

# Semantic Web

# Outline

---

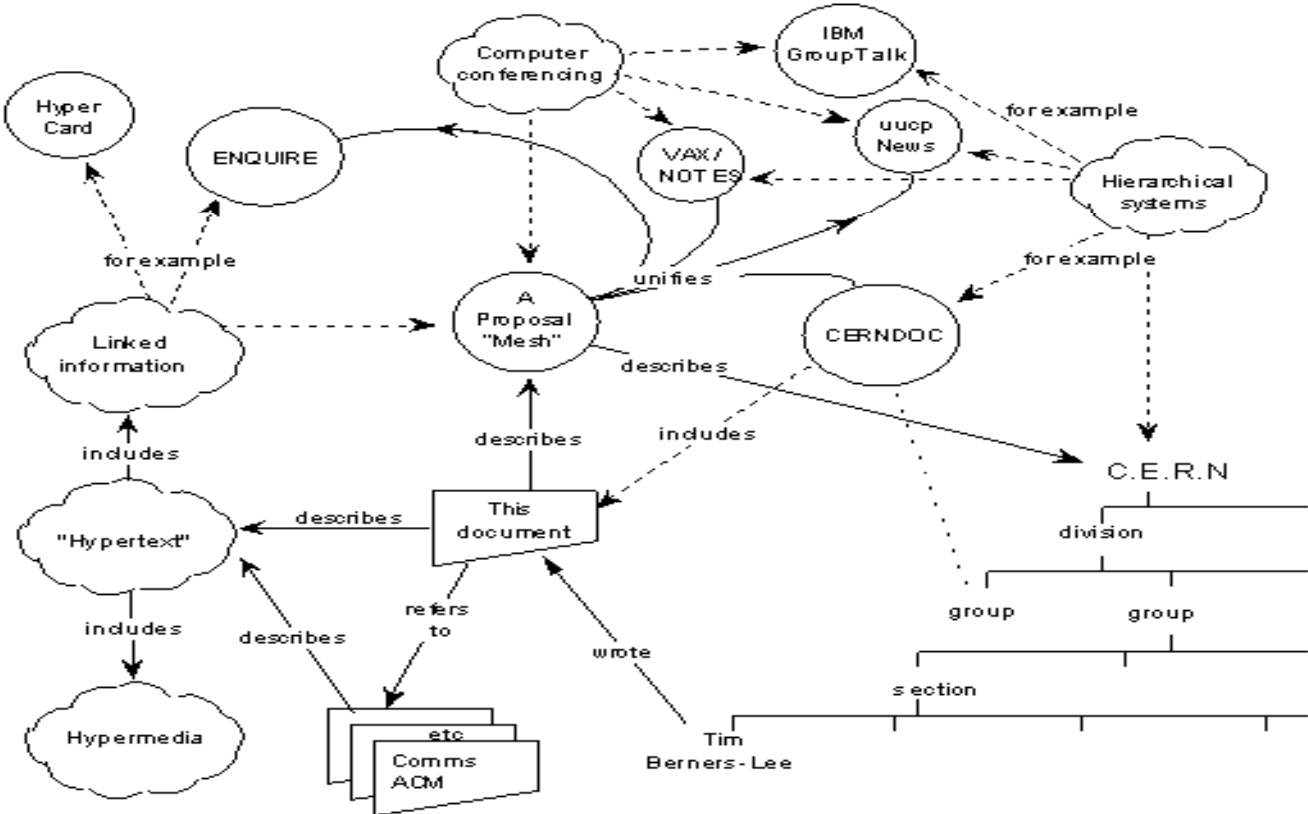
- 1) Background
  - a) challenges
  - b) generic requirements
  - c) solution
- 2) Ontology
  - a) concepts
  - b) methodology
  - c) technologies
- 3) **Semantic Web**
  - a) concepts and principles
  - b) languages and tools
  - c) major projects
- 4) e-Government Case Studies
  - a) semantic search
  - b) interoperability
- 5) Demonstration
- 6) Conclusions

“The bane of my existence is doing what I know the  
computer can do for me”

Dan Connolly, Semantic Web History: Nodes and Arcs  
1989-1999, 2002

# From Syntactic to Semantic Web

Semantic Web is all about realizing the original vision of WWW:



# Semantic Web Definition

---

## Definition [Semantic Web]

An extension of the Web in which information is given well-defined meaning, enabling computers/people to work in better cooperation.

Semantic Web involves:

- 1) Annotating documents with semantic markup, not interpreted for display but for an expression of the document intent.
- 2) Semantically connecting various resources like documents, images and people together.
- 3) Equipping the web with more meaning by explicitly expressing a whole new set of relationships among resources.

