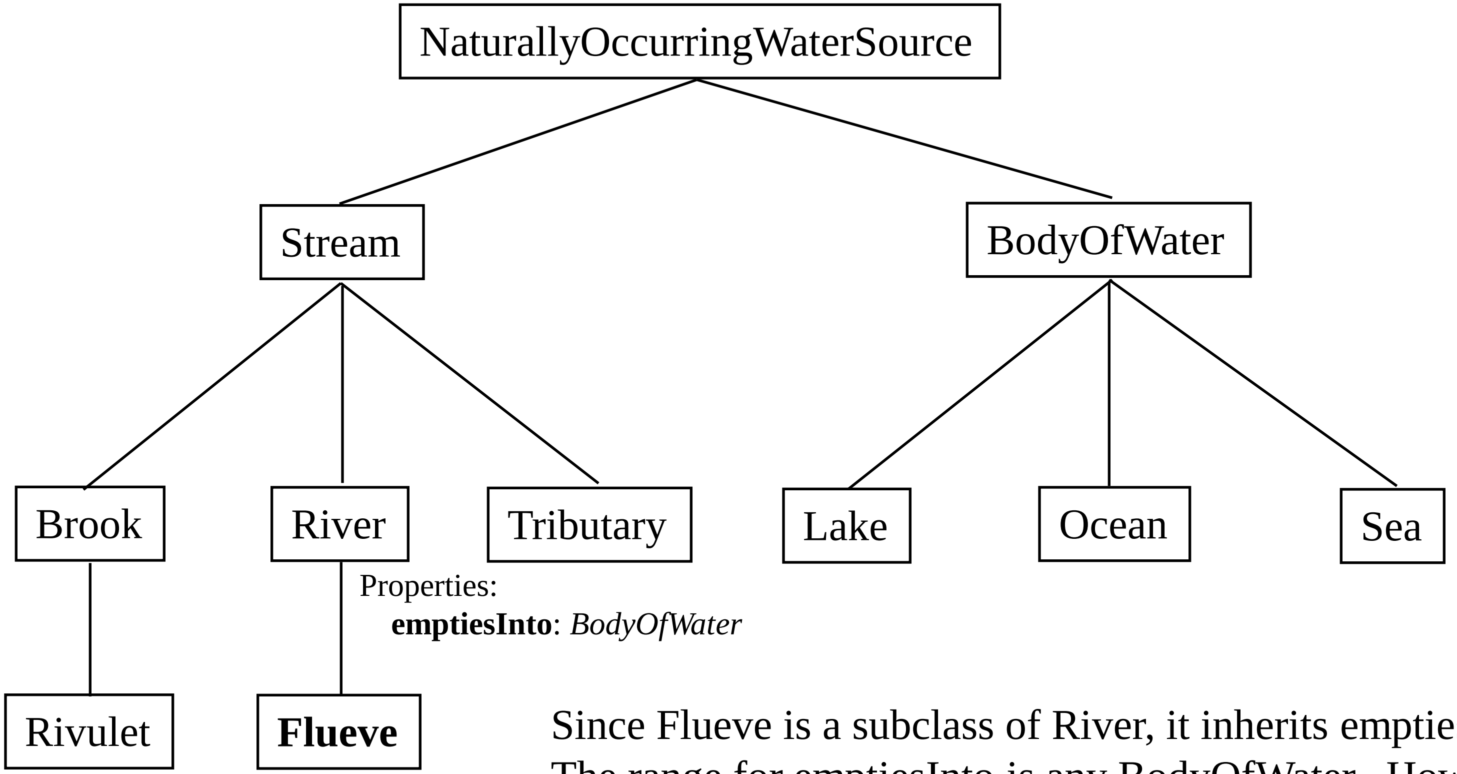
Consequences:

1. SymmetricProperty and TransitiveProperty can only be used to relate Resources to Resources.
2. FunctionalProperty and InverseFunctionalProperty can be used to relate Resources to Resources, or Resources to an RDF Schema Literal or an XML Schema datatype.

Constraining a property based upon its context

- Now we will look at ways to constrain the range of a property based upon the context (class) in which it is used ...

Sometimes a **class** needs to restrict the range of a property



NaturallyOccurringWaterSource

Stream

BodyOfWater

Brook

River

Tributary

Lake

Ocean

Sea

Rivulet

Flueve

Properties:
emptiesInto: *BodyOfWater*

Since Flueve is a subclass of River, it inherits emptiesInto. The range for emptiesInto is any BodyOfWater. However, the definition of a Flueve (French) is: "a River which emptiesInto a Sea". Thus, *in the context of the Flueve class* we want the range of emptiesInto restricted to Sea.

Global vs Local Properties

- rdfs:range imposes a global restriction on the emptiesInto property, i.e., the rdfs:range value applies to River and all subclasses of River.
- As we have seen, *in the context of the Flueve class*, we would like the emptiesInto property to have its range restricted to just the Sea class. Thus, for the Flueve class we want a *local definition of emptiesInto*.
- Before we see how to do this, we need to look at how classes are defined in OWL ...

Defining Classes in OWL

- OWL classes permit much greater expressiveness than RDF Schema classes.
- Consequently, OWL has created their own Class, owl:Class.

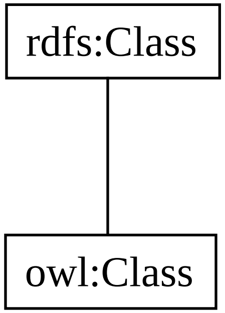
```
<rdfs:Class rdf:ID="River">
      <rdfs:subClassOf rdf:resource="#Stream"/>
</rdfs:Class>
```

RDFS

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

OWL

owl:Class is a subclass of rdfs:Class



Defining emptiesInto (when used in Flueve) to have **allValuesFrom** the Sea class

```
<?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#"
                xmlns:owl="http://www.w3.org/2002/07/owl#"
                xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:Class rdf:ID="Flueve">
        <rdfs:subClassOf rdf:resource="#River"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#emptiesInto"/>
                <owl:allValuesFrom rdf:resource="#Sea"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

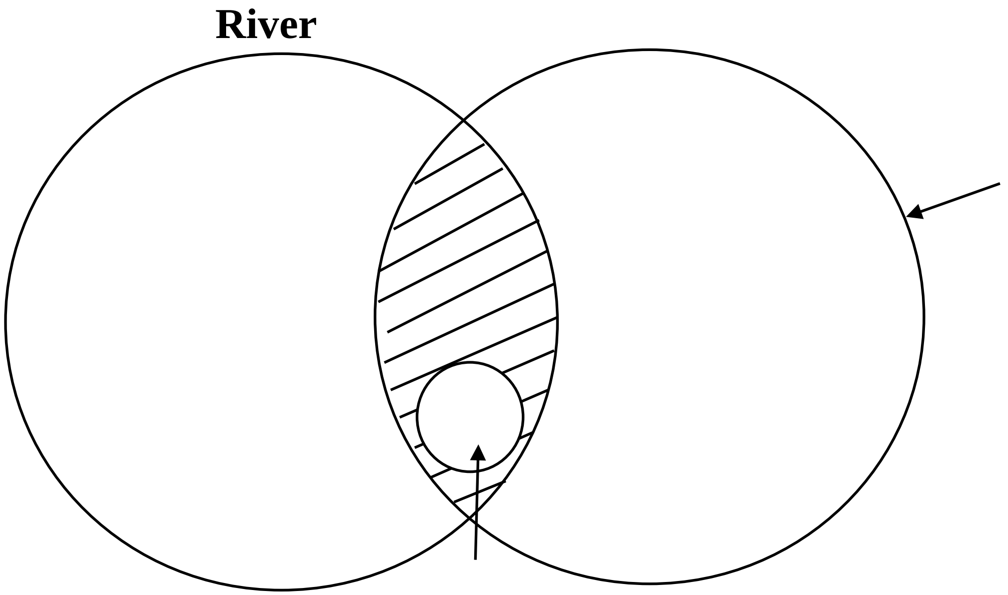
Flueve is a subclass of an "anonymous class"

