

# Sistemas ubicuos e inteligencia ambiental

## Desarrollo de Software Basado en Componentes y Servicios

M.I. Capel

ETS Ingenierías Informática y  
Telecomunicación  
Departamento de Lenguajes y Sistemas Informáticos  
Universidad de Granada  
Email: manuelcapel@ugr.es

**DSBCS**  
Máster en Ingeniería Informática

18 de enero de 2017



# Índice

- 1 Sistemas ubicuos
- 2 Computación UbiCua
- 3 Marcos de trabajo para desarrollo de Sistemas UbiCuos
- 4 Servicios Colaborativos
- 5 Semántica en entornos ubicuos
- 6 Material adicional sobre OWL

# Índice

- 1 Sistemas ubicuos
- 2 Computación UbiCua
- 3 Marcos de trabajo para desarrollo de Sistemas UbiCuos
- 4 Servicios Colaborativos
- 5 Semántica en entornos ubicuos
- 6 Material adicional sobre OWL

# Índice

- 1 Sistemas ubicuos
- 2 Computación UbiCua
- 3 Marcos de trabajo para desarrollo de Sistemas UbiCuos
- 4 Servicios Colaborativos
- 5 Semántica en entornos ubicuos
- 6 Material adicional sobre OWL

# Índice

- 1 Sistemas ubicuos
- 2 Computación UbiCua
- 3 Marcos de trabajo para desarrollo de Sistemas UbiCuos
- 4 Servicios Colaborativos
- 5 Semántica en entornos ubicuos
- 6 Material adicional sobre OWL

# Índice

- 1 Sistemas ubicuos
- 2 Computación UbiCua
- 3 Marcos de trabajo para desarrollo de Sistemas UbiCuos
- 4 Servicios Colaborativos
- 5 Semántica en entornos ubicuos
- 6 Material adicional sobre OWL

# Índice

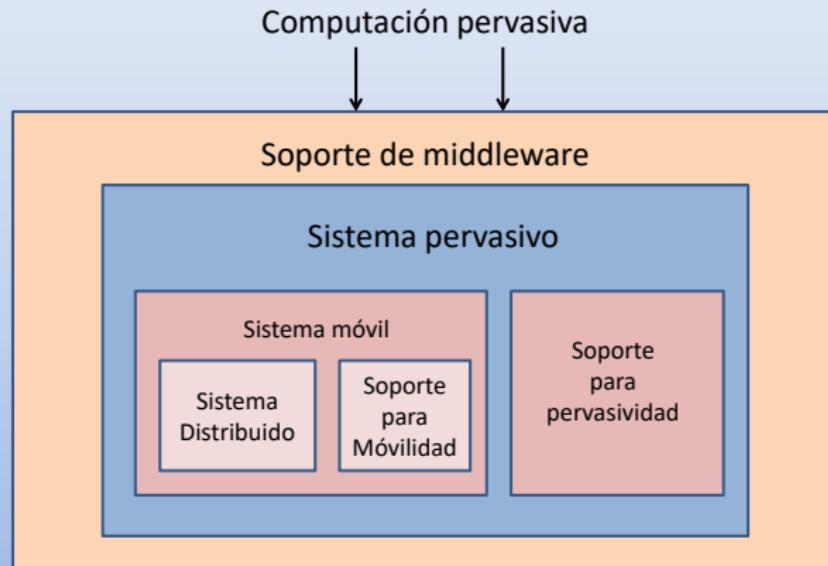
- 1 Sistemas ubicuos
- 2 Computación UbiCua
- 3 Marcos de trabajo para desarrollo de Sistemas UbiCua
- 4 Servicios Colaborativos
- 5 Semántica en entornos ubicuos
- 6 Material adicional sobre OWL

# Introducción-1

## Historia

- Introducción del concepto *Ubiquitous Computing* (UC) por el visionario Mark Weiser ( 1990)
- Sistemas empotrados, interconectados, inteligentes que colaboran entre ellos para realizar tareas cada vez más complejas
- Concepto de *pervasidad*
- Intercambiabilidad conceptual de términos: *computación pervasiva, computación ubicua, inteligencia ambiental, sensibilidad al contexto*

# Soporte para computación pervasiva

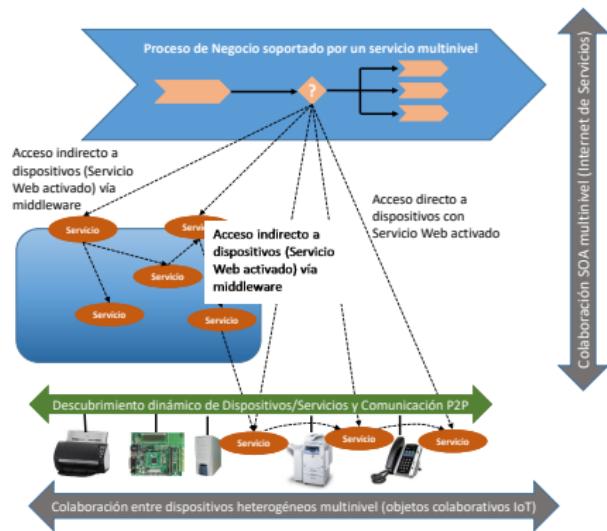


## Introducción-2

- Los dispositivos empotrados con los que interactuamos diariamente poseen ya altas capacidades de comunicación y computación; lo que ha dado lugar a los conceptos:
  - “Internet of Things” [[Atzori et al., 2010](#)]
  - Comunic. “Machine-to-Machine”(M2M) [[Kim et al., 2014](#)]
- La Computación Ubícua (UC) [[Zhang et al., 2013](#)] propicia la convergencia perfecta entre las diferentes tecnologías necesaria para convertir en realidades “IoT” o “M2M”
- No existe todavía un estándar aceptado para el desarrollo de sistemas ubicuos [[Stavropoulos et al., 2013](#)]
- Complejidad “exponencial” respecto del número de dispositivos, derivada de las diferencias de software, redes y de la composicionalidad de servicios

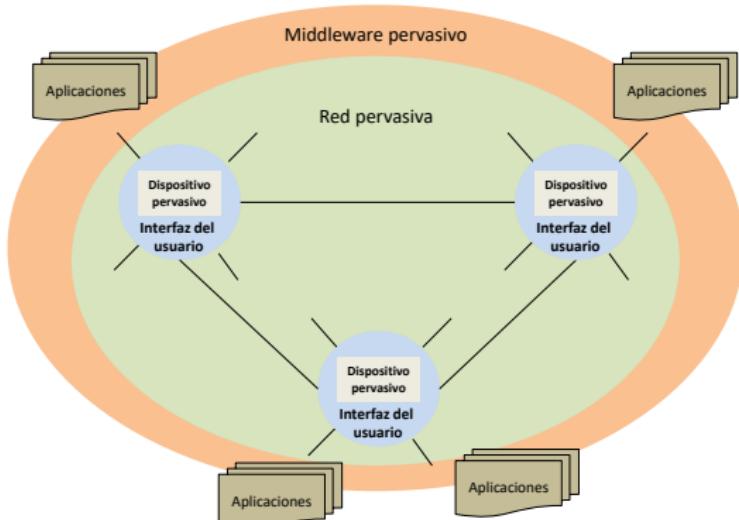


# Arquitectura SOA multinivel para sistema colaborativo



Adecuación del “desarrollo de software” a los requisitos de los sistemas ubicuos

Solución: Mediante el uso de arquitecturas SOA se consigue que la funcionalidad ofrecida por los nodos computacionales del sistema se interprete como un servicio, que puede ser combinado con otros para obtener aplicaciones más complejas

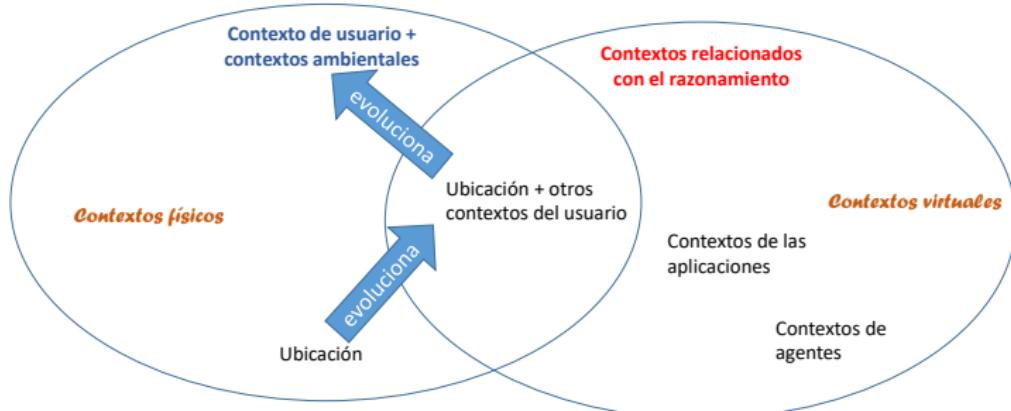


## “Espacio UbiCuo”

- Completamente ocupado por dispositivos
- Funcionamiento “transparente” para sus usuarios
- Comportamiento “colaborativo”
- Proactividad de dispositivos



# Información, semántica y contexto



## Computación Sensible al Contexto

Su objetivo es dar respuesta a situaciones que tienen su origen en el entorno de ejecución de la aplicación, dependiendo de parámetros cambiantes y de la detección de la información que se origina en un contexto. Esta información, una vez obtenida, es utilizada en beneficio del sistema que será capaz de reaccionar adecuadamente [Raz et al., 2006] [Madkour et al., ].



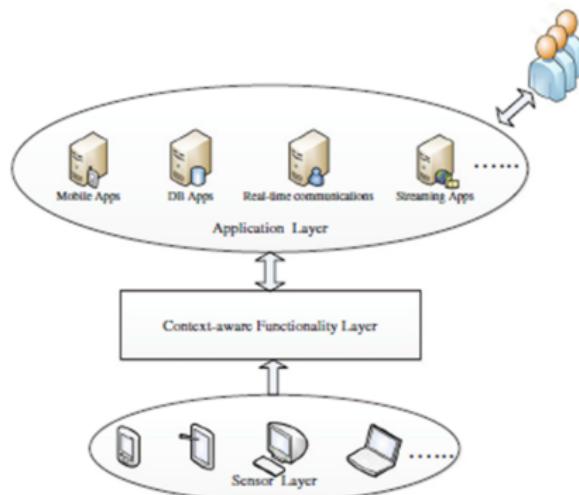
# Información, semántica y contexto-II

## Contexto en entornos de computación ubicua

Conjunto de propiedades que caracterizan el entorno de ejecución de las aplicaciones, relaciones entre componentes y dispositivos en que se suscitan eventos, acciones que afectan a individuos que no afectan directamente a la resolución del problema pero que condicionan su solución.

## Importancia del Contexto

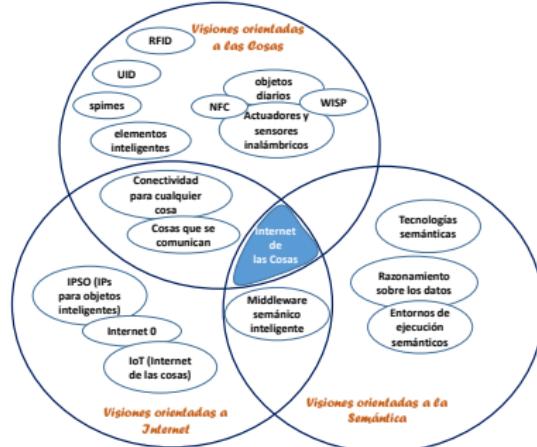
- Propicia la proactividad de los servicios del sistema
- Reduce la participación directa de los usuarios
- Aumenta el grado de '**auto-consciencia**' de los dispositivos
- Modelo del contexto es fundamental para conseguir "inteligencia ambiental"



Forma de conseguir *consciencia de contexto* en sistemas de computación ubicuos



## Importancia del Contexto –II



### Semántica del contexto

El desarrollo de ambientes inteligentes no es sólo un problema de diseño de software, sino que obliga a definir el dominio semántico de información en el que se conformarán tanto los dispositivos como los servicios. Es necesario establecer una metodología que permita manejar la información semántica.

