Metadatos

Metadatos

Significa	Datos sobre los datos
Sirve para	Sirven para expresar datos adicionales sobre un concepto o documento En la Web son esenciales para expresar esta información de forma inequívoca (vía namespaces)
Ejemplos	Datos sobre un documento: El Quijote <autor> Cervantes en la novela la relación autor="Cervantes" no forma parte de la trama, <autor> es un dato adicional, un metadato Datos sobre un concepto: Coche <color> rojo <color> es un dato sobre el objeto "coche", un metadato
Expresión	En la Web se expresa: -Dentro de las etiquetas Meta de HTML -Como Qnames en XML -Como propiedades y clases en RDF y OWL

Metadatos, principios

Metadatos para
- legibilidad e interoperabilidad
- uso para agentes de software y sistemas de búsquedas

Principios de los esquemas de metadatos
- Modularidad: que permita ensamblarse con otros vocabularios
- Extensibilidad: el esquema base debe poderse adecuar con elementos adicionales a las necesidades locales sin perjudicar interoperabilidad
- Refinamiento: por ejemplo con calificadores que especifican un elemento (de autor-coautor) o mediante vocabularios controlados
- Plurilingüísmo

Registros: una declaración de como se utiliza un vocabulario y como se implementa

Perfiles de metadatos: conjunto de elementos extraídos de uno o varios esquemas combinados por implementadores y optimizados para una aplicación particular

Algunos Vocabularios de Metadatos

- SKOS
  - Para crear tesauros y glosarios con XML/RDF
- Dublin Core
  - Para referenciar documentos (quién es el autor, qué titulo tiene,…)
- FOAF
  - Friend Of a Friend, para crear comunidades de usuarios.
- RSS
  - Para publicar simultáneamente en varios sitios noticias y dar alertas sobre su aparición (sindicación)

Elemento DC y cualificadores

Elemento DCMES	Elemento refinado	Sistema de codificación
Title	Alternative	
Creator		
Subject		LCSH MeSH DDC LCC UDC
Description	Table Of Contents Abstract	
Publisher		
Contributor		
Date	Created Valid Available Issued Modified	DCMI Period W3C-DTF
Type		DCMI Type Vocabulary
Format	Extent	
	Medium	IMT
Identifier		URI

Elementos DC y cualificadores

Source		URI
Language		ISO 639-2 RFC 1766
Relation	Is Version Of Has Version Is Replaced By Replaces Is Required By Requires Is Part Of Has Part Is Referenced By References Is Format Of Has Format	URI
Coverage	Spatial	DCMI Point ISO 3166 DCMI Box TGN
	Temporal	DCMI Period W3C-DTF
Rights		

Dublin Core: elementos (3)

Se cambia la designación de las siguientes etiquetas:

- Subject and Keywords
- Author or Creator
- Other Contributors
- ResourceType
- Format
- Resource Identifier

No es obligatorio utilizar todas las etiquetas para describir un documento. Por otra parte, es posible repetir cualquier

Ejemplo de metadatos para HTML utilizando Dublin Core

<META NAME="Title" CONTENT="FrontOffice selects Verity for Microsoft Exchange basad document management system">
<META NAME="DC.Author" CONTENT="Padovani, Marguerite">

<META NAME="DC.Author" CONTENT="Siegel, Gail">

<META NAME=" DC.Publisher" CONTENT="Verity Inc.">

<META NAME=" DC.Date" CONTENT="1996">

<META NAME=" DC.Object" CONTENT="Press Release">

<META NAME=" DC.Form" CONTENT="1 ASCII file">

<META NAME=" DC.Language" CONTENT="English">

<META NAME="DC.title.alternative" CONTENT="Chaos in Shiva-Like Proportions and how the DC can whittle that Puppy Down">

SKOS: ejemplo

Simple Knowledge Organisation System (SKOS) http://www.w3.org/2004/02/skos/

Term: Economic cooperation

Used For:

Economic co-operation

Broader terms:

Economic policy

Narrower terms:

Economic integration

European economic cooperati

European industrial cooperation

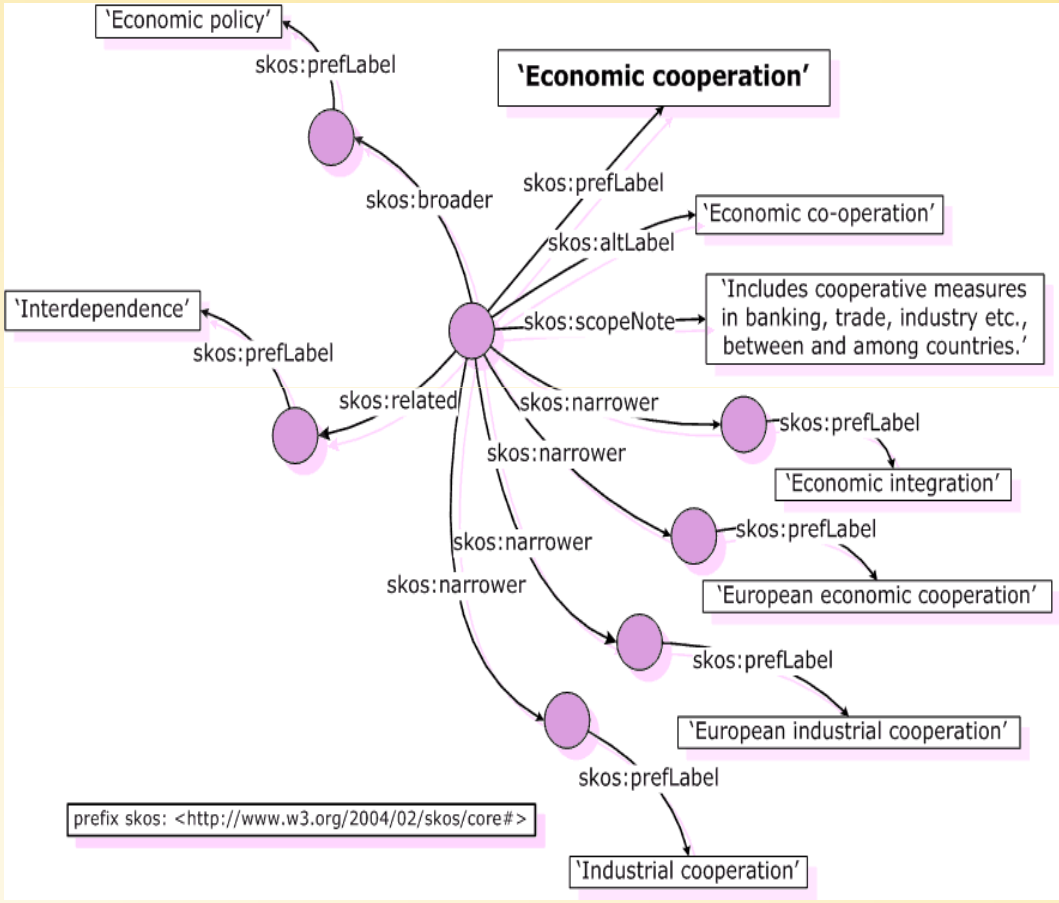
Industrial cooperation

Related terms:

Interdependence

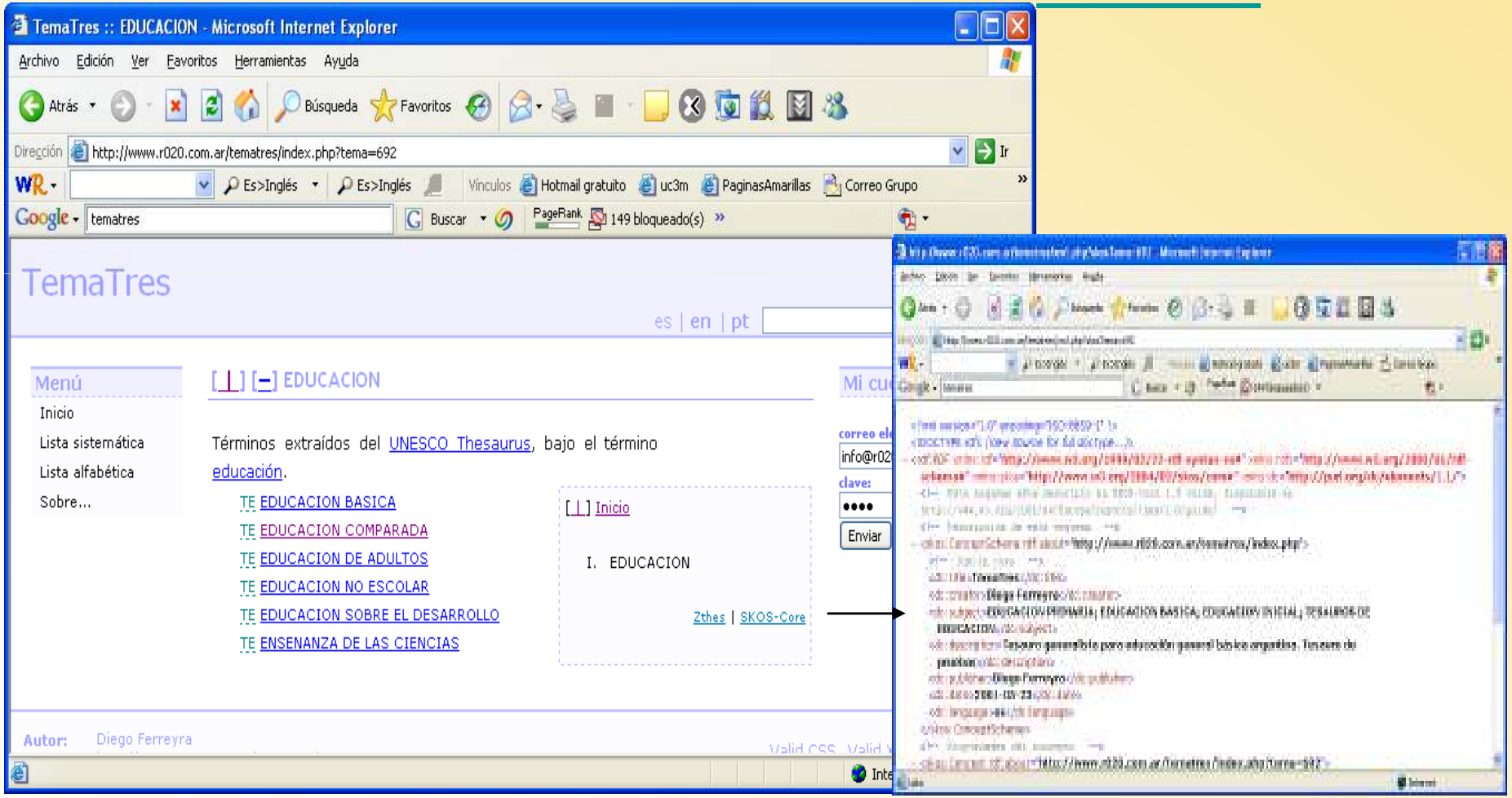
Scope Note:

Includes banking, trade, industry etc., between and among countries.



Tesauros y SKOS: Editores

- TEMATres www.r020.com.ar/tematres/



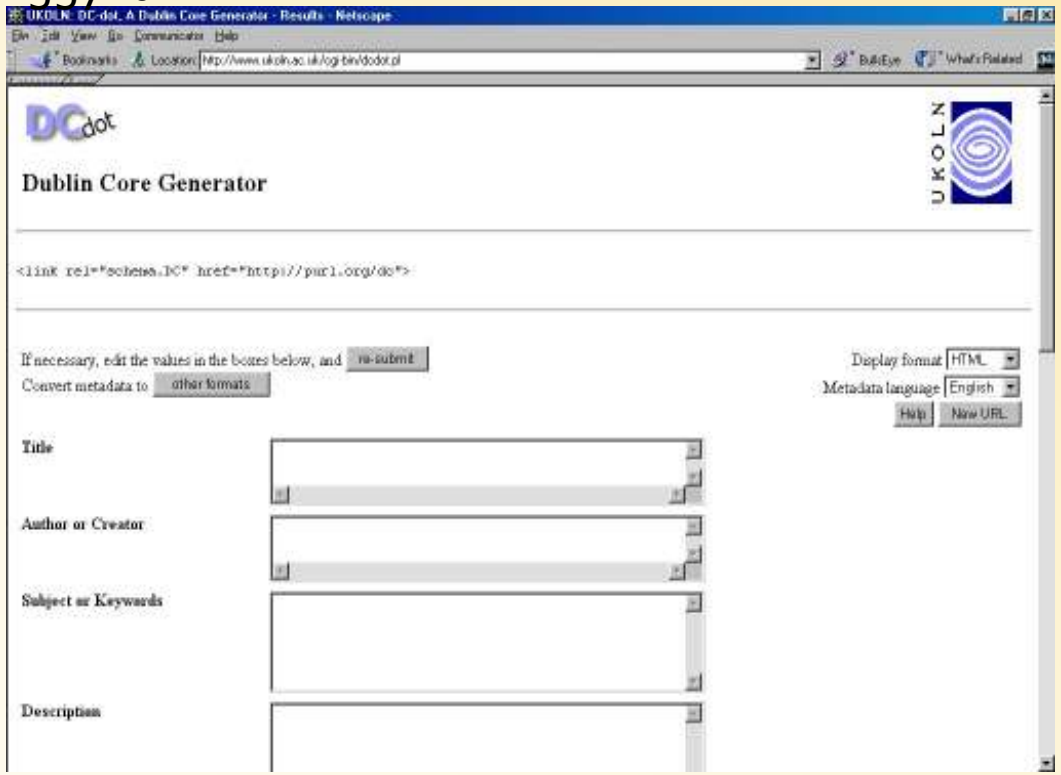
Editores

Metadatos

Generador de Metadatos

http://www.ukoln.ac.uk/cgi-bin/dcdot.pl

Extractor de metadatos de una página y los convierte a otros formatos. Otro recolector que permite incluso incorporar metadatos en forma de ontologías es un plugin de firefox llamado My PiggyBank

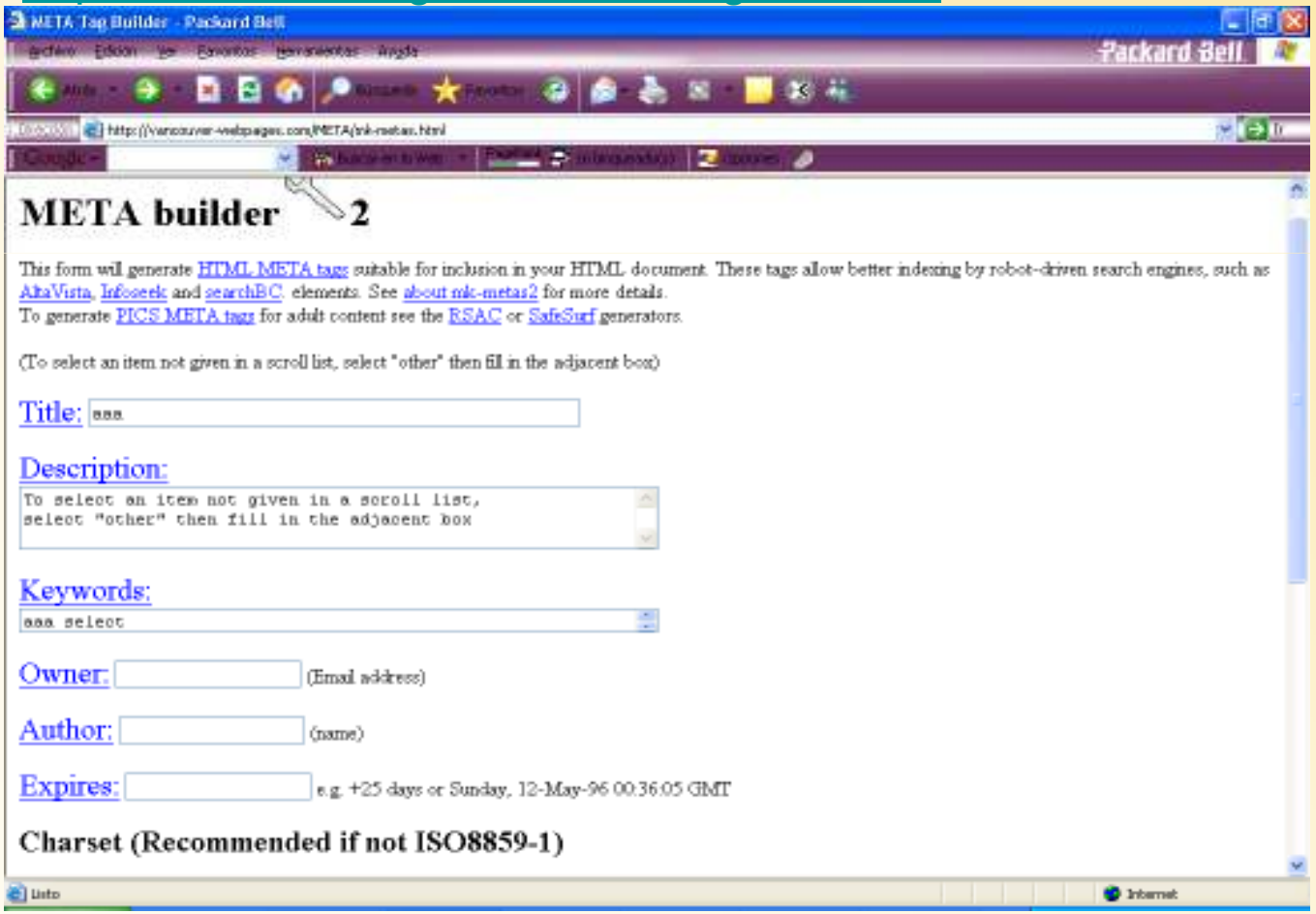


Editores Metadatos

http://vancouver-webpages.com/META/mk-metas.html

Codifica en html los metadatos

Se pueden ver muchos más editores en
http://dublincore.org/tools/#creatingmetadata

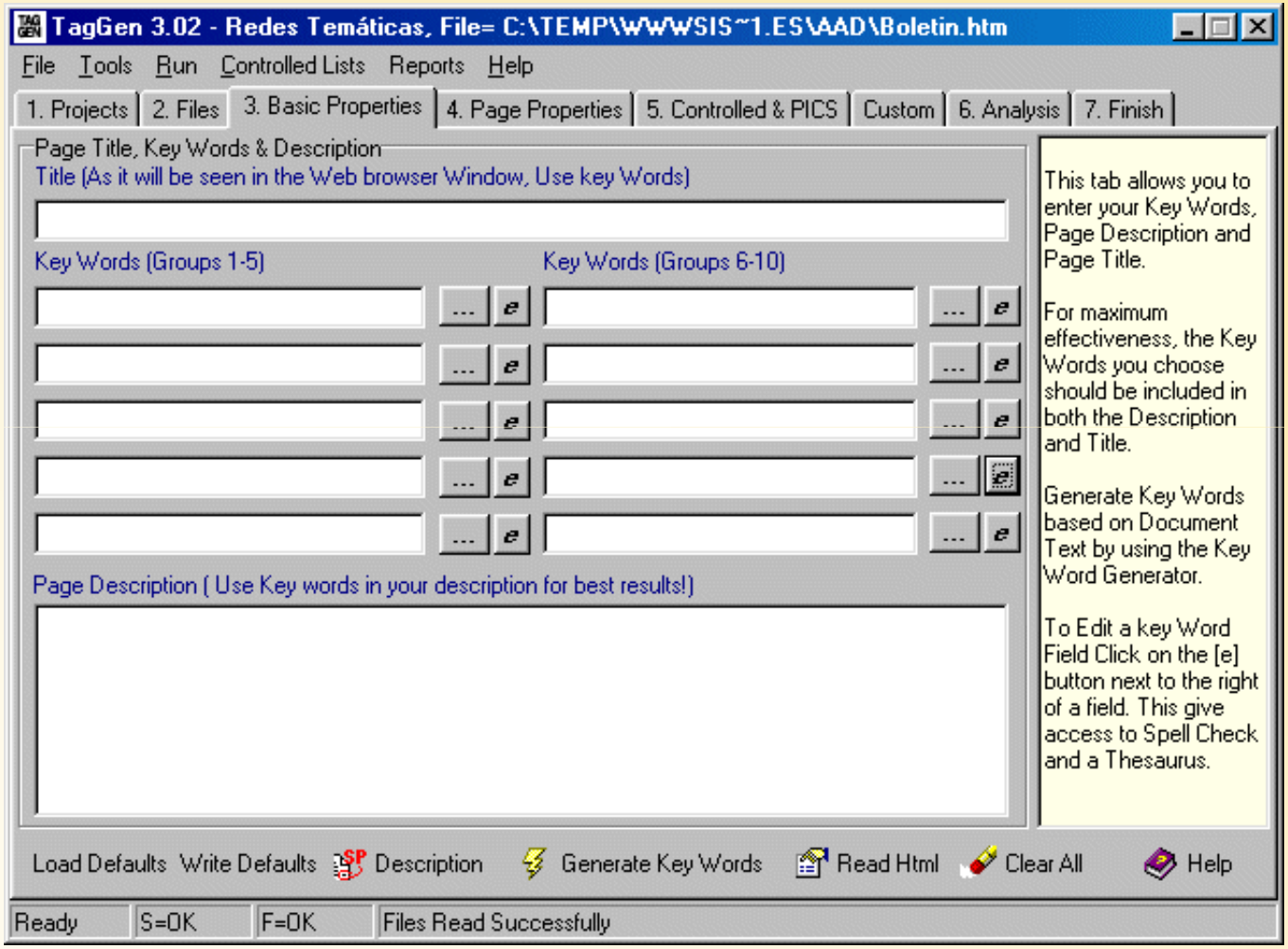


Editores Metadatos

Reggie http://metadata.net/dstc

Editores Metadatos

Clientes (TagGen, Metatag, HotMetal...)



