Semantic Web

$Date: 2006/08/23 13:32:09 $

Ivan Herman, W3C

Overview of the Semantic Web

Slides of the presentation …

This is an evolving slide set, updated regularly.

Towards a Semantic Web

- The current Web represents information using
  - *natural language (English, Hungarian, Chinese,…)*
  - *graphics, multimedia, page layout*
- Humans can process this easily
  - *can deduce facts from partial information*
  - *can create mental associations*
  - *are used to various sensory information*
    - (well, sort of… people with disabilities may have serious problems on the Web with rich media!)

Towards a Semantic Web

- Tasks often require to *combine* data on the Web:
  - *hotel and travel information may come from different sites*
  - *searches in different digital libraries*
  - *etc.*
- Again, humans combine these information easily
  - *even if different terminologies are used!*

However…

- However: machines are ignorant!
  - *partial information is unusable*
  - *difficult to make sense from, e.g., an image*
  - *drawing analogies automatically is difficult*
  - *difficult to combine information*
    - is `<foo:creator>` same as `<bar:author>`?
    - how to combine different XML hierarchies?
  - *…*

Example: Searching

- The best-known example…
  - *Google et al. are great, but there are too many false or missing hits*
    - e.g., if you search in for “yacht racing”, the America’s Cup will *not* be found
  - *adding (maybe application specific) descriptions to resources should improve this*

Example: Automatic Airline Reservation

- Your automatic airline reservation
  - *knows about your preferences*
  - *builds up knowledge base using your past*
  - *can combine the local knowledge with remote services:*
    - airline preferences
    - dietary requirements
    - calendaring
    - etc
- It communicates with *remote* information (i.e., on the Web!)
- (M. Dertouzos: The Unfinished Revolution)

Example: Data(base) Integration

- Databases are very different in structure, in content
- Lots of applications require managing *several* databases
  - *after company mergers*
  - *combination of administrative data for e-Government*
  - *biochemical, genetic, pharmaceutical research*
  - *etc.*
- Most of these data are now on the Web (though not necessarily public yet)
- The *semantics* of the data(bases) should be known (how this semantics is mapped on internal structures is immaterial)

Example: Digital Libraries

- It is a bit like the search example
- It means catalogs on the Web
    - *librarians have known how to do that for centuries*
    - *goal is to have this on the Web, World-wide*
    - *extend it to multimedia data, too*
- But it is more: software agents should also be librarians!
    - *help you in finding the right publications*

Example: Semantics of Web Services

- Web services technology is great
- But if services are ubiquitous, searching issue comes up, for example:
  - *“find me the best differential equation solver”*
  - *“check if it can be combined with the XYZ plotter service”*
- It is necessary to characterize the service
  - *not only in terms of input and output parameters…*
  - *…but also in terms of its semantics*

What Is Needed?

- (Some) data should be available for machines for further processing
- Data should be possibly combined, merged on a Web scale
- Sometimes, data may describe other data (like the library example, using metadata)…
- … but sometimes the data is to be exchanged by itself, like my calendar or my travel preferences
- Machines may also need to *reason* about that data

What Is Needed (Technically)?

- To make data machine processable, we need:
  - *unambiguous names for resources (that may also bind data to real world objects): URI-s*
  - *a common data model to access, connect, describe the resources: RDF*
  - *access to that data: SPARQL*
  - *define common vocabularies: RDFS, OWL, SKOS*
  - *reasoning logics: OWL, Rules*
- *The "Semantic Web" is an extension of the current Web, providing an infrastructure for the integration of data on the Web*

Basic Metadata Architecture: RDF

RDF Triples

- We said “connecting” data…
- But a simple connection is not enough… it should be named somehow
  - *a connection from “me” to my calendar is not the same as the connection from “me” to my CV (even if all of these are on the Web)*
  - *the first connection should somehow say “myCalendar”', the second “myCV”*
- Hence the RDF Triples: a *labelled connection between two resources*

RDF Triples (cont.)

- An RDF Triple (s,p,o) is such that:
  - *“s”, “p” are URI-s, ie, resources on the Web; “o” is a URI or a literal*
  - *conceptually: “p” connects, or relates the “s” and ”o”*
  - *note that we use URI-s for naming: i.e., we can use* `http://www.example.org/myCalendar`
  - *here is the complete triple:*

```
(http://www.ivan-herman.net, http://…/myCalendar, http://…/calendar)
```

- *RDF* is a general model for such triples
  - *… with machine readable formats (RDF/XML, Turtle, n3, RXR, …)*

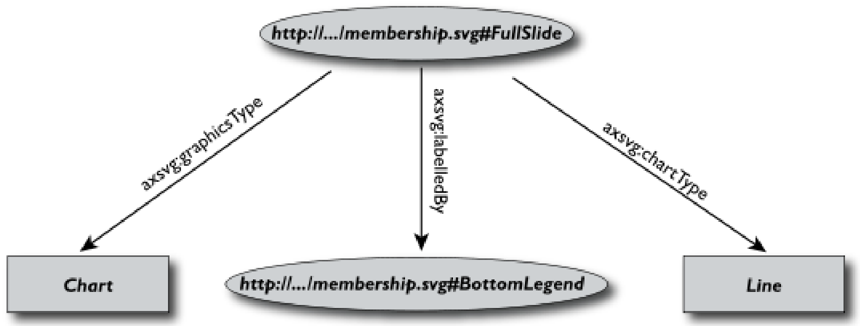
RDF Triples (cont.)

- RDF Triples are also referred to as *“triplets”*, or *“statement”*
- The s, p, o resources are also referred to as *“subject”*, *“predicate”*, *”object”*, or *“subject”*, *”property”*, *”object”*
- Resources can use *any* URI; i.e., it can denote an element *within* an XML file on the Web, not only a “full” resource, e.g.:
  - ***`http://www.example.org/file.xml#xpointer(id('calendar'))`***
  - ***`http://www.example.org/file.html#calendar`***

RDF is a Graph

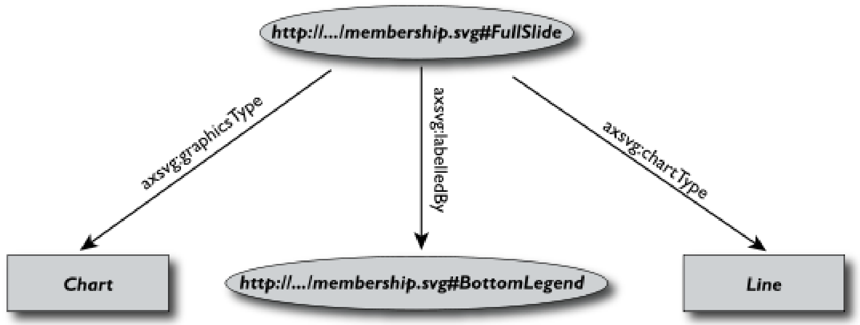
- An (s,p,o) triple can be viewed as a labeled edge in a graph
  - *i.e., a set of RDF statements is a directed, labeled graph*
    - both “objects” and “subjects” are the graph nodes
    - “properties” are the edges
- One should “think” in terms of graphs; XML or Turtle syntax are only the tools for practical usage!
- RDF authoring tools may work with graphs, too (XML or Turtle is done “behind the scenes”)

A Simple RDF Example (in RDF/XML)


http://.../membership.svg#FullSlide
axsvg:graphicsType
axsvg:labelledBy
axsvg:chartType
Chart
http://.../membership.svg#BottomLegend
Line

```
<rdf:Description rdf:about="http://.../membership.svg#FullSlide">
    <axsvg:graphicsType>Chart</axsvg:graphicsType>
    <axsvg:labelledBy rdf:resource="http://...#BottomLegend"/>
    <axsvg:chartType>Line</axsvg:chartType>
</rdf:Description>
```