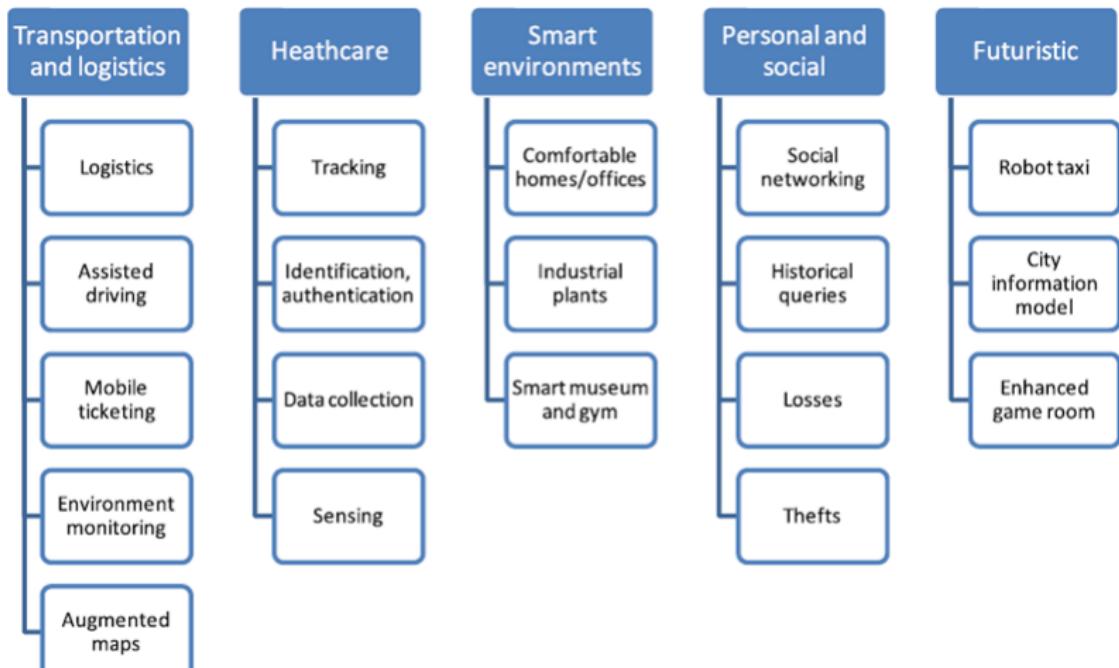


# Programación tradicional vs. en entornos ubicuos

## Ideas fundamentales

- Mayor adaptación al contexto de ejecución: “context-aware computation” [Lassila, 2005] [Perera et al., 2014]
- “Inteligencia ambiental”: independencia respecto del usuario del sistema
- Representación uniforme del contexto, selección de información sensada útil, extracción de conclusiones [Locke, 2006] [Baldauf et al., 2012]
- Limitación en el número de nodos computacionales
- Estandarización de los términos, ontologías de contexto, automatización

# Dominios y principales escenarios de aplicación



# Características de un sistema de computación ubicua

## Elementos imprescindibles

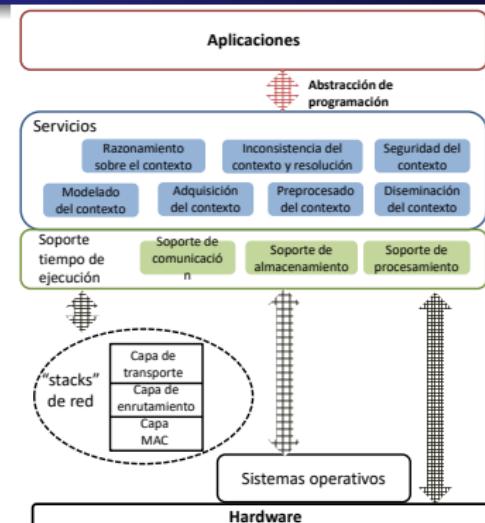
- Interfaz “invisible”
- Uso de “estándares abiertos”
- Utilizar hardware de coste muy bajo
- Reactividad y proactividad del sistema
- Localidad en la ejecución de acciones necesarias
- Adaptación a los entornos

# Plataformas de computación ubicua

## Computación ubicua [[Weiser, 1993](#)]

- Reducción de tamaño de los dispositivos–hardware
- Menor consumo energético
- Incremento capacidad de proceso y memoria
- “Invisibilidad de dispositivos”
- Computación móvil y redes de sensores

# Plataformas de computación ubicua – II



## Características de un entorno de computación ubicua

- capacidad de reconfiguración dinámica
- modularidad
- extensibilidad y transportabilidad

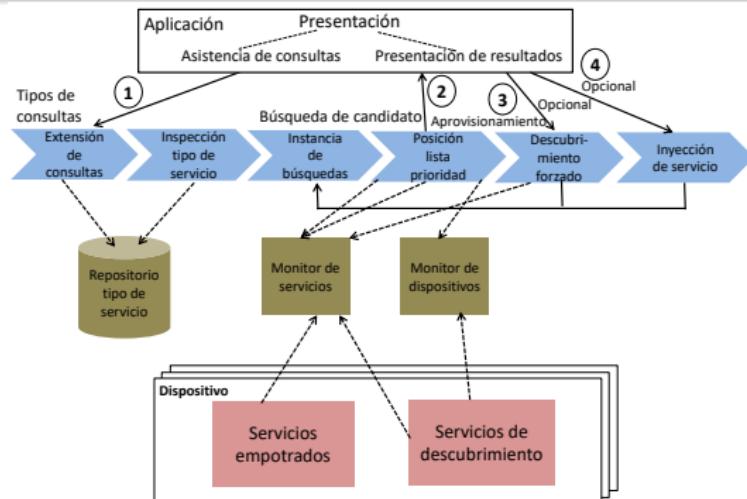


# Plataformas de computación ubicua – III

## Nuevos modelos

- Estilos arquitectónicos: peer-to-peer, pizarra, etc.: se han de evitar arquitecturas centralizadas
- Transparencia, heterogeneidad e interoperabilidad de dispositivos [Banavar and et al., 2000] [Guinard et al., 2010]
- Middleware específico que proporcione: descubrimiento, selección, adaptación y composición de servicios

## Plataformas de computación ubicua – IV



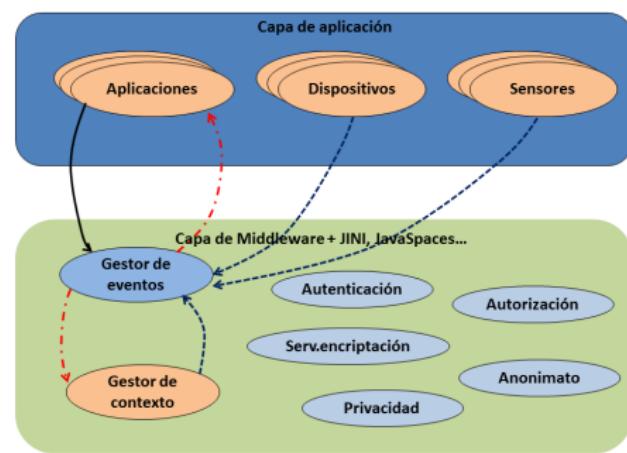
Proceso de descubrimiento de servicios y aprovisionamiento en Device Profile for WS(DPWS) [Guinard et al., 2010]

- SOCRADES Integration Architecture (SIA)
- SIA permite la integración ubicua de servicios de empresa y servicios del mundo real que se ejecutan en empotrados.



# Descripción de tecnologías basadas en Java

- Jini
- JavaSpaces
- Message Oriented Middleware (MOM)
- Web Services (DPWS)
- JXTA



# Jini

## Características fundamentales

- Basado en el concepto de “federación de servicios”
- Arquitectura software Cliente/Servidor [[Apache©, 2014](#)]
- Directorio de servicios mediante *LookUp Service* de Apache, que definen *proxies* para su acceso
- Los proxies implementan interfaces de Java para representar *contratos de servicio*
- Descubrimiento de servicios flexible pero poco *robusto*

# Java Spaces

## Idea fundamental

- Basado en *tuplas* globalmente compartidas por proveedores, clientes, junto con recursos de red y objetos
- El espacio virtual de tuplas es el soporte para definir un Servicio de Directorio SOA
- También para llevar a cabo publicación y descubrimiento de servicios

# Colas de Mensajes Distribuidas

## “Message Oriented Middleware”(MOM)

- Estructura de paso de mensajes no bloqueante para un marco de trabajo de acuerdo con los principios SOA
- Simplifica al máximo la coordinación entre servicios
- Muy bajo acoplamiento entre servicios, pensado para nodos computacionales en redes

# Servicios Web

## Web Services (DPWS)

- La propuesta más difundida actualmente para desarrollo de sistemas que siguen los principios SOA
- Cada aplicación se convierte en un componente integrante de un *servicio Web* en la red global de servicios
- Descripción del “contrato de servicio” utilizando WSDL
- Localización del SW a través de UDDI
- Basado en el protocolo SOAP
- Numerosas publicaciones de investigación que utilizan SW para desarrollar sistemas de computación ubicua

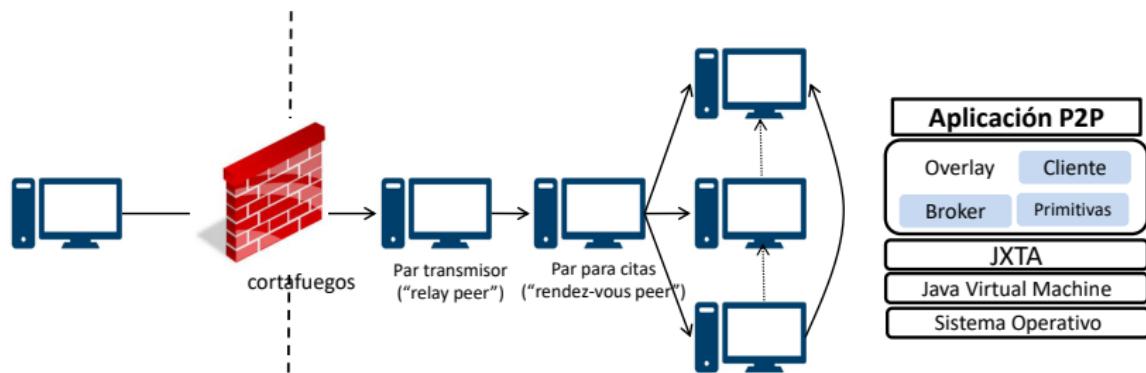
# Marcos de trabajo basados en P2P

## JXTA

- Permite llevar a cabo tareas de publicación, descubrimiento, colaboración e interacción entre servicios [[Oracle©, 2014](#)] de acuerdo con SOA
- Los nodos computacionales de una arquitectura P2P son ambivalentes: se intercambian como clientes o servidores
- Necesita menos trabajo de gestión que arquitecturas centralizadas como Jini
- Mejor para sistemas distribuidos con recursos muy limitados (empotrados)
- Indexación de publicación y consultas: supernodos, *Rendez-Vous* y *Relay-peers* muy eficiente para SOAs

