# Linguaggi

*Corso M-Z - Laurea in Ingegneria Informatica*
*A.A. 2007-2008*

Alessandro Longheu
*http://www.diit.unict.it/users/alongheu*
*alessandro.longheu@diit.unict.it*

*- lezione 27 -*

# Semantic Web: XML e RDF



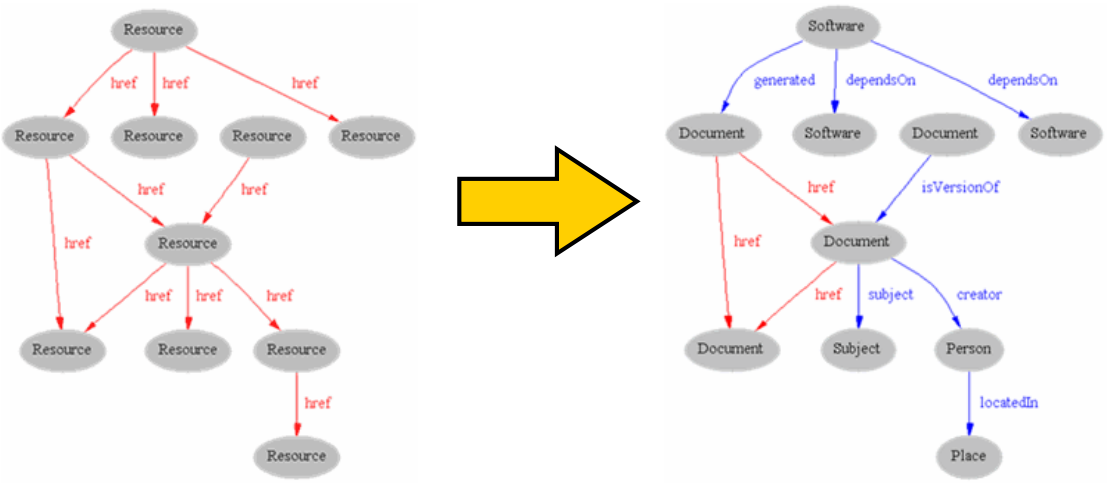


## Web Semantico

- Con il termine Web Semantico si intende la **trasformazione** del World Wide Web in un ambiente dove i documenti pubblicati (pagine HTML, file, immagini, e così via) siano associati ad informazioni e dati (metadati) che ne specifichino il contesto semantico in un formato adatto all'interrogazione, all'interpretazione e, più in generale, all'elaborazione automatica.
- Con l'interpretazione del contenuto dei documenti che il Web Semantico propugna, saranno possibili ricerche molto più evolute delle attuali, basate sulla presenza nel documento di parole chiave, ed altre operazioni specialistiche come la costruzione di reti di relazioni e connessioni tra documenti secondo logiche più elaborate del semplice link ipertestuale, permettendo un approccio simile a quello presente nei *sistemi esperti*

# Web Semantico

- Le pagine web sono oggi collegate sintatticamente mediante indici che localizzano la URL della pagina. Uno dei principali limiti di tale impostazione risiede nell'assenza di significato dei collegamenti: i collegamenti dovrebbero anche descriverci il luogo in cui saremmo condotti (link semantico).