A Simple RDF Example (in Turtle)


http://.../membership.svg#FullSlide
axsvg:graphicsType
axsvg:labelledBy
axsvg:chartType
Chart
http://.../membership.svg#BottomLegend
Line

```
<http://.../membership.svg#FullSlide>
    axsvg:graphicsType "Chart";
    axsvg:labelledBy <http://...#BottomLegend>;
    axsvg:chartType "Line".
```

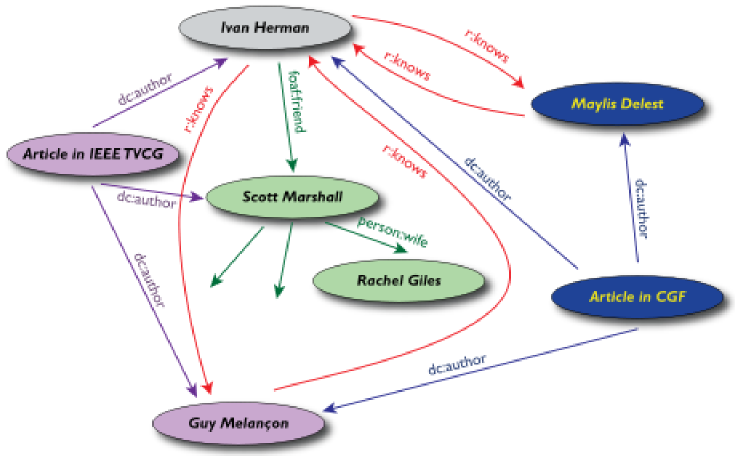
URI-s Play a Fundamental Role

- *Anybody* can create (meta)data on *any* resource on the Web
  - *e.g., the same SVG file could be annotated through other terms*
  - *semantics is added to existing Web resources via URI-s*
  - *URI-s make it possible to link (via properties) data with one another*
- *URI-s ground RDF into the Web*
  - *information can be retrieved using existing tools*
  - *this makes the “Semantic Web”, well… “Semantic Web”*

URI-s: Merging

- It becomes easy to *merge* data
  - *e.g., applications may merge the SVG annotations*
- Merge can be done because statements refer to the *same* URI-s
  - *nodes with identical URI-s are considered identical*
- Merging is a *very* powerful feature of RDF
  - *metadata may be defined by several (independent) parties…*
  - *…and combined by an application*
  - *one of the areas where RDF is much handier than pure XML in many applications*

What Merge Can Do...



What Merge Can Do…

See the “tabulator” example…

RDF Vocabulary Description Language (RDFS)
(a.k.a. RDFS)

Need for RDF Schemas

- Defining the data and using it from a program works… provided the program *knows* what terms to use!
- We used terms like:
  - `Chart`, `labelledBy`, `isAnchor`, …
  - `myCV`, `myCalendar`, …
  - *etc*
- Are they all known? Are they all correct? Are there (logical) relationships among the terms?
- This is where RDF Schemas come in
  - *officially: "RDF Vocabulary Description Language"; the term "Schema" is retained for historical reasons…*

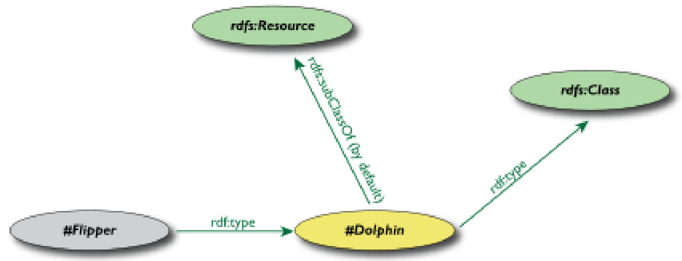
Classes, Resources, …

- Think of well known in traditional ontologies:
  - *use the term "mammal"*
  - *"every dolphin is a mammal"*
  - *"Flipper is a dolphin"*
  - *etc.*
- RDFS defines *resources* and *classes*:
  - *everything in RDF is a "resource"*
  - *"classes" are also resources, but…*
  - *they are also a collection of possible resources (i.e., "individuals")*
    - "mammal", "dolphin", …

Classes, Resources, … (cont.)

- Relationships are defined among classes/resources:
  - *“typing”: an individual belongs to a specific class (“Flipper is a dolphin”)*
  - *“subclassing”: instance of one is also the instance of the other (“every dolphin is a mammal”)*
- *RDFS formalizes these notions in RDF*

Classes, Resources in RDF(S)



- RDFS defines **`rdfs:Resource`**, **`rdfs:Class`** as nodes; **`rdf:type`**, **`rdfs:subClassOf`** as properties
  - *(these are all special URI-s, we just use the namespace abbreviation)*

Schema Example in RDF/XML

- The schema part (“application’s data types”):

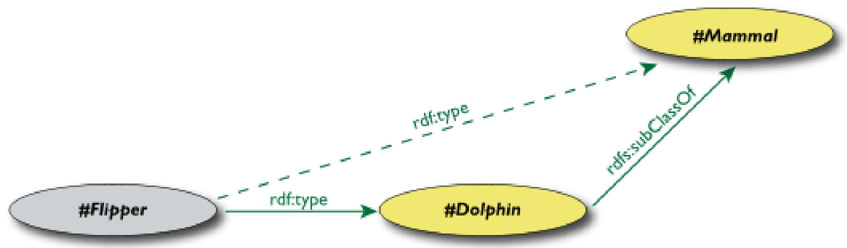
```
<rdf:Description rdf:ID="Dolphin">
  <rdf:type rdf:resource=
    "http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

- The RDF data on a specific animal (“using the type”):

```
<rdf:Description rdf:about="#Flipper">
   <rdf:type rdf:resource="animal-schema.rdf#Dolphin"/>
</rdf:Description>
```

- In traditional knowledge representation this separation is often referred to as: “Terminological axioms” and “Assertions”

Inferred Properties



- - `(#Flipper rdf:type #Mammal)`
- is *not* in the original RDF data…
- …but can be *inferred* from the RDFS rules
- Better RDF environments return that triplet, too

Inference: Let Us Be Formal…

- The RDF Semantics document has a list of (44) *entailment rules*:
  - *"if such and such triplets are in the graph, add this and this triplet"*
  - *do that recursively until the graph does not change*
  - *this can be done in polynomial time for a specific graph*
- The relevant rule for our example:

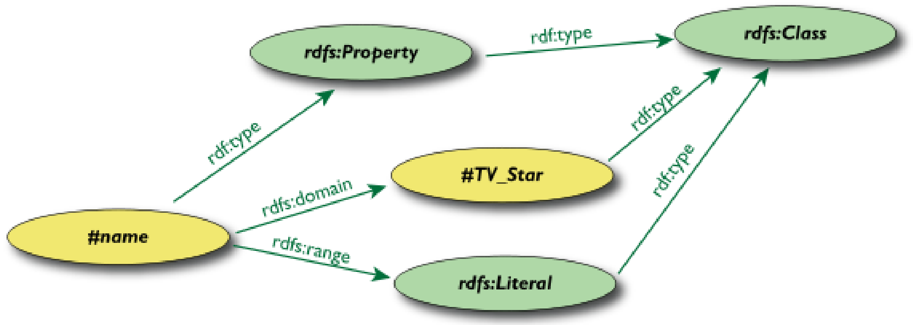
```
If:
  uuu rdfs:subClassOf xxx .
  vvv rdf:type uuu .
Then add:
  vvv rdf:type xxx .
```

- Whether those extra triplets are *physically added* to the graph, or *deduced* when needed is an implementation issue

Properties

- Property is a special class (`rdf:Property`)
  - *properties are also resources identified by URI-s*
- Properties are constrained by their range and domain
  - *i.e., what individuals can serve as object and subject*
- There is also a possibility for a "sub-property"
  - *all resources bound by the "sub" are also bound by the other*

Property Specification Example



- Note that one cannot define within the RDF(S) framework *what* literals can be used

Literals

- Literals may have a data type
  - *floats, integers, booleans, etc, defined in XML Schemas*
    - one can also define complex structures and restrictions via regular expressions, …
  - *full XML fragments*
- (Natural) language can also be specified (via **`xml:lang`**)

Literals Serialized

In RDF/XML