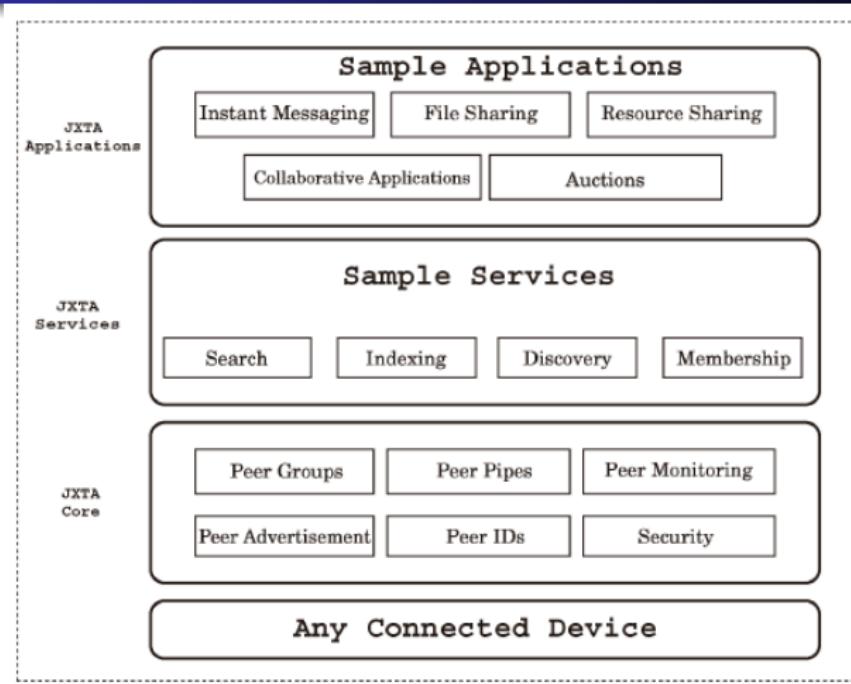


# Middleware JXTA para sistemas colaborativos [Barolli and Xhafa, 2011]



Middleware JXTA para sistemas colaborativos

# Middleware JXTA para sistemas colaborativos



Capas y servicios que ofrece la arquitectura de JXTA

# Tecnologías basadas en Java: resumen

- Jini, JavaSpaces, MOM, Web Services y JXTA

	<b>Jini</b>	<b>JavaSpaces</b>	<b>MOM(JMS)</b>	<b>WS (DPWS)</b>	<b>JXTA</b>
Representación datos	Java Marshal. Objects	Colecciones de Información	Mensajes	XML	XML
Transporte	Protocolos basados RMI	Serialización	Protocolos basados RMI	SOAP sobre HTTP	TCP/IP o HTTP
Descripción Servicios	Interfaz de Java	Contrato de Interfaz	Objetos administrados	WSDL	Anuncios
Servicios Localización	Servicio Lookup	Puntos entrada	Publish Subscribe	UDDI	Descubr. servicios
Ligaduras lenguajes	Java	Java	Java	Java,.NET Perl	Java,C
Referencias remotas	Objetos proxy	Objetos proxy	Identificación	URL	R-Vous de peers
Sincronicidad	sinc/asinc	-	asinc	sinc	sinc/asinc
Tipo de arquitect.	Cliente servidor	Cliente servidor	Cliente servidor	Cliente servidor	Peer-to peer

# Colaboratividad

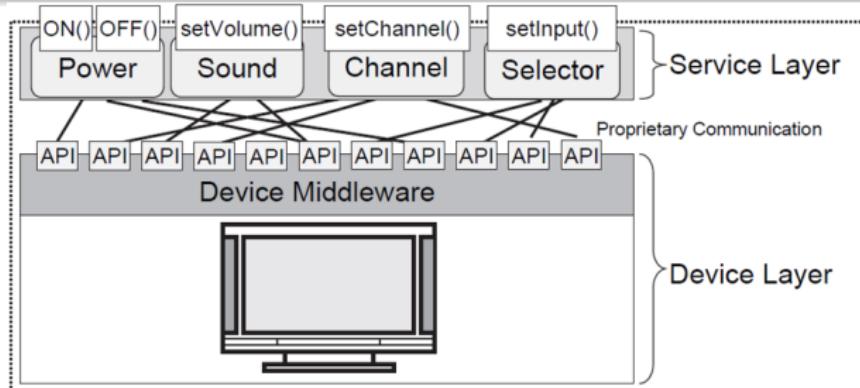
## Ideas fundamentales

- **Proactividad** de los dispositivos en espacios ubicuos, propicia sistemas basados en IA
- Dispositivo en espacios ubicuos: autónomos, distribuidos, interconectados, y **colaborativos**
- **Servicio** según las arquitecturas SOA es una ayuda fundamental para el desarrollo de aplicaciones ubicuas
- **Composición** de servicios orientada a modelado de negocios no resulta adecuada en Computación UbiCua
- Adecuación principios SOA a **requerimientos específicos** de los sistemas ubicuos

# Composición de servicios

## Ideas fundamentales

- Objetivo de la “*composición de servicios*” para sistemas ubicuos
- Inteligencia Ambiental  $\Leftrightarrow$  Composición de Servicios
- Composición de Servicios  $\Leftrightarrow$  Colaboratividad



# Composición de servicios-II

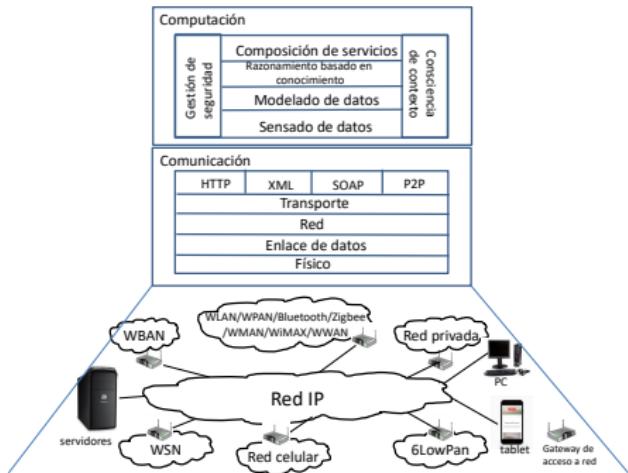
¿quién o qué es responsable de la misr

¿cuándo se lleva a cabo la composiciór

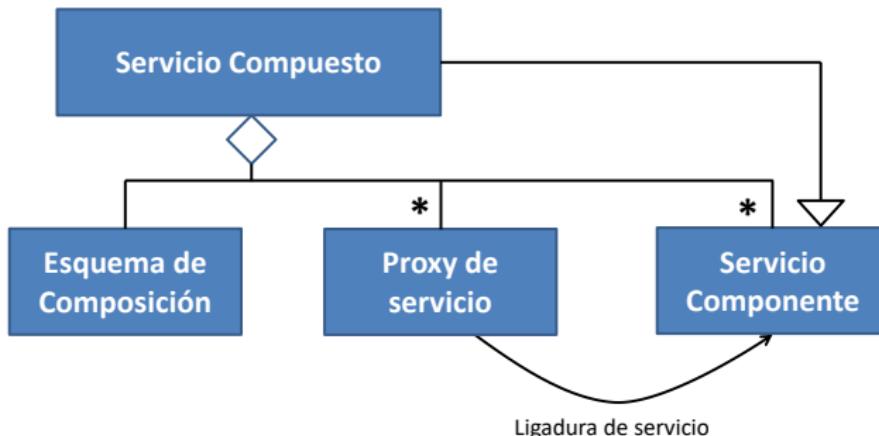
¿dónde se llevará a cabo? y

¿cómo se gestiona?

Sistema de inteligencia ambiental en un escenario real, transcendiendo redes y con varias ubicaciones

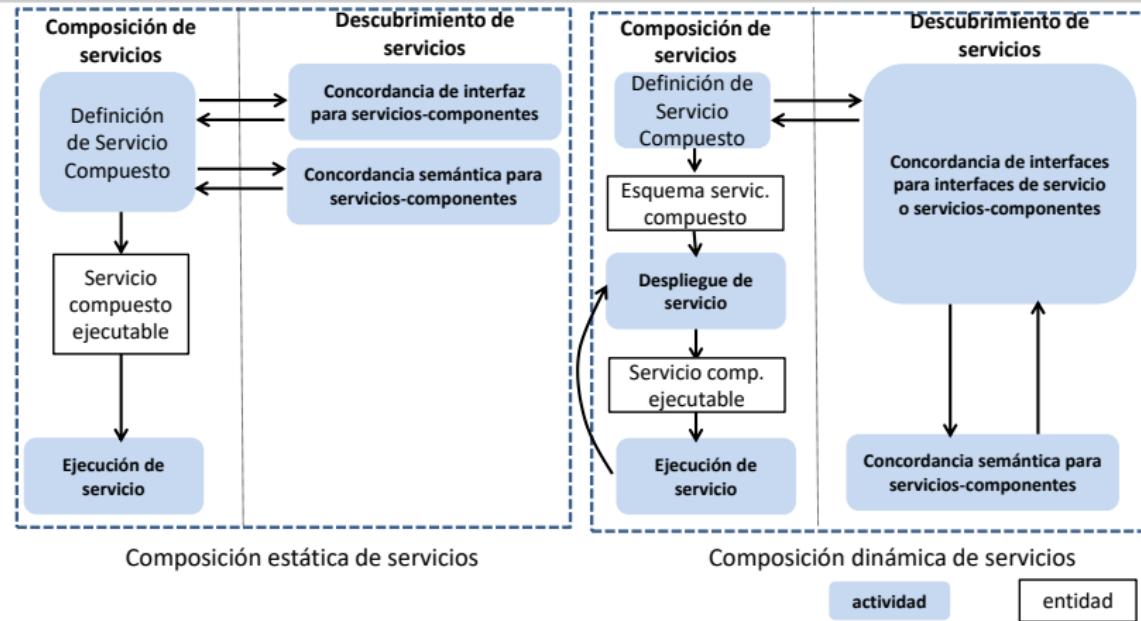


# Modelo conceptual de Composición de Servicios



Modelo conceptual de servicios para sistemas ubícuos

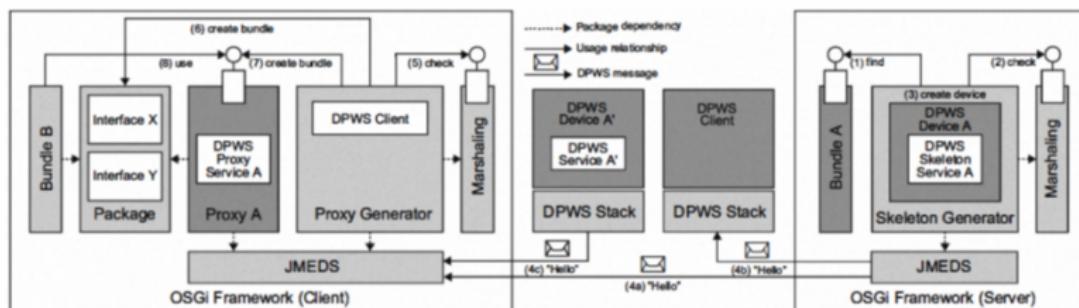
# Tiempo de la Composición de Servicios



Tipos de composición de servicios en sistemas ubicuos

# Lugar de la Composición de Servicios

Composición de Servicios centralizada [Dohndorf et al., 2010]