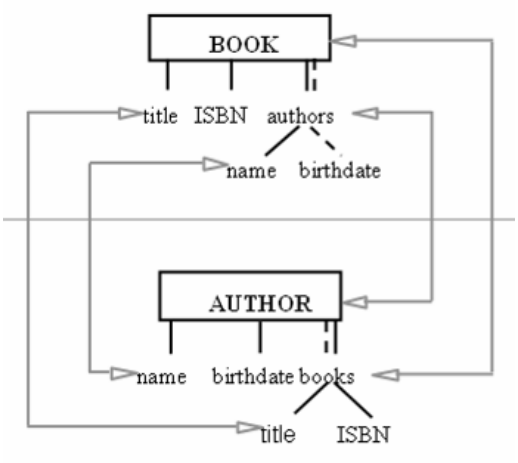



# Web Semantico

- Il WS **non implica una qualche forma di intelligenza** perché Invece di richiedere ai computer di comprendere il linguaggio umano e la sua logica, **si richiede all'uomo di fare uno sforzo in più in fase di progettazione web**.
- Questo sforzo renderebbe il web **machine-understandable** e non semplicemente machine-readable, e risolverebbe uno dei problemi cronici dei motori di ricerca, quello del vocabolario, ad esempio casi di sinonimia e polisemia che rendono praticamente impossibile per i motori di ricerca restituire esclusivamente i risultati attesi, questo a causa della notevole ricchezza ed ambiguità) del linguaggio naturale, ad esempio la parola albero riguarda informatica, botanica, nautica? e ancora, un documento che parla di finanziamento del governo alle società calcistiche in pericolo di fallimento in che ambito ricade? Sport, politica, finanza?

# Web Semantico

- Lo scenario futuro cerca di **riprodurre sul Web quello che già in parte esiste nel mondo dei database**: quando interroghiamo una base di dati, possiamo infatti fare ricerche piuttosto raffinate, ad esempio, chiedere "quali autori hanno scritto almeno due libri sull'IR"
- L'utente può formulare una richiesta che imponga precise relazioni ("almeno due libri sull'IR"), e tali relazioni sono stabilite fra concetti ("autore" e "libro") non fra parole chiave (non si ricerca la stringa "autore" o "libro"). Questo è possibile perché esiste uno **schema del DB**, cioè un modello ed un insieme di regole che stabiliscono coe debbano essere organizzati i dati