Web Semántica

Evolución de la Web

- Se observan dos trayectorias del Web: la Web Semántica y la Web Social (W2.0)
- La relación entre ambas es confusa, pero las dos proveen de mecanismos para compartir información y recursos
- En esta presentación se analiza su naturaleza y las vías de evolución

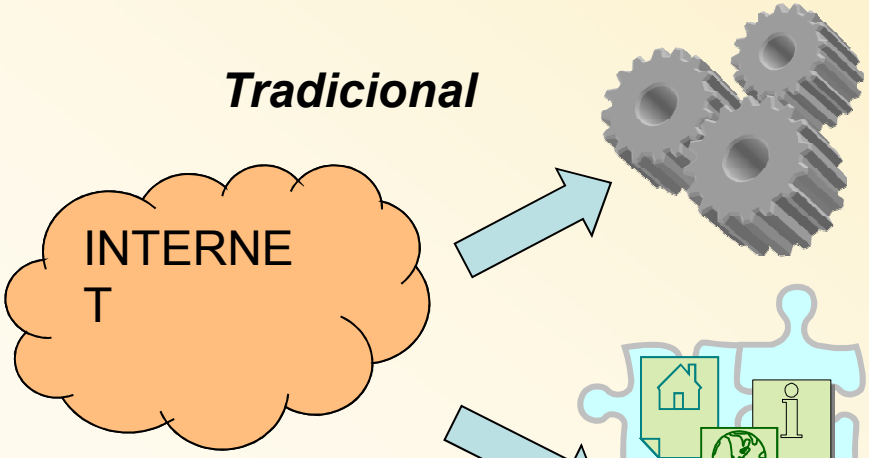
La Web Semántica

- Ejemplo:
  - Página 1: “Página personal sobre la vida de Marilyn Monroe”
  - Página 2: “..supón que eres Marilyn Monroe y que…”

  Ahora busca con un buscador Web información relevante sobre “biografía de Norma Jeane Mortenson ”

  Dos formas de **mejorar la Web**:

Tradicional



Web Semántica

Mejora Funcionamiento con:

-Procesamiento automático del lenguaje (PLN)

- Algoritmos de Inteligencia Automática (IA) y de posicionamiento

Mejora interoperabilidad y legibilidad automática:

-Crear documentos estructurados semánticamente y formalizados

- Crear software y estándares para incorporar estos documentos

Web 2.0

- Propuesta por O´Really 2004 por constatar la evolución del Web
- Alto grado de colaboración
- Voluntad de compartir recursos
- El usuario es el centro de atención: decide qué y cómo usarlo
- Interfaces y usabilidad altas. Representación semántica poco compleja
- Sin autoridad central
- Dificultad de crear aplicaciones para utilizarlas al no estar normalizados

¿Qué son las Folksonomías?

- Thomas van der Wal quién fusionó las palabras *folk* (gente, popular) y -*taxis*-
- *Clasificación gestionada popularmente*
  - Son conjuntos de **palabras clave** incorporadas y asignadas por los **internautas** para colaborar en la indización de cualquier tipo de contenidos en un **espacio compartido y abierto**
  - Indización sin ánimo de lucro y sin la supervisión de un organismo centralizador
  - Ausencia de relaciones entre los términos (Mathes, 2004)
- Popular en la Web 2.0

Desventajas de las Folksonomías

Los creadores de las etiquetas no son necesariamente expertos

- Etiquetas inexactas (p.e. por unitérminos en Del.icios.us)
- Etiquetas ambiguas
- Etiquetas imprecisas

} (debido a sinonimias y homonimias)

- Etiquetas subjetivas

Ventajas Folksonomías

- Simplicidad en la gestión y utilización
- Economía en la construcción por la cooperación
- Asignación de etiquetas flexible

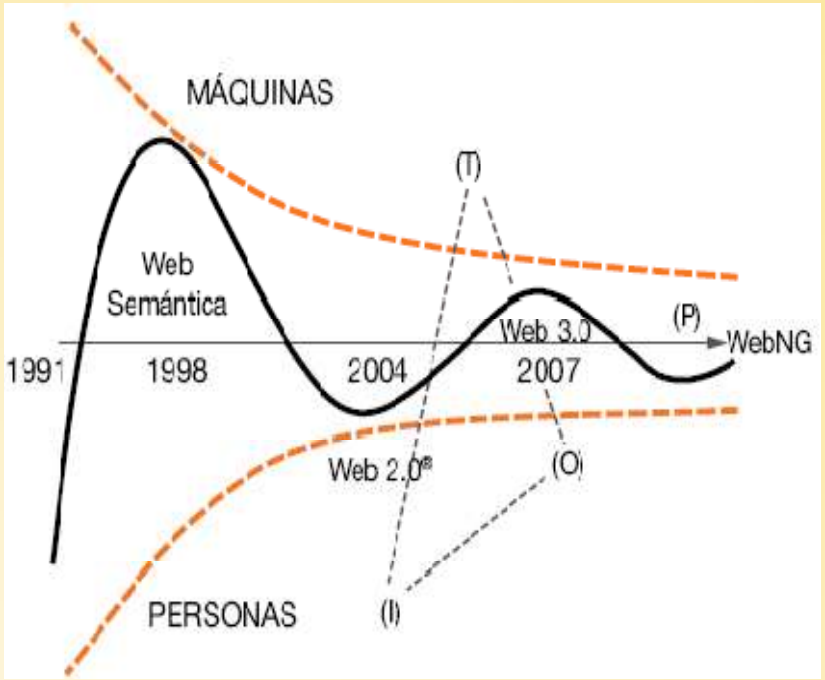
- Adecuación a las enormes dimensiones de la Web
- Ejecución de consultas especificas y adecuadas al vocabulario del usuario
- Capacidad de recuperación de recursos de la Web Invisible

- Con nuevos usos: estudios sociolingüísticos y recursos para crear ontologías

La adecuación de los sistemas de clasificación al Web: la Web Semántica

- Berners-Lee, 1999
- Propuesta con poca implantación
- Medio para que futuras aplicaciones puedan interpretar cualquier dato del Web->Interoperabilidad

- La Web Semántica (WS) propone:
  - Utilización de una sintaxis común. XML y la expresión del conocimiento en estructuras simples predefinidas (p.e. RDF)
  - Utilización de vocabularios de metadatos y ontologías (SKOS)
  - Referenciar los términos a recursos que expliquen su contenido mediante la URI
  - La WS propone la utilización por personas y máquinas (RSS)



[tomado de Fumero, 2007]

Comparativa W2.0 y Web Semántica

	Web 2.0	Web Semántica
Origen	Constatación de la evolución natural de la Web	Propuesta de Tim Berners para evolucionar la Web
Implantación	Muy alta	Escasa (Palacios et al, 2006)
Coordinación	No existe	Centralizada, sobre todo por el W3C
Foco	Personas	Aplicaciones informáticas
Creación	2003, 1ª conferencia 2004	1999 (Berners-Lee, 1999)
Expresión	Lenguaje libre, expresado mediante folksonomías, palabras clave denominadas etiquetas (tags), con problemas de sinonimia y polisemia	Lenguaje controlado, mediante lenguajes para expresión de ontologías, KOS y vocabularios de metadatos
Algunas características	•Descripción de los recursos para mejorar su distribución gratuita, se comparte conocimiento y desarrollos •Arquitectura de colaboración •Usabilidad alta •Recurso más útil cuanto más uso tenga	•Utilización de un lenguaje estandarizado con sintaxis uniforme y semántica no ambigua •Interoperabilidad: Intercambio de información entre cualquier repositorio •Usabilidad escasa

Web Social y Web Semántica

- Tratan diferentes dimensiones del Web
  - La Web Semántica, crea ontologías con semántica muy formalizada y consensuada dirigida a aplicaciones software.
  - La Web Social crea mecanismos locales de colaboración con gran usabilidad y dirigidos al usuario.
- Entran en confrontación por:
  - Las ontologías son poco legibles (cuello de botella por RDF y OWL) por personas y costosas de crear. Las folksonomías son difíciles de interpretar por aplicaciones (polisemia y ambigüedad) pero su creación tiene bajo coste y esfuerzo
  - No hay herramientas de la Web Semántica amigables para los usuarios. Los recursos de la Web Social no son amigables para las aplicaciones
  - Técnicas automáticas de creación de ontologías inmaduras
  - Duplicidades de vocabularios de metadatos (p.e. SKOS-Core, los PSI, Zthes y MADS)

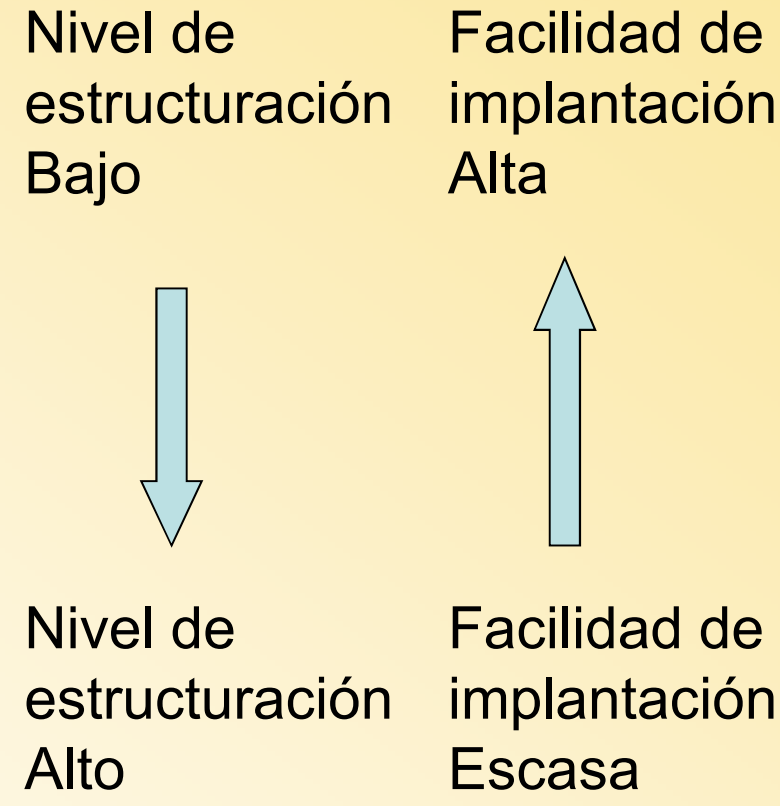
Lenguajes utilizados en RI documental

Palabras-clave
independientes (p.e. folksonomías)

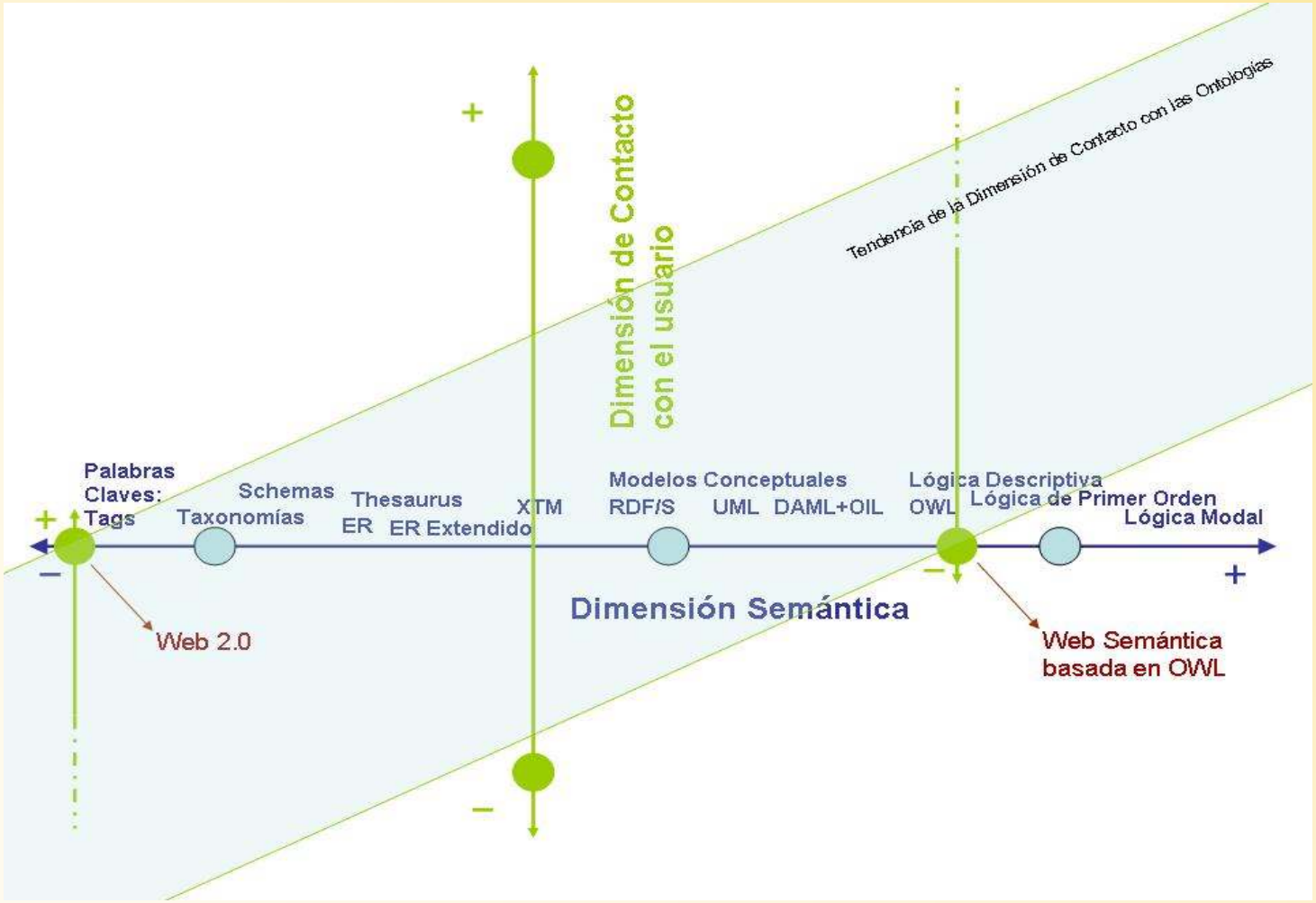
Listas de palabras (glosarios, listas de nombres, diccionarios)

Facetas, categorizaciones y clasificaciones

Grupos de relaciones
(tesauros, los *Topic maps*, y las Ontologías).

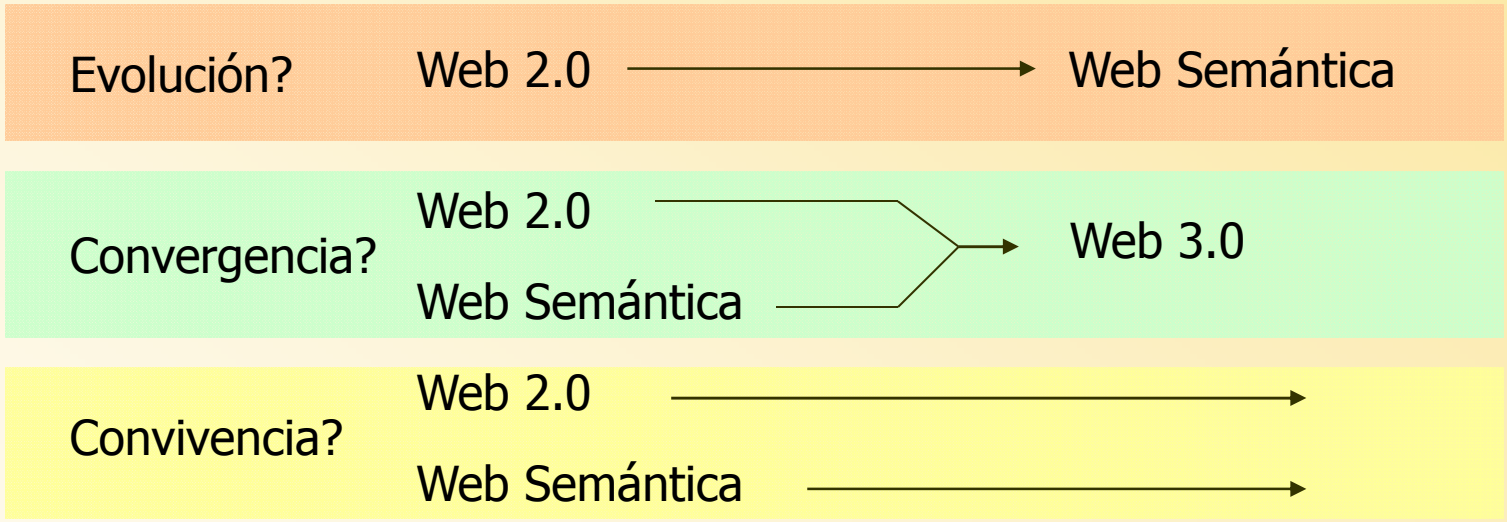


Dimensiones de la Web Social y la Web Semántica



Problemas

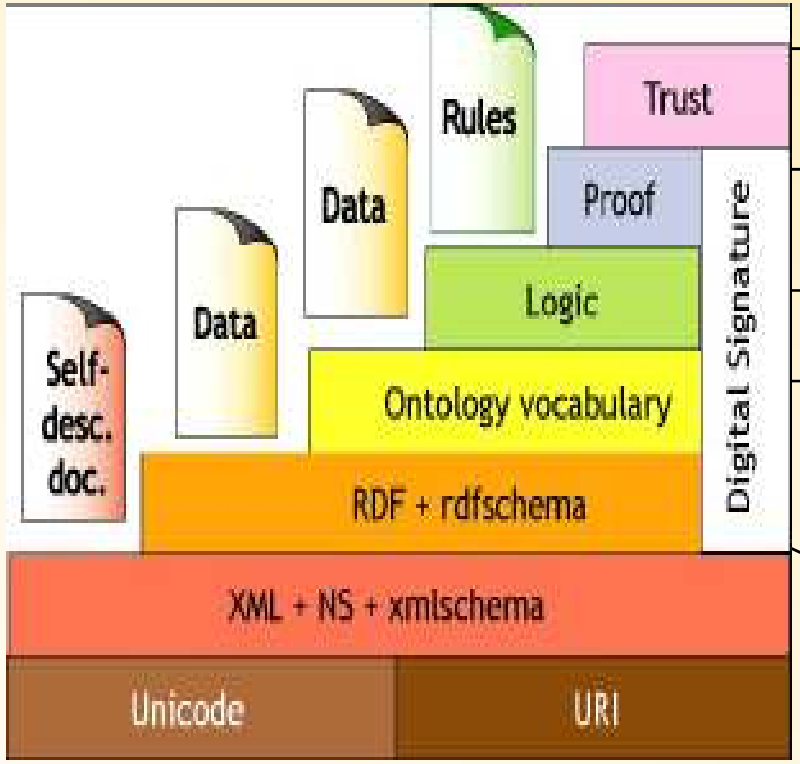
- Posibilidades de Evolución



La evolución no se hará de forma natural pues parten de diferentes dimensiones
La convivencia tiene sentido como mecanismos de bajo coste para representar recursos de la Web Invisible con escaso valor
La convergencia implica tomar medidas para que la evolución sea posible