# Semantic Web Principles

---

Six basic principles underpinning the Semantic Web:

- 1) **identity** - everything is identifiable on the SW through a URI
- 2) **typing** - resources and links can have types
- 3) **partiality** - partial information is tolerated
- 4) **relativity** - there is no need for absolute truth
- 5) **evolution** - evolution is supported
- 6) **brevity** - design is minimal

# Principle 1: Identity

---

Everything is identifiable on the Semantic Web through a Uniform Resource Identifier (URI):

- anyone who has control over a part of the Web namespace can create a URI
- URIs can be de-referenced
- referenced object could be further described through its URI

# Example: Identity

---



*Resources from the physical world and their useful identifications in the semantic web*

*Courtesy: Semantic Web Activity*

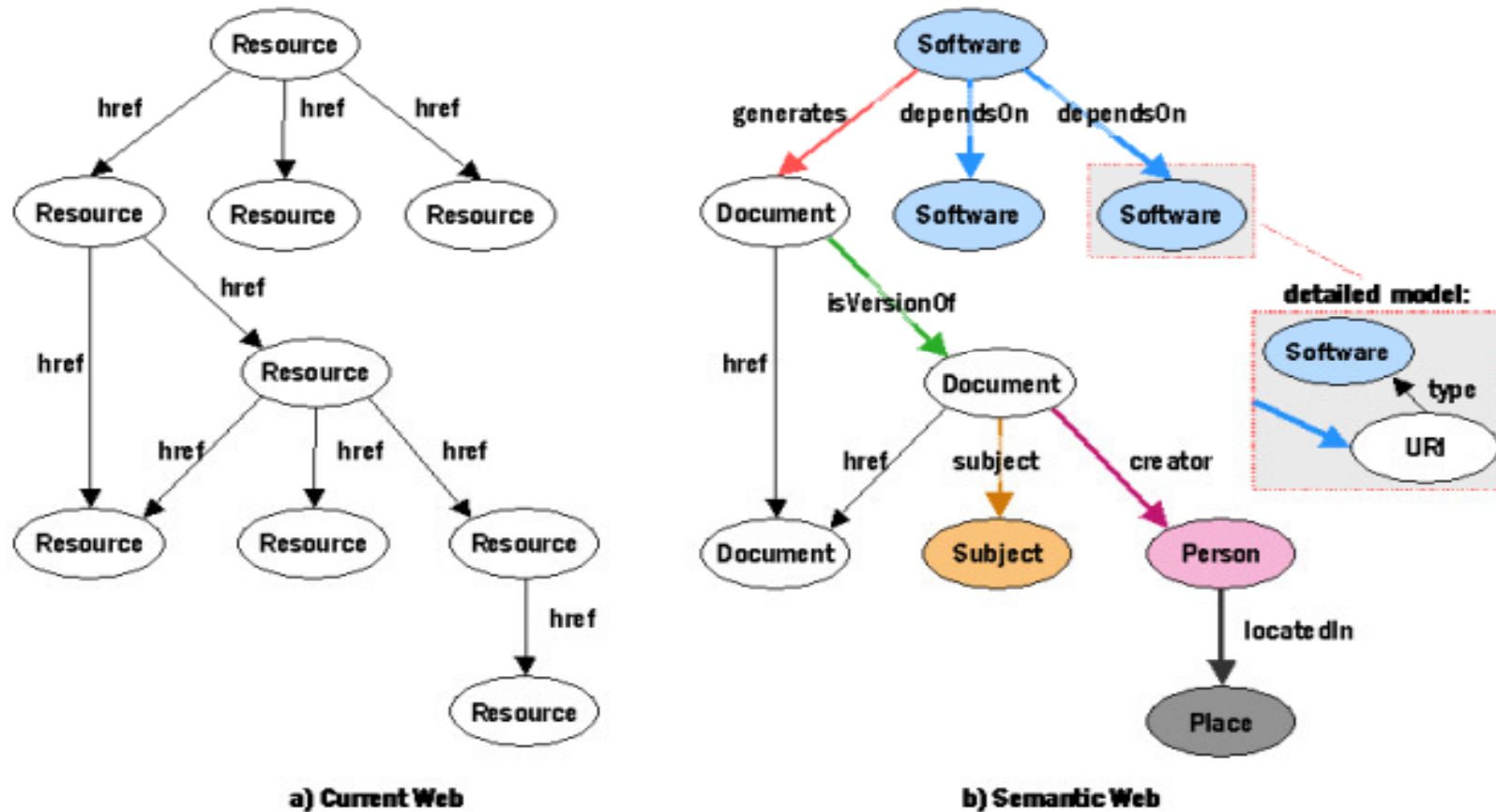
# Principle 2: Typing

---

Resources and links can have types:

- 1) current web consists of resources and links with no metadata
- 2) unlike humans, machines cannot guess the contents of documents and nature of links without being told
- 3) semantic web allows links to be described and typed