```
<owl:Class rdf:ID="Flueve">
      <rdfs:subClassOf rdf:resource="#River"/>
       <rdfs:subClassOf>
              <owl:Restriction>
anonymous            <owl:onProperty rdf:resource="#emptiesInto"/>
   class             <owl:allValuesFrom rdf:resource="#Sea"/>
              </owl:Restriction>
       </rdfs:subClassOf>
</owl:Class>
```

This is read as: "The Flueve class is a subClassOf River, and a subClassOf an *anonymous class* which has a property emptiesInto and all values for emptiesInto must be instances of Sea."

Here's an easier way to read this: "The Flueve class is a subClassOf River. It has a property emptiesInto. All values for emptiesInto must be instances of Sea."

Definition of Flueve



The members of this **anonymous class** are instances which have an emptiesInto property in which all values are instances of Sea.

Flueve - a River that emptiesInto a Sea.

An instance of Flueve

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

Yangtze.rdf

We can infer that this value must be a Sea!

All values for emptiesInto must be an instance of Sea, *in the context of the Flueve class*.

Two forms of rdfs:subClassOf

1

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

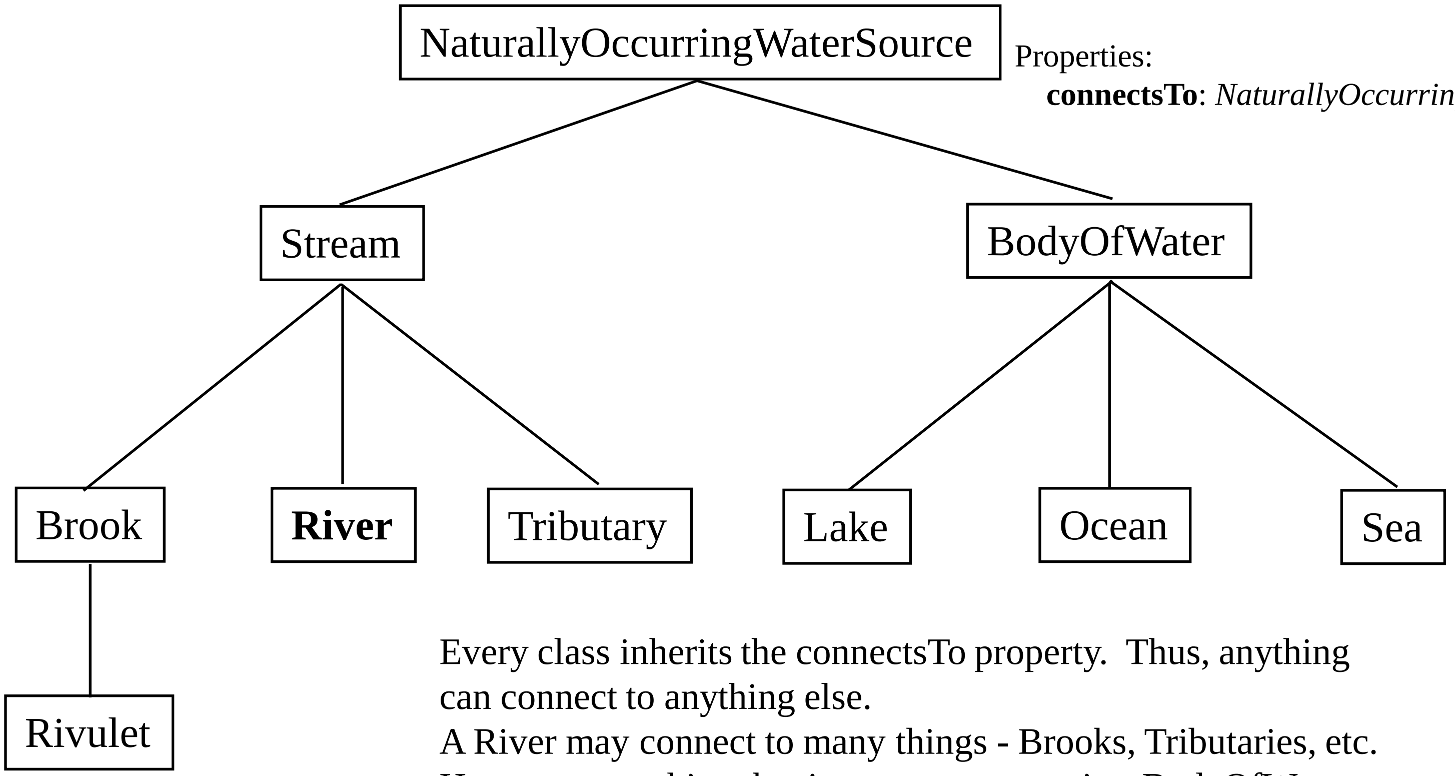
Specify the class using the rdf:resource attribute.

Specify the class using owl:Restriction.

2

```
<rdfs:subClassOf>
        <owl:Restriction>
              <owl:onProperty rdf:resource="#emptiesInto"/>
              <owl:allValuesFrom rdf:resource="#Sea"/>
        </owl:Restriction>
</rdfs:subClassOf>
```

To be a River at least one value of connectsTo must be BodyOfWater



Properties:
connectsTo: *NaturallyOccurringWaterSource*

Every class inherits the connectsTo property. Thus, anything can connect to anything else.
A River may connect to many things - Brooks, Tributaries, etc.
However, one thing that it *must* connect to is a BodyOfWater (Lake, Ocean, or Sea). Thus, *in the context of* the River class the connectsTo property should have at least one value that is a BodyOfWater.

Defining connectsTo (when used in River) to have **someValuesFrom** the BodyOfWater class

```
<?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#"
                xmlns:owl="http://www.w3.org/2002/07/owl#"
                xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
         <rdfs:subClassOf>
              <owl:Restriction>
                     <owl:onProperty rdf:resource="#connectsTo"/>
                     <owl:someValuesFrom rdf:resource="#BodyOfWater"/>
              </owl:Restriction>
         </rdfs:subClassOf>
    </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Understanding owl:someValuesFrom

```
<owl:Class rdf:ID="River">
      <rdfs:subClassOf rdf:resource="#Stream"/>
       <rdfs:subClassOf>
              <owl:Restriction>
                     <owl:onProperty rdf:resource="#connectsTo"/>
                     <owl:someValuesFrom rdf:resource="#BodyOfWater"/>
              </owl:Restriction>
       </rdfs:subClassOf>
</owl:Class>
```

This is read as: "The River class is a subClassOf Stream, and a subClassOf an anonymous class which has a property connectsTo and some values (at least one) of connectsTo must be instances of BodyOfWater."