```
<rdf:Description rdf:about="#Flipper">
   <animal:is_TV_Star
      rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
         True
   </animal:is_TV_Star>
</rdf:Description/>
```

In Turtle

```
:Flipper
   animal:is_TV_Star "True"^^<http://www.w3.org/2001/XMLSchema#boolean>.
```

Querying RDF Data: SPARQL

How to retrieve triples?

- In simple cases: a programming environment gives direct access. E.g.:

```
triples = graph.triples((Resource,Property,None))
```

- to retrieve all objects for `(Resource,Property)`
- This is programming language dependent
- It becomes too complex for complex data
- Does not work if the data are remote
- We need *query* facilities

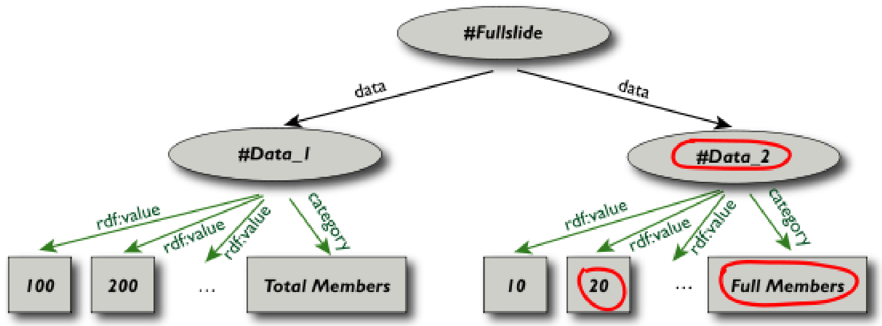
Querying RDF Graphs

- The fundamental idea: use *graph patterns* to define subgraphs:
  - *a pattern contains unbound symbols*
  - *by binding the symbols, subgraphs of the RDF graph may be matched*
  - *if there is such a match, the query returns the bound resources or a subgraph*
- This is the how SPARQL (Query Language for RDF) is defined
  - *is programming language-independent query language*
  - *is in Candidate Recommendation phase (Recommendation in 2006?)*

Simple SPARQL Example

```
SELECT ?cat ?val # note: not ?x!
WHERE { ?x rdf:value ?val. ?x category ?cat }
```

- Returns: `[["Total Members",100],["Total Members",200],…,["Full Members",10],…]`



Other SPARQL Features

- Define *optional* patterns
- Limit the number of returned results; remove duplicates, sort them,…
- Add functional constraints to pattern matching
- Return a full *subgraph* (instead of a list of bound variables)
- Use datatypes and/or language tags when matching a pattern
- SPARQL is a Candidate Recommendation, i.e. the technical aspects are now finalized (modulo implementation problems)
  - *but there are a number of implementations already!*

SPARQL Usage in Practice

- *Locally*, i.e., bound to a programming environments like Jena
- *Remotely*, e.g., over the network or into a database
  - *separate documents define the protocol and the result format*
    - SPARQL Protocol for RDF with HTTP and SOAP bindings
    - SPARQL Results XML Format
    - there is also a JSON binding (soon a W3C note…)
- There are already a number of applications, demos, etc.,

Get to RDF(S) Data

Simplest: Write your own RDF Data…

- The simplest aproach: write your own RDF data in your preferred syntax
- Using URI-s in RDF binds you automatically to the real resources
- You may add RDF to XML directly (in its own namespace)
  - *e.g., in SVG:*

```
<svg ...>
  ...
  <metadata>
    <rdf:RDF xmlns:rdf="http://../rdf-syntax-ns#">
      ...
    </rdf:RDF>
  </metadata>
    ...
</svg>
```

- Works in some cases, but not satisfactory for a real deployement!

RDF Can Also Be Extracted/Generated

- Use intelligent “scrapers” or “wrappers” to extract a structure (hence RDF) from a Web page…
  - *using conventions in, e.g., class names or header conventions like* `meta` *elements*
- … and then *generate* RDF automatically (e.g., via an XSLT script)
- Although they may not say it: this is what the “microformat” world is doing
  - *they may not extract RDF but use the data directly instead, but that depends on the application*
  - *other applications may extract it to yield RDF (e.g., RSS1.0)*

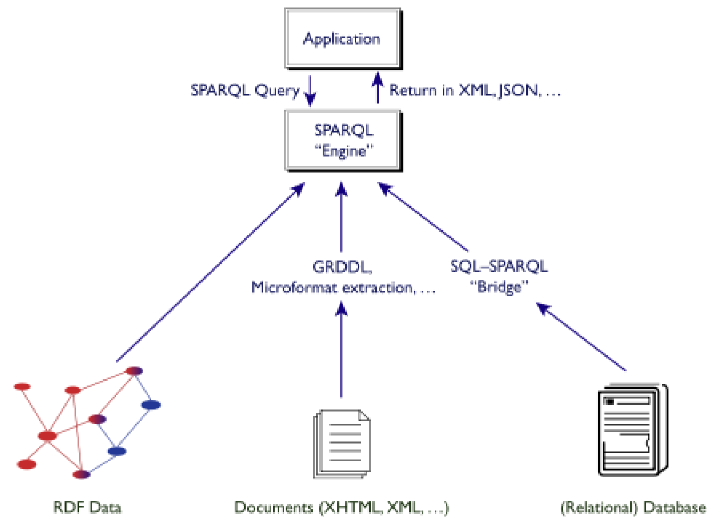
RDF Can Also Be Extracted/Generated (cont.)

- There are projects at W3C in this direction:
  - *RDFa: a scalable and more rigorous microformat approach (annotations are intermixed with XHTML content)*
  - *GRDDL: “formalizing” the scraping process (instructing the client where to find a transformator engine to extract the RDF data)*
- The two projects are complementary

Bridge to Relational Databases

- Most of the data are stored in relational databases
- “RDFying” them is an impossible task
- “Bridges” are being defined:
  - *a layer between RDF and the database*
  - *RDB tables are “mapped” to RDF graphs on the fly*
  - *in some cases the mapping is generic (columns represent properties, cells are, e.g., literals or references to other tables via blank nodes)…*
  - *… in other cases separate mapping files define the details*
- This is a *very* important source of RDF data
- Is instrumental to the fundamental goal of RDF: *qualified relationship among data on the Web*

SPARQL As a Unifying Force



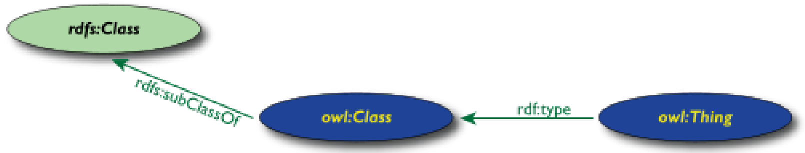
Ontologies (OWL)

Ontologies

- RDFS is useful, but does not solve all the issues
- Complex applications may want more possibilities:
  - *can a program reason about some terms? E.g.:*
    - “if «A» is left of «B» and «B» is left of «C», is «A» left of «C»?”
    - programs should be able to *deduce* such statements
  - *if somebody else defines a set of terms: are they the same?*
  - *construct classes, not just name them*
  - *restrict a property range when used for a specific class*
  - *disjointness or equivalence of classes*
  - *etc.*

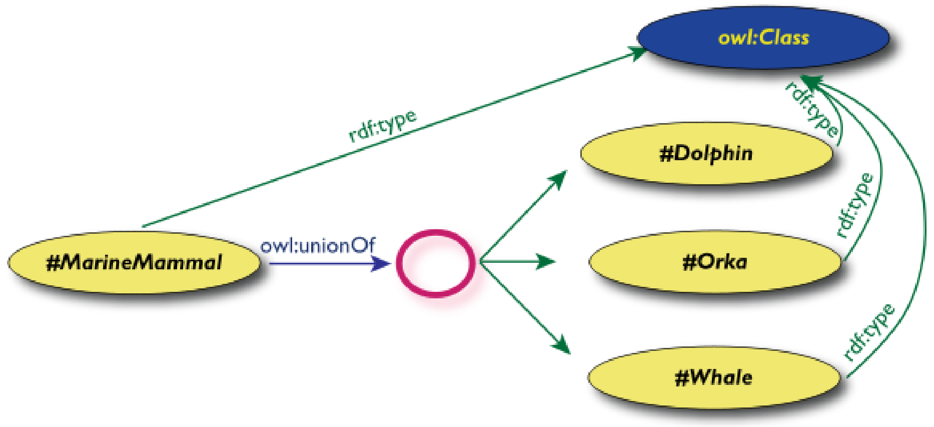
Classes in OWL

- In RDFS, you can subclass existing classes… that's all
- In OWL, you can *construct* classes from existing ones:
  - *enumerate its content*
  - *through intersection, union, complement*
  - *through property restrictions*
- To do so, OWL introduces its own **`Class`** and **`Thing`** to differentiate the *classes* from *individuals*



Union of Classes

- Essentially, like a set-theoretical union:



Same Serialized