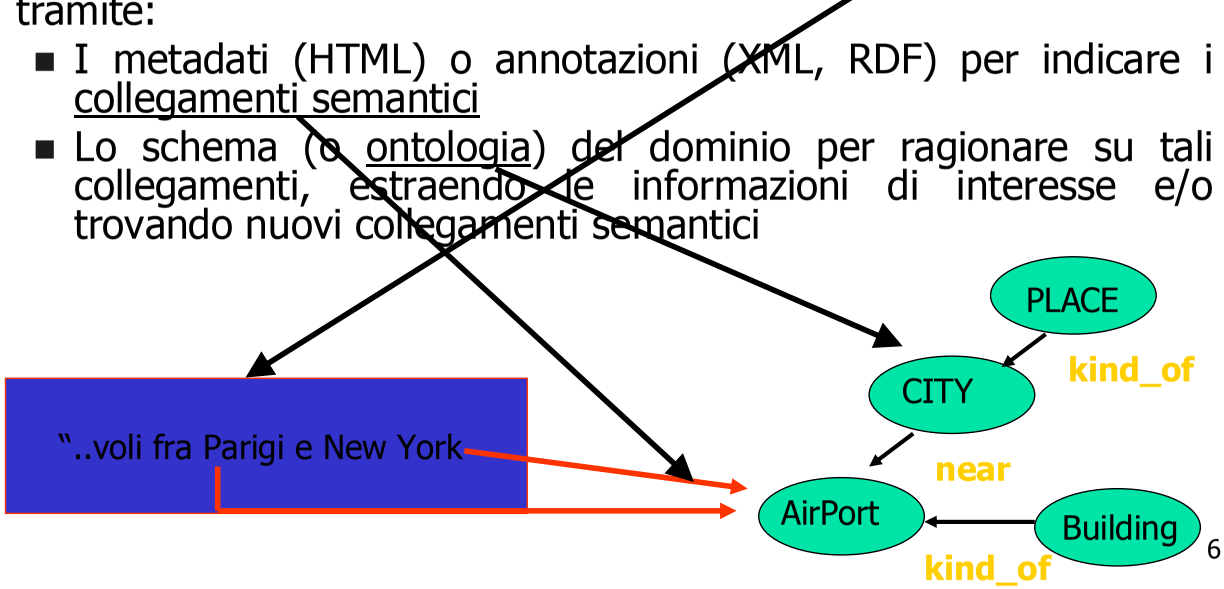


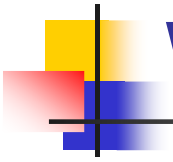





# Web Semantico

- Nel web, invece, le informazioni sono in genere NON strutturate; è quindi necessario fornire tale struttura ai dati (le pagine web) tramite:
  - I metadati (HTML) o annotazioni (XML, RDF) per indicare i collegamenti semantici
  - Lo schema (o ontologia) del dominio per ragionare su tali collegamenti, estraendo le informazioni di interesse e/o trovando nuovi collegamenti semantici



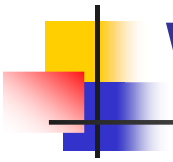


# Web Semantico

- Il linguaggio HTML fornisce una soluzione primitiva al problema della descrizione semantica dei contenuti utilizzando il tag META (che sta per meta-dato) nella sezione HEAD. I tag meta non vengono visualizzati a chi fruisce della pagina HTML come lettore bensì da taluni motori di ricerca per valutare l'appropriatezza di una pagina come risposta ad una certa domanda.
- In generale un tag meta ha la seguente forma:
- <meta NAME="nome" CONTENT="valore">
- Con questa indicazione l'autore di una pagina associa la stringa "valore" alla stringa "nome", dove "nome" indica una proprietà del documento e "valore" il valore di tale proprietà.







# Web Semantico

- Esempi di utilizzo di metatag sono:
- <head>
- <meta NAME="AUTHOR" CONTENT="C. Bianchi">
- <meta NAME="KEYWORDS" CONTENT="XML, programmazione web, query">
- <meta NAME="DESCRIPTION" CONTENT="introduzione all'XML">
- </head>
- L'idea è quindi trasferire nella sezione HEAD le meta-informazioni relative ad un documento, privilegiando il contenuto di tale sezione come informazione utile durante una ricerca.

# Web Semantico

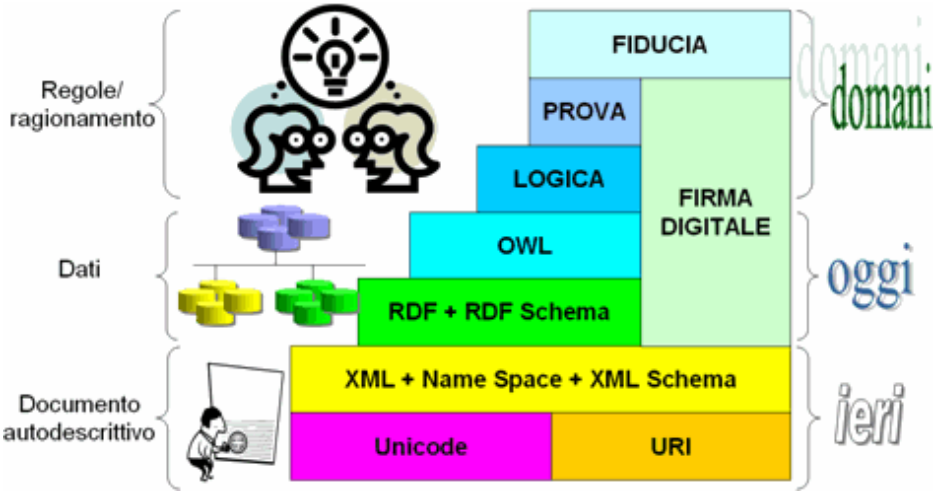
- L'utilizzo del tag HTML <meta> presenta un primo limite, ossia non esistono convenzioni né sui nomi né sulla struttura del contenuto delle proprietà da definire, limitando fortemente l'utilizzo di sistemi automatici.
- Se un sistema non è in grado di identificare o distinguere nomi di proprietà non potrà, per esempio, distinguere il caso in cui le meta-informazioni specificano che il sito è stato scritto da "C. Bianchi" da quello in cui si dichiara che il sito riguarda la vita e le opere di "C. Bianchi"
- Il secondo limite dell'uso del tag <meta> è che non permette di esprimere legami semantici.