Web Semántica

- Las capas de la Web Semántica (“layer cake”)



Recomendaciones sobre fiabilidad (agentes, agencias de certificación,…)

Procesos deductivos y validación de las aseveraciones

Axiomas y Reglas monotónicas salvo para dominios cerrados

Propiedades relaciones (transitiva, …), cardinalidad,.. Permite inferencia y reglas de consistencia (conversión entre vocabularios)

RDFS: vocabulario para definir clases, propiedades, subclases, dominio, rangos, comentarios y etiquetas
RDF: modelo de datos

Siempre hay compatibilidad con capas inferiores y entendimiento hacia las superiores (un documento RDF será entendido por aplicaciones XML, y aplicaciones para OWL podrán extraer la información de un documento RDF)

Ontologías

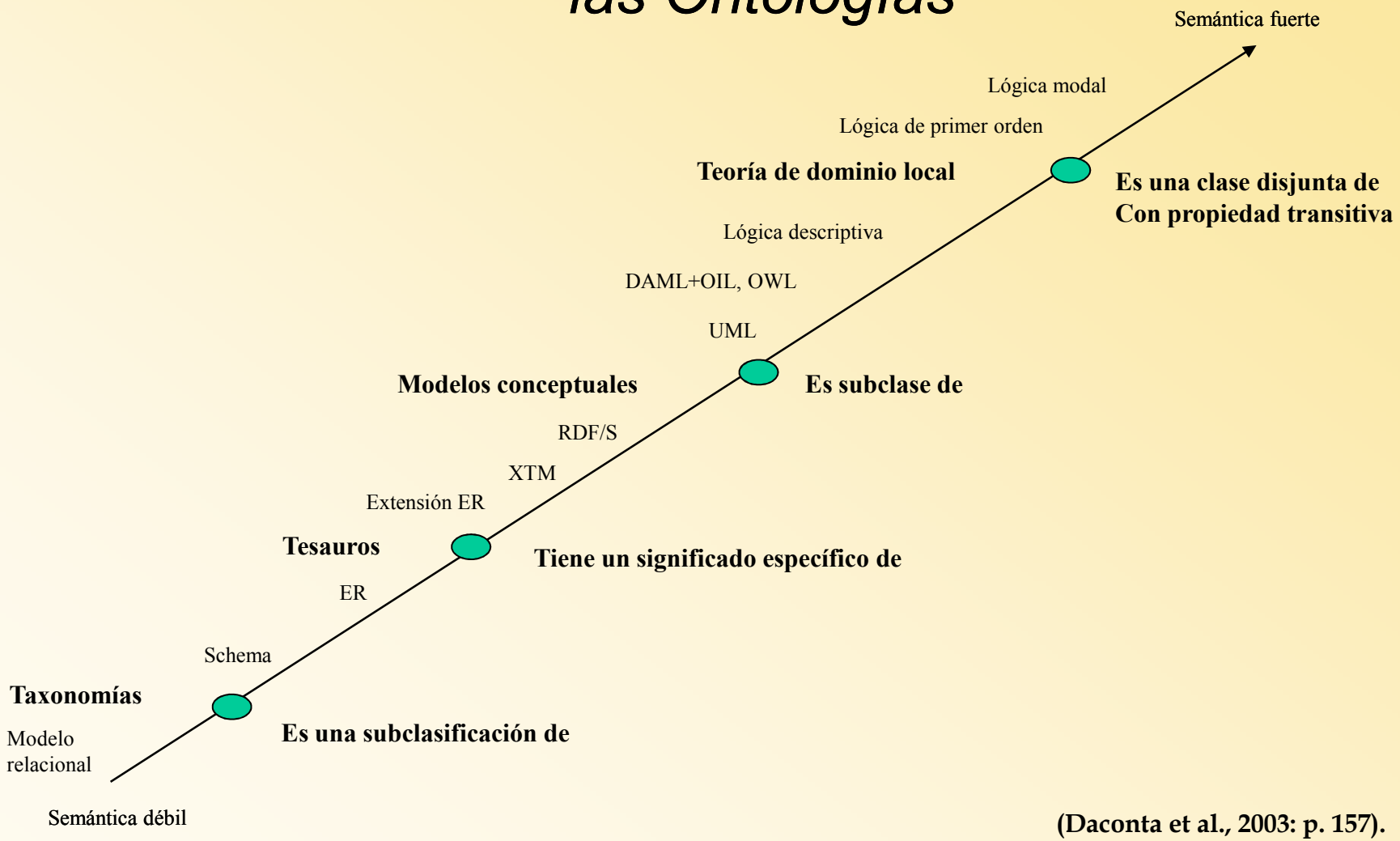
2. Sistemas de RI vs. RC:

Ontologías

- Neches (1991)
  - términos y relaciones de un dominio y reglas para combinar términos y relaciones para extender el vocabulario
- Gruber (1993)
  - *an ontology is an explicit specification of a conceptualization*
- Borst (1997)
  - *a formal specification a shared conceptualization*
- Studer (1998)
  - *Conceptualization*: modelo abstracto de un fenómeno de la realidad con sus conceptos relevantes
  - *Explicit*: los conceptos, sus tipos y *constraints* se definen explícitamente
  - *Formal*: legible por una máquina
  - *Shared*: con conocimiento consensuado (aceptado por la comunidad)

Lightweight vs *heavyweight*

2. Sistemas de RI vs. RC: *Espectro de las Ontologías*



(Daconta et al., 2003: p. 157).

3 Modelos de RC

Disciplina	Objetivos	Elementos máx.	Ejemplos de representación del conocimiento	Carga Semántica
Documentación	- Representar el conocimiento de un dominio	• *términos*	• listados de términos	- complejo
	- Clasificar objetos de información	• *conceptos*	• taxonomías	
	- Recuperar información	• *sinonimias*	• tesauros	
		• *taxonomías*	• Topic Maps	
		• *relaciones asociativas*		+ complejo

3 Modelos de RC

Disciplina	Objetivos	Elementos máx.	Ejemplos de representación del conocimiento	Carga Semántica
Ingeniería del Software	- Modelado de bases de datos	• *conceptos*	• modelos ER	- complejo
	- Modelado de aplicaciones	• *taxonomías*	• diagrama ER extendido	
	- Ayudas gráficas para comunicación entre clientes, analistas y desarrolladores	• *relaciones asociativas*	• diagramas de clases de UML	+ complejo
		• *funciones*		
		• *restricciones básicas*		

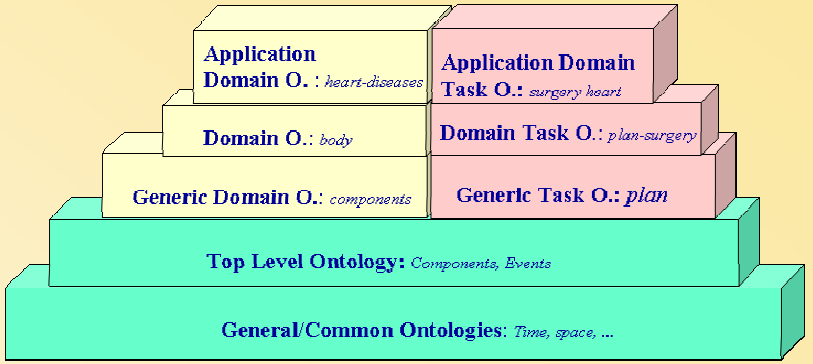
3 Modelos de RC

Disciplina	Objetivos	Elementos máx.	Ejemplos de representación del conocimiento	Carga Semántica
Inteligencia Artificial	- Imitar la mente humana - Almacenar conocimiento común - Con mecanismos para realizar inferencias e inferir nuevo conocimiento - Con mecanismos de aprendizaje - Con capacidad de operar en sistemas informáticos	• *conceptos* • *sinonimias* • *taxonomías* • *relaciones asociativas* • *funciones* • *restricciones formales* • *reglas de inferencia*	• Redes semánticas • OWL DL (Lóg. Descriptiva) • SCL (Lóg. Prm.Orden)	- complejo + complejo

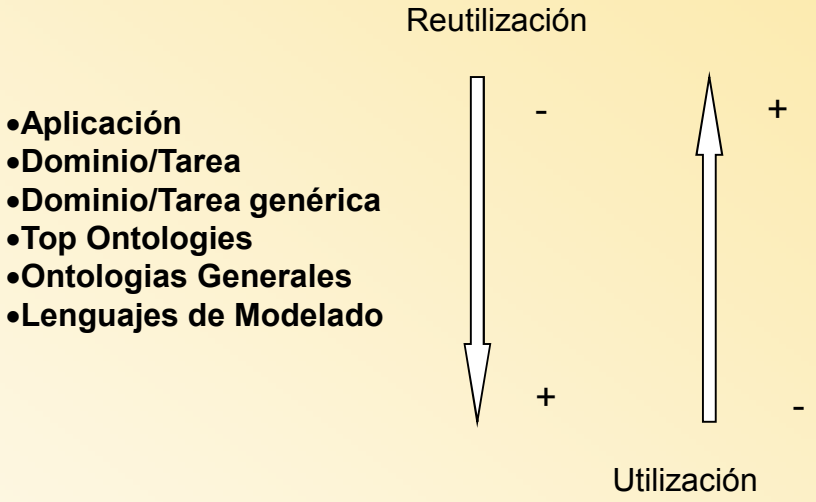
2.Sistemas de RC: *Ontologías*

Componentes de Ontologías

- **Clases** representan conceptos generales en muchos dominios de interés
- **Instancias** conceptos particulares es decir, elementos únicos en una ontología.
- **Relaciones** entre conceptos de un dominio.
- **Propiedades** (y los valores de las propiedades) de estos conceptos
- **Funciones** y desarrollo de los procesos. Es un tipo especial de relación en el uno de los elementos de la relación es el resultado de una fórmula.
- **Restricciones** y reglas o axiomas formales que sirven para modelar sentencias que son verdad



Tipos de Ontologías y su nivel de reusabilidad-usabilidad



2. Sistemas de RC: *ventajas y desventajas*

Ventajas de Ontologías en Web

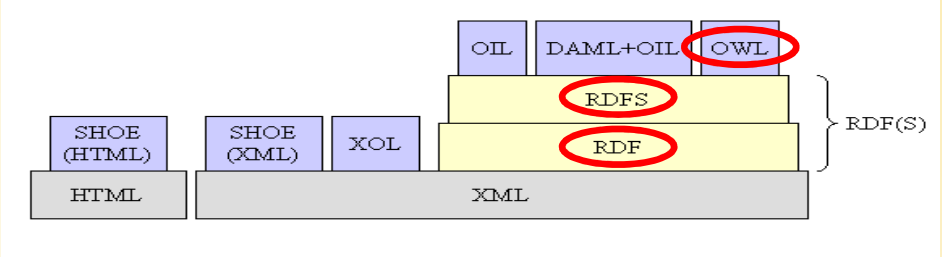
- Mejoran la reusabilidad y la interoperatividad
- Mejora de búsquedas
- Mejora de navegación y la arquitectura de los *sites*
- Pueden permitir inferencia
- Aportan reglas de coherencia y consistencia

Desventajas de Ontologías en Web

- Más útiles cuanto más complejas, pero:
  - Crece la dificultad en la creación
  - Problemas de visualización
  - Problemas para encontrar ontologías desde la misma perspectiva
  - El nivel de detalle y autoridad está en proporción inversa al tamaño del recurso (excepciones SNOMED)
- Todas las metodologías tienen dos grandes problemas:
  - Cuello de Botella de la adquisición de conocimiento
  - Dificultades en la validación por parte de ingenieros en el dominio, agravado con lenguajes como OWL.

Lenguajes para la Web Semántica

RDF/RDFS



RDF

Significa:	Resource Description Framework
Sirve para:	modelo de datos para expresar siempre los datos sobre cualquier cosa de la misma forma
Expresión:	RDF es independiente de XML, aunque suele expresarse con esta sintaxis
Validadores:	http://www.w3.org/RDF/Validator/

Problemas XML

```
<!ELEMENT Libros (Libro)+>
<!ELEMENT Libro (Titulo, Autor)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Autor (#PCDATA)>
<!ATTLIST Autor Nacionalidad
CDATA #IMPLIED>
<!ATTLIST Autor Nombre CDATA
#IMPLIED>
```

```
<!ELEMENT Articulos (Revista)+>
<!ELEMENT Revista (Articulo)+>
<!ELEMENT Articulo (Titular, AutorArticulo)>
<!ELEMENT Titular (#PCDATA)>
<!ELEMENT AutorArticulo (#PCDATA)>
<!ATTLIST Revista Nombre CDATA
#REQUIRED>
<!ATTLIST Articulo Idioma CDATA #IMPLIED>
<!ATTLIST AutorArticulo Titulo CDATA
#IMPLIED>
```

- ¿Puedo crear un listado único de autores de libros y revistas a partir de documentos XML basadas en estas DTDs?, ¿cómo?

-¿Cómo puedo saber que AutorArticulo y Autor son el mismo concepto?

-¿Cómo puedo saber que Titulo o Nombre en cada DTD son conceptos distintos?

-¿Existe algún recurso que me permita saber si existen DTDs normalizadas para describir libros y artículos?, y ¿cuál es el más utilizado entre los existentes?

-¿Existe algún recurso que me permita conocer elementos de descripción estandarizados similares a los descritos aquí?

Diferencia con XML

Un mismo documento se puede modelar de distintas formas con XML:

```
<Libro idTitulo="El_Quijote"
        xmlns="http://www.ejemplo.org/libro">
  <autor>Cervantes</autor>
</Libro>
```

```
<obra>
  <libro>El Quijote</libro>
  <autor>Cervantes</autor>
</obra>
```

```
<autor nombre="Cervantes">
  <Libro>El Quijote</Libro>
</autor>
```

Pero con RDF*…

```
<rdf:RDF rdf:ID="El_Quijote"
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:dc="http://purl.org/dc/elements/1.1/"
         xml:base="http://www.mi_ejemplo.org/libro#">
        <rdf:Description ID="quijote">
                <dc:title>El Quijote</dc:title>
                <dc:creator>Miguel de Cervantes</dc:creator>
        </rdf:Description>
</rdf:RDF>
```

Ventajas RDF sobre XML

- Muy utilizado y mayor capacidad de interoperar
  - Clases y propiedades son fácilmente identificadas
  - Vocabulario estandarizado (en cuanto a clases y en cuanto a propiedades) [problema: se deben conocer los namespaces para referenciarlos]
- RDF se escribe de forma normalizada a diferencia de XML [quita la libertad en la creación de documentos de XML]
- Da un formato legible en la Web Semántica
- Los editores XML y RDF pueden entender la codificación RDF (no al contrario)

Elementos

Recursos

cosas sobre las que queremos hablar
- URI

Propiedad

relaciona la cosa sobre la que queremos hablar con otra cosa o un valor
- URI

Valor

valor de la propiedad para el recurso. Puede ser a su vez un recurso
- URI
- Cadena de caracteres

Sentencia

Recurso	Propiedad	Valor
www.pepe.com	Autor	Pepe

Expresar

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:uc3m="http://www.uc3m.es/yo-ns#"
  xmlns:base="http://aqui.esta.laontologia/identificacion">
 <rdf: Description rdf:about="#300003333">
  <uc3m:nombre>        Pepe García      </uc3m:nombre>
  <uc3m:colectivo>     PAS              </uc3m:colectivo>
 </rdf:Description>
</rdf:RDF>
```

Recursos

Propiedades

Valores

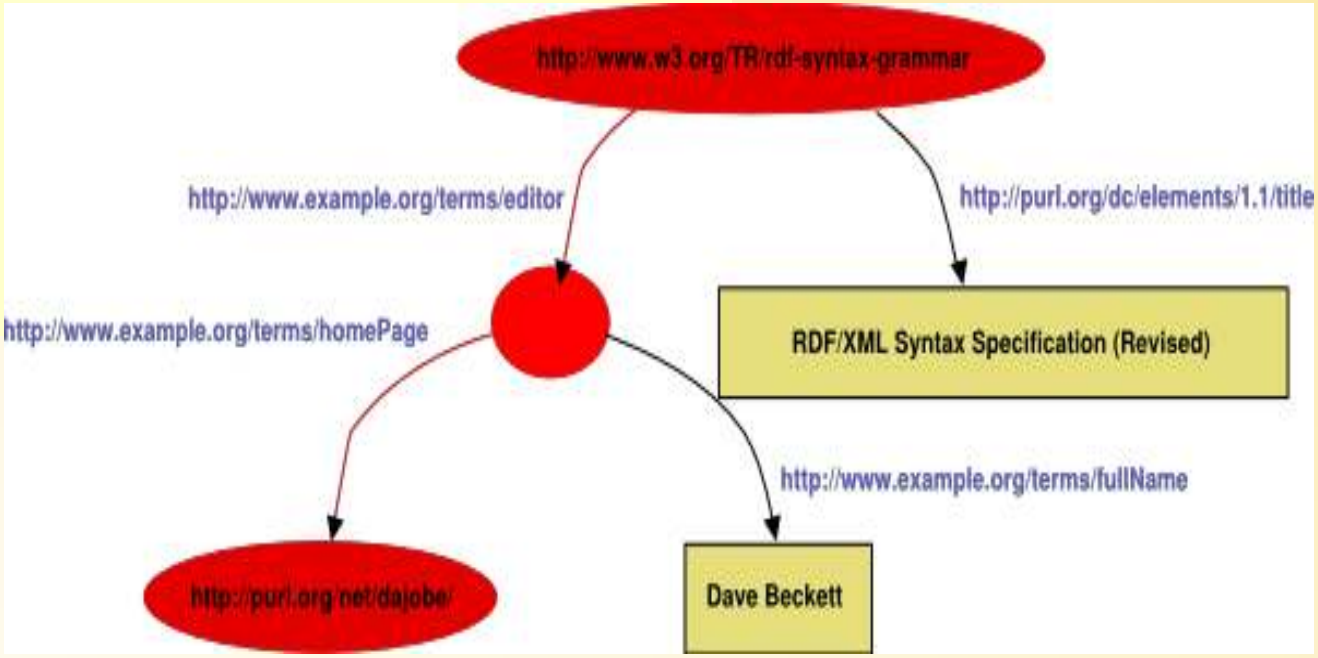
- Para referirnos a:
  - Una propiedad como nombre escribiremos http://www.uc3m.es/yo-ns#nombre
  - Un recurso respecto al lugar donde se encuentra el documento: http://aqui.esta.laontologia/identificacion#300003333

Grafos

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/...-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ex="http://example.org/stuff/1.0/">
<rdf:Description rdf:about=
    "http://www.w3.org/rdf-syntax-grammar"
  dc:title="RDF/XML Syntax Specification">
<ex:editor>
 <rdf:Description ex:fullName="Dave Beckett">
     <ex:homePage
     rdf:resource="http://purl.org/net/dajobe/" />
 </rdf:Description>
</ex:editor>
</rdf:Description>
</rdf:RDF>
```

Valor como cadena

Valor como URI



Forma abreviada y clases

Forma abreviada

Clase (con mayúsculas)

Propiedades
(en minúsculas)

```
<?xml version="1.0"?>
<Rio rdf:about="http://www.china.org/geography/rivers#Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

Forma extendida

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://www.china.org/geography/rivers#Yangtze"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.geodesy.org/river#">
    <rdf:type rdf:resource="http://www.geodesy.org/river#River"/>
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</rdf:Description>
```

Ejercicio:estructura RDF

```
<?xml version="1.0"?>
<Libro id="Quijote"
        xmlns="http://www.libreria.org/libro">
    <paginas>630</paginas>
    <idioma>castellano</idioma>
     <Pelicula id="DonQuijote" x
       "http://www.libreria.org/ci
       <titulo>
        Don Quijote de la Mancha
       </titulo>
       <año>1967</año>
      </Pelicula>
</Libro>
```

```
<?xml version="1.0"?>
 <rdf:RDF
 xmlns:rdf="…" xmlns=http://www.libreria.org/libro#">
 <rdf:Description
     rdf:about="http://www.libreria.org/libro#Quijote">
     <paginas>630</paginas>
     <idioma>castellano</idioma>
     <cinematografia rdf:resource=
     "http://www.libreria.org/cine#DonQuijote" />
</rdf:Description>
<rdf:Description rdf:about=
    "http://www.libreria.org/cine#DonQuijote">
    <titulo>Don Quijote de la Mancha</titulo>
    <año>1967</año>
</rdf:Description>
<rdf:RDF>
```