```
<owl:Class rdf:ID="MarineMammal">
   <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Dolphin"/>
       <owl:Class rdf:about="#Orca"/>
       <owl:Class rdf:about="#Whale"/>
       …
   </owl:unionOf>
</owl:Class>
:Dolphin  rdf:type owl:Class.
:Orca     rdf:type owl:Class.
:Whale    rdf:type owl:Class.
:MarineMammal rdf:type owlClass;
   owl:unionOf (:Dolphin, :Orca, :Whale).
```

- Other possibilities: **`complementOf`**, **`intersectionOf`**

Property Restrictions

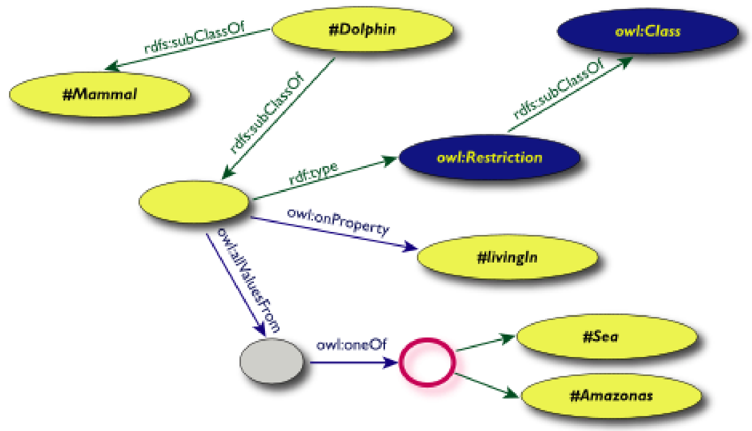
- (Sub)classes created by restricting the property value *on that class*
- For example, “a dolphin is a mammal living in sea or in the Amazonas” means:
  - *restrict the value of “living in” when applied to “mammal” to a specific set…*
  - *…thereby define the class of “dolphins”*

Property Restrictions in OWL

- Restriction may be by:
  - *value constraints (i.e., further restrictions on the range)*
    - *all* values must be from a class (like the dolphin example)
    - *some* value must be from a class
  - *cardinality constraints*
  - *(i.e., how many times the property can be used on an instance?)*
    - minimum cardinality
    - maximum cardinality
    - exact cardinality

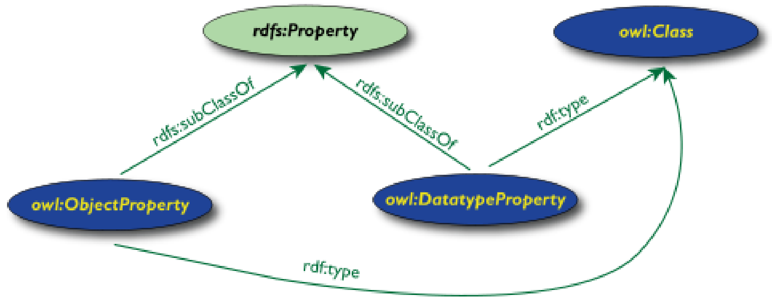
Property Restriction Example

- “A dolphin is a mammal living in the sea or in the Amazonas”:



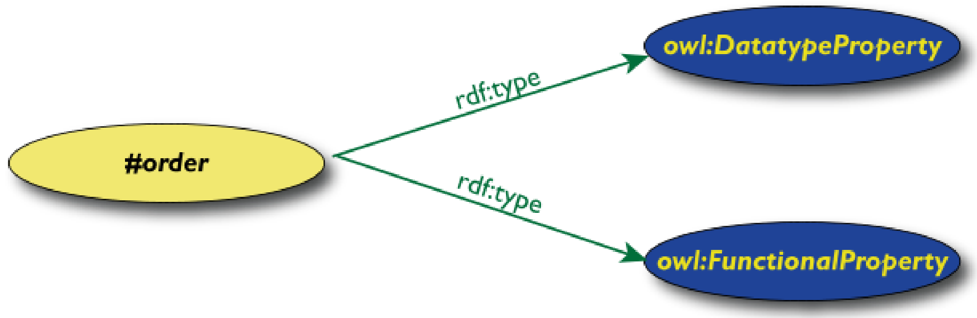
Property Characterization

- In OWL, one can characterize the *behavior* of properties (symmetric, transitive, …)
- OWL also separates data properties
  - *"datatype property" means that its range are typed literals*



Characterization Example

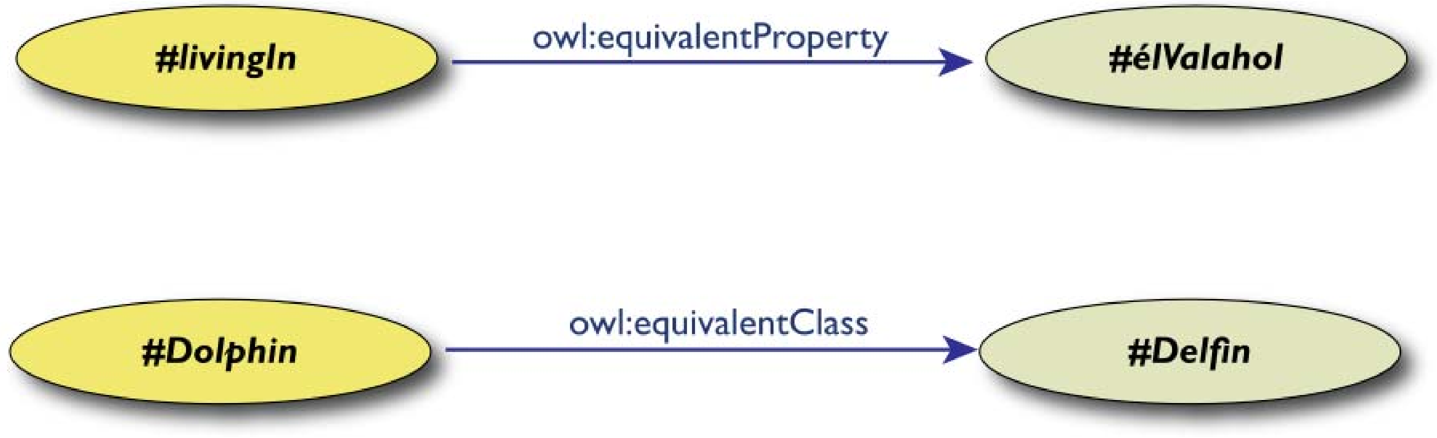
- "There should be only one order for each animal class" (in scientific classification)



OWL: Additional Features

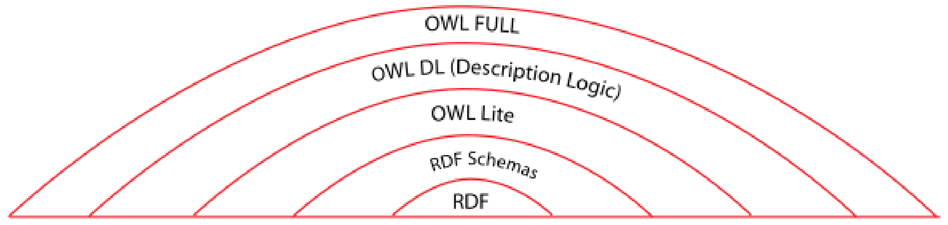
- Ontologies may be extremely a large:
  - *their management requires special care*
  - *they may consist of several modules*
  - *come from different places and must be integrated*
- Ontologies are *on the Web*. That means
  - *applications may use several, different ontologies, or…*
  - *… same ontologies but in different languages*
  - *equivalence of, and relations among terms become an issue*
- OWL includes possibilites for class/property equivalence, version and deprecation control, etc.

Example: Connecting to Hungarian



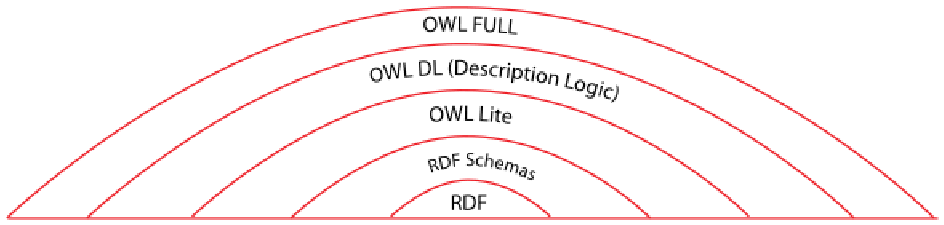
However: Ontologies are Hard!

- Hard to implement a full ontology management system
  - *may be superfluous for some applications*
- Hence the “onion” model of increasingly complex specs:
  - *no property expressions or datatypes in RDF Schemas*
  - *not all set operators, restricted cardinality in OWL Lite*
  - *some restrictions, but a computational guarantee in OWL DL*
  - *full expressive power in OWL Full (but no computational guarantee)*



Ontologies are Hard! (cont)

- "Lite" < "DL" < "Full", but not completely true for RDFS
  - *RDFS is "almost" a subcategory*
  - *not all RDFS statements are valid in DL…*
  - *…but they are for Full*
- Applications may take what they really need!



RDF/OWL in Practice

RDF in Programming Practice

- Programming environments usually provide:
  - *creation of a “Graph”, “Triple Store”, or “Model” (the terms used are different in various implementations)*
  - *the RDF file(s) are parsed and results stored in the Graph*
  - *the Graph offers methods to retrieve and/or remove triples, resources, etc*
  - *“merge” is automatic if several files are parsed*
  - *the rest is conventional programming…*

Example: Jena

- RDF toolkit in Java from HP’s Bristol lab