Here's an easier way to read this: "The River class is a subClassOf Stream. It has a property connectsTo. At least one value for connectsTo must be an instance of BodyOfWater."

An instance of River

```
<?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#">
    <connectsTo rdf:resource="http://www.china.org/rivers#Wu"/>
    <connectsTo rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>
```

Yangtze.rdf

At least one of these values must
be a BodyOfWater (Lake, Ocean, or Sea)!
(Assume that there are no other documents
which describe the Yangtze.)

At least one value for connectsTo must be an instance of BodyOfWater, *in the context of the River class*.

allValuesFrom vs. someValuesFrom

```
<owl:onProperty rdf:resource="#emptiesInto"/>
<owl:allValuesFrom rdf:resource="#Sea"/>
```

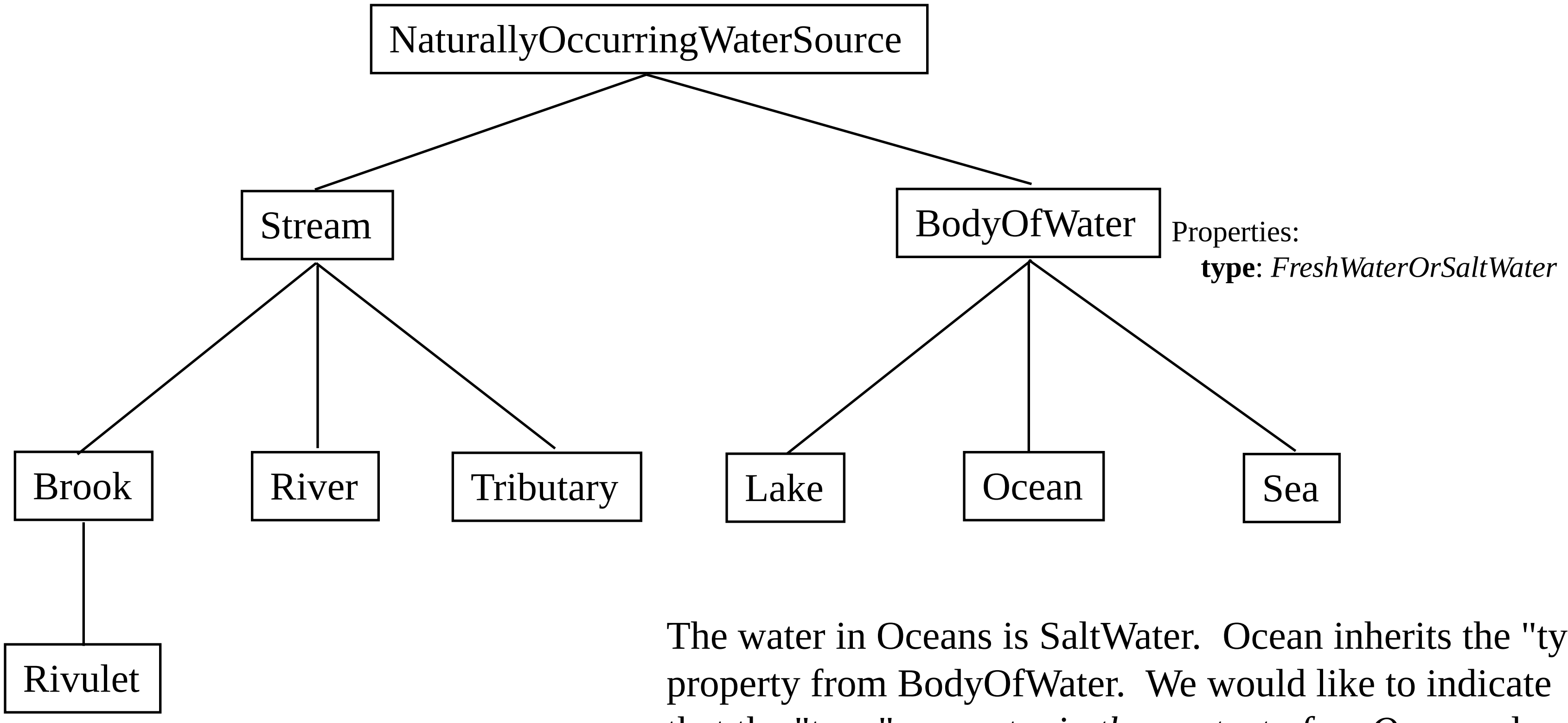
Wherever there is an emptiesInto property, all its values must be instances of Sea. [There may be zero emptiesInto properties.]

versus:

```
<owl:onProperty rdf:resource="#connectsTo"/>
<owl:someValuesFrom rdf:resource="#BodyOfWater"/>
```

There must be at least one connectsTo property whose value is BodyOfWater. [There must be at least one connectsTo property.]

All Oceans are SaltWater



Properties:
type: *FreshWaterOrSaltWater*

FreshWaterOrSaltWater

The water in Oceans is SaltWater. Ocean inherits the "type" property from BodyOfWater. We would like to indicate that the "type" property, *in the context of an Ocean*, always has a value of SaltWater.

Defining the "type" property to have the value SaltWater (when used in Ocean)

```
<?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#"
               xmlns:owl="http://www.w3.org/2002/07/owl#"
               xml:base="http://www.geodesy.org/water/naturally-occurring">

   <FreshWaterOrSaltWater rdf:ID="SaltWater"/>

    <owl:Class rdf:ID="Ocean">
        <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
         <rdfs:subClassOf>
              <owl:Restriction>
                     <owl:onProperty rdf:resource="#type"/>
                     <owl:hasValue rdf:resource="#SaltWater"/>
              </owl:Restriction>
         </rdfs:subClassOf>
    </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Understanding owl:hasValue

```
 <FreshWaterOrSaltWater rdf:ID="SaltWater"/>

<owl:Class rdf:ID="Ocean">
      <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
       <rdfs:subClassOf>
              <owl:Restriction>
                     <owl:onProperty rdf:resource="#type"/>
                     <owl:hasValue rdf:resource="#SaltWater"/>
              </owl:Restriction>
       </rdfs:subClassOf>
</owl:Class>
```

Note that this is an **instance** of the class FreshWaterOrSaltWater.

This is read as: "The Ocean class is a subClassOf BodyOfWater, and a subClassOf an anonymous class which has a property - type - that has the value SaltWater."

Here's an easier way to read this: "The Ocean class is a subClassOf BodyOfWater. Every Ocean has a 'type' property whose value is SaltWater."

An instance of Ocean