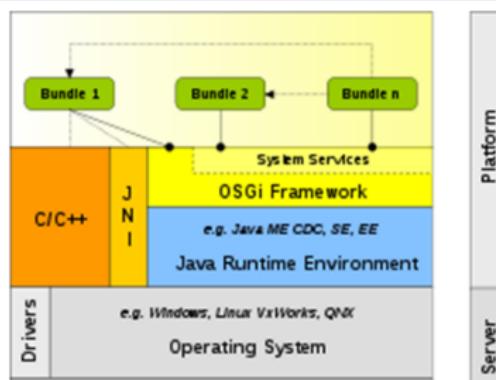


Generación de proxies y skeletons en la arquitectura OSGi

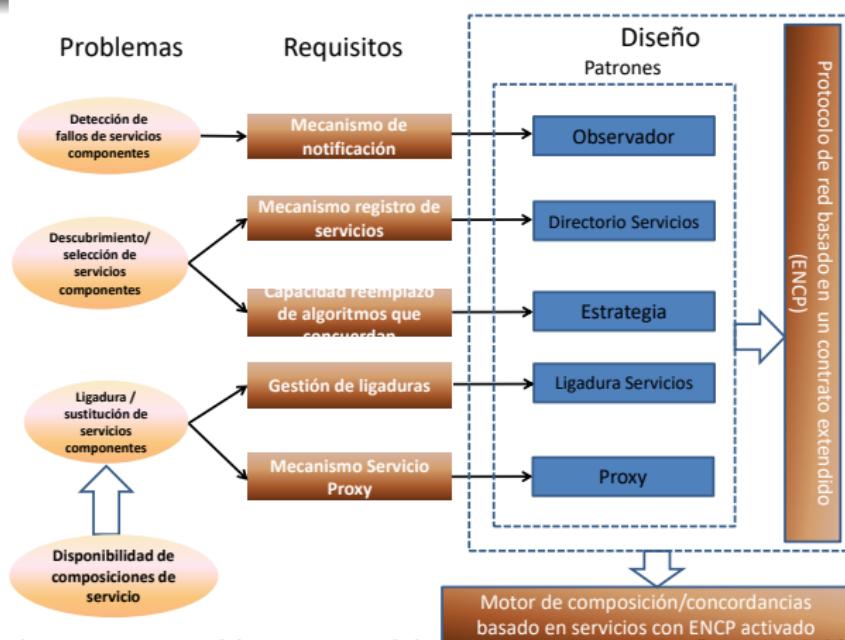
# Arquitectura Open Services Gateway initiative (OSGi)

## Idea fundamental

- Con OSGi se implementa un modelo de componentes dinámico y completo, que no existe en un entorno Java/VM considerado aisladamente
- Los componentes “bundle” para su despliegue y pueden ser instalados, arrancados, parados o actualizados remotamente sin necesidad de reiniciar
- OSGi inicialmente una especificación para gateways, ahora se utiliza en el IDE Eclipse y multitud de entornos de desarrollo, tales como apps para móviles



# Validación de la Composición de Servicios



Relaciones entre problemas, requisitos y patrones en la Composición de Servicios

# Inclusión de información semántica en los servicios

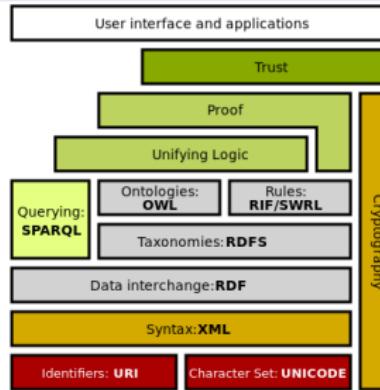
## Ideas fundamentales

- Procesable directamente por dispositivos sin intervención humana
- Los servicios encuentran de forma dinámica dispositivos y servicios con los que colaborar
- Necesidad de contar con notaciones para expresar semánticamente entidades y relaciones dentro de un dominio



# Web semántica

- Lenguajes para la representación de conceptos (casi todos sobre XML)
- XML no sirve para indicar restricciones que afectan al significado de los documentos estructurados
- RDF es un modelo de datos para definir recursos y relaciones entre ellos
- Los lenguajes de ontologías (RDF, OWL, OWL-S) permite definir relaciones complejas entre clases y caracterización de propiedades



Estructura de la web Semántica para el W3C

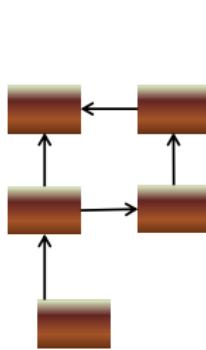
# RDF

## Resource Description Framework (RDF)

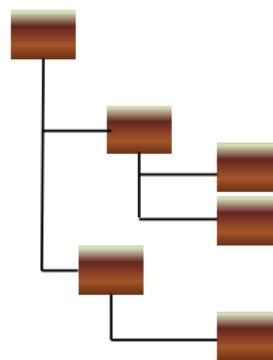
- Consideración como el *lenguaje de la Web Semántica*
  - Modelo estándar para intercambio de datos
- Características que propician la fusión de modelos de datos
- Evolución de esquemas

*"If the graph data model is the model the semantic web uses to store data, RDF is the format in which it is written".*

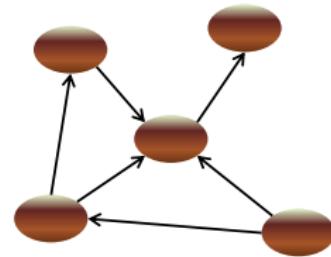
## RDF – II



Base de datos relacional.  
Tablas relacionadas  
mediante una clave primaria



Base de datos jerárquica.  
Los nodos-padre son  
más importantes



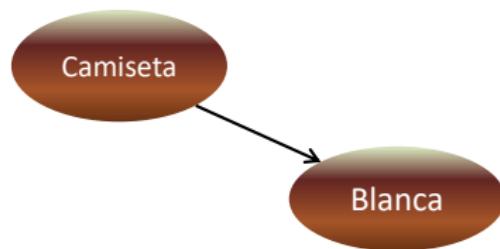
Base de datos en grafo.  
Se establecen relaciones  
arbitrarias entre objetos

Base de datos organizada como un grafo

# Elementos de RDF/XML

## Tripletas

- Proporcionar significado a los datos
- Composición:
  - Sujeto
  - Predicado
  - Objeto



# La sentencia RDF/XML

```
1 01.<?xml version="1.0" encoding="UTF-8"?>
2 02.
3 03.<rdf:RDF
4 04. xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5 05. xmlns:feature="http://www.linkeddatatools.com/clothing-
   features#">
6 06.
7 07.<rdf:Description rdf:about="http://www.linkeddatatools.com/
   clothes#t-shirt">
8 08.<feature:size>12</feature:size>
9 09.<feature:color rdf:resource="http://www.linkeddatatools.com/
   colors#white"/>
10 10.
11 11.</rdf:Description>
12 12.
13 13.</rdf:RDF>
```

## Formalización de la sentencia RDF

```
1.<rdf:Description rdf:about="sujeto">  
2.<predicate rdf:resource="objeto" />  
3.<predicate>valor literal </predicate>  
4.<rdf:Description>
```

El elemento `rdf:Description` de RDF/XML nos permite agrupar 1 ó más sentencias en un solo contenedor. La forma general contiene realmente 2 sentencias que se refieren al mismo sujeto, cada una con 1 predicado y 1 objeto.

# Ejercicio

```
1 01.<?xml version="1.0" encoding="UTF-8"?>
2 03.<rdf:RDF
3 04. xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4 05. xmlns:dc="http://purl.org/dc/elements/1.1/"
5 06. xmlns:region="http://www.country-regions.fake/"
6 08.<rdf:Description rdf:about="http://en.wikipedia.org/wiki/
    Granada">
7 09.<dc:title>Granada</dc:title>
8 10.<dc:coverage>Andalucia</dc:coverage>
9 11.<dc:publisher>Wikipedia</dc:publisher>
10 12.<region:population>236,982</region:population>
11 13.<region:principaltown rdf:resource="http://www.country-
    regions.fake/granada"/>
12 14.</rdf:Description>
13 16.</rdf:RDF>
```

Identificar: sujeto de la sentencia RDF/XML, predicados: *recursos o literales* y los objetos a los que se refieren.



# Ejercicio: Solución

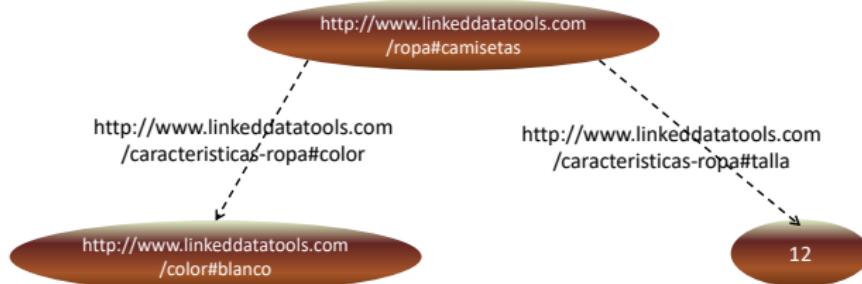
Triples of the Data Model

Number	Subject	Predicate	Object
1	<a href="http://en.wikipedia.org/wiki/Granada">http://en.wikipedia.org/wiki/Granada</a>	<a href="http://purl.org/dc/elements/1.1/title">http://purl.org/dc/elements/1.1/title</a>	"Granada"
2	<a href="http://en.wikipedia.org/wiki/Granada">http://en.wikipedia.org/wiki/Granada</a>	<a href="http://purl.org/dc/elements/1.1/coverage">http://purl.org/dc/elements/1.1/coverage</a>	"Andalucia"
3	<a href="http://en.wikipedia.org/wiki/Granada">http://en.wikipedia.org/wiki/Granada</a>	<a href="http://purl.org/dc/elements/1.1/publisher">http://purl.org/dc/elements/1.1/publisher</a>	"Wikipedia"
4	<a href="http://en.wikipedia.org/wiki/Granada">http://en.wikipedia.org/wiki/Granada</a>	<a href="http://www.country-regions.fake/population">http://www.country-regions.fake/population</a>	"236,982"
5	<a href="http://en.wikipedia.org/wiki/Granada">http://en.wikipedia.org/wiki/Granada</a>	<a href="http://www.country-regions.fake/principaltown">http://www.country-regions.fake/principaltown</a>	<a href="http://www.country-regions.fake/granada">http://www.country-regions.fake/granada</a>

*Triples del modelo obtenidos con el validador W3C de RDF*

## Espacio de nombres XML: URLs

Los espacios-XML de URLs en RDF se utilizan para distinguir entre propiedades con el mismo nombre de XML ("tag").



Obtención del URI cualificado mediante sustitución de su prefijo.

Si se define un espacio de nombres:

`xmlns:feature="http://www.linkeddatatools.com/caracteristicas-ropa#"`

Entonces la etiqueta del grafo sólo sería, `caracteristica:color`



# Modelado semántico

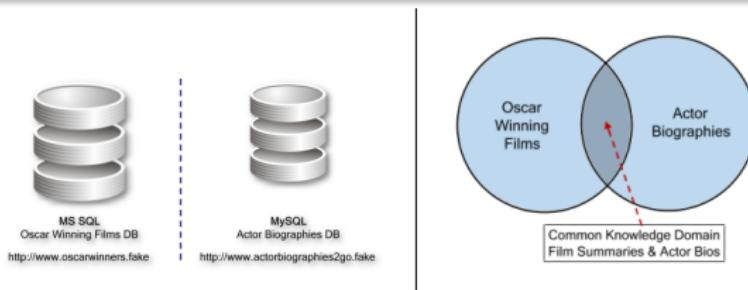
## Ideas fundamentales sobre RDF

- Modelo para registrar datos, globalmente intercambiables
- No prevé nada para guardar el significado de los datos

Modelo	Formato	Datos	Metadatos	Identificación	Consultas	Semánt.
Serialización objetos	.NET CLR obj.	Valores propiedad	Valores propiedad	Nombre archivo	LINK	N.D.
Relacional	Oracle MySQL	Valores celdas	Definición colum.yfilas	Valores clave única	SQL	N.D.
Jerárquico	XML	Valores tag/atrib.	XSD/ DTD	Valores clave/atrib.	XPath	N.D.
Grafo	RDF/XML Turtle	RDF	RDFS/ OWL	URI	SPARQL	sí, con OWL

# Integración de conocimiento

Razón para incluir información semántica asociada a los datos.