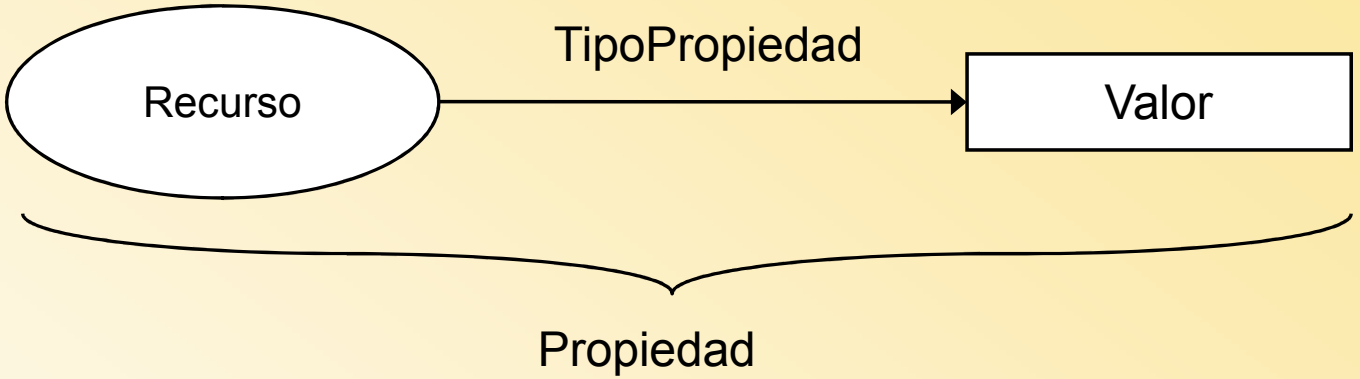
Notas

- En las URI **no** se ponen **fichero para identificar el recurso**. Si el recurso desaparece un agregador podría intentar encontrar URI de libros, poniendo un nombre de un fichero podríamos confundir al **agregador**
- RDF no admite poner atributos a propiedades

ID vs About:

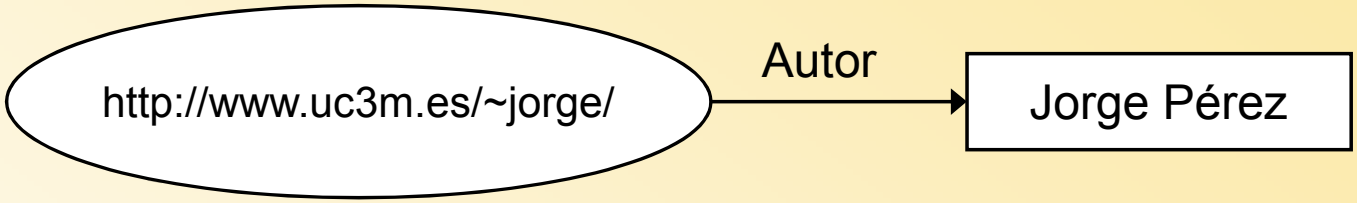
- **ID** se utiliza cuando queremos hablar por primera vez de un recurso y darle un nombre único <titulo año="1967">
- **About** se utiliza cuando queremos expandir la información sobre un recurso del que ya hemos hablado
- Si no referenciamos un recurso público (mediante id o about) la clase será **ambigua**.

RDF: modelo



- Basado en un modelo matemático=triple
- Recursos Web representados por nodos URI
- Los conjuntos de propiedades se conocen como “descripciones”

RDF: ejemplo básico



"Jorge Pérez es el autor del recurso identificado por http://www.uc3m.es/~jperez/"

RDF: Ejemplo

<?xml version="1.0" ?>

<RDF xmlns = **"http://w3.org/TR/1999/PR-rdf-syntax-19990105#"** xmlns:DC = **"http://purl.org/DC#"** >

<Description about = **"http://www.amazon.com"** >

<DC:Title> **Ontologia** </DC:Title>

<DC:Creator> **Jorge Pérez** </DC:Creator>

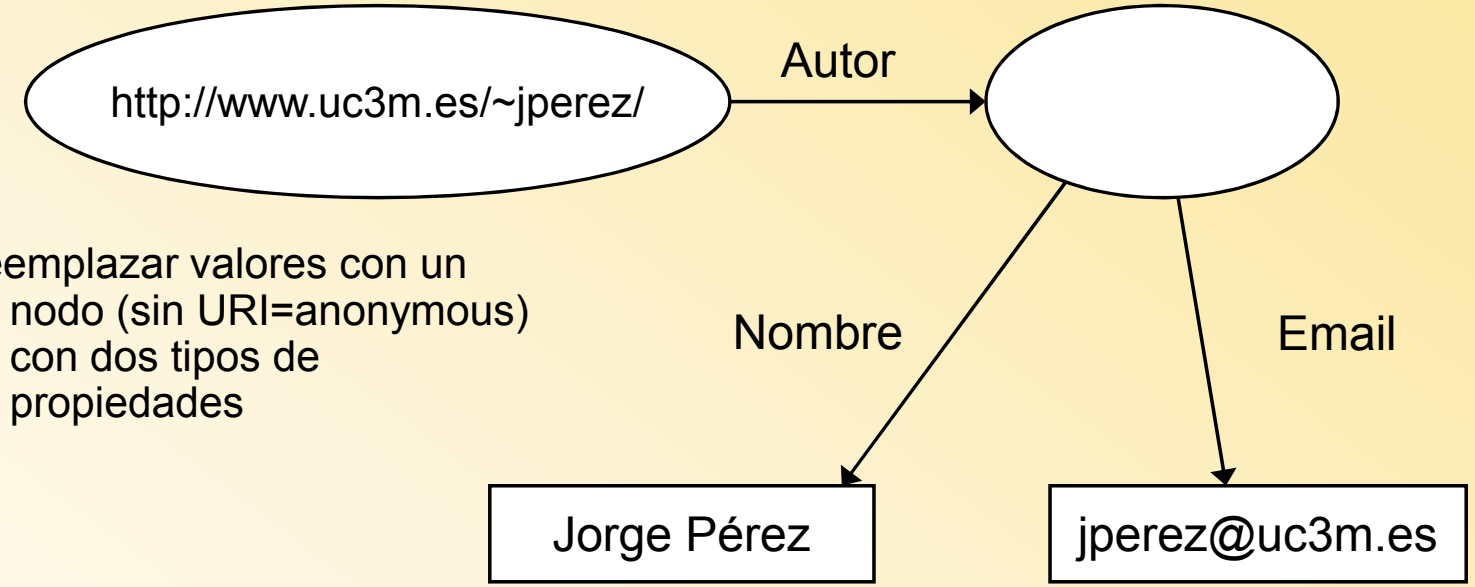
<DC:Date> **1999-12-31** </DC:Date>

<DC:Subject> **Metadata, RDF, Dublin Core** </DC:Subject>

</Description>

</RDF>

RDF: estructuración

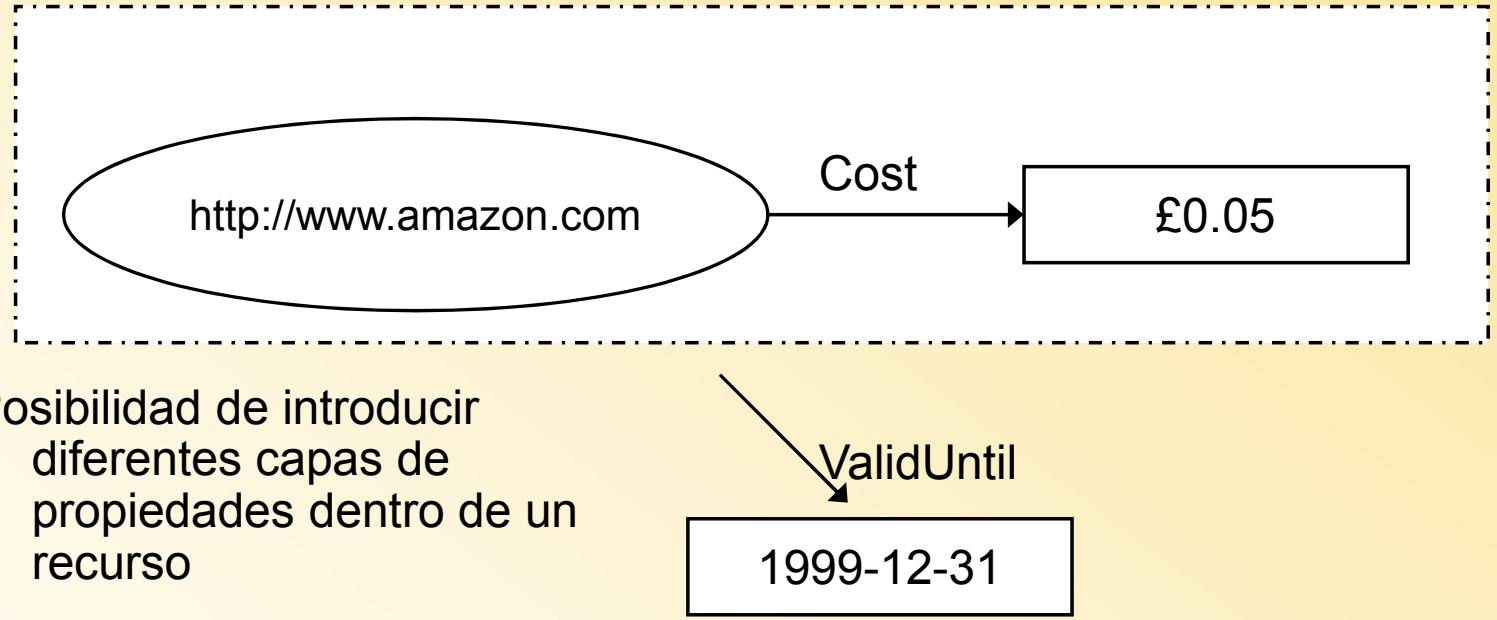


Reemplazar valores con un nodo (sin URI=anonymous) con dos tipos de propiedades

```
...
<DC:Creator parseType="Literal">
   <b>Jorge Pérez</b>
   <center>director</center>
   <i>jperez@uc3m.es</i>
   <b>Madrid</b>
</DC:Creator>
...
```

```
<DC:Creator parseType="Resource">
   <vCard:FN> Jorge Pérez </vCard:FN>
   <vCard:TITLE> Director</vCard:TITLE>
   <vCard:EMAIL> jperez@uc3m.es </vCard:EMAIL>
</DC:Creator>
...
```

RDF: reification



Posibilidad de introducir diferentes capas de propiedades dentro de un recurso

```
...
<Description about = http://www.amazon.com
                bagID = "ID001" >
  <DC:Title> Ontologias </DC:Title>
  <DC:Creator> Ruben Prieto Diaz</DC:Creator>
  <ECOMM:Price>£0.05</ECOMM:Price>
</Description>
```

```
<Description aboutEach = "#ID001" >
  <ADMIN:ValidFrom> 1998-01-01 </ADMIN:ValidFrom>
  <ADMIN:ValidTo> 1999-12-31 </ADMIN:ValidTo>
</Description>
...
```

RDF: mútiples propiedades

```
...
<DC:Creator>
  <Bag>
    <li> Maddie Azzurii </li>
    <li> Corky Brown </li>
    <li> Jacky Crystal </li>
  </Bag>
</DC:Creator>
...
```

```
...
<DC:Creator>
  <Seq>
    <li> Maddie Azzurii </li>
    <li> Corky Brown </li>
    <li> Jacky Crystal </li>
  </Seq>
</DC:Creator>
...
```

```
...
<SOFT:Location>
  <Alt>
    <li> ftp://soft-sales.com.us/abc.exe </li>
    <li> ftp://soft-sales.com.au/abc.exe </li>
    <li> ftp://soft-sales.com.de/abc.exe </li>
    <li> ftp://soft-sales.com.uk/abc.exe </li>
  </Alt>
</SOFT:Location>
...
```

RDF: ejemplo con SKOS

```
<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:skos="http://www.w3.org/2004/02/skos/core#">
    <skos:Concept rdf:about="http://www.ukat.org.uk/thesaurus/concept/1750">
        <skos:prefLabel>Economic cooperation</skos:prefLabel>
        <skos:altLabel>Economic co-operation</skos:altLabel>
        <skos:scopeNote>Includes cooperative measures in banking, trade, industry
    etc., between and among countries.</skos:scopeNote>
        <skos:inScheme rdf:resource="http://www.ukat.org.uk/thesaurus"/>
        <skos:broader rdf:resource="http://www.ukat.org.uk/thesaurus/concept/4382"/>
        <skos:narrower rdf:resource="http://www.ukat.org.uk/thesaurus/concept/2108"/>
        <skos:narrower rdf:resource="http://www.ukat.org.uk/thesaurus/concept/9505"/>
        <skos:narrower rdf:resource="http://www.ukat.org.uk/thesaurus/concept/15053"/>
        <skos:narrower rdf:resource="http://www.ukat.org.uk/thesaurus/concept/18987"/>
        <skos:related rdf:resource="http://www.ukat.org.uk/thesaurus/concept/3250"/>
    </skos:Concept> </rdf:RDF>
```

TM vs RDF

TOPIC MAPS

- TM optimizado para información y documentos
- TM tiene sentido fuera del Web
- Orientada a la fusión de mapas de conocimiento e índices
- Relaciones ternarias complicadas en RDF, no en TM
- La relación en RDF es direccional en TM no (OWL inverse_of)
- de primitivas de modelado
- TM tiene elementos para realizar vistas y niveles en la ontología
- Permite distinguir la naturaleza de los recursos con los que se asocia

RDF

- RDF/OWL optimizado para software
- Permite representaciones más flexibles que Topic Maps
- RDF pensado solo para el Web
- RDF/OWL tiene un mayor número de primitivas
- No incluye recursos no Web
- Dificultades en fusión de documentos
- RDF fusión con URIs (problema: varios conceptos pueden tener mismo URI)
- Todas las implementaciones para la Web Semántica se basan en RDF
- RDF no permite distinguir entre dos recursos asociados y entre una URI para ampliar información

RDF: **Ventajas y desventajas**

Desventaja

- Aunque RDF es el lenguaje más difundidos tienen deficiencias en:
  - Expresión en ternas, dos argumentos mediante una propiedad, implica problemas en ternarias y navegabilidad
  - Visualización no intuitiva

Ventajas

- Es el lenguaje con más futuro en la Web Semántica, y mediante OWL se han eliminado muchas deficiencias de partida mediante primitivas.

RDFS: RDF *Schema* (1/6)

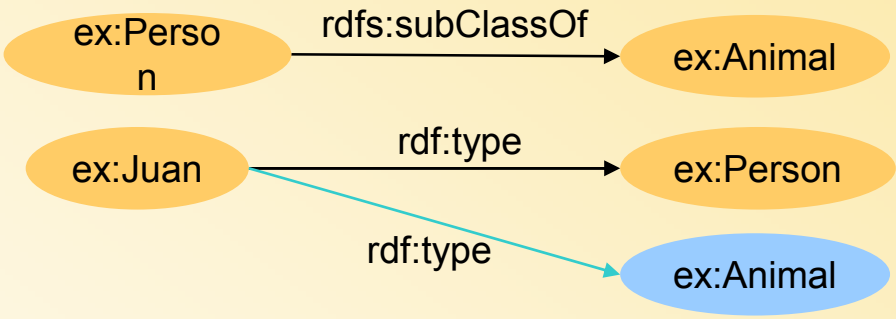
- RDF *Schema* extiende RDF con primitivas de *frames* para poder hablar de clases de recursos y propiedades asociadas a ellos.
  - Clases: rdfs:Class, rdfs:subClassOf
  - Propiedades: rdfs:subPropertyOf, rdfs:range, rdfs:domain
  - Otras primitivas: rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
- RDF *Schema* permite describir vocabularios RDF específicos de la aplicación.
- La relación entre instancias y clases se hace con rdf:type.

RDFS: RDF *Schema* (2/6)

- Clases principales:
  - rdfs:Resource
  - rdfs:Literal
  - rdfs:XMLiteral
  - rdfs:Class
  - rdfs:Datatype
  - rdf:Property
- Propiedades principales:
  - rdfs:subClassOf
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
  - rdfs:label
  - rdfs:comment
- Reificación:
  - rdf:Statement
  - rdf:predicate
  - rdf:subject
  - rdf:object
- Clases y propiedades contenedoras:
  - rdfs:Container
  - rdfs:ContainerMembershipProperty
  - rdf:Bag
  - rdf:Seq
  - rdf:Alt
  - rdfs:member
- Colecciones:
  - rdf:List
  - rdf:first
  - rdf:rest
  - rdf:nil (lista vacía)
- Propiedades de utilidad:
  - rdfs:seeAlso
  - rdfs:isDefinedBy
  - rdfs:value

RDFS: RDF *Schema* (3/6)

- RDFS añade así axiomas y restricciones en los modelos que antes no teníamos con RDF:
  - Herencia: $\forall x,y,z$ type(x,y) and subClassOf(y,z) $\rightarrow$ type(x,z)



  - Restricciones de rango: “Queremos que los cursos sean dados sólo por profesores”: restricción en los valores de la propiedad “es impartido por”.
  - Restricciones en el dominio: “Sólo los cursos pueden ser impartidos”: Impone una restricción en los objetos a los que se le puede aplicar esta propiedad.