# Example: Typing



*Resources and Links can have types in the Semantic Web*

*Courtesy: Semantic Web Activity*

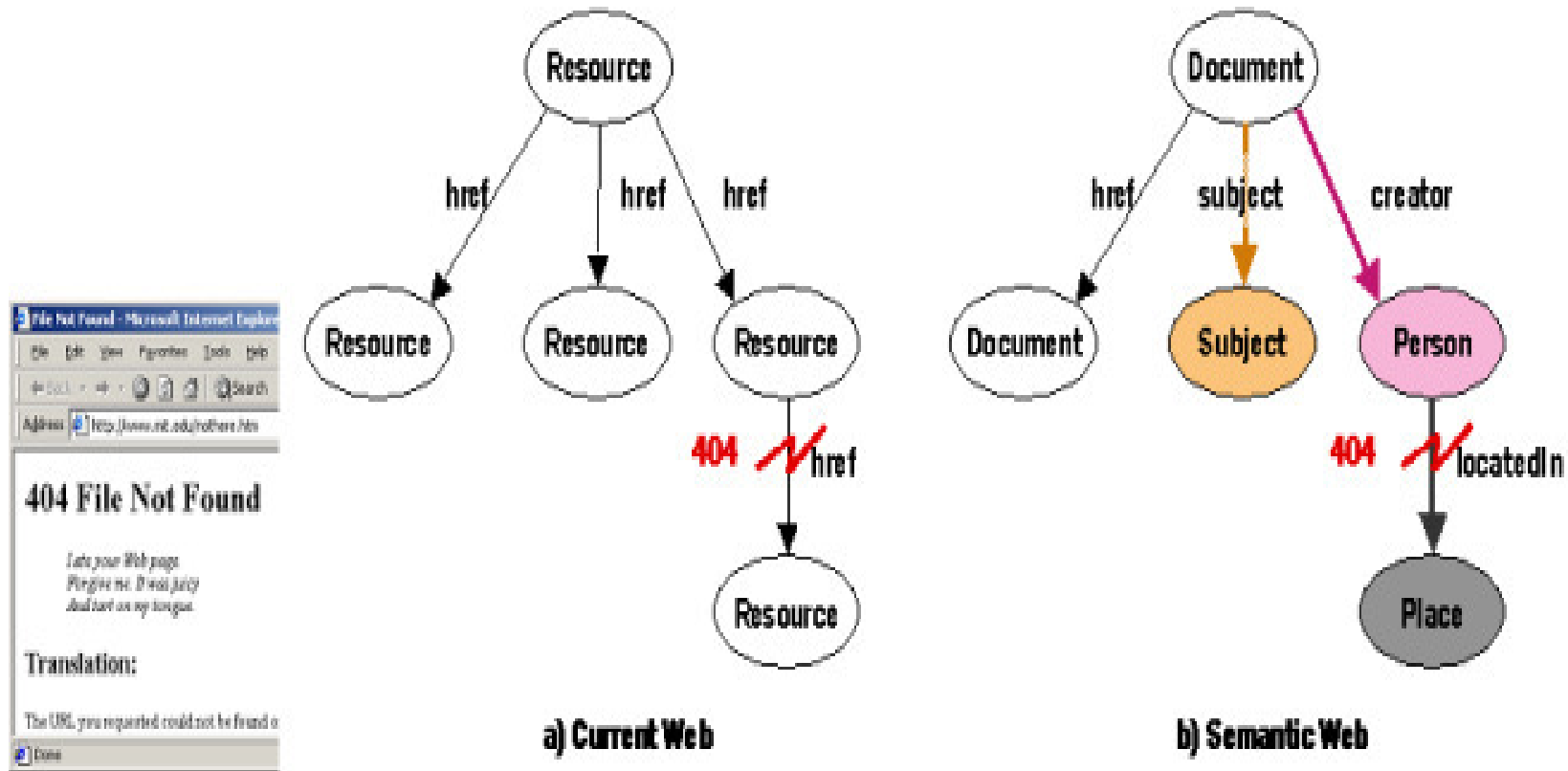
# Principle 3: Partiality

---

Partial information is tolerated:

- 1) current web permits links with non-existing targets
- 2) semantic web is also partial, allowing anyone to say anything about web resources by creating different types of links
- 3) semantic web tools tolerate this partiality

# Example: Partiality



*Current web and Semantic web handle partial information*

*Courtesy: Semantic Web Activity*

# Principle 4: Relativity

---

There is no absolute truth:

- 1) semantic web does not guarantee absolute truth, only the truth derived from the existing statements in a given context
- 2) applications that process information decide what can be trusted based on the context:

“who said what, when, how...”

# Example: Relativity

URI variable



- 1) If X is AC rep of Y, X can delegate W3C member access rights in Y.
- 2) *Kari* is AC rep of *Elisa* .



- 1) If X is employee of *Elisa*, X has W3C member access rights.
- 2) *Tiina* is employee of *Elisa*.