This sort of information interchange across incompatible, independently designed data systems takes time, money and human contextual interpretation of the different datasets. It also is restrictive to the data domains of only these two websites, any further additions to their knowledge from elsewhere will demand similar efforts. It requires humans to understand the meaning of the data and agree on common formats to collaborate the two databases appropriately.



# Modelado semántico de los datos

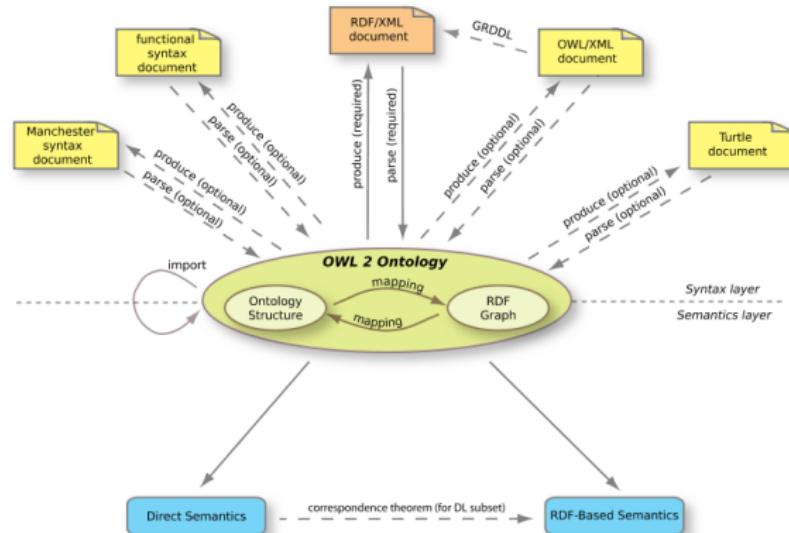
Ontología base + punto-final consultable públicamente:

- Los sitios de ambas bases de datos se pueden consultar autónomamente
- Extensión posterior de la información integrada
- Facilitar búsquedas más completas
- Los sitios que contienen las bases de datos no necesitan ser transformados o reorganizados de ninguna manera

## Iniciativas internacionales – metadatos

- Dublin Core Metadata Initiative (DCMI)
- Friend Of A Friend (FOAF)
- OpenCyc

# Lenguajes de ontologías



La estructura de OWL 2.0

## Ejemplo preliminar con OWL

```
1 <rdf :RDF
2 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4 xmlns:owl="http://www.w3.org/2002/07/owl#"
5 xmlns:dc="http://purl.org/dc/elements/1.1/">
6 <!-- Ejemplo de OWL Header -->
7 <owl:Ontology rdf:about="http://www.linkeddatatools.com/plants">
8 <dc:title>El ejemplo LinkedDataTools.com de ontologia de plantas
9 <dc:description>Un ejemplo de ontologia escrito para
    LinkedDataTools.com RDFS & OWL </dc:description>
10 </owl:Ontology>
11 <!-- Ejemplo Definicion Clase OWL -->
12 <owl:Class rdf:about="http://www.linkeddatatools.com/plants#
    planttype">
13 <rdfs:label>El tipo de planta</rdfs:label>
14 <rdfs:comment>La clase de los tipos de plantas.</rdfs:comment>
15 </owl:Class>
16 </rdf:RDF>
```

## Clases, subclases e *individuals* en OWL

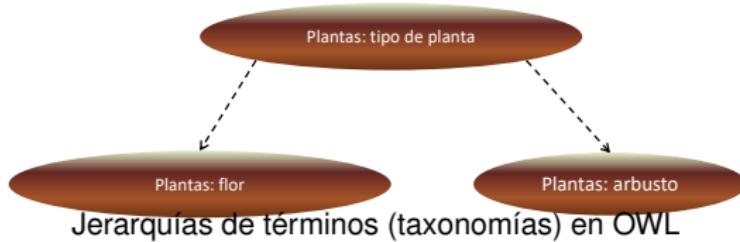
```
1 <!-- OWL Definicion de subclase – Flor -->
2 <owl:Class rdf:about="http://www.linkeddatatools.com/plants#
  flowers">
3 <!-- Flores es una subclase de planttype -->
4 <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/
  plants#planttype"/>
5 <rdfs:label>Plantas que dan flores </rdfs:label>
6 <rdfs:comment>A estas plantas tambien se las conoce como
  angiospermas.</rdfs:comment>
7 </owl:Class>
8 <!-- Definicion de una subclase de OWL – Arbusto -->
9 <owl:Class rdf:about="http://www.linkeddatatools.com/plants#
  shrubs">
10 <!-- Un arbusto (shrub) es una subclase de planttype -->
11 <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/
  plants#planttype"/>
12 <rdfs:label>Arbustos </rdfs:label>
13 <rdfs:comment>Los arbustos son un tipo de planta que poseen
  ramas desde el pie.</rdfs:comment>
14 </owl:Class> ...
```



## Definición de propiedades con OWL

### Propiedades: relación entre *individuals*

- Propiedades entre objetos (`owl:ObjectProperty`): relaciona individuos (instancias) de 2 clases OWL.
- Propiedades de ADTs (`owl:DatatypeProperty`): relaciona individuos (instancias) de clases OWL con valores literales.



## Ejemplo de propiedad de *tipo de datos* con OWL

```
1 <rdf :RDF
2 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4 xmlns:owl="http://www.w3.org/2002/07/owl#"
5 xmlns:dc="http://purl.org/dc/elements/1.1/"
6 xmlns:plants="http://www.linkeddatatools.com/plants#">
7 <!-- Cabecera OWL omitida por brevedad -->
8 <!-- Clases de OWL omitidas por brevedad -->
9 <!-- Definir la propiedad de la familia -->
10 <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/
    plants#family"/>
11 <rdf:Description rdf:about="http://www.linkeddatatools.com/
    plants#magnolia">
12 <!-- Magnolia es un tipo (instancia) de la clase: flowers -->
13 <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
    flowers"/>
14 <!-- La magnolia forma parte de la familia 'Magnoliaceae'-->
15 <plants:family>Magnoliaceae</plants:family>
16 </rdf:Description>
17 </rdf:RDF>
```



## Clases, subclases e *individuals* en OWL-II

```
1 1<!-- Ejemplo de sentencia 'individual' (Instancia) de RDF -->
2 <rdf:Description rdf:about="http://www.linkeddatatools.com/
3   plants#magnolia">
4 <!-- Magnolia es un tipo (instancia) de la clasificacion de las
5   flores -->
6 <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
7   flowers"/>
8 </rdf:Description>
9 </rdf:RDF>
```

## Diferencias entre clases–POO y tipo de clase–OWL

- Las propiedades que poseen las instancias (*individuals*) no se describen en la definición de tipo de su clase
- Consecuencias para la definición observable en las instancias (independencia de las propiedades)
- Tiempo de definición de las propiedades

## Ejemplo de propiedad entre objetos con OWL

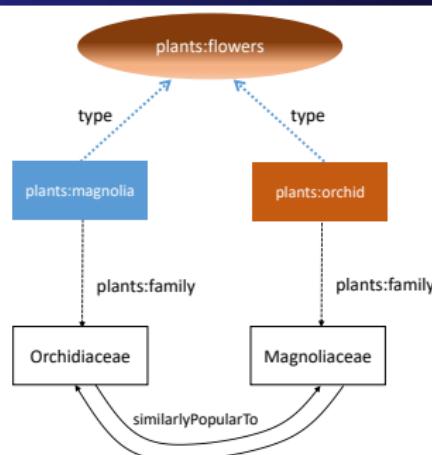
```
1 <owl:ObjectProperty rdf:about="http://www.linkeddatatools.com/
  plants#similarlyPopularTo"/>
2 <rdf:Description rdf:about="http://www.linkeddatatools.com/
  plants#orchid">
3 <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
  flowers"/>
4 <plants:family>Orchidaceae</plants:family>
5 <!-- Las orquideas son tan populares como las magnolias -->
6 <plants:similarlyPopularTo rdf:resource="http://www.
  linkeddatatools.com/plants#magnolia"/>
7 </rdf:Description>
8
9 <rdf:Description rdf:about="http://www.linkeddatatools.com/
  plants#magnolia">
10 <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#
  flowers"/>
11 <plants:family>Magnoliaceae</plants:family>
12 <plants:similarlyPopularTo rdf:resource="http://www.
  linkeddatatools.com/plants#orchid"/>
13 </rdf:Description>
```



## Ejercicio

Dibujar un grafo que muestre a las instancias de las clases Magnolia y Orchid y sus respectivos predicados, de acuerdo con la especificación OWL definida anteriormente.

## Ejercicio: solución



*similarlyPopularTo:* relación bidireccional entre 2 instancias de una misma clase OWL

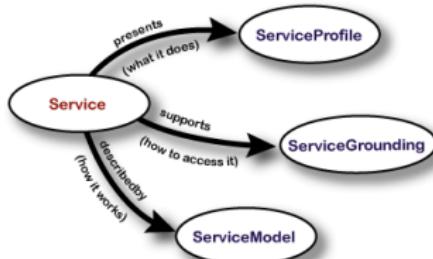
Notar que:

- Las propiedades de objeto no tienen por qué ser bidireccionales
- Tampoco tienen por qué relacionar 2 instancias de la misma clase: se puede establecer entre clases OWL completamente distintas



## Web Ontology Language for Services (OWL-S)

- OWL-S es un ontología con los elementos fundamentales para describir semánticamente las capacidades de los servicios
- Servicios atómicos y servicios compuestos en OWL-S
- Su objetivo es hacer posible el descubrimiento, invocación y composición automática de servicios



Parte superior de la ontología de servicios OWL-S

# Web Ontology Language for Services (OWL-S)–ii

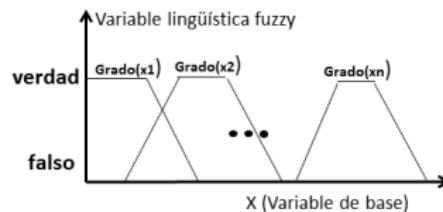
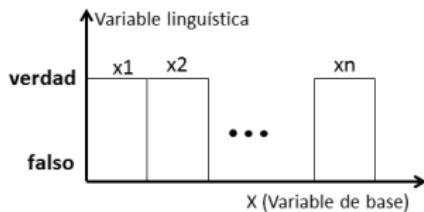
## Conceptos fundamentales del servicio

- Service Profile: ¿qué hace el servicio?
- Service Model: ¿cómo se usa el servicio?
- Service Grounding: ¿cómo interaccionar con el servicio?