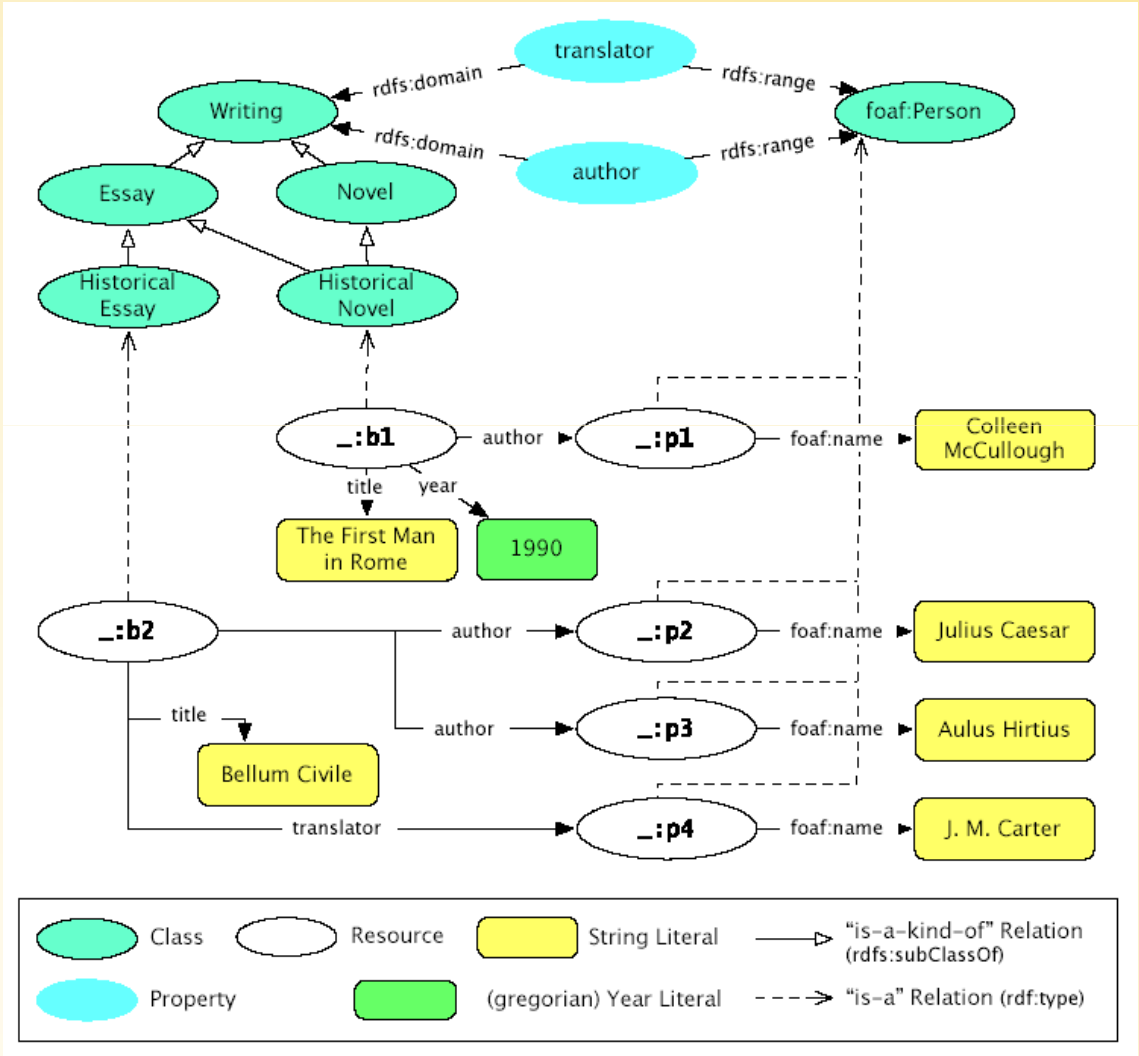
RDFS: RDF *Schema* (4/6)

- **Ejemplo: “Una universidad”:**

```
<rdfs:Class rdf:ID="course">
        <rdfs:comment>The class of courses</rdfs:comment>
</rdfs:Class>
<rdfs:Class rdf:about="#lecturer">
    <rdfs:comment>
        The class of lecturers. All lecturers are academic staff members.
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="#staffMember"/>
</rdfs:Class>
<rdf:Property rdf:ID="phone">
        <rdfs:domain rdf:resource="#staffMember"/>
        <rdfs:range rdf:resource="http://www.w3.org/
                    2000/01/rdf-schema#Literal"/>
</rdf:Property>
<rdf:Property rdf:ID="isTaughtBy">
    <rdfs:comment>
        Inherits its domain ("course") and range ("lecturer")
        from its superproperty "involves"
    </rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="#involves"/>
</rdf:Property>
```

RDFS: RDF *Schema* (5/6)

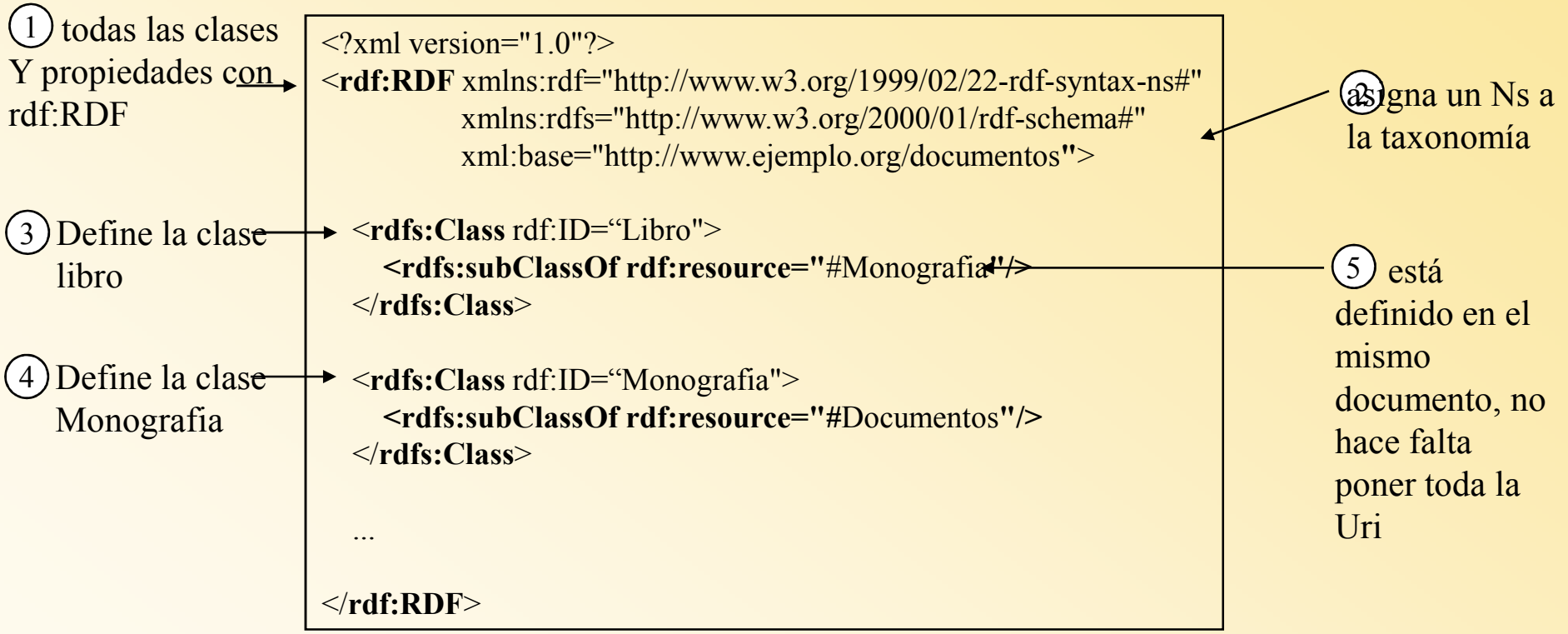
- Ejemplo:



RDFS: RDF *Schema* (6/6)

- ¿Problemas?
  - RDF *Schema* es un lenguaje de construcción de ontologías muy primitivo para modelar.
    - No hay restricciones en el rango o dominio: No puedo decir que el rango de hasChild es una “persona” si se aplica a personas y es un “elefante” si se aplica a elefantes.
    - No hay restricciones de existencia/cardinalidad: No puedo decir que “todas” las instancias de persona tienen una madre que es también en una persona, o que las personas tienen “exactamente 2 padres”.
  - Muchas primitivas que serían deseables no están:
    - No hay propiedades transitivas, inversas o simétricas: No puedo decir que “esParteDe” es una propiedad transitiva, que “tieneParte” es la inversa de “esParteDe” o que “esColindante” es simétrica.
    - No hay disjunción: Estaría bien decir que “macho y hembra” son disjuntos.
  - El razonamiento que podemos hacer pues es muy pobre
  - Necesitamos pues una capa más rica en semántica encima de RDF y RDF *Schema*.

RDFS definición de clases

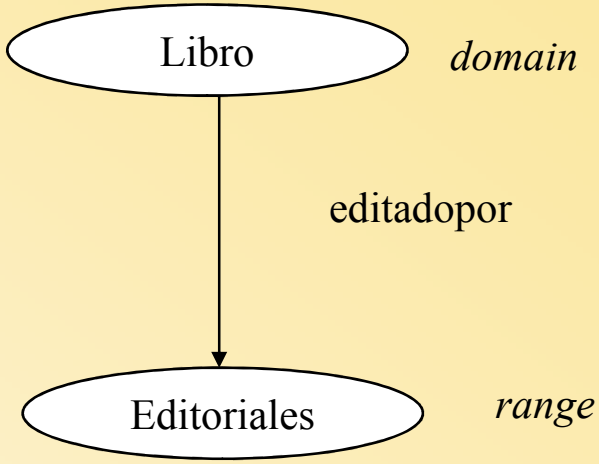


Definición de propiedades en RDFS

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

  <rdf:Property rdf:ID="editadopor">
    <rdfs:domain rdf:resource="#Libro"/>
    <rdfs:range rdf:resource="#editoriales"/>
  </rdf:Property>
  ...