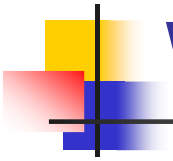




# Web Semantico

- **L'architettura del semantic web**, necessaria allora per permettere ai dati di essere machine-understandable, ha diversi livelli:



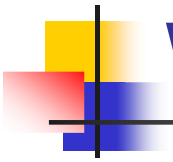


# Web Semantico

- URL vs. URN vs. URI
  - The difference between the three is subtle. An URL refers to a Web page, including the scheme, but without a name location. An URN may also include the location of a code fragment. An URI refers to a Web page including the location of the code fragment, if one exists, and the scheme.
  - URL http://www.cnn.org/iis/review1.htm
  - URN www.cnn.org/iis/review1.htm#one
  - URI http://www.cnn.org/iis/review1.htm#one
  - Because Web servers allow for default documents and do not require a scheme to retrieve a document, the subtle difference between an URL, URN and URI is hard to tell.

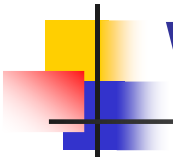






# Web Semantico

- **XML, Name Space e XML Schema**:
  - XML (eXstensible Markup Language) è un meta-linguaggio di markup. In pratica fornisce un insieme di regole sintattiche per modellare la struttura di documenti e dati. Questo insieme di specifiche definiscono le modalità con cui crearsi un proprio linguaggio di markup. XML reca tra i suoi vantaggi fondamentali quello di garantire un'alta interoperabilità dei dati (e dunque di consentire l'interscambio di dati tra piattaforme ed applicativi diversi).
  - La struttura e la grammatica soggiacenti ad un documento XML possono essere stabilite attraverso un DTD (Document Type Definition) o (meglio) attraverso XML Schema, che fornisce un metodo per comporre vocabolari XML.
  - Un Namespace non è altro che un insieme di nomi di elementi e/o attributi identificati in modo univoco da un identificatore. La presenza di un identificatore univoco individua così un insieme di nomi distinguendoli da eventuali omonimie presenti in altri namespaces.

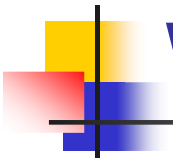


# Web Semantico

- **RDF e RDF Schema**:
  - RDF (Resource Description Framework) fornisce un insieme di regole per definire informazioni descrittive sui dati, più precisamente sugli elementi costitutivi un documento web; queste asserzioni sono realizzate tramite triple che legano tra loro gli elementi in una relazione binaria. Le triple sono del tipo: Soggetto (la risorsa), Predicato (la proprietà) e Oggetto (il valore). Un modello RDF è rappresentabile da un grafo orientato sui cui nodi ci sono risorse o tipi primitivi e i cui archi rappresentano le proprietà.
  - RDF Schema fornisce, a sua volta, un metodo per combinare queste descrizioni in un singolo vocabolario. Il modo per sviluppare vocabolari specifici per un dato dominio di conoscenza è rappresentato dalle ontologie.

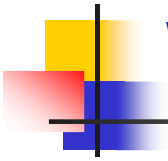






# Web Semantico

- Nell'ambito del Web Semantico, il W3C ha sostenuto lo sviluppo di **OWL (Web Ontology Language)** quale linguaggio per la definizione di ontologie strutturate basate sul Web.
- OWL è un linguaggio di markup per rappresentare esplicitamente significato e semantica di termini con vocabolari e relazioni tra i termini. Tale rappresentazione dei termini e delle relative relazioni costituisce un'ontologia.
- L'obiettivo è permettere ad applicazioni software di elaborare il contenuto dei documenti scritti in OWL.