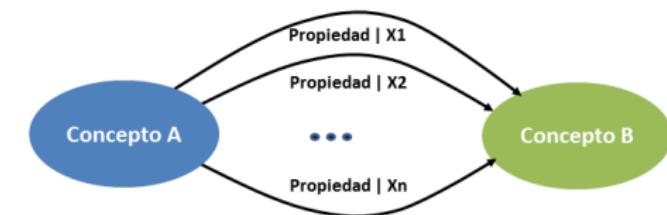


## Composición de servicios sensibles al contexto

Representación abstracta del mecanismo de *pertenencia* de una propiedad –expresada por una variable base (X)– a un intervalo, para una descripción precisa o ambigua de certeza de la propiedad.

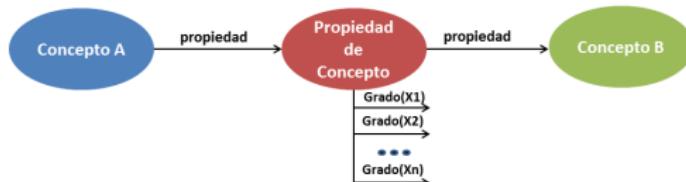


## Composición de servicios sensibles al contexto-II



Representación abstracta del mecanismo “propiedad”

## Composición de servicios sensibles al contexto–III



Grado de pertenencia de una “propiedad” a intervalo de certeza

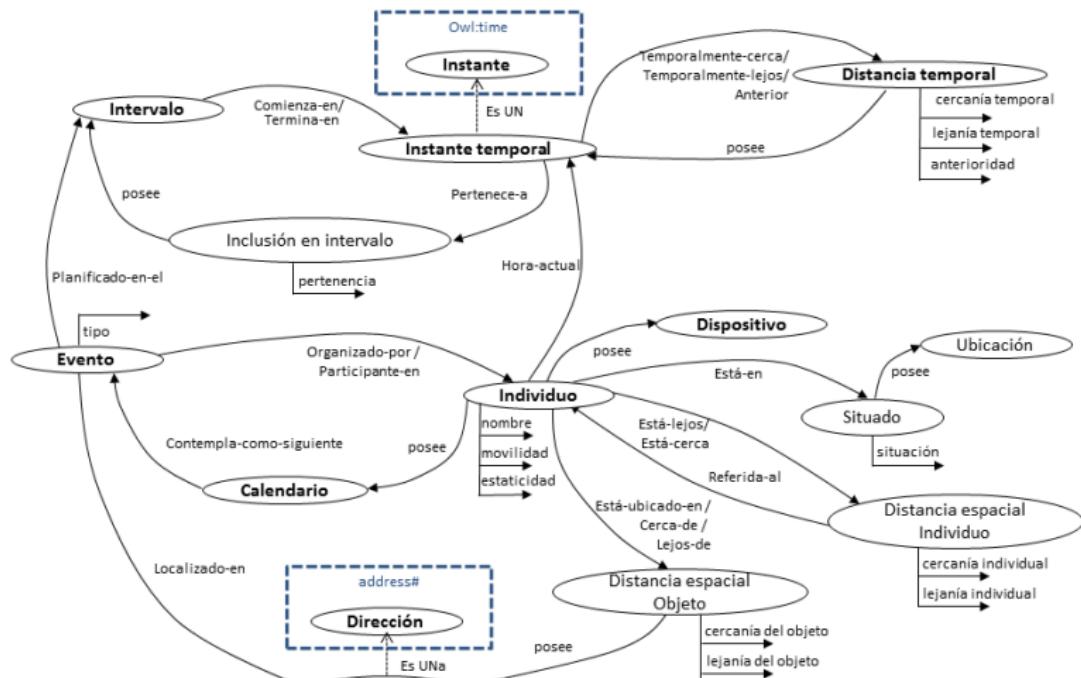
Patrón de representación aplicable a las propiedades relacionadas con la misma variable base y con el mismo par de conceptos.

## Composición de servicios sensibles al contexto–IV

### Ejemplo de representación ontológica de propiedades

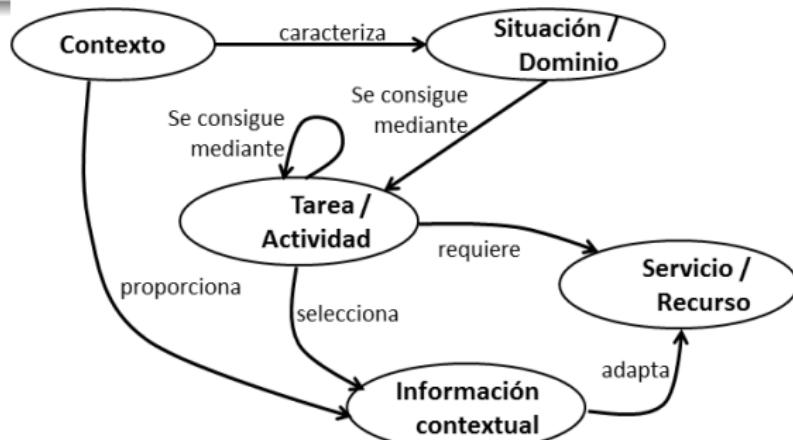
- “Cercanía/lejanía” de un individuo a un lugar / tiempo
- Variable de base: “distancia” (espacial o temporal)
- Conceptos: Individuo, Lugar, Instante temporal, etc.
- Propiedades
- La certeza de en el cumplimiento de las propiedades dependerá del grado de pertenencia de “distancia” a un intervalo de valores establecido previamente.

# Representación ontológica de “cercanía”/“lejanía”

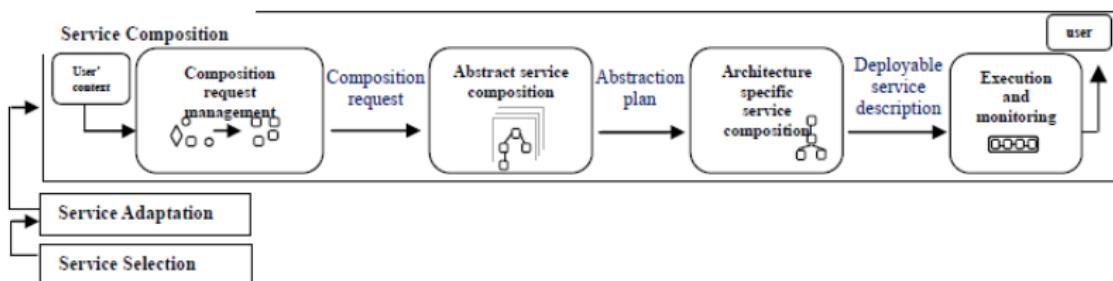


## Reglas asociadas al modelo ontológico

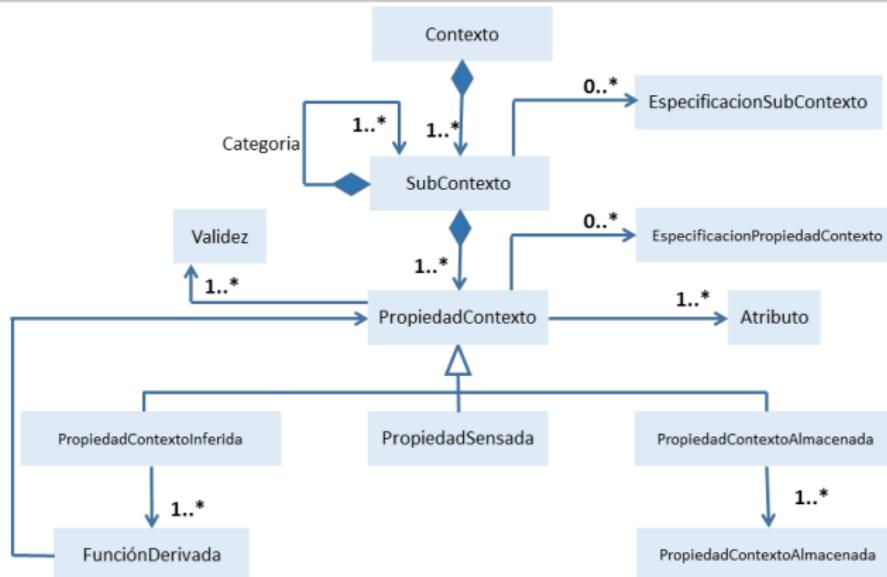
Conjunto de reglas para inferir situaciones actuales del modelo respecto de la ontología de representación de propiedades utilizando variables-base y el concepto de pertenencia a un intervalo de valores.



# Composición de servicios–fuzzy



# Modelos de representación del conocimiento y sistemas ubicuos



Metamodelo del contexto en entornos ubicuos

# Modelos de representación del conocimiento y sistemas ubicuos-II

Enfoque Requerimientos	Composición distribuida	Verificación parcial	Riqueza y calidad info.	Ambigüedad no compl.	Nivel formalidad	Aplicabilidad a frameworks
Modelos de BD clave-valor	—	—	—	—	—	+
Esquemas de marcado	+	++	—	—	+	++
Modelos gráficos	—	-	+	—	+	+
Modelos OO	++	+	+	+	+	+
Modelos lógicos	++	-	-	-	++	-
Modelos ontológicos	++	++	+	+	++	+

## Ontologías como dominio de conocimiento

- CONON [Wang et al., 2004]—CONtext ONtology
- CoDAMoS [Preuveneers et al., 2004] –Context–Driven Adaptation of MOBILE Services
- COBRA-ONT [Chen et al. , 2004]
- SOUPA [Chen et al., 2004] –Standard Ontology for Ubiquitous and Pervasive Applications—

## Resumen de ontologías que representan a dominios conocimiento característicos de ubicuos

	CONON	CoDAMoS	Cobra-ONT	SOUPA
Dispositivo	X	X	X	-
Entorno	-	X	X	-
Interfaz	-	-	-	-
Situación	X	X	X	X
Red	X	-	-	-
Rol	-	X	X	-
Servicio	X	X	-	-
Tiempo	-	X	X	X
Usuario	X	X	X	X

## Proyectos de investigación que utilizan ontologías para modelar el contexto ubicuo

- Utilización de agentes inteligentes + adaptación a los cambios de contexto en tiempo real [Machuca et al., 2005]
- SOAM [Vazquez et al., 2006] y la creación de objetos inteligentes que utilizan los servicios de la Web semántica
- Plataforma de servicios en un contexto híbrido [Ejigu et al., 2008]

## Ejercicio

Un diagrama ontológico que describe la información semántica de contexto de un usuario de un sistema ubicuo, necesaria para la correcta composición de servicios en la búsqueda de un restaurante apropiado para celebrar una reunión con sus amigos. El usuario está suscrito a un proveedor de red para móviles, el cual mantiene un portal con un canal de información+entretenimiento *Infoainment* para sus usuarios.

*Infoainment* ofrece un amplio espectro de recursos y servicios: un punto de información de interés, comercio electrónico y buscadores especializados. El usuario utiliza su *smart-phone* mientras pasea por la plaza principal de su ciudad y accede a *Infoainment*, que le proporciona el acceso a un servicio de recomendación de restaurantes. El mencionado portal utiliza una infraestructura de adaptación para personalizar el servicio adaptándolo al contexto del usuario que lo solicita. En este caso, la petición de servicio tiene como resultado la construcción de un servicio compuesto a partir de 2 servicios atómicos adaptados al contexto del usuario. El contexto se ve enriquecido directamente por los servicios de la suscripción al portal *Infoainment*.

# OWL

## Ideas fundamentales:

- Se propone como respuesta a la necesidad de contar con un nuevo lenguaje de modelado de ontologías de la Web Semántica La expresividad de RDF y de los esquemas de éste (“RDF Schema”) es muy limitada

## Deficiencias de RDF

- RDF está aproximadamente limitado a la expresión de predicados evaluables como *verdadero/falso*
- RDF Schema está limitado a la descripción de una jerarquía de subclases y de propiedades

# Utilidad de la semántica formal

## Idea fundamental:

Permitir a las personas poder razonar acerca del *conocimiento* adquirido/almacenado.