```
<?xml version="1.0"?>
<Ocean rdf:ID="PacificOcean"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <type rdf:resource="http://www.geodesy.org/water/naturally-occurring#SaltWater"/>
</Ocean>
```

PacificOcean.rdf

Every instance of Ocean must have a property "type" whose value is SaltWater.
Note: it is not necessary to put the type property in an Ocean instance document - the "type" may be inferred from hasValue. That is, *the Ontology indicates that if it's an Ocean then its type is SaltWater.*

At least one "type" property must have the value SaltWater, *in the context of an Ocean class.*

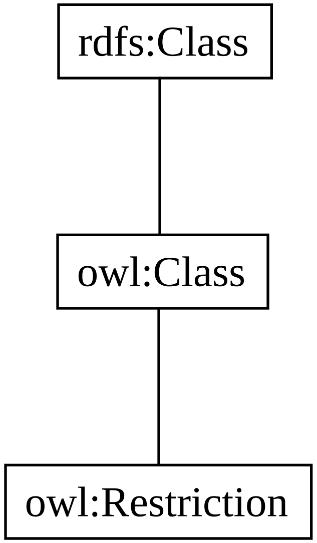
owl:hasValue means **there exists** a property with the specified value

- The owl:hasValue property restriction simply asserts that **there exists** a property with the value.
- In fact, there may be other instances of the same property that do not have the value.
- For the Ocean example, we know that every Ocean is of type of SaltWater.

Summary of the different ways a class can constrain a property

- In the preceding slides we have seen the different ways that a class can constrain a global property. We saw that a property can be constrained such that:
  - All values must belong to a certain class (use allValuesFrom).
  - At least one value must come from a certain class (use someValuesFrom).
  - It has a specific value (use hasValue).

Properties of the Restriction Class



rdfs:Class

owl:Class

owl:Restriction

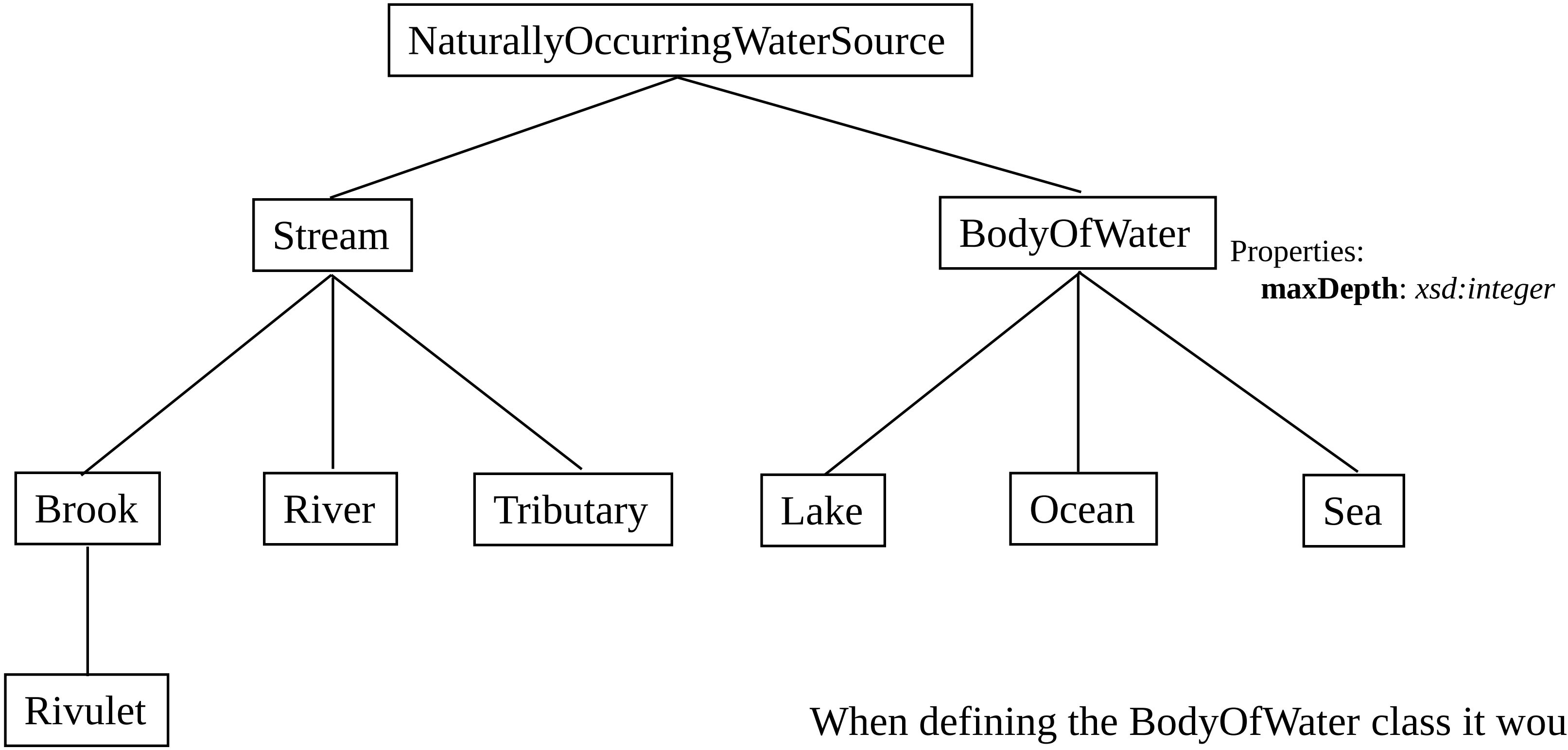
Properties:

onProperty: *rdf:Property*
allValuesFrom: *rdfs:Class*
hasValue:
someValuesFrom: *rdfs:Class*

Context-specific cardinality constraints

- Definition of cardinality: the number of occurrences.
- Now we will look at ways to constrain the cardinality of a property based upon the context (class) in which it is used ...

A BodyOfWater can have only one maxDepth (cardinality = 1)



When defining the BodyOfWater class it would be useful to indicate that there can be only one maxDepth for a BodyOfWater.

Defining the cardinality of the maxDepth property to be 1

```
<?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#"
              xmlns:owl="http://www.w3.org/2002/07/owl#"
              xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:Class rdf:ID="BodyOfWater">
        <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
         <rdfs:subClassOf>
              <owl:Restriction>
                     <owl:onProperty rdf:resource="#maxDepth"/>
                     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardinality>
              </owl:Restriction>
         </rdfs:subClassOf>
    </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Understanding owl:cardinality

```
<owl:Class rdf:ID="BodyOfWater">
      <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
       <rdfs:subClassOf>
              <owl:Restriction>
                     <owl:onProperty rdf:resource="#maxDepth"/>
                     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardinality>
              </owl:Restriction>
       </rdfs:subClassOf>
</owl:Class>
```

This is read as: "The BodyOfWater class is a subClassOf NaturallyOccurringWaterSource, and a subClassOf an anonymous class which has a property maxDepth. There can be only one maxDepth for a BodyOfWater. This is indicated by a cardinality of 1."