**Tiina:** I have W3C member access rights  
**Proof:** Alan 1, Alan 2, Kari 1, Kari 2



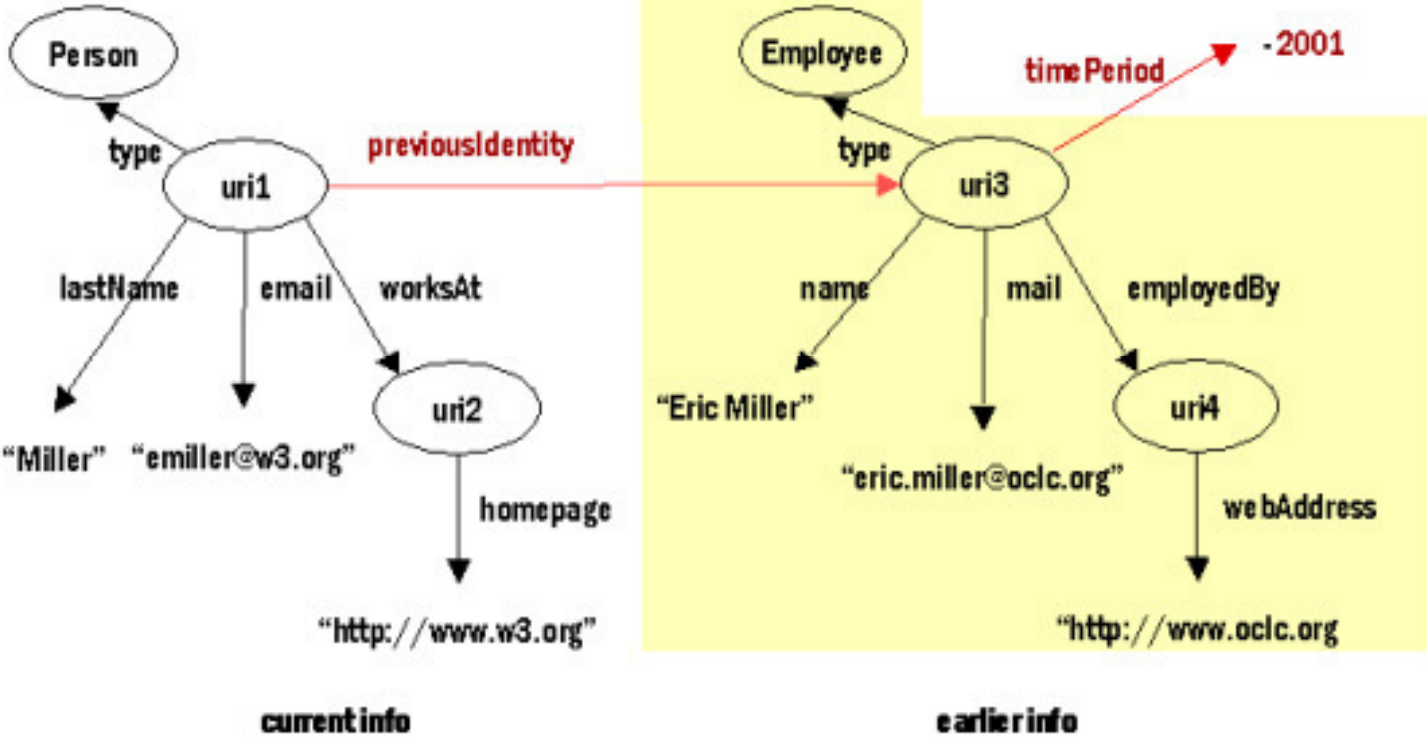
# Principle 5: Evolution

---

Evolution is supported:

- 1) combination of the independent work of diverse communities
- 2) support the ability to add new information without insisting that the old information be up-to-date
- 3) ability to resolve ambiguities and clarify inconsistencies
- 4) conventions that expand as human understanding expands

# Example: Evolution



*Combining new information with old when the old information cannot be changed.*

*Courtesy: Semantic Web Activity*

# Principle 6: Brevity

---

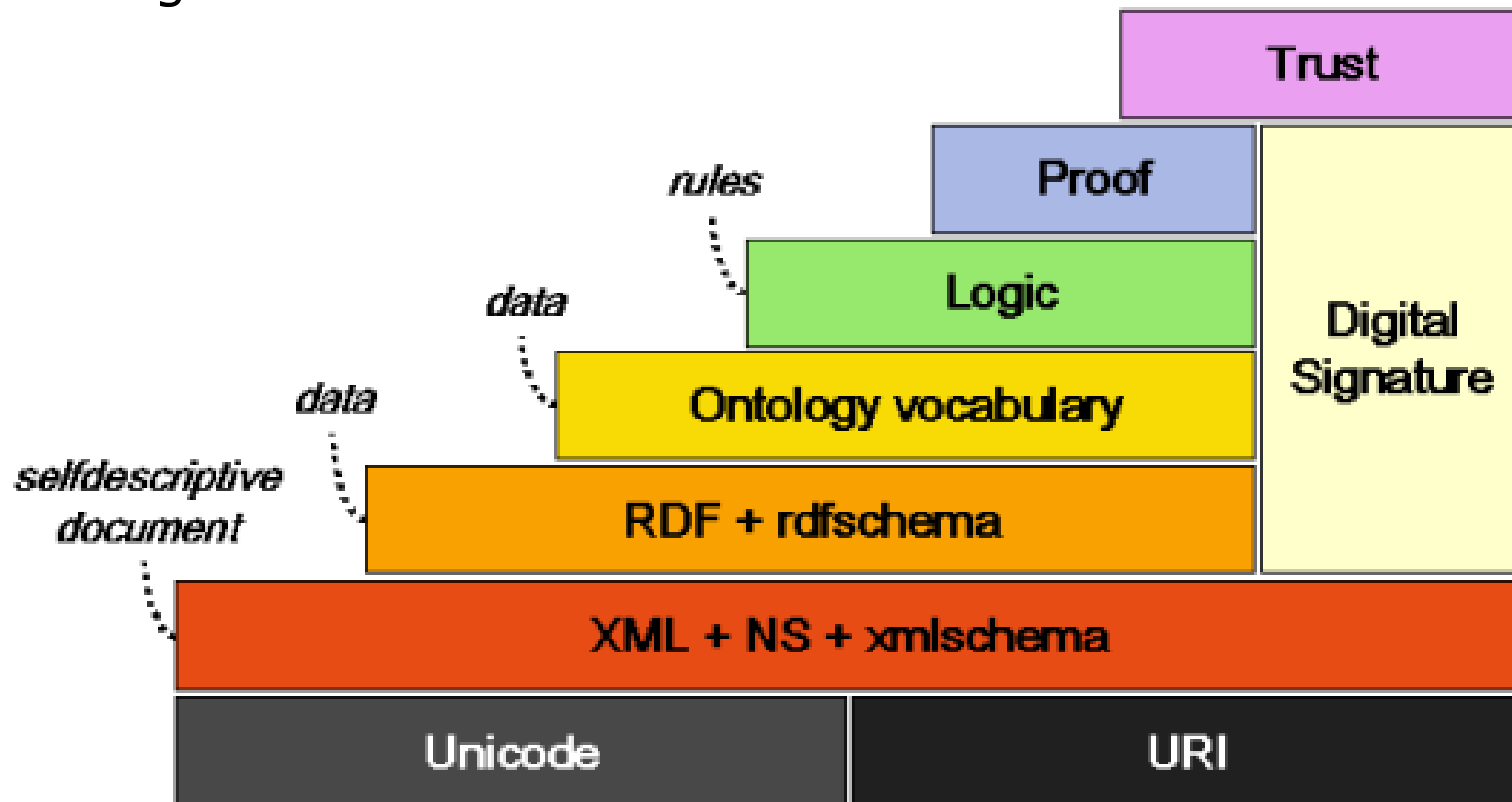
Minimalist design:

- 1) Semantic Web enables simple applications based on already existing standards ([RSS](#), [Dublin Core](#) and [MusicBrainz](#))
- 2) Semantic web will not standardize more than is necessary

# Semantic Web Layers

---

The principles are implemented in the layers of the web technologies and standards:



*Semantic Web Layers*

*Courtesy: Semantic Web Activity*

# Layer 1: Unicode + URI

The layer ensures that:

- 1) International character set is used.
- 2) Objects in the Semantic Web are identifiable.

# Example: Unicode/URI Layer

- Unicode character references:

“Prot&#225;g&#225;” expands to “Protégé”

- URI:

<http://www.e-macao-seminar13.com/wine.owl#Wine>

Identifies the Wine concept in the ontology.

# Layer 2: XML + Namespaces

XML layer with namespace and schema definition allows:

- 1) creation and use of metadata vocabulary and syntactic interoperability of these metadata vocabulary
- 2) the integration of semantic web definitions with other XML standards

# Example: XML

---

## Beaujolais class in XML:

```
<Class ID="Beaujolais">  
  
  <comment datatype="string">  
    A kind of wine  
  </comment>  
  
  <subClassOf>  
    <Class ID="Wine"/>  
  </subClassOf>  
  
</Class>
```

# Example: XML + Namespaces

Beaujolais class in XML with namespace and schema:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.e-macao-seminar13.com/wine.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

  <owl:Class rdf:ID="Beaujolais">

    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      A kind of wine
    </rdfs:comment>

    <rdfs:subClassOf>
      <owl:Class rdf:ID="Wine"/>
    </rdfs:subClassOf>

  </owl:Class>
```

# Layer 3: RDF

---

The RDF layer allows for:

- 1) interoperability at the semantic level
- 2) statements to be made about objects with URIs
- 3) defining vocabularies that can be referred to by URIs
- 4) associating types with resources and links
- 5) classifying and describing objects through class hierarchies

# Example: RDF

---

## Beaujolais class in RDF:

```
<rdf:Description rdf:about="#Beaujolais">

  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    A kind of wine
  </rdfs:comment>

  <rdfs:subClassOf rdf:resource="#Wine"/>

  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>

</rdf:Description>
```

# Layer 4: Ontology

---

The ontology layer supports:

- 1) evolution of vocabularies
- 2) definition of relations between different concepts
- 3) specification of taxonomic and logical theories about concepts upon which inferences can be made
- 4) specification of meta-information such as transitivity, symmetry, uniqueness and cardinality
- 5) wider interoperability than the RDF layer

The language used for Ontology layer is OWL.

# Example: Ontology

---

Wine class is a subclass of PortableLiquid.

OWL definition:

```
<owl:Class rdf:about="#Wine">  
  
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">  
    A portable liquid made from fermented fruit, usuall grapes  
  </rdfs:comment>  
  
  <rdfs:subClassOf>  
    <owl:Class rdf:ID="PortableLiquid"/>  
  </rdfs:subClassOf>  
  
</owl:Class>
```

# Example: Ontology

---

Beaujolais as a sub-class of Wine:

```
<owl:Class rdf:ID="Beaujolais">  
  
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">  
    A kind of wine  
  </rdfs:comment>  
  
  <rdfs:subClassOf>  
    <owl:Class rdf:ID="Wine"/>  
  </rdfs:subClassOf>  
  
</owl:Class>
```

# Layer 5: Logic

---

The logic layer:

- 1) provides logic system for expressing rules
- 2) provides a universal language for monotonic logic  
(what is already valid should remain valid with new information)

Development of this layer is ongoing.

# Example: Ontology

---

In addition to being a PortableLiquid, Wine must have at least one WineDescriptor:

```
<owl:Class rdf:about="#Wine">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasWineDescriptor"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="www.w3.org/2001/XMLSchema#int">
        1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

# Layer 6: Proof

---

The proof layer:

- 1) allows the execution of rules facilitated by the logic layer
- 2) provides explanations or evidence for statements
- 3) allows formal proofs to be shared

Development of this layer is ongoing.

# Example: Proof

---

Reasoning in the wine domain:

- Given that:
  - 1) Wine is a subclass of PortableLiquid
  - 2) Beaujolais is a subclass of Wine
  - 3) ChateauMorgonBeajolais is a Beaujolais Wine
- We can deduce that:

ChateauMorgonBeajolais is a PortableLiquid

# Layer 7: Trust

---

The web of trust assumes that:

- 1) all statements on the web occur in some context
- 2) applications need this context in order to evaluate the trustworthiness of the statements

The trust layer provides a machinery to establish the trustworthiness of statements based on proofs and context.