## Ejemplos de razonamiento basado en semántica formal:

- Pertenencia a una clase
- Equivalencia de clases
- Consistencia:  
detectar incosistencias:  $x \in A; A \subseteq B \cap C;$   
 $A \subseteq D \wedge B \cap D = \{0\}$
- Clasificación:  
Si una propiedad (*propiedad, valor*) es condición suficiente de pertenencia a clase, entonces si un elemento la satisface, ha de ser instancia de la clase.



## Utilidad de la semántica formal-II

### Idea fundamental:

- La semántica es un pre-requisito para el *soporte al razonamiento*

### Soporte al razonamiento:

- Comprobación de la consistencia de ontologías
- Detección de relaciones no deseadas entre clases
- Clasificación automática de las instancias de clases

Las denominadas “description logics” son subconjuntos de la Lógica de Predicados para las que es posible contar un soporte al razonamiento eficiente.



# Conjunto completo de requisitos de un lenguaje para ontologías

*Soporte para razonamiento eficiente y facilidad de expresión de propiedades complejas*

RDF Schema + Lógica completa:

Proporciona el requisito anterior pero el lenguaje es tan potente que se convierte en *indecidable*

Solución de W3C Web Ontology Working Group:

Definir tres lenguajes:

- OWL Full
- OWL DL
- OWL Lite



## OWL Full

### Características del lenguaje:

- Utiliza todas las primitivas de OWL y permite mezclarlas de forma arbitraria con RDF y RDF Schema
- Permite aplicar las primitivas del lenguaje unas a otras
- Se puede imponer una restricción a la *clase de todas las clases*, p.e., para limitar el número de clases de una ontología
- Es un lenguaje tan potente que resulta ser *indecidable*

## OWL DL

### OWL Description Logic

- Pone restricciones a la forma de utilizar los constructores de OWL Full y RDF
- No permite que los constructores de OWL se apliquen unos a otros
- El lenguaje resultante es compatible con una *lógica de descripción* bien definida
- Permite obtener herramientas de soporte al razonamiento eficientes
- Se pierde la compatibilidad con RDF

## OWL Lite

Subconjunto aún más restrictivo:

- Restringe aún más los constructores de OWL DL respecto de las clases: no permite *enumeradas*, ni *sentencias de disyunción*, ni *cardinalidad arbitraria*
- Fácil de aprender para sus usuarios
- Fácil de implementar en el desarrollo de herramientas
- Expresividad muy restrictiva

# Compatibilidad de los lenguajes

Ontologías desarrolladas:

OWL Lite ⊑ OWL DL ⊑ OWL Full

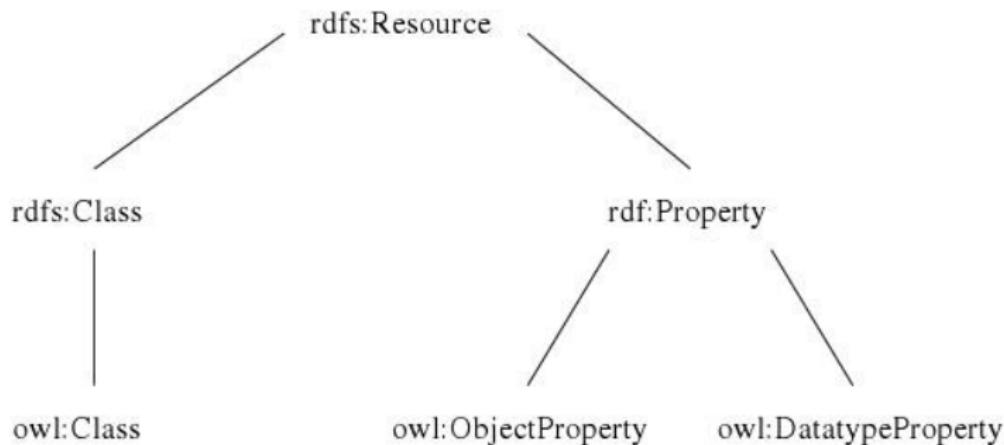
Conclusiones obtenidas:

Conclusiones con OWL Lite ⊑ OWL DL ⊑ OWL Full

Compatibilidad sintáctica:

- Todas las variedades de OWL utilizan sintaxis de RDF
- Instancias declaradas como en RDF, utilizando:  
descripciones RDF e información de tipos

## Relaciones entre subclases de OWL y RDF/RDFS



## Sintaxis de OWL

El elemento raíz de OWL es también un elemento `rdf:RDF`

```
1 <rdf:RDF xmlns:owl = "http://www.w3.org/2002/07/owl#"  
2   xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
3   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
4   xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">
```

Puede comenzar con una colección de asertos, útiles para organizar el documento:

```
1 <owl:Ontology rdf:about="">  
2   <rdfs:comment>An example OWL ontology </rdfs:comment>  
3   <owl:priorVersion rdf:resource="http://www.mydomain.org/uni-ns  
4     -old">  
5   </owl:priorVersion>  
6   <owl:imports rdf:resource="http://www.mydomain.org/persons">  
7   </owl:imports>  
8   <rdfs:label>University Ontology </rdfs:label>  
9 </owl:Ontology>
```

## Elementos Class

```
1 <owl:Class rdf:id="associateProfessor">
2   <rdfs:subClassOf rdf:resource="#academicStaffMember">
3   </rdfs:subClassOf>
4 </owl:Class>
```

Expresión de disyunción entre clases:

```
1 <owl:Class rdf:about="#associateProfessor">
2   <owl:disjointWith rdf:resource="#professor">
3   </owl:disjointWith>
4   <owl:disjointWith rdf:resource="#assistantProfessor">
5   </owl:disjointWith>
6 </owl:Class>
```

## Elementos Class-II

Equivalencia de clases:

```
1 <owl:Class rdf:id="faculty">
2   <owl:equivalentClass rdf:resource="#academicStaffMember">
3     </owl:equivalentClass>
4 </owl:Class>
```

Toda clase OWL es subclase de `owl:Thing` y *superclase* de `owl:Nothing`.

# Elementos Property

## Clases de propiedades en OWL:

- De objeto (ejemplos: `isTaughtBy`, `supervises`)
- De tipos de datos: relacionan objetos con tipos de datos (ejemplos: `telefono`, `titulo`, `edad`)

### Propiedad de tipos de datos:

```
1 <owl:DatatypeProperty rdf:id="age">
2   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
3   </rdfs:range>
4 </owl:DatatypeProperty>
```

### Propiedad de objetos:

```
1 <owl:ObjectProperty rdf:id="isTaughtBy">
2   <rdfs:domain rdf:resource="#course"></rdfs:domain>
3   <rdfs:range rdf:resource="#academicStaffMember"></rdfs:range>
4   <rdfs:subPropertyOf rdf:resource="#involves">
5   </rdfs:subPropertyOf>
6 </owl:ObjectProperty>
```



## Elementos Property-II

Para relacionar propiedades inversas:

```
1 <owl:ObjectProperty rdf:id="teaches">
2   <rdfs:range rdf:resource="#course">
3   </rdfs:range>
4   <rdfs:domain rdf:resource="#academicStaffMember">
5   </rdfs:domain>
6   <owl:inverseOf rdf:resource="#isTaughtBy">
7   </owl:inverseOf>
8 </owl:ObjectProperty>
```

Equivalencia de propiedades:

```
1 <owl:ObjectProperty rdf:id="lecturesIn">
2   <owl:equivalentProperty rdf:resource="#teaches"/>
3 </owl:ObjectProperty>
```

## Restricciones sobre las propiedades

Los valores de `onProperty` han de tomarse de la dada por `allValuesFrom`

```
1 <owl:Class rdf:about="#firstYearCourse">
2   <rdfs:subClassOf>
3     <owl:Restriction>
4       <owl:onProperty rdf:resource="#isTaughtBy"/>
5       <owl:allValuesFrom rdf:resource="#Professor"/>
6     </owl:Restriction>
7   </rdfs:subClassOf>
8 </owl:Class>
```

Utilizando la cuantificación existencial:

```
1 <owl:Class rdf:about="#academicStaffMember">
2   <rdfs:subClassOf>
3     <owl:Restriction>
4       <owl:onProperty rdf:resource="#teaches"/>
5       <owl:someValuesFrom rdf:resource="#undergraduateCourse">
6     </owl:Restriction>
7   </rdfs:subClassOf>
8 </owl:Class>
```



## Restricciones sobre las propiedades-II

Restricción sobre los valores de una propiedad: `hasValue`:

```
1 <owl:Class rdf:about="#mathCourse">
2   <rdfs:subClassOf>
3     <owl:Restriction>
4       <owl:onProperty rdf:resource="#isTaughtBy"/>
5       <owl:hasValue rdf:resource="#949318"/>
6     </owl:Restriction>
7   </rdfs:subClassOf>
8 </owl:Class>
```

## Restricciones sobre las propiedades-III

Restricciones que afectan a la cardinalidad:

```
1 <owl:Class rdf:about="#department">
2   <rdfs:subClassOf>
3     <owl:Restriction>
4       <owl:onProperty rdf:resource="#hasMember"/>
5         <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger"> 10
6         </owl:minCardinality>
7     </owl:Restriction>
8   </rdfs:subClassOf>
9   <rdfs:subClassOf>
10    <owl:Restriction>
11      <owl:onProperty rdf:resource="#hasMember"/>
12        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger"> 30
13        </owl:maxCardinality>
14    </owl:Restriction>
15  </rdfs:subClassOf>
16 </owl:Class>
```



# Expresión de propiedades especiales

## Definidas directamente

- owl:TransitiveProperty
- owl:SymmetricProperty
- owl:FunctionalProperty: **esta propiedad sólo puede tener un valor**
- owl:InverseFunctionalProperty: **define una inyección (2 elementos distintos poseen  $\neq$  valor)**

```
1 <owl:ObjectProperty rdf:id="hasSameGradeAs">
2   <rdf:type rdf:resource="&owl;TransitiveProperty" />
3   <rdf:type rdf:resource="&owl;SymmetricProperty" />
4   <rdfs:domain rdf:resource="#student" >
5   <rdfs:range rdf:resource="#student" >
6 </owl:ObjectProperty>
```

## Combinaciones booleanas de clases

Se pueden definir operaciones sobre las clases: unión, intersección, complementación  
*curso* es instancia del complementario de *staff member*:

```
1 <owl:Class rdf:about="#course">
2   <rdfs:subClassOf>
3     <owl:Class>
4       <owl:complementOf rdf:resource="#staffMember"/>
5     </owl:Class>
6   </rdfs:subClassOf>
7 </owl:Class>
```

O como una unión de clases:

```
1 <owl:Class rdf:ID="peopleAtUni">
2   <owl:unionOf rdf:parseType="Collection">
3     <owl:Class rdf:about="#staffMember"/>
4     <owl:Class rdf:about="#student"/>
5   </owl:unionOf>
6 </owl:Class>
```

## Combinaciones booleanas de clases-II

Las combinaciones booleanas se pueden anidar arbitrariamente:

```
1 <owl:Class rdf:id="adminStaff">
2   <owl:intersectionOf rdf:parseType="Collection">
3     <owl:Class rdf:about="#staffMember"/>
4     <owl:Class>
5       <owl:complementOf>
6         <owl:Class>
7           <owl:unionOf rdf:parseType="Collection">
8             <owl:Class rdf:about="#faculty"/>
9             <owl:Class rdf:about="#techSupportStaff"/>
10            </owl:unionOf>
11          </owl:Class>
12        </owl:complementOf>
13      </owl:Class>
14    </owl:intersectionOf>
15  </owl:Class>
```