Here's an easier way to read this: "The BodyOfWater class is a subClassOf NaturallyOccurringWaterSource. It has a property maxDepth. There can be only one maxDepth for a BodyOfWater."

maxDepth of the PacificOcean

```
<?xml version="1.0"?>
<Ocean rdf:ID="PacificOcean"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns="http://www.geodesy.org/water/naturally-occurring#">
    <maxDepth rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2300</maxDepth>
</Ocean>
```

PacificOcean.rdf

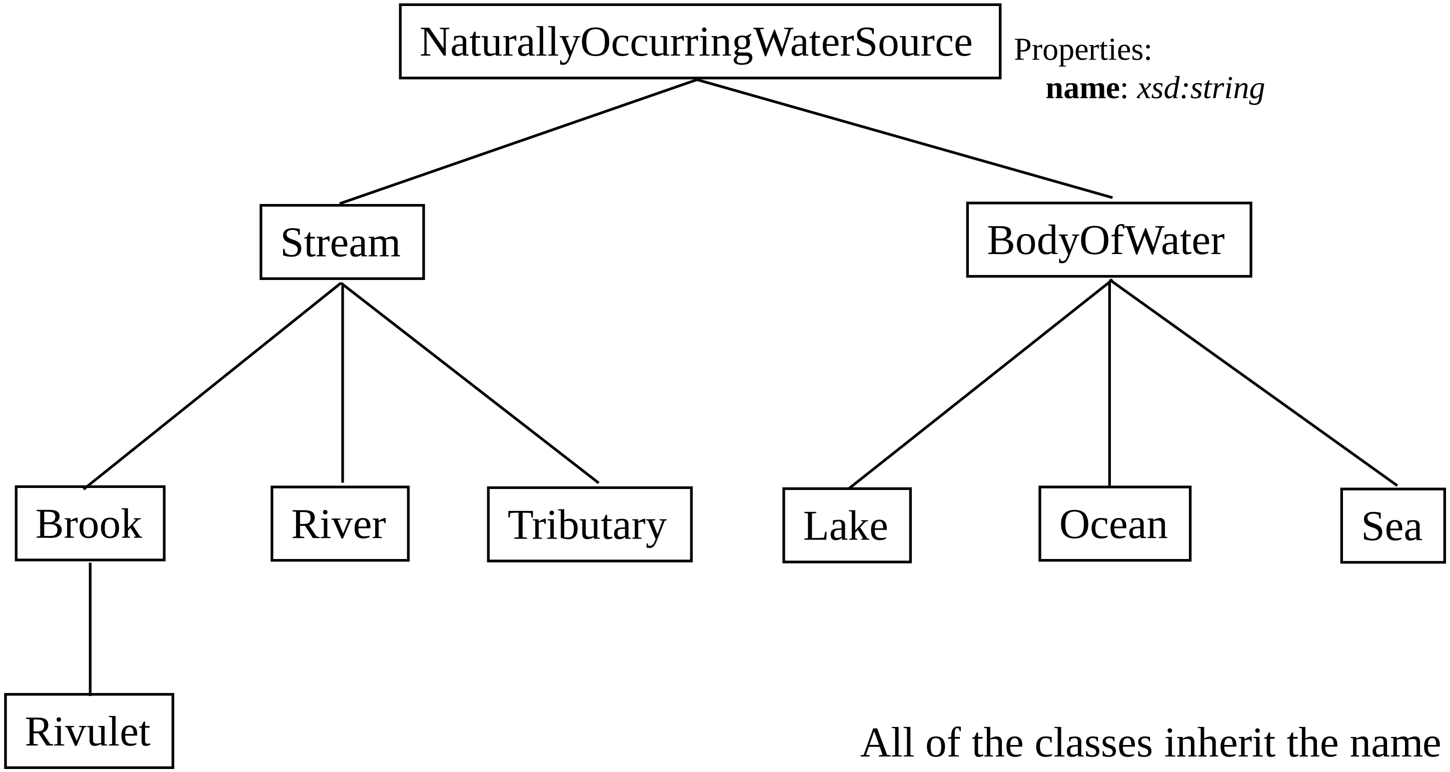
The PacificOcean has only one maxDepth.

There is only one maxDepth, *in the context of a BodyOfWater (e.g., Ocean) class*.

The cardinality is **not** mandating the number of occurrences of a property in an instance document!

- Differentiate between these two statements:
  - 1. In an instance document there can be only one maxDepth property for a BodyOfWater.
  - 2. A BodyOfWater has only one maxDepth.
- Do you see the difference?
  - 1. The first statement is something that you would find in an XML Schema.
  - 2. The second statement is a statement of information. It places no restrictions on the number of occurrences of the maxDepth property in an instance document. In fact, any resource may have multiple maxDepth properties. They must all be equal, however, since there can be only one maxDepth per resource.

Some Brooks have no name (minCardinality = 0)



Properties:
name: *xsd:string*

All of the classes inherit the name property. When defining the Brook class it would be useful to indicate that a Brook might not have a name.

Defining the minCardinality of the name property to be 0

```
<?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#"
            xmlns:owl="http://www.w3.org/2002/07/owl#"
            xml:base="http://www.geodesy.org/water/naturally-occurring">

   <owl:Class rdf:ID="Brook">
      <rdfs:subClassOf rdf:resource="#Stream"/>
       <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#name"/>
                  <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">0</owl:minCardinality>
            </owl:Restriction>
       </rdfs:subClassOf>
   </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Defining the cardinality of the name property to be a range (0-10)

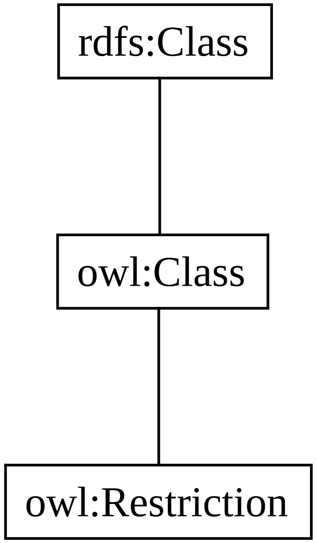
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xml:base="http://www.geodesy.org/water/naturally-occurring">

   <owl:Class rdf:ID="Brook">
      <rdfs:subClassOf rdf:resource="#Stream"/>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#name"/>
            <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">0</owl:minCardinality>
            <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">10</owl:maxCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>

   ...

</rdf:RDF>
```

naturally-occurring.owl (snippet)

Summary of the different ways to express the cardinality of a property

- In the preceding slides we have seen the ways that a class can specify the cardinality of a property, using:
  - cardinality
  - minCardinality
  - maxCardinality