</rdf:RDF>
```



Otra notación equivalente es:

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

Hay herencia de propiedades a las subclases!

Lenguajes para la Web Semántica

OWL

OWL

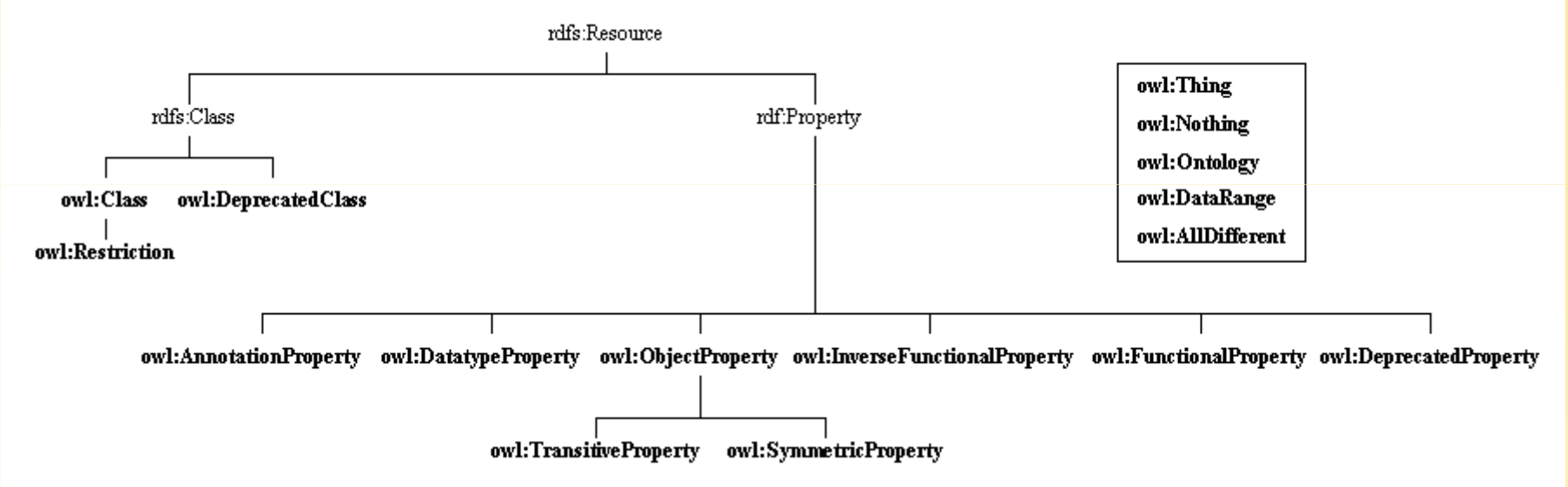
- Recordemos:
  - Los lenguajes de ontologías deben permitir escribir conceptualizaciones explícitas y formales.
  - Los principales requisitos son:
    - Una sintáxis bien definida.
    - Posibilidad de razonamiento eficiente.
    - Suficiente riqueza semántica.
  - Sin embargo, cuanto más rico es el lenguaje, más ineficiente es su razonamiento, llegando incluso hasta el punto de ser “incomputable”.
    - Necesitamos un compromiso entre ambas cosas.

OWL

- Web Ontology Language (2004): Construido encima de RDF(S).
- Tiene 3 capas:
  - OWL Lite: Pequeño subconjunto, basado en frames, pero con algo de razonamiento.
  - OWL DL: Subconjunto de la lógica de primer orden denominado *Description logics*. Su capacidad de inferencia es ya potente y decidible.
  - OWL Full: Extensión RDF, permitiendo metaclases. Es tan rico que su razonamiento puede ser no decidible.
- Varias sintáxis:
  - Abstract syntax (conceptualización): Corresponde a la común de DL, fácil de escribir y leer.
  - RDF/XML (implementación): Se puede escribir como un documento RDF.

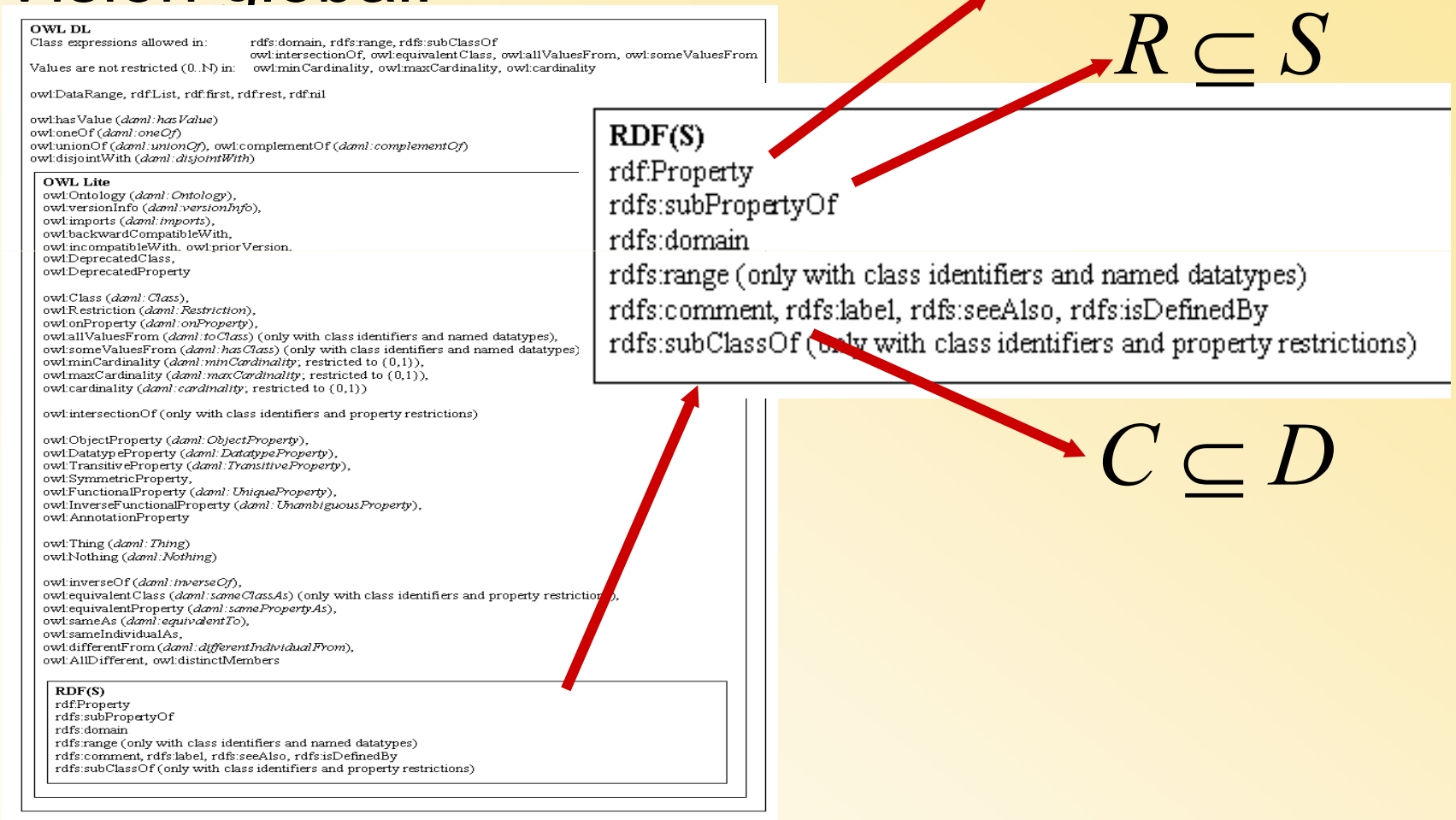
OWL

- Taxonomía: Compatible con RDFS:



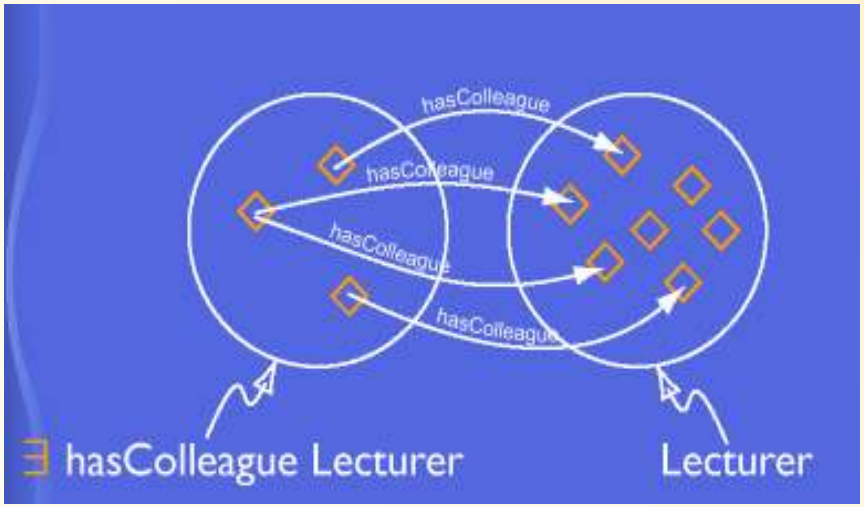
OWL

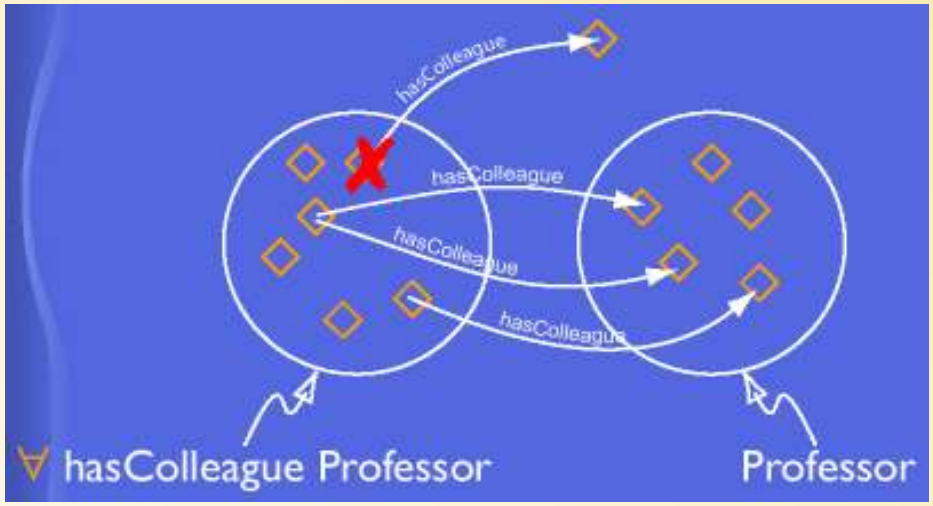
- Visión global:



OWL

- Visión global: Cuantificación existencial y universal:
  - someValuesFrom
  - allValuesFrom





OWL Lite Sinopsis

- RDF Schema Caract.
  - Class (Thing, Nothing)
  - rdfs:subClassOf
  - rdf:Property
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
  - Individual
- (In)igualdades:
  - equivalentClass
  - equivalentProperty
  - sameAs
  - differentFrom
  - AllDifferent
  - distinctMembers

OWL Lite Synopsis (Cont.)

- Propiedades:
  - ObjectProperty
  - DatatypeProperty
  - inverseOf
  - TransitiveProperty
  - SymmetricProperty
  - FunctionalProperty
  - InverseFunctionalProperty
- Restricciones:
  - Restriction
  - onProperty
  - allValuesFrom
  - someValuesFrom

- Cardinalidad
  - minCardinality (0 o 1)
  - maxCardinality (0 o 1)
  - cardinality (0 o 1)
- Cabeceras:
  - Ontology
  - imports
- Interseccion de clases:
  - intersectionOf

OWL Lite Sinopsis (Cont.)

- Versiones:
  - versionInfo
  - priorVersion
  - backwardCompatibleWith
  - incompatibleWith
  - DeprecatedClass
  - DeprecatedProperty

- Notas. Propiedades:
  - rdfs:label
  - rdfs:comment
  - rdfs:seeAlso
  - rdfs:isDefinedBy
  - AnnotationProperty
  - OntologyProperty
- Tipos de datos
  - xsd datatypes

OWL DL y Full

- Axiomas:
  - oneOf, dataRange
  - disjointWith
  - equivalentClass
  - rdfs:subClassOf
- Expresiones booleanas:
  - unionOf
  - complementOf
  - intersectionOf

- Cardinalidad:
  - minCardinality
  - maxCardinality
  - cardinality

Editores

Ontologías

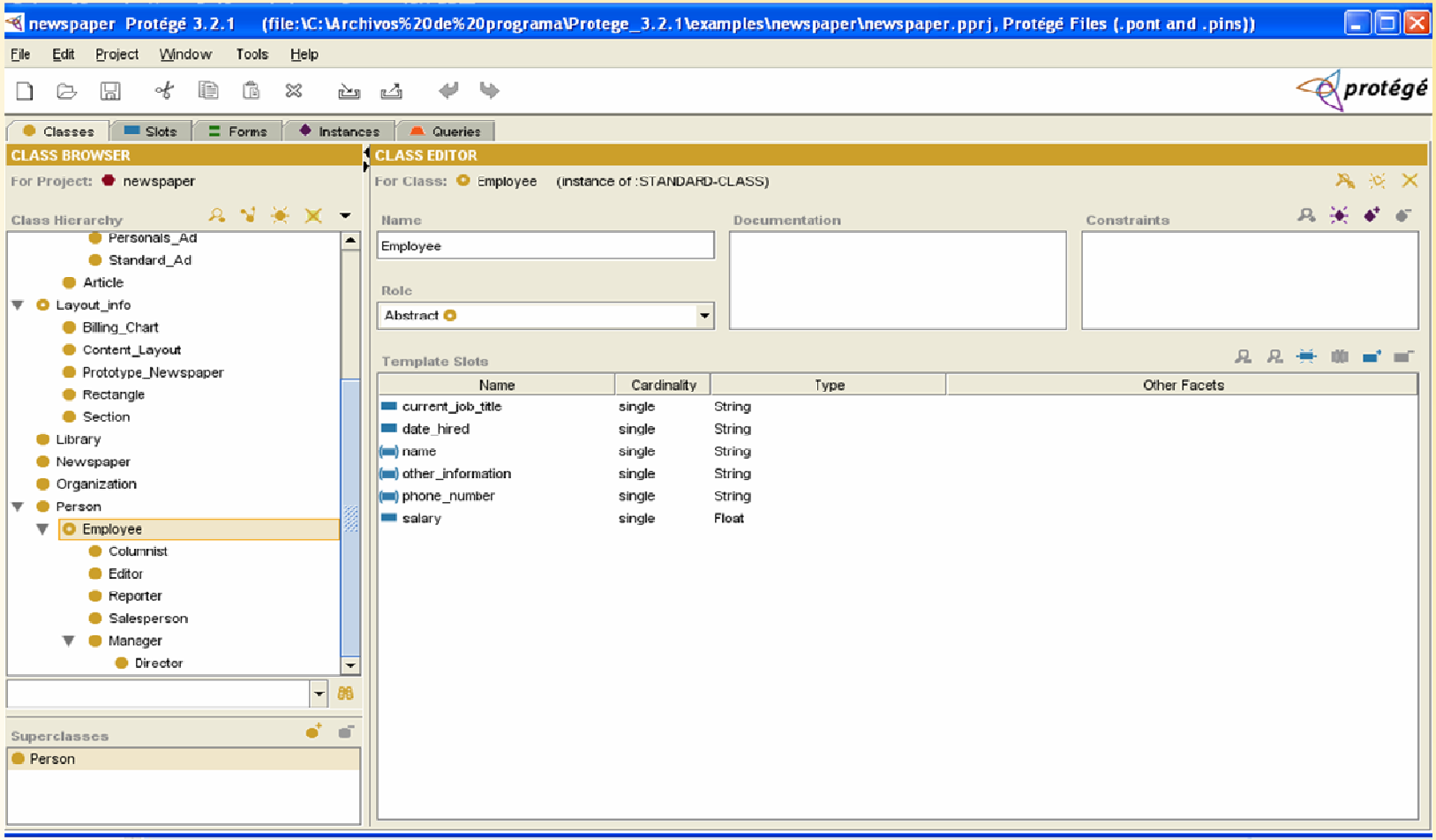
Herramientas de desarrollo

- Dependientes del lenguaje:
  - Ontolingua Server (1997):
    - Ontologías Ontolingua.
    - Web: Editor, browser, plugin Chimaera.
    - Ofrece edición colaborativa y repositorio de ontologías.
  - OntoSaurus (1997):
    - Ontologías LOOM.
    - Web: Editor, navegación.
  - WebOnto (1998):
    - Ontologías OCML.
    - Web: Editor (applets en Java). Edición colaborativa.
  - OilEd (2001):
    - Ontologías OIL y DAML+OIL.
    - Aplicación *standalone* realizada en Java.
    - Conocimientos sobre DL. Razonador FaCT.
    - Puede exportar a OWL.

Herramientas de desarrollo

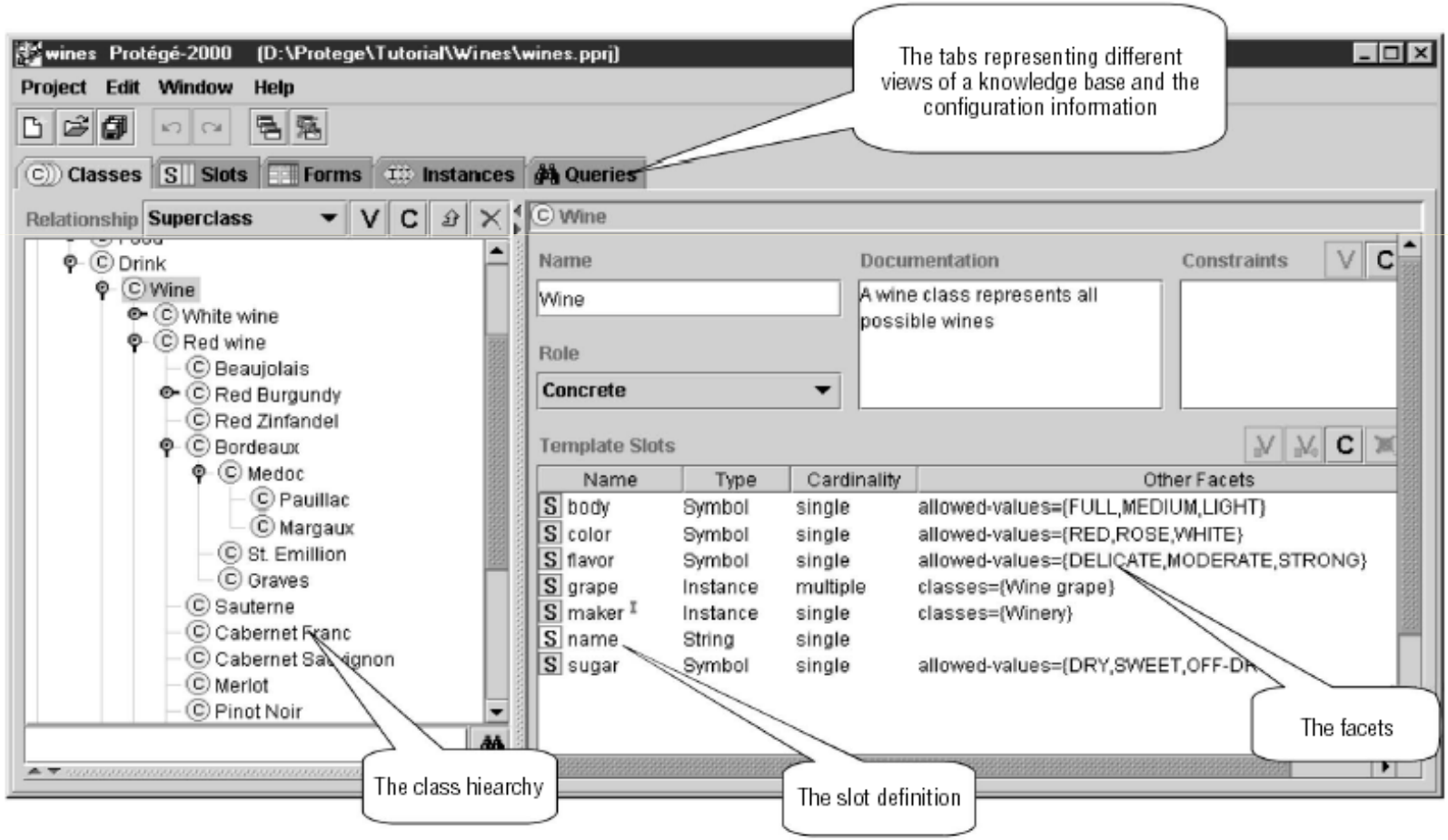
- Independientes del lenguaje:
  - Protege (2000):
    - Aplicación gratuita, arquitectura extensible.
    - Editor, navegación.
    - Plugins: Visualización gráfica, *merging*, razonamiento.
  - WebODE (2003):
    - Web: Edición, navegación (HTML y applets JAVA).
    - Creación de axiomas, documentación, evolución, aprendizaje, *merging*, razonamiento, evaluación (ODEClean).
  - OntoEdit (2002):
    - Aplicación: Editor, navegación.
    - Funciones: Edición colaborativa, inferencia.
    - Versión gratuita y comercial.
  - KAON (2003):
    - Aplicación en Java gratuita.
- Otros:
  - SemanticWorks
  - Swoop: Sólo visualización.

Herramientas de desarrollo

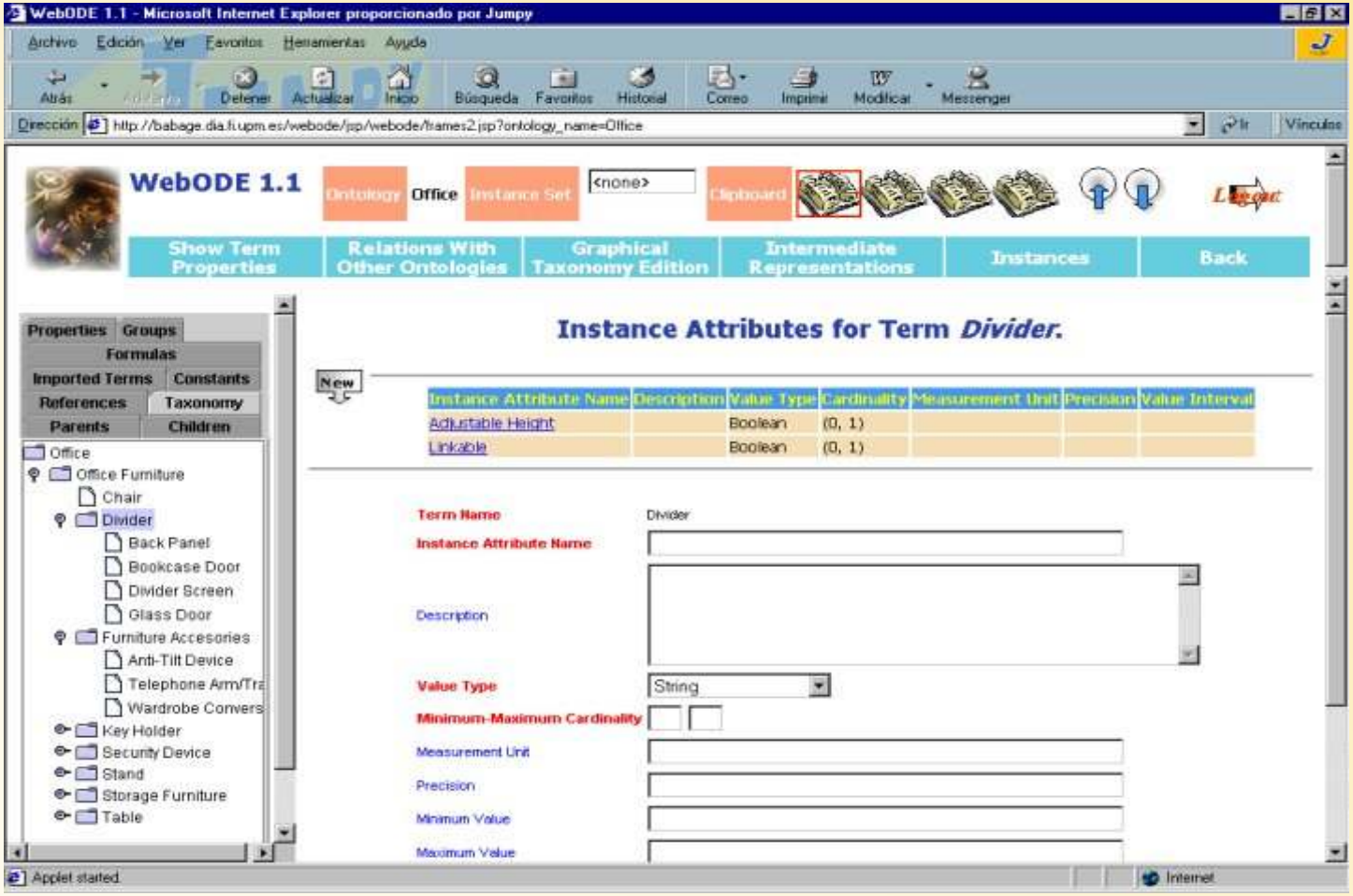


Editores de Ontologías

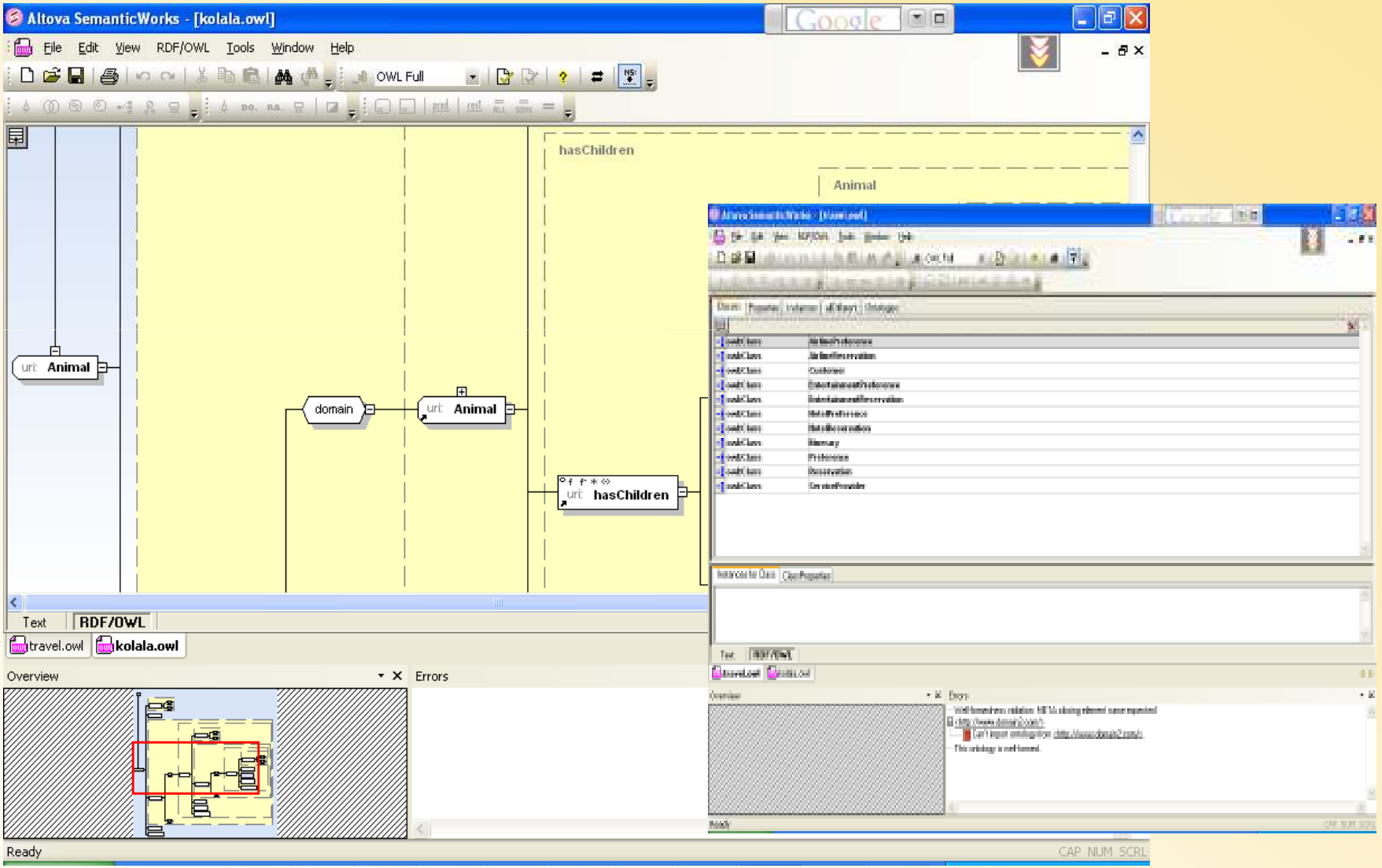
Verity, Ontolingua, WebOnto, WebODE, Ontosaurus, Chimaera, OilEd, Protégé, SWOOP, Sesame...



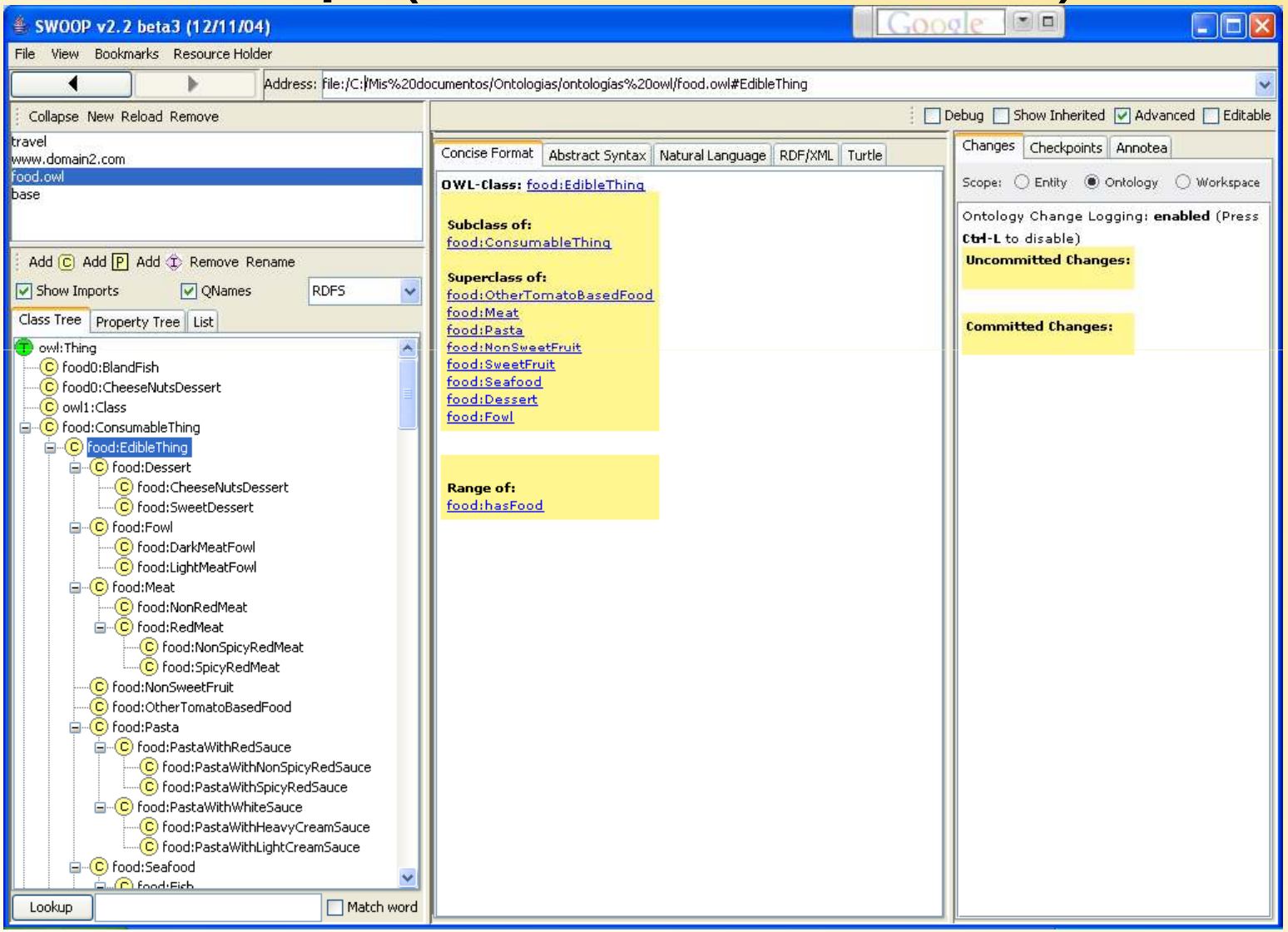
Herramientas de desarrollo



Editores de Ontologías
SemanticWorks de Altova

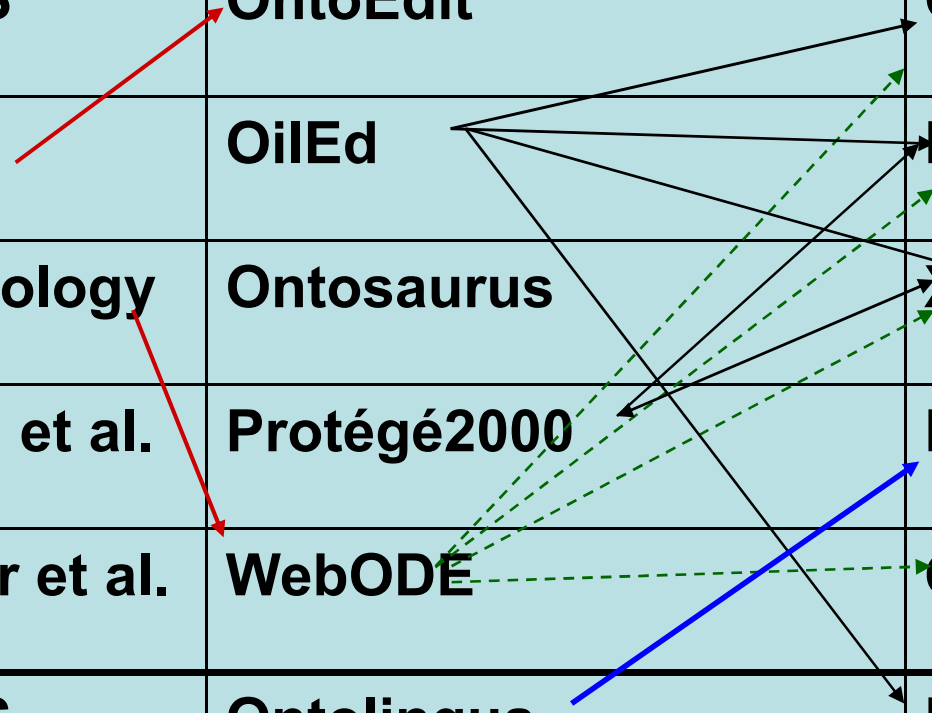


Visualización de Ontologías Swoop (basado en Jena)



Herramientas creación Ontologías

Método	Herramientas	Idioma
SENSUS	OntoEdit	OWL,DAML+OIL
OTK	OilEd	RDF(S)
Methontology	Ontosaurus	XML
Uschold et al.	Protégé2000	KIF
Gruniger et al.	WebODE	OCML
SENSUS	Ontolingua	FLogic



Herramientas de *merging* y alineamiento

- Principios para su elección:
  - ¿Qué componentes se pueden fundir o alinear?
  - ¿Las ontologías fuentes deben estar en el mismo lenguaje?
  - ¿Cuál es el grado de automatización?
  - ¿Están integradas en alguna herramienta de desarrollo anterior?

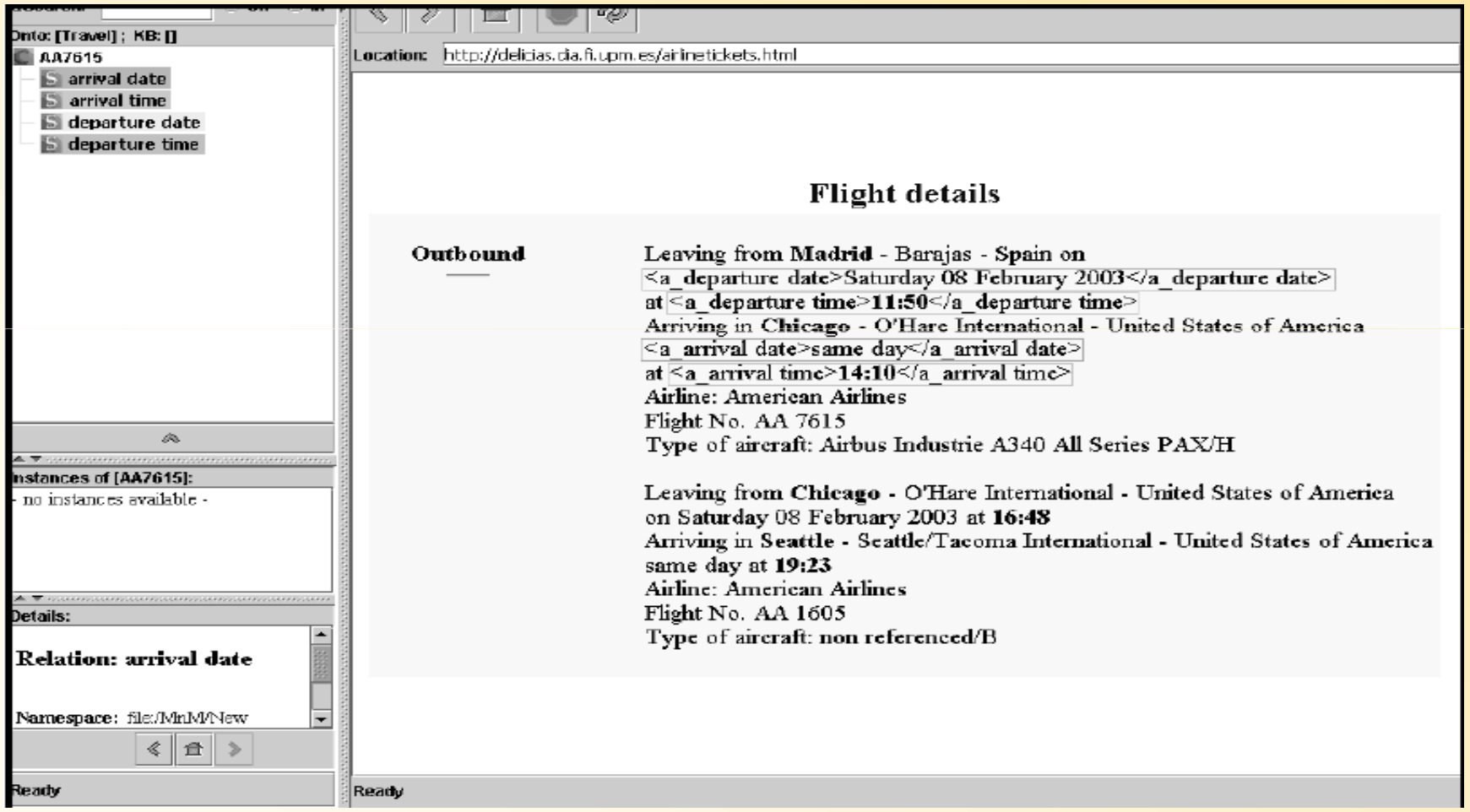
Herramientas de *merging* y alineamiento

- Herramientas:
  - Observer (1996):
    - *Merging* automático de ontologías del mismo dominio.
    - Proceso invisible al usuario.
  - Chimaera (2000):
    - Integrado en Ontolingua Server.
  - PROMPT Plug-in (2000):
    - Integrado en Protege.
    - Soporta el método PROMPT.
  - FCA-Merge toolset (2001):
    - Soporta el método FCA-Merge.
  - GLUE (2002):
    - Mapea conceptos automáticamente.

Herramientas de anotación

- Herramientas:
  - SHOE Knowledge Annotator (2001):
    - Anotación manual de documentos HTML.
    - Da paso a SMORE, que anota con RDF(S) y DAML+OIL
  - OntoMat-Annotizer (2001):
    - Aplicación Java con un visualizador de ontologías y una navegador Web.
    - Anotaciones manuales de DAML+OIL.
    - Da paso a OntoAnnotate.
  - MnM (2001):
    - Aplicación Java con un editor y un navegador.
    - Ontologías en RDF(S), DAML+OIL, OCML.
    - Añade soporte semiautomático.
  - COHSE (2002):
    - Se añade en la barra lateral del Mozilla.
    - Se basa en servidores (no es una aplicación *standalone*).
    - Anotaciones basadas y no basadas en ontologías.
    - Permite instanciar conceptos, pero no sus atributos.
  - UBOT AeroDAML (2001):
    - Aplicación Web.
    - El usuario manda la página, y se la devuelve con anotaciones basadas en ontologías generales y de alto nivel.

Herramientas de anotación



Algunos Razonadores

- CEL es un razonador basado en LISP, gratuito para uso no comercial.
- Cerebra Engine es un razonador comercial basado en C++.
- FaCT++ es un razonador basado en C++, gratuito open-source.
- KAON2 es un razonador basado en Java, gratuito para uso no comercial.
- MSPASS es un razonador basado en C, gratuito y open-source.
- Pellet es un razonador basado en Java, gratuito open-source.
- RacerPro es un razonador basado en LISP comercial, pero con trials gratuitos y licencias de investigación.

Metodologías para crear ontologías

Principios de Diseño de Ontologias (Gruber 93)

1. Claridad: definiciones objetivas, claras, completas con condiciones necesarias y suficientes. Con documentación en lenguaje natural
2. Coherencia: p.e. las inferencias a través de axiomas no deben contradecir las definiciones de los conceptos
3. Extensibilidad
4. Especificación es independiente de codificación por símbolos. P.e. una moneda no se definiria como tipo número sino como moneda
5. Mínimo compromiso ontológico: cuantos menos compromisos mejor para llegar a un acuerdo. se debe procurar el uso consistente del vocabulario esencial para comunicar el conocimiento.

Además

- Conceptos similares con definiciones similares
- Estandarizar nombres

Principios de diseño (1/6)

- Claridad: Comunicar el significado de los términos.