Development of this layer is ongoing.

# Example: Trust

---

Do we trust that

“ChateauMorgonBeajolais is a PortableLiquid”?

Yes:

- We trust the Wine Ontology (it was created by ourselves)
- We can deduce this fact based on the Wine Ontology

What if the Wine Ontology was created by somebody else?

# Digital Signature

---

Digital signatures supports the notion of trust.

They can be used to:

- 1) authenticate the identity of source of the ontology
- 2) ensure that the original content of the ontology is unchanged

# Semantic Web Languages

---

There are two major families of languages:

- 1) Resource Description Framework – RDF
- 2) Web Ontology Language – OWL

# RDF

---

Resource Description Framework (RDF):

- 1) generally describes resources
- 2) an infrastructure for encoding, exchanging and reusing structured metadata

The most common syntax for RDF is RDF/XML.

The W3C RDF specifications: <http://www.w3.org/RDF/>

# RDF Model

---

The RDF model is a triple of:

- 1) **subject** – resource that the RDF statement describes, uniquely identified by a URI
- 2) **predicate** – property of the subject, uniquely identified by a URI
- 3) **object** – value for the property, which can be any valid RDF data type (all XML data types are valid in RDF)

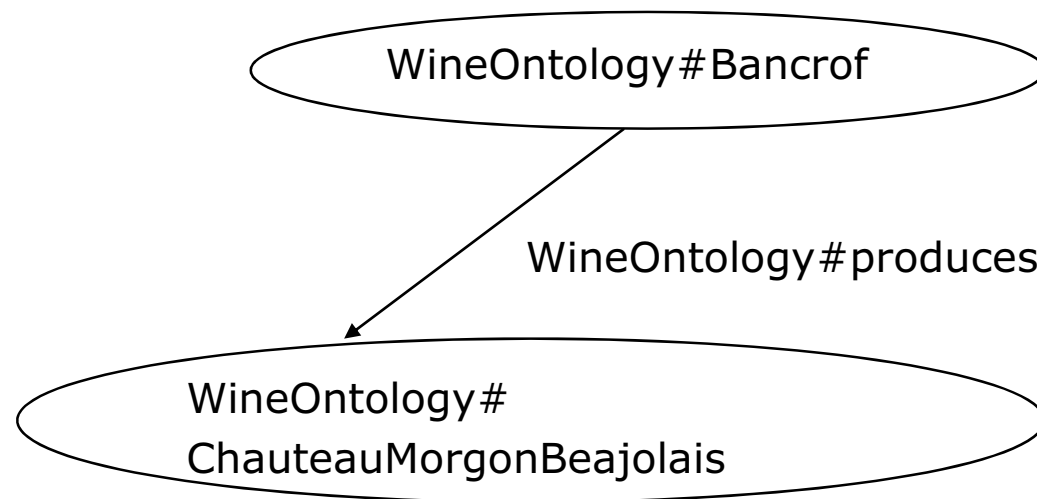
# RDF Model Example

---

RDF models states that:

Bankcrof produces ChateauMorgonBeajolais

- Subject is Bancrof
- Predicate is produces
- Object is ChateauMorgonBeajolais



# Notes on RDF Model

---

- 1) each RDF triple is a complete and unique fact
- 2) the subject is either URI or a blank node, the predicate is a URI, and the object is a URI, blank node or literal
- 3) an RDF triple can be joined to other RDF triples while retaining its original meaning

# RDF/XML

---

- 1) the most common serialization notation for the RDF
- 2) well-formed XML with additional constraints
- 3) can be parsed with common XML technology
- 4) requires well-formedness but not XML-style validation

# RDF/XML Example

---

## RDF/XML representation of the description of ChateauMorgonBeajolais wine

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://www.e-macao-seminar13.com/wine.rdf#"
  xml:base="http://www.e-macao-seminar13.com/wine.rdf" >

  <rdf:Description rdf:about="#ChateauMorgonBeajolais">
    <madeFromFruit rdf:resource="#Grapes"/>
    <hasWineDescriptor rdf:resource="#Red"/>
    <rdf:type rdf:resource="#Beaujolais"/>
  </rdf:Description>
  ...
</rdf:RDF>
```

# RDF Element

---

- 1) The RDF element is the root of the RDF document
- 2) It defines the XML document to be an RDF document and contains a reference to the `xmlns:rdf` namespace:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  ...
  Description of resource is specified here
  ...
</rdf:RDF>
```

# Basic Elements and Attributes

---

- 1) **Description** - element which describes a resource
- 2) **About** - an attribute which identifies the resource
- 3) **Property** - used to describes the resource within Description

```
<rdf:Description rdf:about="#ChateauMorgonBeajolais">  
  <madeFromFruit rdf:resource="#Grapes"/>  
  <hasWineDescriptor rdf:resource="#Red"/>  
  <rdf:type rdf:resource="#Beaujolais"/>  
</rdf:Description>
```

# RDF Schema

---

RDF defines metadata with no special meanings for vocabulary:

RDF Schema:

- 1) defines the vocabulary for RDF statements
- 2) provides a lightweight ontology
- 3) specifies valid properties in a given RDF description
- 4) restricts property-type values