```
// create a model (a.k.a. Triple Store)
Model model = new ModelMem();
Resource subject = model.createResource("URI_of_Subject")
// 'in' refers to the input file
model.read(new InputStreamReader(in));
StmtIterator iter = model.listStatements(subject,null,null);
while(iter.hasNext()) {
   st = iter.next();
   p = st.getProperty();
   o = st.getObject();
   do_something(p,o);
}
```

Jena (cont)

- But Jena is *much* more than just the basics; it has
  - *a huge number of classes/methods*
    - listing, removing associated properties, objects, comparing full RDF graphs, manage typed literals, etc.
  - *an “RDFS Reasoner”*
    - all the “inferred” properties, types are accessible and errors are checked
  - *a SPARQL implementation*
  - *an OWL reasoner*
  - *a layer (Joseki) for remote access of triples (essentially, and triple database)*
- Probably the most widely used RDF environment in Java today

Lots of Other tools

- There are *lots* of other tools:
  - *RDF frameworks for specific languages: RDFStore (Perl), RAP (PHP, includes a SPARQL engine), SWI-Prolog (Prolog), RDFLib for Python,…*
  - *Redland: general RDF Framework, with bindings to C, C++, C#, Python, …, and with a SPARQL engine (Rasqal)*
  - *RDF storage systems: (Sesame, Kowari, Tucana, Gateway, @Semantics RDFStore, Virtuoso, 3Store, Jena's Joseki, InferEd, Oracle Database 10g , Allegro…)*
    - some of these are based on an internal sql engine (3Store, Oracle), others are made bottom up as triple stores
    - most of them have, or plan for, SPARQL facilities
  - *RDB→RDF layers, tools: D2R Server, SquirrelRDF*
- See the tool list at W3C or the Free University of Berlin list

Current/future Developments

Simple Knowledge Organization System

- Goal: porting ("Webifying") thesauri: representing and sharing classifications, glossaries, thesauri, etc, as developed in the "Print World". For example:
  - *Dewey Decimal Classification, Art and Architecture Thesaurus, ACM classification of keywords and terms…*
  - *DMOZ categories (a.k.a. Open Directory Project)*
- The system must be simple to allow for a quick port of traditional data (done by "traditional" people…)
- *This is where SKOS comes in*

Example: Entries in a Glossary (1)

"Assertion"

> "(i) Any expression which is claimed to be true. (ii) The act of claiming something to be true."

"Class"

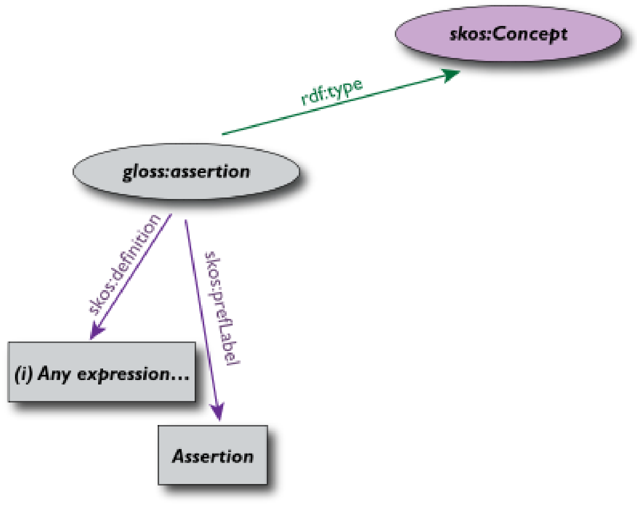
> "A general concept, category or classification. Something used primarily to classify or categorize other things."

"Resource"

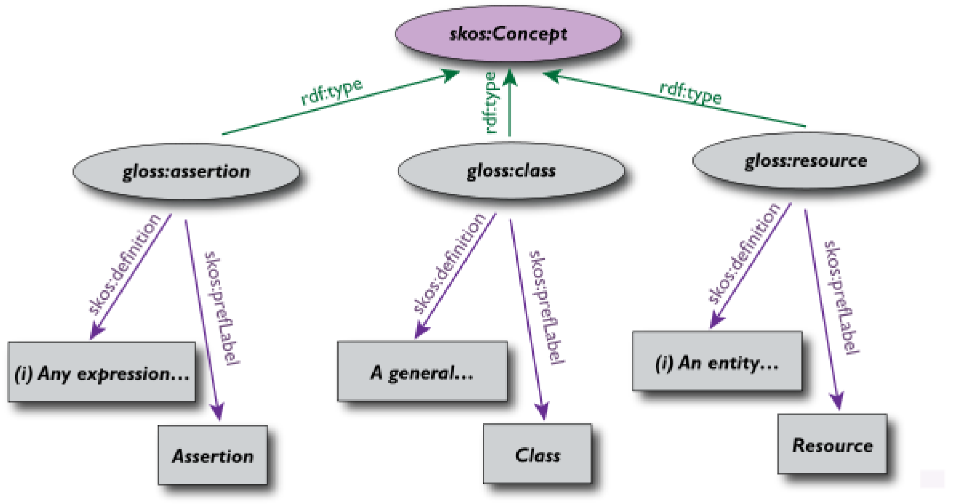
> "(i) An entity; anything in the universe. (ii) As a class name: the class of everything; the most inclusive category possible."

(from the RDF Semantics Glossary)

Example: Entries in a Glossary (2)



Example: Entries in a Glossary (3)

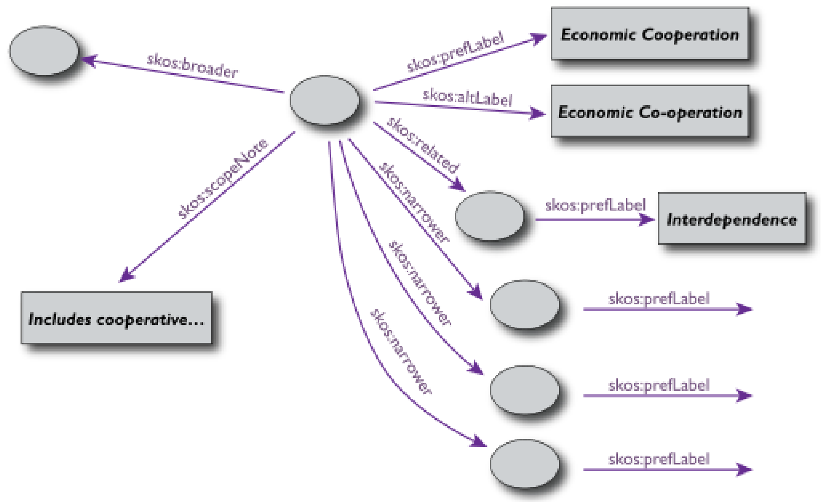


Example: Thesaurus (1)

Term	
	Economic cooperation
Used For	
	Economic co-operation
Broader terms	
	Economic policy
Narrower terms	
	Economic integration, European economic cooperation, …
Related terms	
	Interdependence
Scope Note	
	Includes cooperative measures in banking, trade, …

(from UK Archival Thesaurus)

Example: Thesaurus (2)



Why Having SKOS OWL?

- OWL’s precision not always necessary or even appropriate
    - *“OWL a sledge hammer / SKOS a nutcracker”, or “OWL a Harley / SKOS a bike”*
    - *complement each other, can be used in combination to optimize cost/benefit*
- Role of SKOS is
    - *to bring the worlds of library classification and Web technology together*
    - *to be simple and undemanding enough in terms of cost and required expertise*
- A typical example: the Glossary of project of W3C stores all terms in SKOS (and extracted from W3C documents)

“Core” Vocabularies

- A number of public “core” vocabularies evolve to be used by applications, e.g.:
  - *SKOS Core: about knowledge systems*
  - *Dublin Core: about information resources, digital libraries, with extensions for rights, permissions, digital right management*
  - *FOAF: about people and their organizations*
  - *DOAP: on the descriptions of software projects*
  - *MusicBrainz: on the description of CDs, music tracks, …*