Complete List of Properties of the Restriction Class



rdfs:Class

owl:Class

owl:Restriction

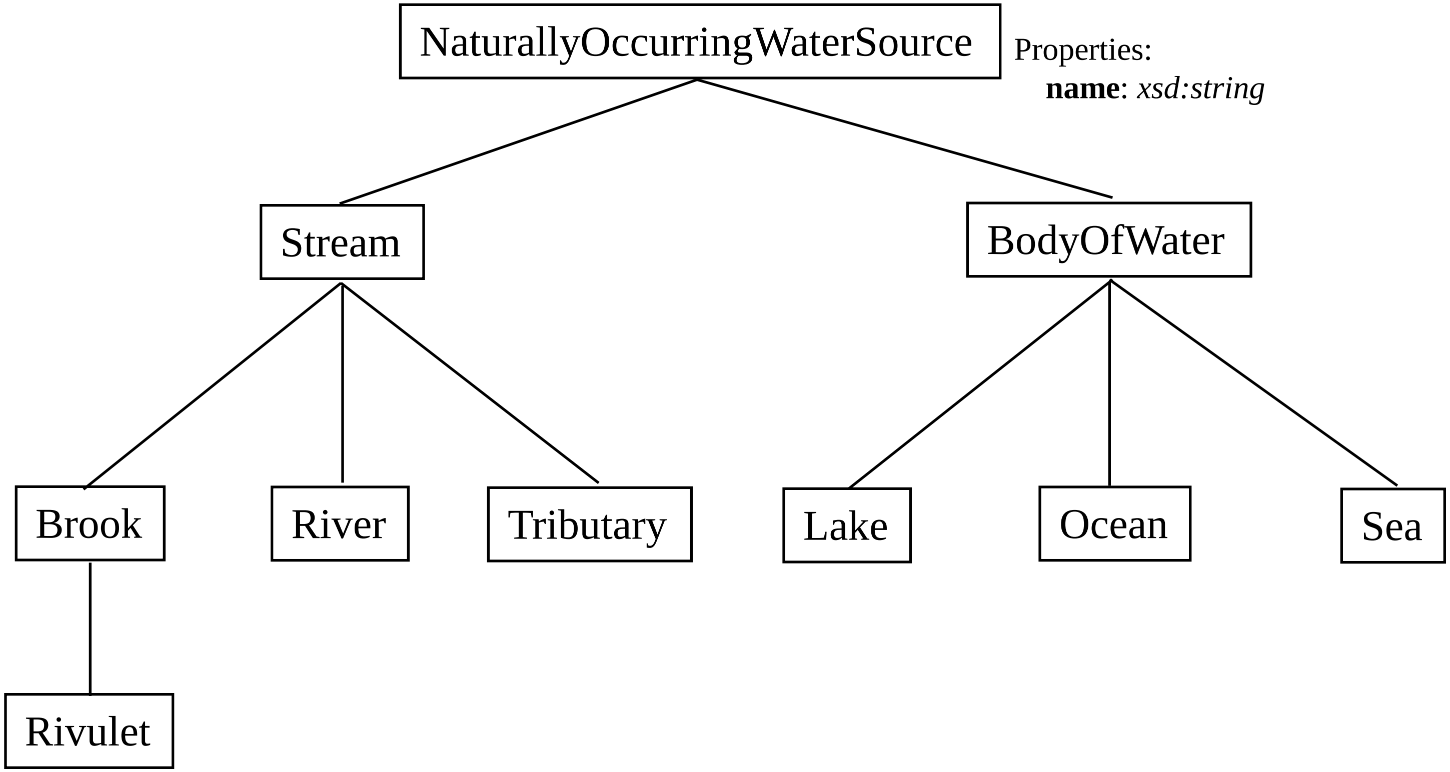
Properties:

onProperty: *rdf:Property*
allValuesFrom: *rdfs:Class*
hasValue:
someValuesFrom: *rdfs:Class*
cardinality: *xsd:nonNegativeInteger*
minCardinality: *xsd:nonNegativeInteger*
maxCardinality: *xsd:nonNegativeInteger*

Equivalent Properties

- Now we will look at the ways to express that two properties are equivalent ...

name is equivalent to the Title property in Dublin Core



Properties:
name: *xsd:string*

Defining name to be equivalent to dc:Title

```
<?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#"
             xmlns:owl="http://www.w3.org/2002/07/owl#"
             xml:base="http://www.geodesy.org/water/naturally-occurring">

    <owl:DatatypeProperty rdf:ID="name">
        <owl:equivalentProperty rdf:resource="http://pur1.org/dc/terms/title"/>
        <rdfs:domain rdf:resource="#NaturallyOccurringWaterSource"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

   ...

</rdf:RDF>
```

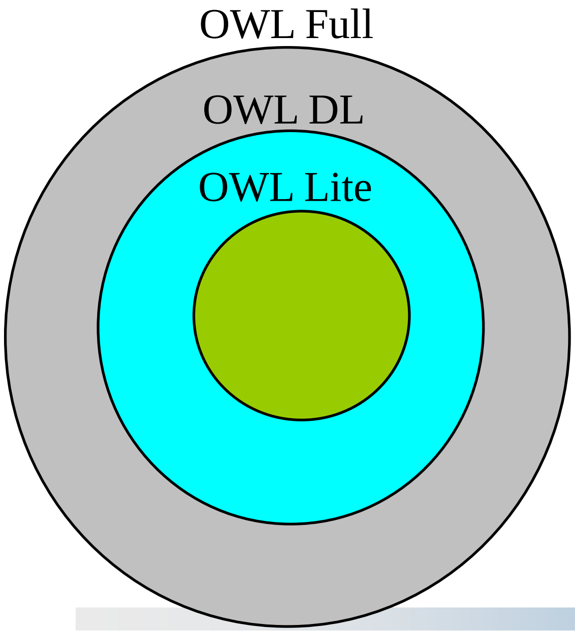
naturally-occurring.owl (snippet)

Note that we are using owl:DatatypeProperty to define name.

The Three Faces of OWL

OWL Full, OWL DL, and OWL Lite

- Not everyone will need all of the capabilities that OWL provides. Thus, there are three versions of OWL:



DL = Description Logic

Comparison

OWL Full	OWL DL	OWL Lite
Everything that has been shown in this tutorial is available. Further, you can mix RDF Schema definitions with OWL definitions.	You cannot use owl:cardinality with TransitiveProperty. A DL ontology cannot import an OWL Full ontology. You cannot use a class as a member of another class, i.e., you cannot have metaclasses. FunctionalProperty and InverseFunctionalProperty cannot be used with datatypes (they can only be used with ObjectProperty).	You cannot use owl:minCardinality or owl:maxCardinality. The only allowed values for owl:cardinality is 0 and 1. Cannot use owl:hasValue. Cannot use owl:disjointWith. Cannot use owl:oneOf. Cannot use owl:complementOf. Cannot use owl:unionOf.

Advantages/Disadvantages

- Full:
  - The advantage of the Full version of OWL is that you get the full power of the OWL language.
  - The disadvantage of the Full version of OWL is that it is difficult to build a Full tool. Also, the user of a Full-compliant tool may not get a quick and complete answer.
- DL/Lite:
  - The advantage of the DL or Lite version of OWL is that tools can be built more quickly and easily, and users can expect responses from such tools to come quicker and be more complete.
  - The disadvantage of the DL or Lite version of OWL is that you don't have access to the full power of the language.

Related Documents

- The **OWL Guide** provides a very nice description of OWL, with many examples:
  - http://www.w3.org/TR/owl-guide/
- Here is the URL to the **OWL Reference** document:
  - http://www.w3.org/TR/owl-ref/
- For all other OWL documents, and information on the Semantic Web see:
  - http://www.w3.org/2001/sw/

Examples

The Robber and the Speeder (version 2)

- An expanded version of the Robber and the Speeder example is shown on the following slides. This version was created by Ian Davis. Thanks Ian!

Robber drops gun while fleeing!