# RDFS Example

---

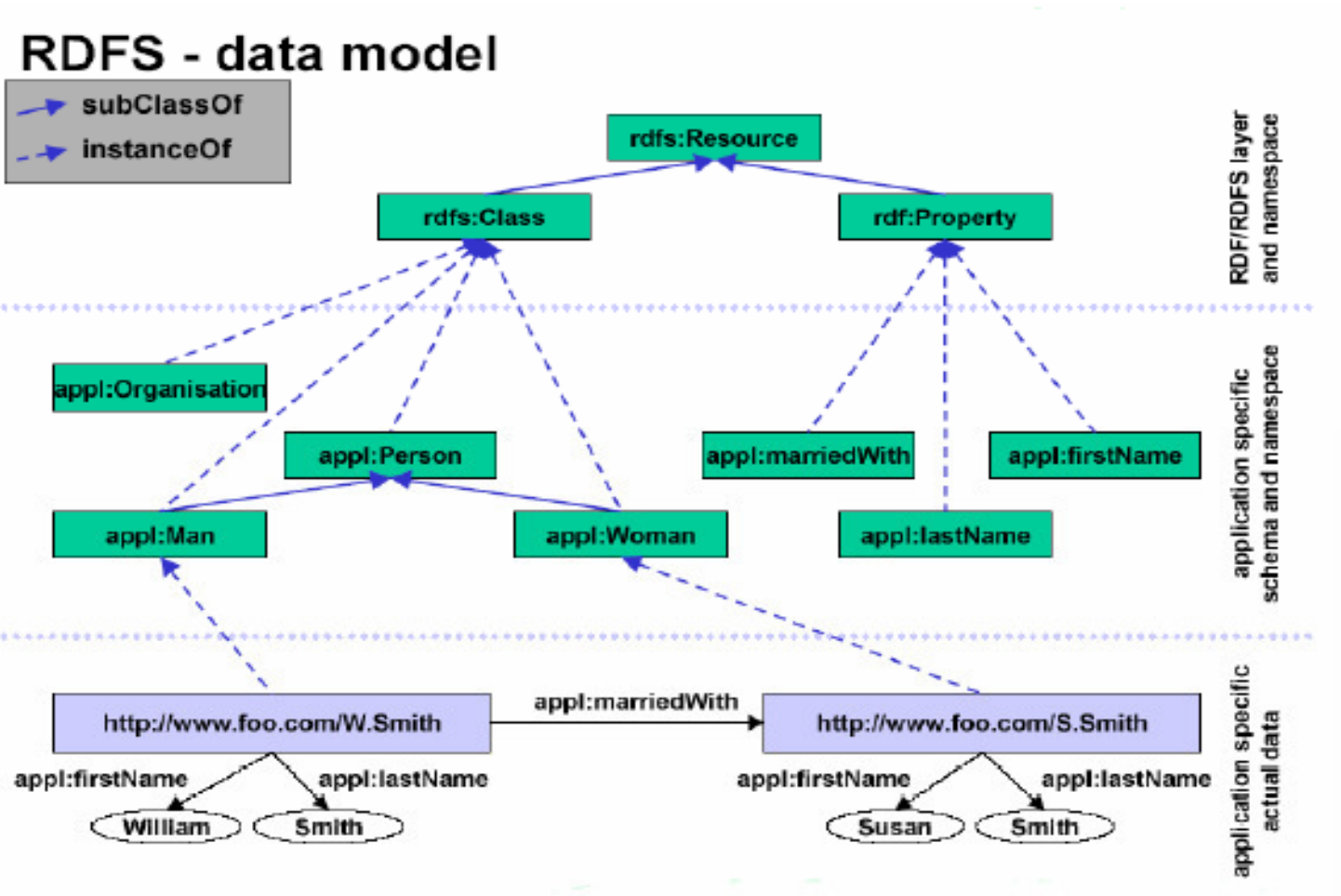
## RDF Schema terms

- 1) Class
- 2) Property
- 3) type
- 4) subClassOf
- 5) range
- 6) domain

## Using the terms:

- 1) `<Wine, type, Class>`
- 2) `<produces, type, Property>`
- 3) `<Wine, subClassOf, PortableLiquid>`
- 4) `<Bancrof, type, Winery>`
- 5) `<produces, range, Wine>`
- 6) `<produces, domain, Winery>`

# RDF-S Model Example



Courtesy: *Ontology languages for the web*  
Dr. Aida Boukottaya

# OWL Overview

---

Features of OWL:

- 1) extend existing web standards (XML/XML-S, RDF/RDF-S)
- 2) easy to understand and use
- 3) formally specified semantics
- 4) adequate expressive power
- 5) support for automated reasoning support

# OWL Background

---

- 1) Two languages preceded OWL as a Web Ontology Language
- 2) Ontology Inference Language (OIL) - developed within the OntoKnowledge project in Europe
- 3) DARPA Agent Markup Language (DAML-ONT) - developed within DARPA DAML Programme in the US
- 4) DAML-ONT and OIL led to DAML+OIL
- 5) DAML+OIL was submitted to W3C for standardisation
- 6) WebOnt working group developed OWL based on DAML+OIL

# OWL Example

---

## An OWL specification for the Wine Class

```
<owl:Class rdf:about="#Wine">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    A portable liquid made from fermented fruit
  </rdfs:comment>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasWineDescriptor"/>
      </owl:onProperty>
      <owl:minCardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PortableLiquid"/>
  </rdfs:subClassOf>
</owl:Class>
```