## Enumeraciones

una enumeración es un elemento: `owl:oneOf`, usado para definir una clase listando todos sus elementos:

```
1 <owl:Class rdf:id="weekdays">
2   <owl:oneOf rdf:parseType="Collection">
3     <owl:Thing rdf:about="#Monday"/>
4     <owl:Thing rdf:about="#Tuesday"/>
5     <owl:Thing rdf:about="#Wednesday"/>
6     <owl:Thing rdf:about="#Thursday"/>
7     <owl:Thing rdf:about="#Friday"/>
8     <owl:Thing rdf:about="#Saturday"/>
9     <owl:Thing rdf:about="#Sunday"/>
10    </owl:oneOf>
11 </owl:Class>
```

# Instancias

## Ideas fundamentales:

- Se declaran como en RDF o de forma abreviada:  
`<academicStaffMember rdf:ID="949352"/>`
- OWL no asumen la convención de unicidad de nombres de los sistemas de bases de datos: 2 instancias con distintos ID no tienen por qué ser individuos diferentes
- OWL posee operadores para indicar explícitamente desigualdades entre individuos: `owl:DifferentFrom` y `owl:AllDifferent`

## Ejemplos con declaración de instancias

Cada curso es enseñando, como máximo, por 1 profesor:

```
1 <owl:ObjectProperty rdf:id="isTaughtBy">  
2   <rdf:type rdf:resource="#owl:FunctionalProperty"/>  
3 </owl:ObjectProperty>
```

no produce un error del razonador de OWL:

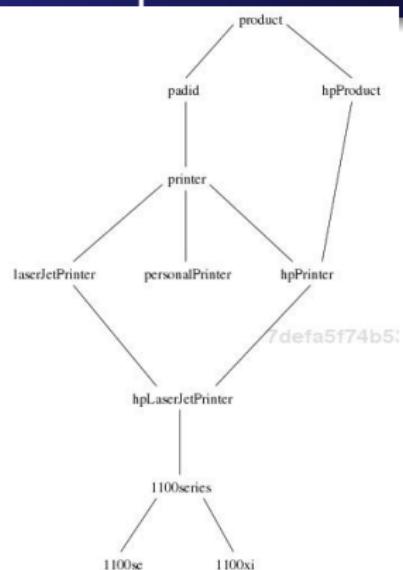
```
1 <course rdf:id="CIT1111">  
2   <isTaughtBy rdf:resource="#949318"/>  
3   <isTaughtBy rdf:resource="#949352"/>  
4 </course>
```

distintos IDs de profesores representan diferentes profesores:

```
1 <owl:AllDifferent>  
2   <owl:distinctMembers rdf:parseType="Collection">  
3     <lecturer rdf:about="#949318"/></lecturer>  
4     <lecturer rdf:about="#949352"/></lecturer>  
5     <lecturer rdf:about="#949111"/></lecturer>  
6   </owl:distinctMembers>  
7 </owl:AllDifferent>
```



## Ejemplo:ontología de impresoras



Ejemplo desarrollado en: "Cooperative Information Systems : Semantic Web Primer".  
*Antoniou, Grigoris, Van Harmelen, Frank, MIT Press(2008)*

## Ejemplo:ontología de impresoras-2

### Cabecera RDF/OWL:

```
1 <!DOCTYPE owl
2   [ <!ENTITY xsd ]> "http://www.w3.org/2001/XMLSchema#" >
3 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
6   xmlns:owl = "http://www.w3.org/2002/07/owl#"
7   xmlns="http://www.cs.vu.nl/~frankh/spool/printer.owl#">
```

### Comienzo documento OWL:

```
1 <owl:Ontology rdf:about="">
2   <owl:versionInfo> Mi ejemplo version 1.3, 17 Enero 2017
3   </owl:versionInfo>
4 </owl:Ontology>
```

## Ejemplo:ontología de impresoras-3

```
1 <owl:Class rdf:id="product">
2   <rdfs:comment>Products form a class.</rdfs:comment>
3 </owl:Class>
4 <owl:Class rdf:id="padid">
5   <rdfs:comment> Printing and digital imaging devices are a
6     subclass of product.
7   </rdfs:comment>
8   <rdfs:label>Device</rdfs:label>
9   <rdfs:subClassOf rdf:resource="#product">
10  </rdfs:subClassOf>
11 </owl:Class>
```

## Ejemplo:ontología de impresoras-4

```
1 <owl:Class rdf:id="hpProduct">
2   <rdfs:comment>
3     HP products are exactly those products manufactured by Hewlett
4       Packard.
5   </rdfs:comment>
6   <owl:intersectionOf rdf:parseType="Collection">
7     <owl:Class rdf:about="#product"></owl:Class>
8     <owl:Restriction>
9       <owl:onProperty rdf:resource="#manufactured_by">
10      </owl:onProperty>
11      <owl:hasValue rdf:datatype="&xsd:string">
12        Hewlett Packard
13      </owl:hasValue>
14    </owl:Restriction>
15  </owl:intersectionOf>
</owl:Class>
```

## Ejemplo:ontología de impresoras-5

```
1 <owl:Class rdf:ID="printer">
2   <rdfs:comment>
3     Printers are printing and digital imaging devices.
4   </rdfs:comment>
5   <rdfs:subClassOf rdf:resource="#padid"/>
6 </owl:Class>
```

```
1 <owl:Class rdf:ID="personalPrinter">
2   <rdfs:comment>
3     Printers for personal use form a subclass of printers.
4   </rdfs:comment>
5   <rdfs:subClassOf rdf:resource="#printer"/>
6 </owl:Class>
```

```
1 <owl:Class rdf:ID="hpPrinter">
2   <rdfs:comment>
3     HP printers are HP Products and printers.
4   </rdfs:comment>
5   <rdfs:subClassOf rdf:resource="#printer"/>
6   <rdfs:subClassOf rdf:resource="#hpProduct"/>
7 </owl:Class>
```



## Ejemplo:ontología de impresoras-6

```
1 <owl:Class rdf:id="laserJetPrinter">
2   <rdfs:comment>
3     Laser jet printers are exactly those printers that use laser
4       jet printing technology .
5   </rdfs:comment>
6   <owl:intersectionOf rdf:parseType="Collection">
7     <owl:Class rdf:about="#printer">
8       <owl:Restriction>
9         <owl:onProperty rdf:resource="#printingTechnology">
10        </owl:onProperty>
11        <owl:hasValue rdf:datatype="&xsd:string">
12          laser jet
13        </owl:hasValue>
14      </owl:Restriction>
15    </owl:intersectionOf>
</owl:Class>
```

## Ejemplo:ontología de impresoras-7

```
1 <owl:Class rdf:id="hpLaserJetPrinter">
2   <rdfs:comment>
3     HP laser jet printers are HP products and laser jet printers .
4   </rdfs:comment>
5   <rdfs:subClassOf rdf:resource="#laserJetPrinter"/>
6   <rdfs:subClassOf rdf:resource="#hpPrinter"/>
7 </owl:Class>
```

## Ejemplo:ontología de impresoras-8

```
1 <owl:Class rdf:id="1100series">
2   <rdfs:comment>
3     1100series printers are 8ppm printing speed and HP laser jet
4       printers with 600dpi printing resolution.
5   </rdfs:comment>
6   <rdfs:subClassOf rdf:resource="#hpLaserJetPrinter"/>
7   <rdfs:subClassOf>
8     <owl:Restriction>
9       <owl:onProperty rdf:resource="#printingSpeed">
10      </owl:onProperty>
11      <owl:hasValue rdf:datatype="&xsd:string"> 8ppm
12      </owl:hasValue>
13    </owl:Restriction>
14  </rdfs:subClassOf>
15  <rdfs:subClassOf>
16    <owl:Restriction>
17      <owl:onProperty rdf:resource="#printingResolution">
18      </owl:onProperty>
19      <owl:hasValue rdf:datatype="&xsd:string"> 600dpi
        </owl:hasValue>
```



## Ejemplo:ontología de impresoras-9

```
1 <owl:Class rdf:id="1100se">
2   <rdfs:comment>
3     1100se printers belong to the 1100 series and cost 450 USD.
4   </rdfs:comment>
5   <rdfs:subClassOf rdf:resource="#1100series">
6   </rdfs:subClassOf>
7   <rdfs:subClassOf>
8     <owl:Restriction>
9       <owl:onProperty rdf:resource="#price">
10      </owl:onProperty>
11      <owl:hasValue rdf:datatype="&xsd;integer"> 450
12      </owl:hasValue>
13    </owl:Restriction>
14  </rdfs:subClassOf>
15</owl:Class>
```

## Ejemplo:ontología de impresoras-10

```
1 <owl:Class rdf:id="1100xi">
2   <rdfs:comment>
3     1100xi printers belong to the 1100 series and cost 350 USD.
4   </rdfs:comment>
5   <rdfs:subClassOf rdf:resource="#1100series">
6   </rdfs:subClassOf>
7   <rdfs:subClassOf>
8     <owl:Restriction>
9       <owl:onProperty rdf:resource="#price">
10      </owl:onProperty>
11      <owl:hasValue rdf:datatype="&xsd;integer"> 350
12      </owl:hasValue>
13    </owl:Restriction>
14  </rdfs:subClassOf>
15</owl:Class>
```

## Ejemplo:ontología de impresoras-11

```
1 <owl:DatatypeProperty rdf:id="manufactured_by">
2   <rdfs:domain rdf:resource="#product"></rdfs:domain>
3   <rdfs:range rdf:resource="&xsd:string"></rdfs:range>
4 </owl:DatatypeProperty>
5 <owl:DatatypeProperty rdf:id="price">
6   <rdfs:domain rdf:resource="#product"></rdfs:domain>
7   <rdfs:range rdf:resource="&xsd:nonNegativeInteger"></rdfs:range>
8 </owl:DatatypeProperty>
```

## Ejemplo:ontología de impresoras-12

```
1 <owl:DatatypeProperty rdf:id="printingTechnology">
2   <rdfs:domain rdf:resource="#printer"></rdfs:domain>
3   <rdfs:range rdf:resource="&xsd:string"></rdfs:range>
4 </owl:DatatypeProperty>
5 <owl:DatatypeProperty rdf:id="printingResolution">
6   <rdfs:domain rdf:resource="#printer"></rdfs:domain>
7   <rdfs:range rdf:resource="&xsd:string"></rdfs:range>
8 </owl:DatatypeProperty>
9 <owl:DatatypeProperty rdf:id="printingSpeed">
10  <rdfs:domain rdf:resource="#printer"></rdfs:domain>
11  <rdfs:range rdf:resource="&xsd:string"></rdfs:range>
12 </owl:DatatypeProperty>
13 </rdf:RDF>
```

# Bibliografía Fundamental

-  Apache©(2014).  
Jini technology.  
<http://river.apache.org>.  
[Online; accessed July-2014].
-  Atzori, L., Iera, A., and Morabito, A. (2010).  
The internet of things: A survey.  
*Computer Networks*, 54(15):2787–2805.
-  Baldauf, M., Dustdar, S., and Rosenberg, F. (2012).  
A survey of context aware systems.  
*International Journal of Adhoc and Ubiquitous Computing*, 2(4):263–267.
-  Banavar, G. and etal. (2000).  
Challenges: An application model for pervasive computing.  
In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 266–274.
-  Barolli, L. and Xhafa, F. (2011).  
Jxta–overlay: A p2p platform for distributed, collaborative, and ubiquitous computing.