  - *…*
- They share the underlying RDF model (provides mechanisms for extensibility, sharing, …)

Rules

- OWL can be used for simple inferences
- Applications may want to express domain-specific knowledge, like “Horn clauses”:
  - *$(P_1 \land P_2 \land \ldots) \rightarrow C$*
  - *e.g.: for any «X», «Y» and «Z»: “if «Y» is a parent of «X», and «Z» is a brother of «Y» then «Z» is the uncle of «X»”*
- There is also a large corpus of rule–based systems and languages, though not necessarily bound to the Web (yet)
- Several attempts already to combine Semantic Web with Rules (Metalog, RuleML, SWRL, WRL, cwm, …)
  - *note: cwm, for example, defines Horn predicates in terms of graph patterns, a connection to SPARQL…*

Rules Interchange Format Working Group

- The W3C Working Group started at the beginning of November 2005
- Work is planned in two “phases”:
  1. *construct an extensible format for rule interchange*
  2. *define more complex extensions*
- Great interest from financial services, business rules, life science community…

RIF Phase 1 Goals

- An *interchange format* to exchange rules among rule engines and systems
  - *probably based on "full Horn Logic" with some simple datatypes (int, boolean, strings,…)*
  - *make it relatively simple, leave the more complex issues to Phase 2*
  - *make a new type of data accessible for the Web…*
- An *extensible format* to allow more complex alternatives to be defined
  - *e.g., fuzzy and/or temporal logic*
- Recommendation planned in May 2007

RIF Use Cases and Requirements

- The first draft has just been published
- Contains a number of use cases, e.g.:
  - *negotiating eBusiness contracts across rule platforms: supply vendor-neutral representation of your business rules so that others may find you*
  - *describing privacy requirements and policies, and let client “merge” those (e.g., when paying with a credit card)*
  - *medical decision support, combining rules on diagnoses, drug prescription conditions, etc,*
  - *extending OWL with rule-based statements (e.g., the uncle example)*

RIF Phase 2 Goals

- Define more complex extensions
  - *towards First Order Logic (FOL), Logic Programming systems…*
  - *syntactic extensions to Horn logic like Lloyd-Topor*
  - *actions, i.e., running procedural codes as part of rules*
- First recommendation(s) planned in May 2008

Lots of Theoretical Questions to Solve

- Open vs. Closed Worlds, monotonicity vs. non-monotonicity (see, e.g., WW2006 paper for more details on this)
- How to use various logic systems (Description Logic, F-Logic, Horn, Business Rules,…) in a coherent framework
- Relationships to RDFS, OWL

Beyond Rules: Trust

- Can I trust a (meta)data on the Web?
  - *is the author the one who claims he/she is, can I check his/her credentials?*
  - *can I trust the inference engine?*
  - *etc.*
- There are issues to solve, e.g.,
  - *how to "name" a full graph*
  - *protocols and policies to encode/sign full or partial graphs (blank nodes may be a problem to achieve uniqueness)*
  - *how to "express" trust? (e.g., trust in context)*
- It is on the "future" stack of W3C and the SW Community …

Other Issues…

- Improve the inference algorithms and implementations, scalability, reasoning with OWL Full
- Better modularization (import or refer to *part of* ontologies)
- Ontology management on the Web
- Extensions of RDF and/or OWL (based on experience and theoretical advances)
- “Sub OWL Lite” ontology level for some applications (RDFS++, OWL Feather, pD*,…)
- Temporal & spatial reasoning
- Probabilistic reasoning and/or fuzzy logic
- …

Available Documents

Some Books

- J. Davies, D. Fensel, F. van Harmelen: Towards the Semantic Web (2002)
- S. Powers: Practical RDF (2003)
- D. Fensel, J. Hendler: Spinning the Semantic Web (2003)
- F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider: The Description Logic Handbook (2003)
- G. Antoniu, F. van Harmelen: Semantic Web Primer (2004)
- A. Gómez-Pérez, M. Fernández-López, O. Corcho: Ontological Engineering (2004)
- …

Further Information

- Dave Beckett’s Resources at Bristol University