# OWL Sub-languages

---

OWL provides three sublanguages :

- 1) **OWL-Lite** - classification hierarchy and simple constraints
- 2) **OWL-DL** - maximum expressiveness with desirable computational properties for reasoning
- 3) **OWL-Full** - maximum expressiveness and syntactic freedom of RDF with no computational guarantees

# OWL Lite Basic Constructs

<b>Schema Constructs</b>	<b>Equality Constructs</b>	<b>Headers</b>
owl:Class rdf:Property rdfs:subClassOf rdfs:domain rdfs:range Individual	equivalentClass equivalentProperty sameIndividualAs differentFrom allDifferent	Imports priorVersion backwardCompatibleWith incompatibeWith
<b>Property Characteristics</b>	<b>Cardinality</b>	<b>Property type restrictions</b>
inverseOf TransitiveProperty FunctionalProperty InverseFunctionalProp erty SymetricProperty	minCardinality (0,1) maxcardinality (0,1) Cardinality (0,1)	allValuesFrom someValuesFrom

# OWL DL and Full Constructs

---

<b>Class Axioms</b>	<b>Boolean Combinations of Class Expressions</b>
<code>oneOf</code> <code>disjointWith</code> <code>equivalentClass</code> <code>rdfs:subClassOf</code>	<code>unionOf</code> <code>complementOf</code> <code>intersectionOf</code>
<b>Arbitrary Cardinality</b>	<b>Filler Information</b>
<code>minCardinality</code> <code>maxCardinality</code> <code>cardinality</code>	<code>hasValue</code>

# OWL Constructs Examples

---

<b>Class</b>	<pre>&lt;owl:Class rdf:ID="Winery"/&gt; &lt;owl:Class rdf:ID="Region"/&gt; &lt;owl:Class rdf:ID="ConsumableThing"/&gt;</pre>
<b>Individual</b>	<pre>&lt;Region rdf:ID="CentralCoastRegion"/&gt;</pre>
<b>Object Property</b>	<pre>&lt;owl:ObjectProperty rdf:ID="madeFromGrape"&gt;   &lt;rdfs:domain rdf:resource="#Wine"/&gt;   &lt;rdfs:range rdf:resource="#WineGrape"/&gt; &lt;/owl:ObjectProperty&gt;</pre>

# OWL Construct Examples

---

Sub-Classes	<pre> &lt;owl:Class rdf:ID="WineDescriptor" /&gt;  &lt;owl:Class rdf:ID="WineColor"&gt;   &lt;rdfs:subClassOf rdf:resource="#WineDescriptor"/&gt;   ... &lt;/owl:Class&gt; </pre>
Property Hierarchy	<pre> &lt;owl:ObjectProperty rdf:ID="hasWineDescriptor"&gt;   &lt;rdfs:domain rdf:resource="#Wine" /&gt;   &lt;rdfs:range rdf:resource="#WineDescriptor"/&gt; &lt;/owl:ObjectProperty&gt;  &lt;owl:ObjectProperty rdf:ID="hasColor"&gt;   &lt;rdfs:subPropertyOf rdf:resource="#hasWineDescriptor"/&gt;   &lt;rdfs:range rdf:resource="#WineColor" /&gt;   ... &lt;/owl:ObjectProperty&gt; </pre>

# Semantic Web Tools

---

## RDF tools:

- Jena Java RDF API and toolkit – manipulating RDF models
- Sesame – Java based storage and querying middleware
- IsaViz – Visual authoring tools for browsing and authoring

## OWL tools:

- Protégé – ontology development environment
- Racer – A Description Logic based reasoner
- SemanticWorks – ontology editor

## General tools:

Swoogle – semantic search engine

# Semantic Web in e-Government

Drivers for Semantic Web in e-government include:

- 1) content and services integration across platforms
- 2) capacity for advanced information management
- 3) cost benefits and usability
- 4) improved work processes through collaborative and dynamic environment

# Semantic Web Tools

---

## RDF tools:

- Jena Java RDF API and toolkit – manipulating RDF models
- Sesame – Java based storage and querying middleware
- IsaViz – Visual authoring tools for browsing and authoring

## OWL tools:

- Protégé – ontology development environment
- Racer – A Description Logic based reasoner
- SemanticWorks – ontology editor

## General tools:

Swoogle – semantic search engine

# Semantic Web Projects 1

---

The following are some Semantic Web projects related to e-Government and Information Society in Europe:

- 1) **OntoGov** - aims to develop, test and validate a semantically-enriched (ontology-enabled) platform that will facilitate the consistent composition, re-configuration and evolution of e-government services.
- 2) **Semantic Knowledge Technologies** - aims to make web information machine processable.

# Semantic Web Projects 2

---

- 3) **Data Information and Process Integration Project** - addresses Enterprise Application Integration, allowing disparate systems to share information.
- 4) **Knowledge Web** – strengthen European industry and service providers in the area of e-work and e-commerce based on Semantic Web

# Conclusions

---

- 1) Semantic Web aims to provide the necessary semantic infrastructure to realize the original promise of the web
- 2) Semantic enables machine processable contents on the web
- 3) The Semantic Web is underpinned by six basic principles and supported by an XML-based technology stack
- 4) RDF and OWL are currently the major semantic web languages for annotation and ontology development
- 5) semantic web addresses the problem of semantic interoperability in e-government