First of all a robbery takes place. The robber drops his gun while fleeing. This report is filed by the investigating officers:

```
<RobberyEvent>
    <date>...</date>
    <description>...</description>
    <evidence>
         <Gun>
              <serial>ABCD</serial>
         </Gun>
    </evidence>
    <robber>
          <Person /> <!-- an unknown person -->
    </robber>
</RobberyEvent>
```

Speeder stopped

Subsequently a car is pulled over for speeding. The traffic officer files this report electronically while issuing a ticket:

```
<SpeedingOffence>
    <date>...</date>
    <description>...</description>
    <speeder>
         <Person>
              <name>Fred Blogs</name>
              <driversLicenseNumber>ZXYZXY</driversLicenseNumber>
        </Person>
    </speeder>
</SpeedingOffence>
```

The speeder owns a gun with the same serial number as the robbery gun!

At police headquarters (HQ), a computer analyzes each report as it is filed. The computer uses the driver's license information to look up any other records it has about Fred Blogs (the speeder) and discovers this gun license:

```
<GunLicense>
   <registeredGun>
      <Gun>
         <serial>ABCD</serial>
      </Gun>
   </registeredGun>
   <holder>
      <Person>
         <driversLicenseNumber>ZXYZXY</driversLicenseNumber>
      </Person>
   </holder>
</GunLicense>
```

Case Solved?

- Not yet! These questions must be answered before the speeder can be arrested as the robbery suspect:
    - Can multiple guns have the same serial number?
        - If so, then just because Fred Blogs owns a gun with the same serial number as the robbery gun does not mean it was his gun that was used in the robbery.
    - Can multiple people have the same driver's license number?
        - If so, then the gun license information may be for someone else.
    - Can a gun be registered in multiple gun licenses?
        - If so, then the other gun licenses may show the holder of the gun to be someone other than Fred Blogs.
    - Can a gun license have multiple holders of a registered gun?
        - If so, then there may be another gun license document (not available at the police HQ) which shows the same registered gun but with a different holder.
- The OWL Ontology (Police.owl) provides the information needed to answer these questions!

Can multiple guns have the same serial number?

This OWL rule (in Police.owl) tells the computer at police HQ that *each gun is uniquely identified by its serial number*:

```
<owl:InverseFunctionalProperty rdf:ID="serial">
      <rdfs:domain rdf:resource="Gun"/>
      <rdfs:range  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:InverseFunctionalProperty>
```

Can multiple people have the same driver's license number?

The following OWL rule tells the computer that *a driver's license number is unique to a Person*:

```
<owl:InverseFunctionalProperty rdf:ID="driversLicenseNumber">
      <rdfs:domain rdf:resource="Person"/>
      <rdfs:range  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:InverseFunctionalProperty>
```

Can a gun be registered in multiple gun licenses?

The next OWL rule tells the computer that the registeredGun property uniquely identifies a GunLicense, i.e., *each gun is associated with only a single GunLicense*:

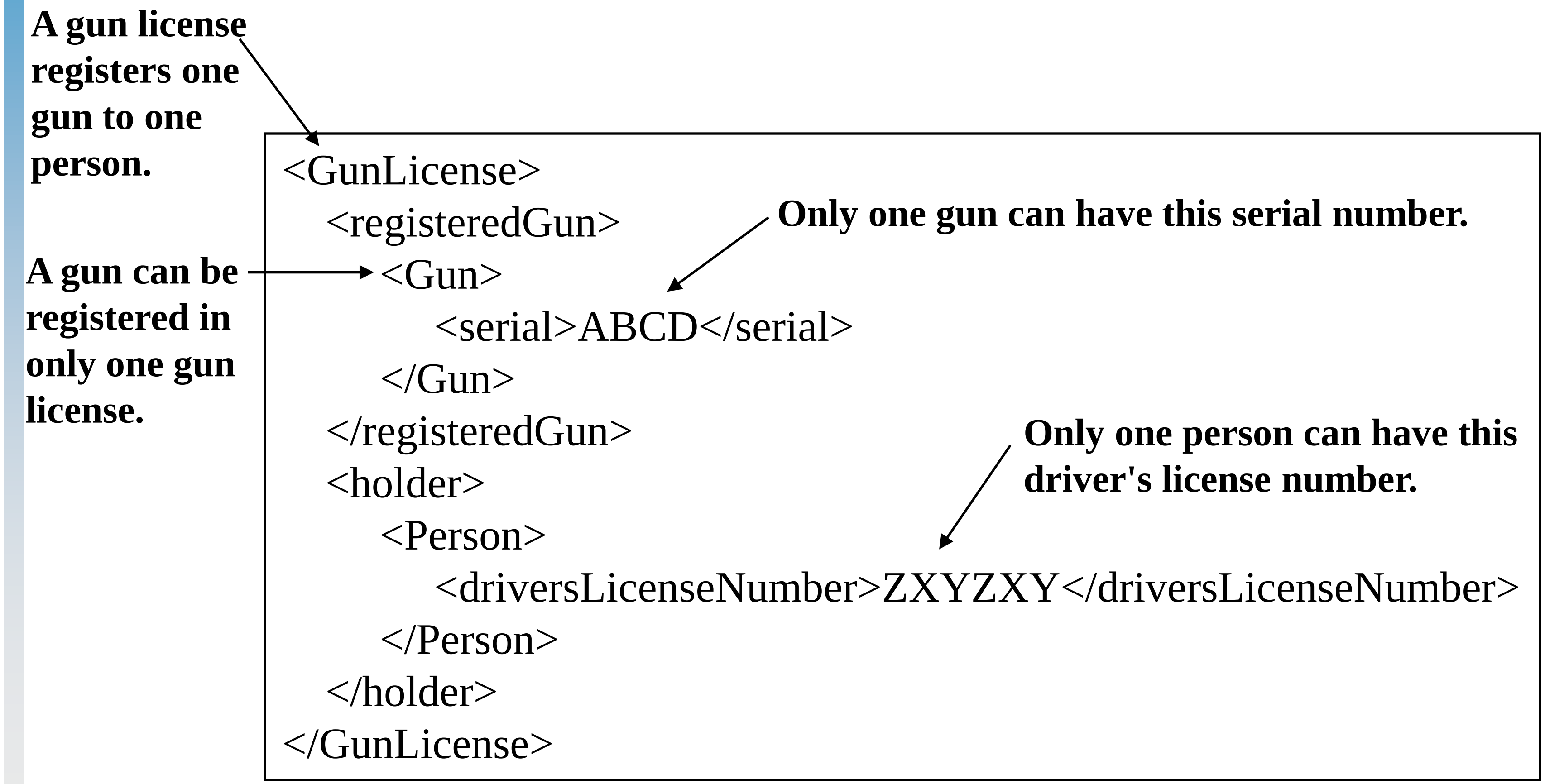
```
<owl:InverseFunctionalProperty rdf:ID="registeredGun">
        <rdfs:domain rdf:resource="GunLicense"/>
        <rdfs:range  rdf:resource="Gun"/>
</owl:InverseFunctionalProperty>
```

Can a gun license have multiple holders of a registered gun?