```
(define-class Travel (?travel)
   "A journey from place to place"
:axiom-def
 (and (Superclass-Of Travel Flight)
      (Subclass-Of Travel Thing)
      (Template-Facet-Value Cardinality
           arrivalDate Travel 1)
      (Template-Facet-Value Cardinality
           departureDate Travel 1)
      (Template-Facet-Value Maximum-Cardinality
           singleFare Travel 1))
:def
 (and (arrivalDate ?travel Date)
      (departureDate ?travel Date)
      (singleFare ?travel Number)
      (companyName ?travel String)))
```

```
(define-class Travel (?travel)
   "A journey from place to place"
:axiom-def
 (and (Superclass-Of Travel Flight)
      (Subclass-Of Travel Thing)
      (Template-Facet-Value Cardinality
           arrivalDate Travel 1)
      (Template-Facet-Value Cardinality
           departureDate Travel 1)
      (Template-Facet-Value Maximum-Cardinality
           singleFare Travel 1))
:ff-def
 (and (arrivalDate ?travel Date)
      (departureDate ?travel Date))
:def
 (and (singleFare ?travel Number)
      (companyName ?travel String)))
```

Principios de diseño (2/6)

- Mínima posición con la codificación: Ser independiente del lenguaje.

```
(define-class Travel (?travel)
   "A journey from place to place"
:axiom-def
 (and (Superclass-Of Travel Flight)
      (Subclass-Of Travel Thing)
      (Template-Facet-Value Cardinality
           arrivalDate Travel 1)
      (Template-Facet-Value Cardinality
           departureDate Travel 1)
      (Template-Facet-Value Maximum-Cardinality
           singleFare Travel 1))
:iff-def
  (and (arrivalDate ?travel Date)
       (departureDate ?travel Date))
:def
  (and (singleFare ?travel Number)
       (companyName ?travel String)))
```

No minimal Encoding Bias

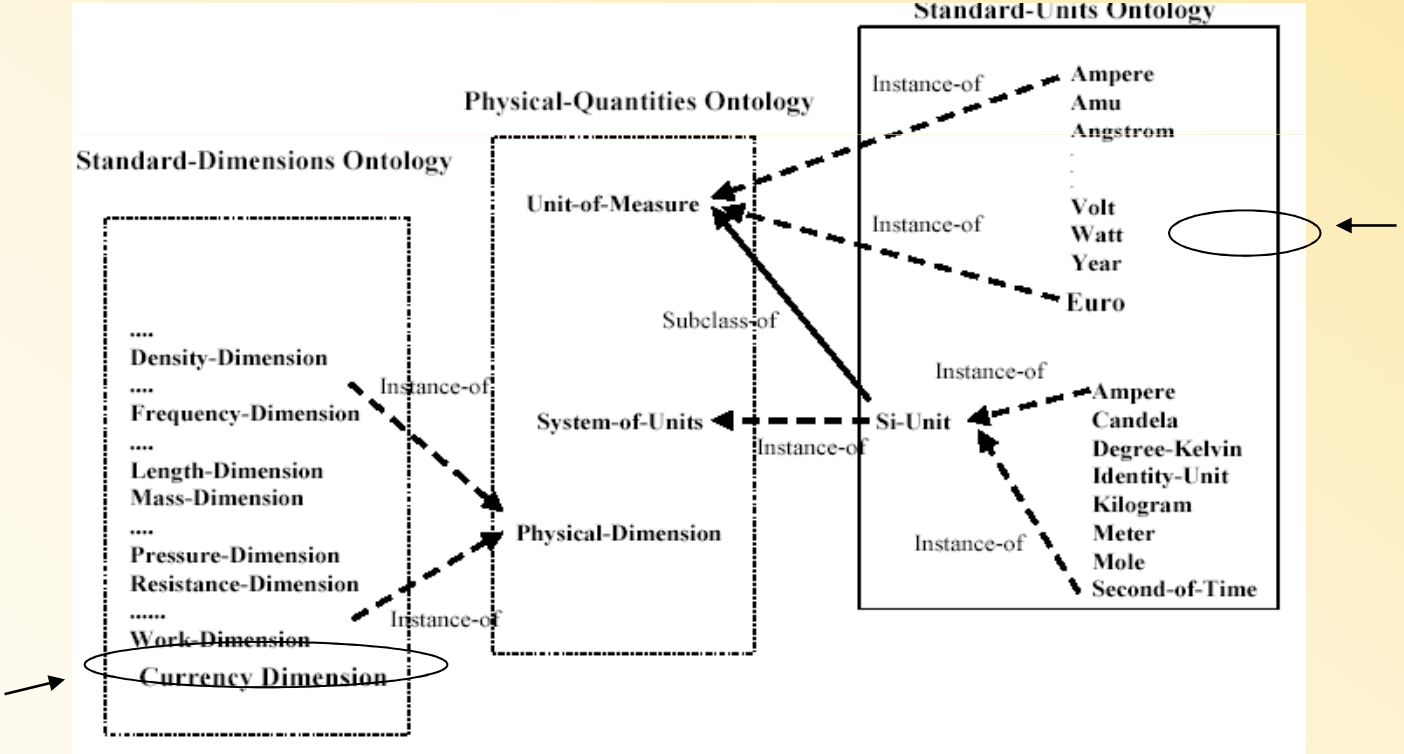
```
(singleFare ?travel Number)
```

should be substituted by:

```
(singleFare ?travel CurrencyQuantity)
```

Principios de diseño (3/6)

- Extensibilidad: Anticipar el uso compartido del vocabulario.



Principios de diseño (4/6)

- Coherencia: Que las inferencias que se realicen sean consistentes con las definiciones.