  - *huge list of documents, publications, tools, …*
- Semantic Web Community Portals, e.g.:
  - *Semanticweb.org*
  - *“Business model IG” (part of semanticweb.org)*
  - *list documents, software, host project pages, etc,…*
- The Semantic Web Activity page at W3C lists a number of commercial tools

SWBP Working Group Documents

- Documents for ontology engineering
- Semantic Web Tutorials (list of references)
- Survey of RDF/Topic Map Maps Interoperability
- “Ontology Driven Architectures in Software Engineering”

Further Information (cont)

- Description Logic links:
  - *online course by Enrico Franconi,*
  - *teaching material and links by Ian Horrocks*
- “Ontology Development 101”
- OWL Reasoning Examples
- *Lots* of papers at WWW2003, WWW2004, WWW2005, and WWW2006; see also the ISWC200X conference proceedings (unfortunately, not on-line…)

Public Fora at W3C

Semantic Web Interest Group

- a forum for discussions on applications

RDF Logic

- public (archived) mailing list for technical discussions

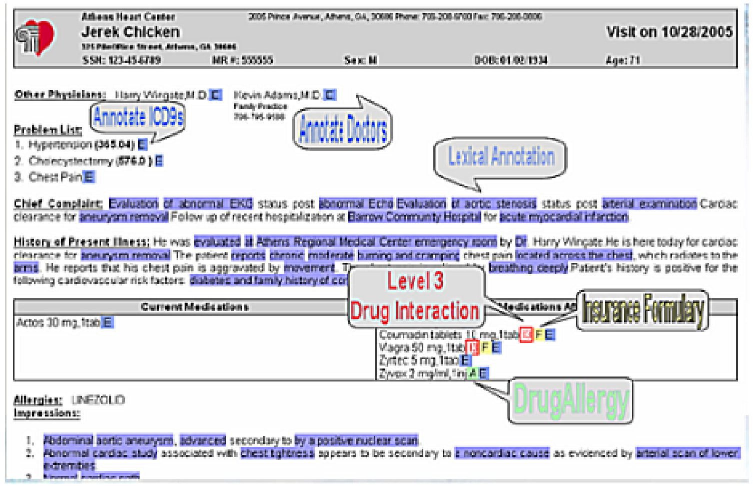
Some Application Examples

SW Applications

- Large number of applications emerge
- Most applications are still “centralized”, not many decentralized applications yet
- Large datasets are accumulating. E.g.:
    - *RDF version of Wikipedia: more than 47 million triplets, based also on SKOS, soon with a SPARQL interface*
    - *tracking the US Congress: data stored in RDF (around 25 million triplets) with a SPARQL interface*
    - *“Département/canton/commune” structure of France published by the French Statistical Institute*
- “Corporate Semantic Web” listed as major technology by Gartner
- For further examples, see, for example, the Semantic Technology Conference series
    - *not a scientific conference, but commercial people making real money!*
    - *speakers in 2006: from IBM, Cisco, BellSouth, GE, Walt Disney, Nokia, Oracle, …*

Applications are not always very complex…

- Eg: simple semantic annotations of patients' data greatly enhances communications among doctors
- What is needed: some simple ontologies, an RDFa/microformat type editing environment
- Simple but powerful!



Athens Heart Center 2005 Prince Avenue, Athens, GA, 30606 Phone: 706-208-6700 Fax: 706-208-0806

Jerek Chicken Visit on 10/28/2005

SSN: 123-45-6789 MR #: 555555 Sex: M DOB: 01/02/1934 Age: 71

Other Physicians: Harry Wingate,M.D. C Kevin Adams,M.D. C Family Practice 706-795-9588

Problem List:
1. Hypertension (365.04) E
2. Cholecystectomy (576.0) E
3. Chest Pain E

Chief Complaint: Evaluation of abnormal EKG status post abnormal Echo Evaluation of aortic stenosis status post arterial examination Cardiac clearance for aneurysm removal Follow up of recent hospitalization at Barrow Community Hospital for acute myocardial infarction.

History of Present Illness: He was evaluated at Athens Regional Medical Center emergency room by Dr. Harry Wingate He is here today for cardiac clearance for aneurysm removal The patient reports chronic moderate burning and cramping chest pain located across the chest, which radiates to the arms. He reports that his chest pain is aggravated by movement. T[illegible] breathing deeply Patient's history is positive for the following cardiovascular risk factors: diabetes and family history of co[illegible]

Current Medications	Medications A[illegible]
Actos 30 mg, 1tab E	Coumadin tablets 10 mg, 1tab I F E Viagra 50 mg, 1tab I F E Zyrtec 5 mg, 1tab E Zyvox 2 mg/ml, 1inj A E

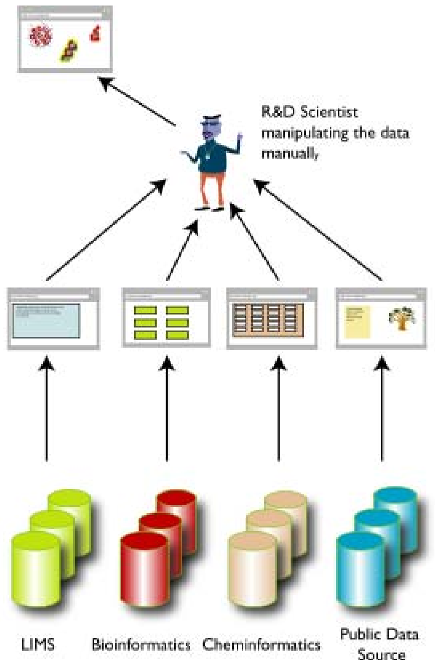
Allergies: LINEZOLID
Impressions:

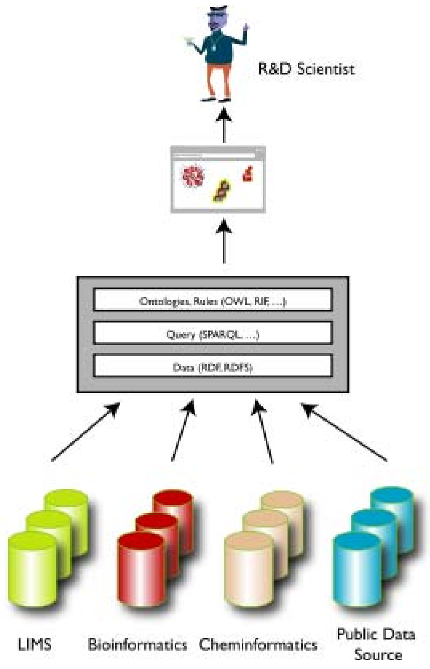
1. Abdominal aortic aneurysm, advanced secondary to by a positive nuclear scan.
2. Abnormal cardiac study associated with chest tightness appears to be secondary to a noncardiac cause as evidenced by arterial scan of lower extremities
3. Normal cardiac cath

Data integration

- Data integration comes to the fore as one of *the* SW Application areas
- Very important for large application areas (life sciences, energy sector, eGovernment, financial institutions), as well as everyday applications (eg, reconciliation of calendar data)
- Life sciences example:
  - *data in different labs…*
  - *data aimed at scientists, managers, clinical trial participants…*
  - *large scale public ontologies (genes, proteins, antibodies, …)*
  - *different formats (databases, spreadsheets, XML data, XHTML pages)*
  - *etc*

Life Sciences (cont.)





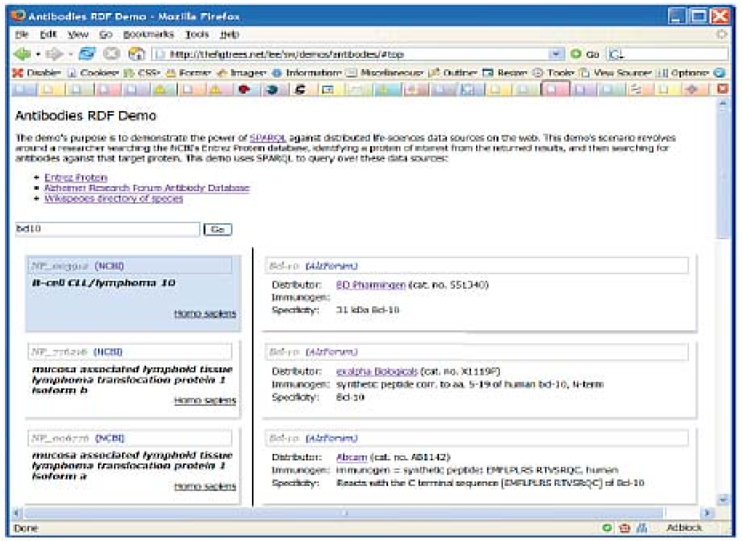
General approach

1. Map the various data onto RDF
    - *“mapping” may mean on-the-fly SPARQL to SQL conversion, “scraping”, etc*
2. Merge the resulting RDF graphs (with a possible help of ontologies, rules, etc, to combine the terms)
3. Start making queries on the whole!

- Remember the role of SPARQL?

Example: antibodies demo

- Scenario: find the known antibodies for a protein in a specific species
- Combine (“scrape”…) three different data sources
- Use SPARQL as an integration tool (see also demo online)



Antibodies RDF Demo - Mozilla Firefox

http://thefigtrees.net/lee/sw/demos/antibodies/#top

Antibodies RDF Demo

The demo's purpose is to demonstrate the power of SPARQL against distributed life-sciences data sources on the web. This demo's scenario revolves around a researcher searching the NCBI's Entrez Protein database, identifying a protein of interest from the returned results, and then searching for antibodies against that target protein. This demo uses SPARQL to query over these data sources:

- Entrez Protein
- Alzheimer Research Forum Antibody Database
- Wikispecies directory of species

bcl10

NP_[illegible] (NCBI)
B-cell CLL/lymphoma 10
Homo sapiens

NP_[illegible] (NCBI)
mucosa associated lymphoid tissue lymphoma translocation protein 1 isoform b
Homo sapiens

NP_[illegible] (NCBI)
mucosa associated lymphoid tissue lymphoma translocation protein 1 isoform a
Homo sapiens

Bcl-10 (AlzForum)
Distributor: BD Pharmingen (cat. no. 551340)
Immunogen:
Specificity: 31 kDa Bcl-10

Bcl-10 (AlzForum)
Distributor: exalpha Biologicals (cat. no. X1119P)
Immunogen: synthetic peptide corr. to aa. 5-19 of human bcl-10, N-term
Specificity: Bcl-10

Bcl-10 (AlzForum)
Distributor: Abcam (cat. no. AB1142)
Immunogen: immunogen = synthetic peptide: EMFLPLRS RTVSRQC, human
Specificity: Reacts with the C terminal sequence [EMFLPLRS RTVSRQC] of Bcl-10

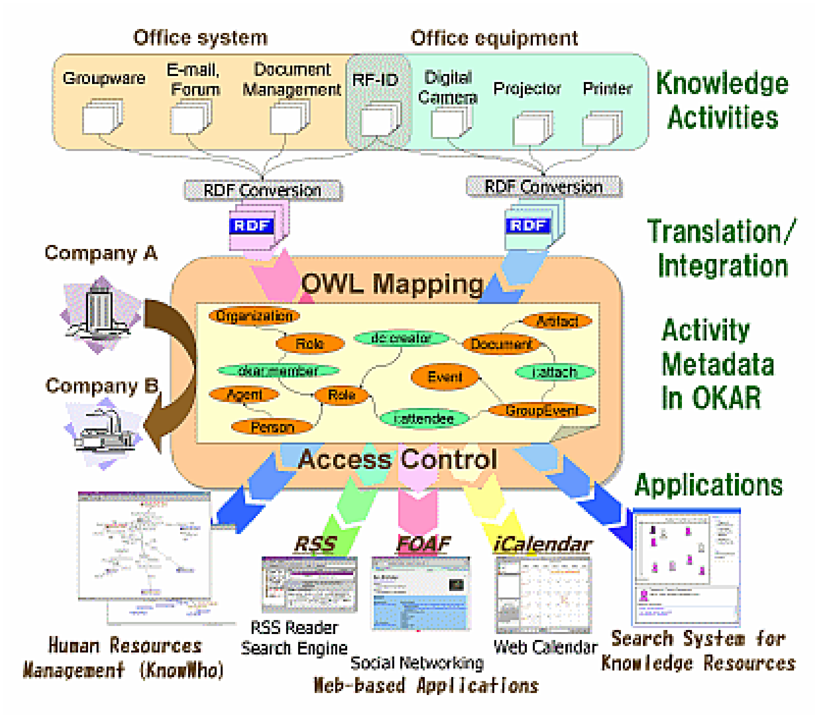
File Edit View Go Bookmarks Tools Help
Go
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Go
Done
Adblock

Portals

- Vodafone's Live Mobile Portal
  - *search application (e.g. ringtone, game, picture) using RDF*
    - page views per download decreased 50%
    - ringtone up 20% in 2 months
- Sun's SwordFish: public queries for support, handbooks, etc, go through an internal RDF engine for White Paper Collections and System Handbook collections
- Nokia has a somewhat similar support portal
- Harper's Online magazine links items together via an internal ontology

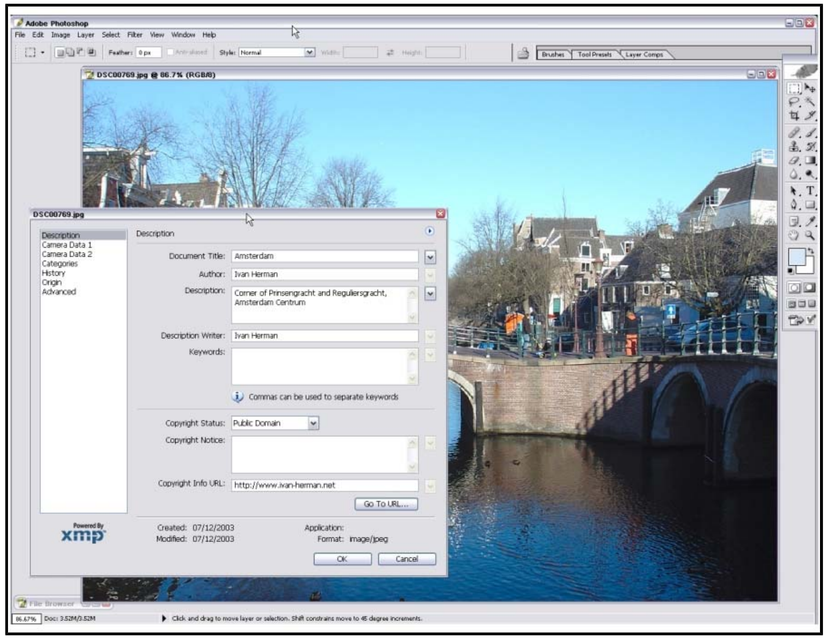
OKAR Fujitsu's and Ricoh's OKAR

- Management of office information, projects, personal skills, calendars, …
  - *e.g., "find me a person with a specific skill"*
- Still an R&D project



Adobe's XMP

- Adobe’s tool to add RDF-based metadata to *most* of their file formats
  - *used for more effective organization*
  - *supported in Adobe Creative Suite*
  - *support from 30+ major asset management vendors, with separate XMP conferences*
- The tool is available for all!



Description

Document Title: Amsterdam
Author: Ivan Herman
Description: Corner of Prinsengracht and Reguliersgracht, Amsterdam Centrum
Description Writer: Ivan Herman
Keywords:

Commas can be used to separate keywords

Copyright Status: Public Domain
Copyright Notice:
Copyright Info URL: http://www.ivan-herman.net

Created: 07/12/2003
Modified: 07/12/2003
Application:
Format: image/jpeg

86.67% Doc: 3.52M/3.52M
Click and drag to move layer or selection. Shift constrains move to 45 degree increments.

Further Information

These slides are at:

http://www.w3.org/People/Ivan/CorePresentations/SW_Advanced/

Semantic Web homepage

http://www.w3.org//2001/sw/

More information about W3C:

http://www.w3.org/Consortium/

Contact information:

http://www.w3.org//Consortium/Contact

Mail me:

ivan@w3.org