The police computer uses the following OWL rule to determine that the gun on the license is the same gun used in the robbery. This final rule seals the speeder's fate. It tells the computer that *each GunLicense applies to only one gun and one person*. So, there is no doubt that the speeder is the person who owns the gun:

```
<owl:Class rdf:ID="GunLicense">
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
            <owl:onProperty rdf:resource="#registeredGun"/>
            <owl:cardinality>1</owl:cardinality>
        </owl:Restriction>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#holder"/>
            <owl:cardinality>1</owl:cardinality>
        </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

Summary of information provided by the Police ontology



A gun license registers one gun to one person.

A gun can be registered in only one gun license.

Only one gun can have this serial number.

Only one person can have this driver's license number.

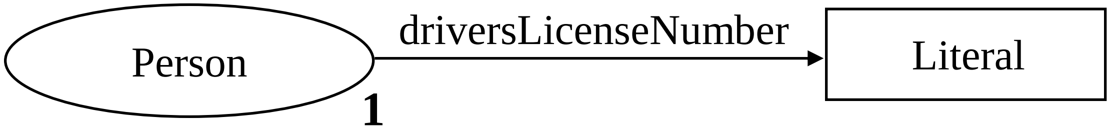
```
<GunLicense>
    <registeredGun>
        <Gun>
            <serial>ABCD</serial>
        </Gun>
    </registeredGun>
    <holder>
        <Person>
            <driversLicenseNumber>ZXYZXY</driversLicenseNumber>
        </Person>
    </holder>
</GunLicense>
```

We now have overwhelming evidence that the speeder is the robber!

Notes

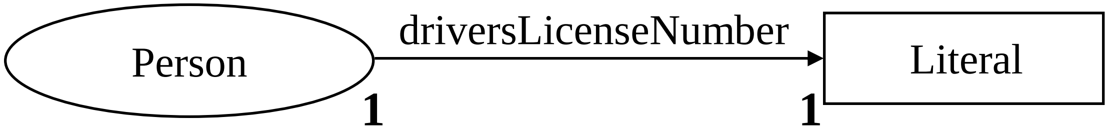
The example showed that a driver's license number applies to only one person:

```
<owl:InverseFunctionalProperty rdf:ID="driversLicenseNumber">
      <rdfs:domain rdf:resource="Person"/>
      <rdfs:range  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:InverseFunctionalProperty>
```



"A driver's license number applies to only one person."

We can make an even stronger statement, because it's also true that a person has only one driver's license number:



"A driver's license number applies to only one person, and a person has only one driver's license number."

Notes (cont.)

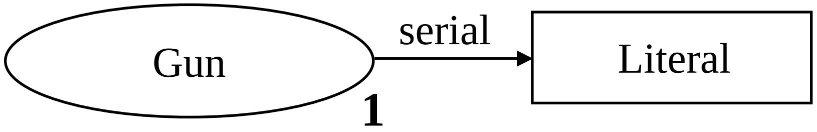
Thus, driversLicenseNumber is also a functional property:

```
<owl:DatatypeProperty rdf:ID="driversLicenseNumber">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="Person"/>
    <rdfs:range  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:DatatypeProperty>
```

Notes (cont.)

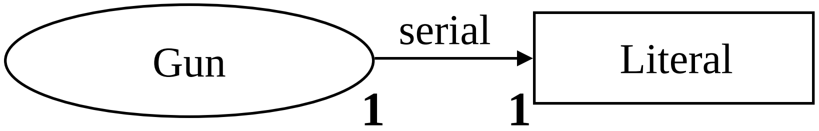
The example also showed that a serial number applies to only one gun:

```
<owl:InverseFunctionalProperty rdf:ID="serial">
      <rdfs:domain rdf:resource="Gun"/>
      <rdfs:range  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:InverseFunctionalProperty>
```



"A serial number applies to only one gun."

We can make an even stronger statement, because it's also true that a gun has only one serial number:



"A serial number applies to only one gun, and a gun has only one serial number."

Notes (cont.)

Thus, serial is also a functional property:

```
<owl:DatatypeProperty rdf:ID="serial">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
      <rdfs:domain rdf:resource="Gun"/>
      <rdfs:range  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</owl:DatatypeProperty>
```

FAQ

Can an OWL Ontology also contain instance data?

In general, it is best to keep instance data separate from the ontology. Sometimes, however, mingling instance data with an ontology may be unavoidable. For example, suppose that you wish to use owl:AllDifferent to indicate that Mary, David, and Roger are all different:

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
        <Person rdf:about="#Mary"/>
        <Person rdf:about="#David"/>
        <Person rdf:about="#Roger"/>
    </owl:distinctMembers>
</owl:AllDifferent>
```

You might wish to provide, in the ontology, a "barebones" definition of the Mary instance, the David instance, and the Roger instance:

```
<Person rdf:ID="Mary"/>
<Person rdf:ID="David"/>
<Person rdf:ID="Roger"/>
```

Can an ontology also contain instance data? (cont.)

Now, instance documents extend the ontologies' barebones instance definitions:

```
<Person rdf:about="http://www.person.org#Roger">
    <hometown>Boston, MA</hometown>
    <workplace>The MITRE Corp.</workplace>
</Person>
```

Difference between a Class with a property that has a maxCardinality=1 versus a Functional Property?

Both forms are equivalent! Let's take an example. Below is shown a Gun Class which is defined to have at most one serial number:

Version 1

```
<owl:Class rdf:ID="Gun">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#serial"/>
                        <owl:maxCardinality>1</owl:maxCardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="serial">
         <rdfs:domain rdf:resource="#Gun"/>
         <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

Continued →

Defining a Class which has a Functional Property is equivalent!

The below serial property is defined to be a Functional Property, and is to be used with a Gun instance. Thus, the Gun Class has at most one serial number. The two forms are equivalent!

Version 2

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

<owl:FunctionalProperty rdf:ID="serial">
        <rdfs:domain rdf:resource="Gun" />
        <rdfs:range  rdf:resource="&rdfs;#Literal"/>
</owl:FunctionalProperty>
```

Difference between a Class with multiple subclasses, each having a property that has a maxCardinality=1 versus multiple Functional Properties?

Both forms are equivalent! Let's take an example. Below is shown a GunLicense Class which is defined to have at most one registeredGun and at most one holder:

Version 1

```
<owl:Class rdf:ID="GunLicense">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#registeredGun"/>
            <owl:maxCardinality>1</owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#holder"/>
            <owl:maxCardinality>1</owl:maxCardinality>
        </owl:Restriction>
    <rdfs:subClassOf>
</owl:Class

<owl:ObjectProperty rdf:ID="registeredGun">
      <rdfs:domain rdf:resource="#GunLicense"/>
      <rdfs:range  rdf:resource="#Gun"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="holder">
      <rdfs:domain rdf:resource="#GunLicense"/>
      <rdfs:range  rdf:resource="#Person"/>
</owl:ObjectProperty>
```

Defining a Class which has multiple Functional Properties is equivalent!

The below registeredGun property and holder property are defined to be Functional Properties, and are to be used with a GunLicense instance. So, the GunLicense Class has at most one registeredGun and at most one holder. Thus, the two forms are equivalent!

Version 2

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

<owl:FunctionalProperty rdf:ID="registeredGun">
      <rdfs:domain rdf:resource="#GunLicense"/>
      <rdfs:range  rdf:resource="#Gun"/>
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="holder">
      <rdfs:domain rdf:resource="#GunLicense"/>
      <rdfs:range  rdf:resource="#Person"/>
</owl:FunctionalProperty>
```