```
(define-axiom No-Train-between-USA-and-Europe
   "It is not possible to travel by train between the USA and Europe"
:= (forall (?travel)
     (forall (?city1)
      (forall (?city2)
       (=> (and (Travel ?travel)
                (arrivalPlace ?travel ?city1)
                (departurePlace ?travel ?city2)
                (or (and (EuropeanLocation ?city1)
                         (USALocation ?city2))
                    (and (EuropeanLocation ?city2)
                         (USALocation ?city1) )))
           (not (TrainTravel ?travel)))))))
(define-instance Madrid (EuropeanLocation))
(define-instance NewYork (USALocation))
```

Principios de diseño (5/6)

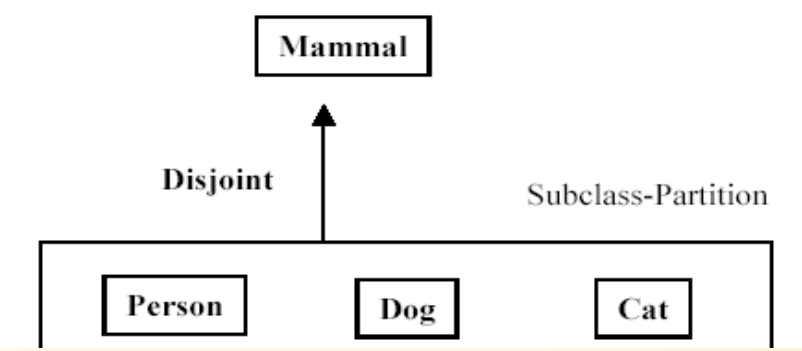
- Mínimos compromisos ontológicos: Los compromisos deben ser los mínimos, pero que aseguren lo esencial.

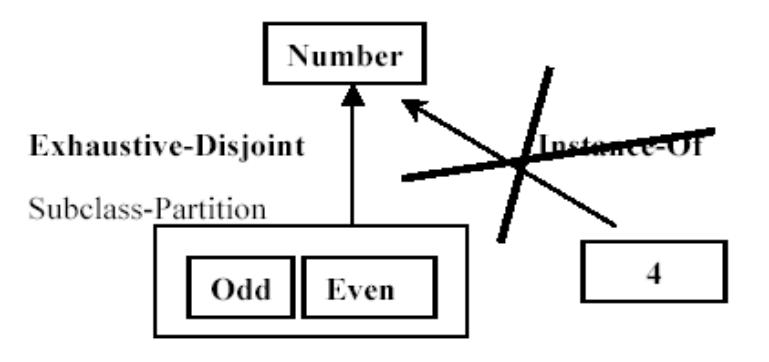
```
(define-class Travel (?travel)
   "A journey from place to place"
:axiom-def
 ( .... )
:iff-def
  (and (arrivalDate ?travel Date)
       (departureDate ?travel Date))
:def
  (and (singleFare ?travel Number)
       (companyName ?travel String)))
```

- What is a date?
- Absolute/relative date?
- could be an interval?
- date= month + year
- date= day + month +year
- date = month +day +year

Principios de diseño (6/6)

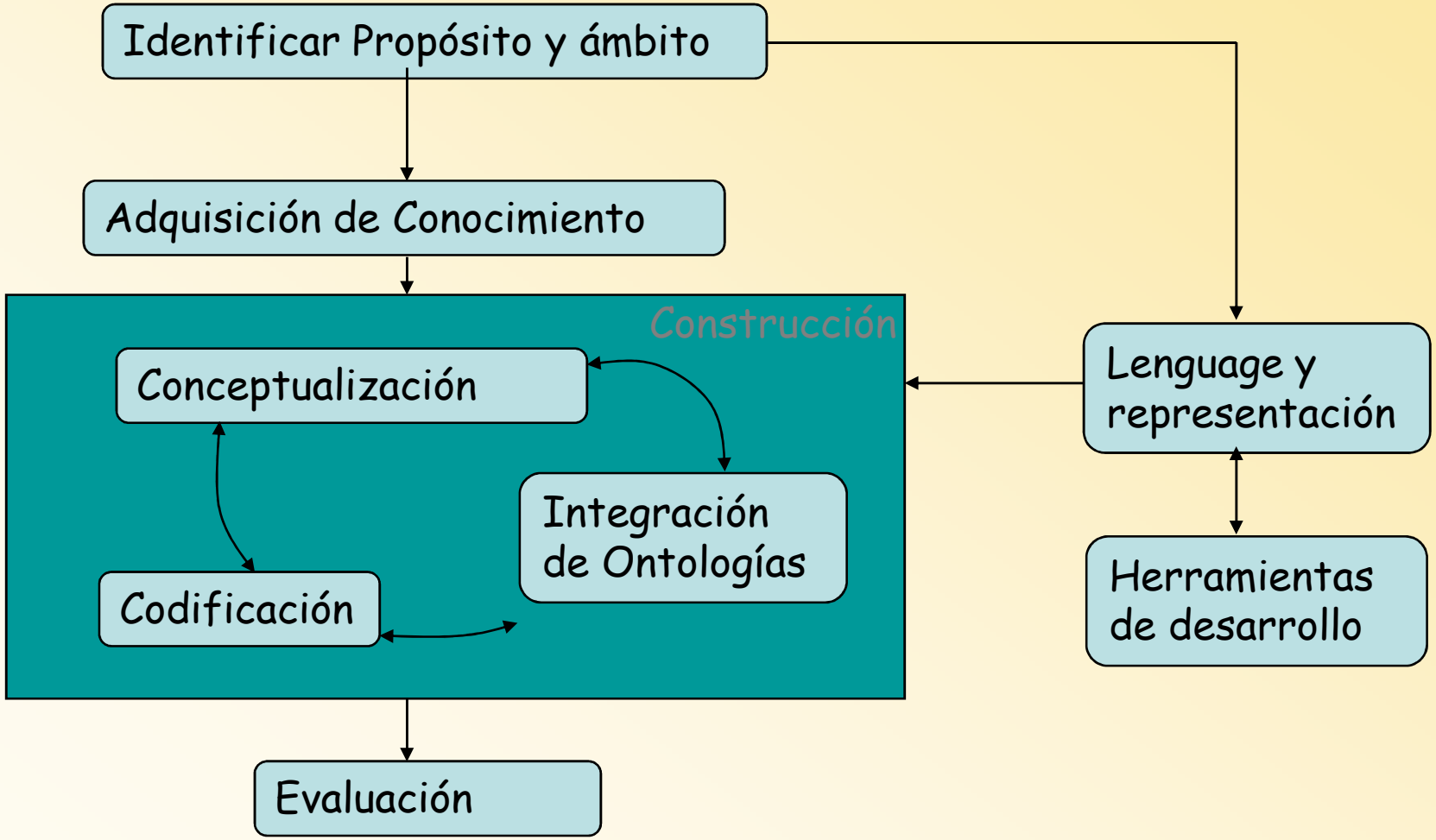
- Otros principios:
  - Ser completos a la hora de definir clases y sus subclases, con conocimiento disjunto y exhaustivo.





  - Estandarización de nombres

Método general (ciclo de vida)



Metodologías

- **Methontology**: actividades para realizar una metodología. Complicada, pero próxima al mundo de Ingeniería del Software. Útil en dominios dinámicos y complejos
- **Uschold's Methodology**
- **101** se popularizó gracias a Protegé pues las versiones iniciales traían un tutorial con esta metología
- **OTK Methodology**
- ***Toronto Virtual Enterprise*** (TOVE): Tiene aspectos de mantenimiento, se utiliza cuando el propósito esta claro
- ***Descriptive Ontology for Linguistic and Cognitive Engineering*** (DOLCE)

Ejemplo: http://klt.dif.um.es/ontologies/jobrecruitment.htm

Estrategias

- **Botton Up**: de lo específico a lo general, muy detallada, pero es más trabajo y se pueden tener inconsistencias por términos generales poco útiles
- **Top_down**: control del nivel de detalle, se pueden tener conceptos raíz inútiles
- **Middle_out**: muy balanceada

Método Uschold

1. Identificar propósito y ámbito
2. Codificarlo en clases y relaciones
3. Integrarlo en un modelo evaluándolo y documentándolo

Notad que no existe etapa de conceptualización (es decir sin ponerlo en un metalenguaje “cuello de botella” de adquisición de información)

Creación de Ontologías: la 101

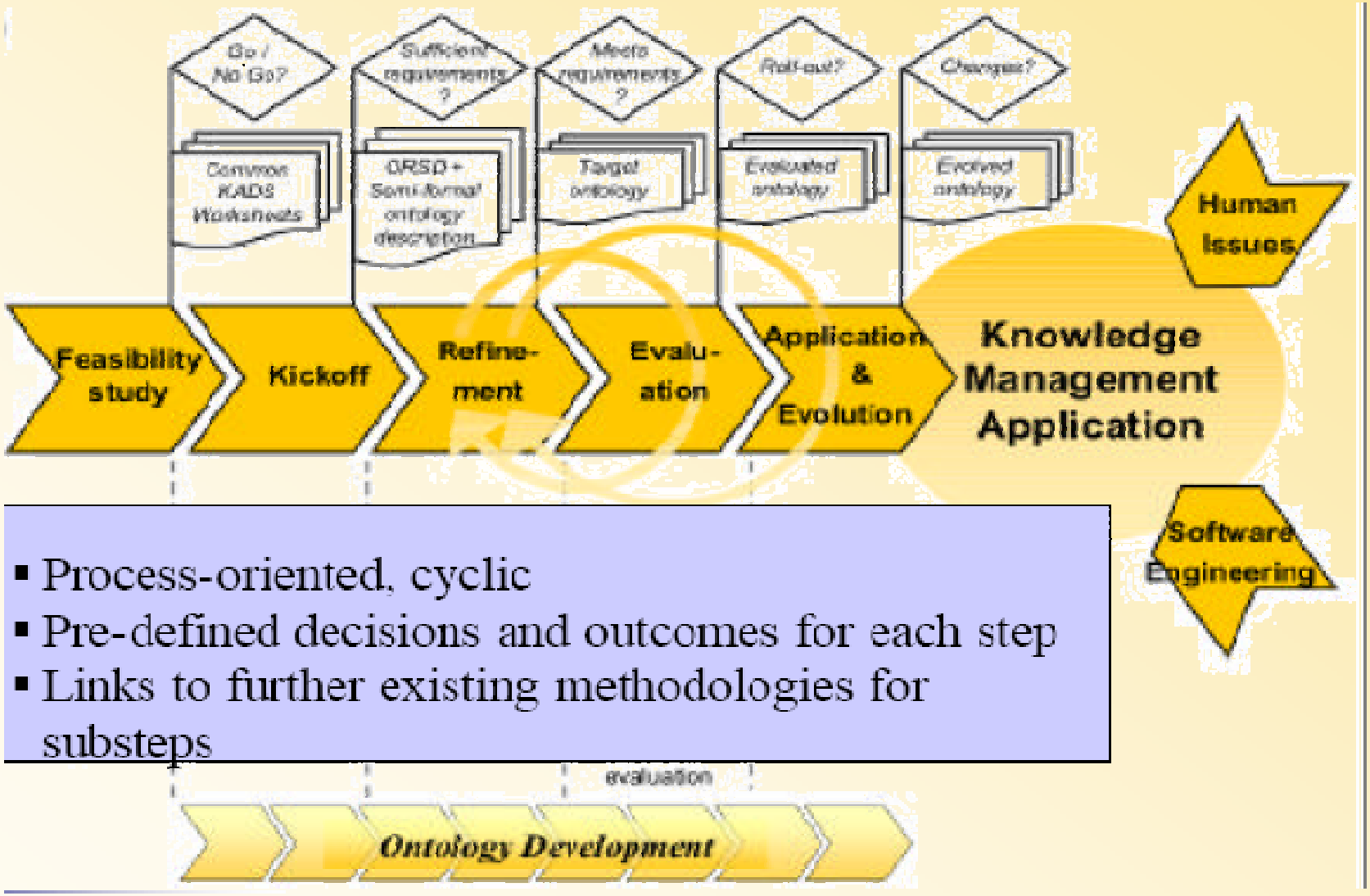
No existe un único modo de modelar un dominio, si no diferentes alternativas. La solución óptima depende de la aplicación. Es esencial tener objetos no ambiguos

- El desarrollo de las ontologías es necesariamente un proceso iterativo.

Fases:

- **Determinar un ámbito** o dominio según: la finalidad de uso de la ontología, el tipo de preguntas y respuestas que se van a realizar y quién usará y mantendrá la ontología.
- **Posibilidad de reutilización** supone un estudio de las ontologías previas existentes y el grado de utilidad que proporcionan.
- **Selección de términos del dominio** están los que determinan conceptos específicos del dominio y aquellos generales que indican propiedades de los términos específicos.
- **Construcción de la taxonomía** con métodos deductivos, inductivos o mixtos. Tener en cuenta que *si una clase A es superclase de B, entonces cada instancia de B lo es de A*.
- **Definición de propiedades** de cada clase. Las subclases heredan las propiedades de las clases.
- **Definición de Slot** (atributos) Se define la cardinalidad, el tipo, dominio y rango.
- **Definición de instancias** perteneciente a una clase y rellenar los slots.
- **Comprobación de posibles anomalías** e inconsistencias, por ejemplo, nombres de clases, ciclos entre las clases y la transitividad de las relaciones jerárquicas.

OTK

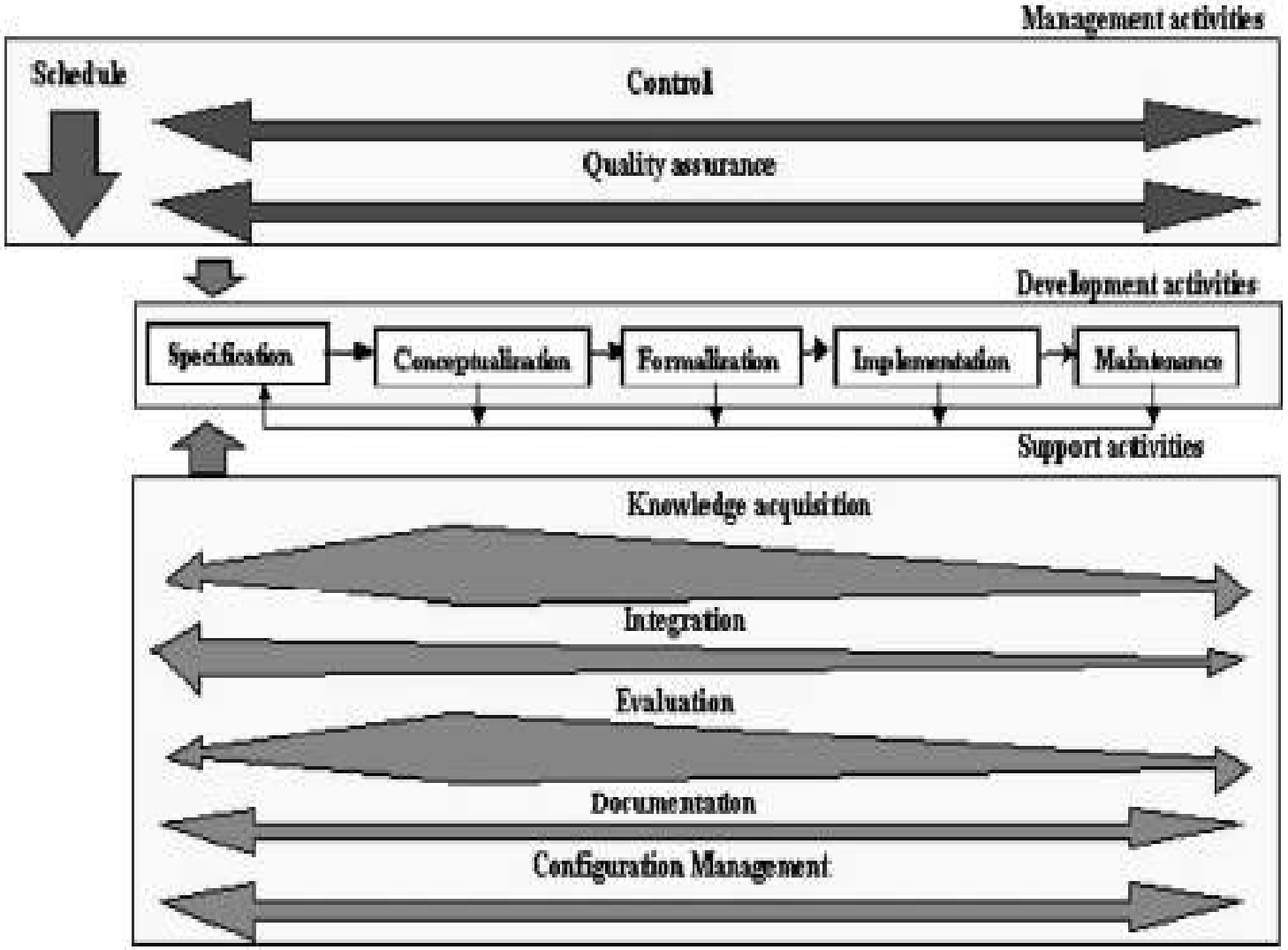


- Process-oriented, cyclic
- Pre-defined decisions and outcomes for each step
- Links to further existing methodologies for substeps

OTK

- **Feasibility study**/ Estudio previo
  - Identificar y delimitar dominio y casos de uso con sus roles
- **Kickoff**/ Inicio
  - Documentos de especificación de requisitos: con usuarios potenciales, recursos y fuentes que se pueden utilizar, cualificación del equipo de desarrollos y reuniones de tormenta de ideas
- **Refinement**/ Refinado
  - Identificación de los elementos de la ontología por los ingenieros en el dominio y formalización
- **Inferencing** / Inferencia
  - En lógica de frames. Se tratan temas de implementación
- **Evaluation** / Evaluación
  - Comprobación de requisitos, pruebas y análisis de calidad (Ontoclean)
- **Application&Evolution**

Methontology I



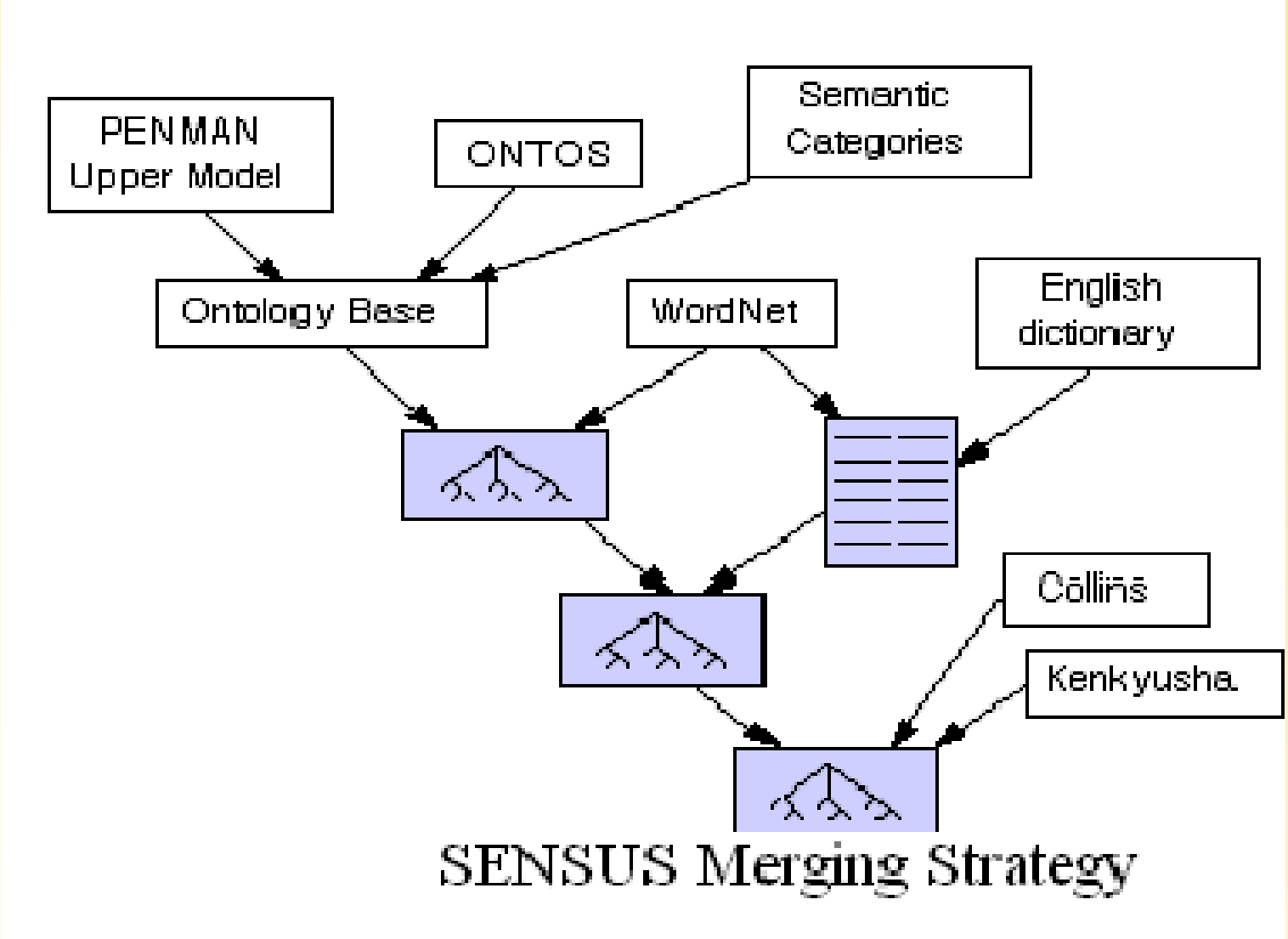
Methontology II

- Es la más utilizada. Utilizada en WebODE y con implementaciones en Protege y OntoEdit
- Etapa de Gestión:
  - Tareas a realizar, orden, tiempos y recursos necesarios
  - Especificar los controles de calidad y tiempos
- Actividades complementarias:
  - Encontrar ontologías que puedan ser reutilizadas (modificándolas o no) para lo que deberán ser evaluadas (puede implicar ingeniería inversa si no hay código)

Methontology III. Diseño

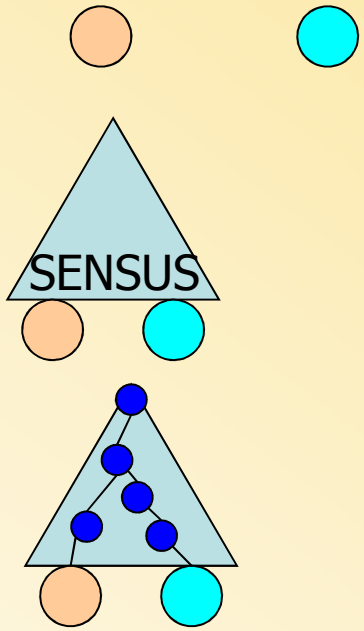
- Conceptualización. Genera gráficos y tablas para que los expertos en el dominio lo supervisen (huye de la expresión en lenguajes de ontologías)
  1. Creación de un glosario con definiciones y sinónimos (y acrónimos) y tipo de dato
  2. Creación de la jerarquía (una o varias). Indicar si completa y disjunta
  3. Relaciones binarias (entre términos y con otras ontologías)
  4. Construir diccionario con instancias, clases, atributos y relaciones
  5. Relaciones binarias en una tabla (nombre, origen, destino, cardinalidad, propiedad transitiva, simétrica, …, navegación)
  6. Atributos de clase e instancias (con unidad de medida si son unidades, cardinalidad, tipo de dato…)
  7. Constantes (nombre, tipo de valor, unidad de medida, inferencias) y funciones
  8. Axiomas (tautologias en lógica de primer orden, rango, dominio, nombre y descripción)
  9. Reglas (expresión lógica “si…y …. Entonces”)
  10. Instancias (nombre, concepto al que pertenece y valores de los atributos)

Creación de Sensus



Crear nuevas ontologías a partir de Sensus

1. Identificar Términos semilla
2. Unir términos semilla a términos que existan en Sensus (a mano)
3. Añadir todos los términos desde el identificado de Sensus al raíz de sensus
4. Añadir nuevos términos que no hayan aparecido (si muchos términos del subárbol son relevantes todo el subárbol lo es)
5. Ir a 2



Algunas aplicaciones (I)

Tipos de kos	***Aplicaciones posibles***	***Ejemplos***
Ontologías (OWL, RDF), metadatos	Rss, foat	http://rss.elmundo.es/rss/ (noticias) http://www.elpais.com/rss/index.html (noticias) http://www.edreams.es/edreams/espanol/rss/eDreams-feeds.jhtml (Agencia de viajes) Alertas (p.e.calendario para uso público o privado) http://www.rsscalendar.com/rss/ **Clasificados** (Casas/Empleos) http://www.masternewmedia.org/es/2006/04/11/rss_usos_y_aplicaciones_no.htm **Seguimiento de Envíos** notificaciones vía alimentadores RSS cuando los paquetes son enviados RSS o Atom para ser escuchados en dispositivos móviles y computadoras personales http://www.apple.com/itunes/ Edición de videos colaborativo pronto puede ser una realidad para muchos http://www.masternewmedia.org/es/2004/12/22/la_segunda_capa_de_la.htm FOAF es un proyecto de Web Semántica, que permite crear páginas Web para describir personas, vínculos entre ellos, y cosas que hacen y crean. http://www.hipertexto.info/documentos/web2.htm Para la bolsa

Algunas aplicaciones (II)

Topic map	Diseño sitios web	Organización de la información (Opera)
		Recuperación de información (Opera)
metadatos	Diseño sitios web	Bibliotecas, catalogación, derecho de propiedad intelectual
		Sistemas de búsqueda, descubrimiento de recursos
		Web semántica, Agentes de software inteligente
		Intercambio normalizado de información (interoperabilidad)
		comercio electrónico, firmas digitales, políticas y preferencias de privacidad
Tesauros	Gestión de la información	Sistemas de organización
		Sistemas de recuperación
Ontologías		Comunicación entre personas y sistemas
		Razonamientos automáticos