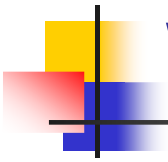


# Web Semantico

- Logica, Prova e Fiducia:
  - **Logica:** Affinché il Web Semantico possa effettivamente aiutarci in una vasta gamma di situazioni, estraendo autonomamente informazioni utili dalla mole di documenti annotati semanticamente, è indispensabile costruire un potente linguaggio logico per realizzare le inferenze (ovvero procedimenti deduttivo mediante cui, a partire da una o più premesse, si ricava, per via logica, una conclusione).
  - **Prova:** Le conclusioni ottenute saranno validate a questo livello tramite motori di validazione costituiti da sequenze di formule derivate da assiomi.
  - **Trust:** Infine il sistema restituirà solo quelle informazioni che secondo il richiedente proverranno da utenti di indubbia attendibilità.

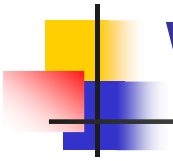






# Web Semantico

- Volendo semplificare il discorso possiamo dire che alla base vi dev'essere una diversa e più attenta filosofia di progettazione delle risorse web - basate su XML -, le quali devono rispettare gli standard definiti e recare con se una descrizione delle proprie caratteristiche (tramite RDF e metadati).
- Ciascuna di queste risorse sarà identificabile in modo non ambiguo grazie all'uso degli URI (risolvendo così i problemi di ambiguità visti quando abbiamo parlato dei motori di ricerca).
- I metadati sono la base informativa su cui potranno operare gli agenti intelligenti per prendere le proprie decisioni.
- Gli agenti, a loro volta, potranno muoversi nello spazio-web sfruttando il sistema di rappresentazione della conoscenza disponibile (ontologie). Le decisioni degli agenti a questo punto saranno consentite grazie all'utilizzo di linguaggi di inferenza logica. Gli agenti, infine, nel prendere le proprie decisioni terranno conto del grado di fiducia attribuito alle risorse (ed ai loro autori identificati da sistemi di firma digitale) dagli utenti stessi.

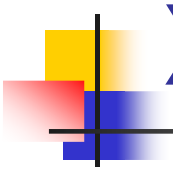


# Web Semantico

- La piena realizzazione dei principi del Web Semantico è probabilmente ancora lontana da una sua realizzazione e gli ostacoli maggiori al suo sviluppo si incontrano proprio al livello ontologico dell'architettura precedentemente vista.
- L'onerosità della mappatura delle risorse, la piena interoperabilità tra i diversi linguaggi utilizzati per la descrizione dei dati e le relazioni tra essi, i cambiamenti, anche culturali, profondi che si richiedono soprattutto in fase di progettazione dei documenti destinati al web richiedono uno sforzo supplementare e quell'adeguamento sociale e tecnologico che fin dagli inizi Berners Lee aveva indicato come chiave del cambiamento.







# XML

**Confronto XML – HTML:**

- Dal punto di vista dell'organizzazione dei dati XML somiglia ad HTML. Infatti anche in questo caso si fa uso di tag, con la notazione <TAG> testo </TAG>. La differenza principale è che mentre in HTML i tag sono predefiniti, in XML ogni utente può definire i propri tag.
- Un'altra differenza (più subdola) è che mentre HTML è case-insensitive, XML è case-sensitive.
- In secondo luogo l'XML è molto più preciso dell'HTML riguardo la chiusura dei tag: ad ogni <TAG> corrisponde sempre una chiusura </TAG>.
- Possono inoltre essere definiti tag vuoti ma la loro sintassi è <TAG/>.

# XML

- **Esempio di file XML:**

Rex Stout è l'autore di molti romanzi gialli
aventi come protagonisti il famoso Nero Wolfe
e il suo aiutante e narratore Archie Goodwin

**<?xml version="1.0"?>**
**<autore>**Rex Stout**</autore>** è l'autore di molti
**<genere>**romanzi gialli**</genere>** aventi come protagonisti
il famoso **<protagonista>**Nero Wolfe**</protagonista>**
e il suo aiutante e narratore
**<narratore><protagonista>**
Archie Goodwin
**</protagonista></narratore>**





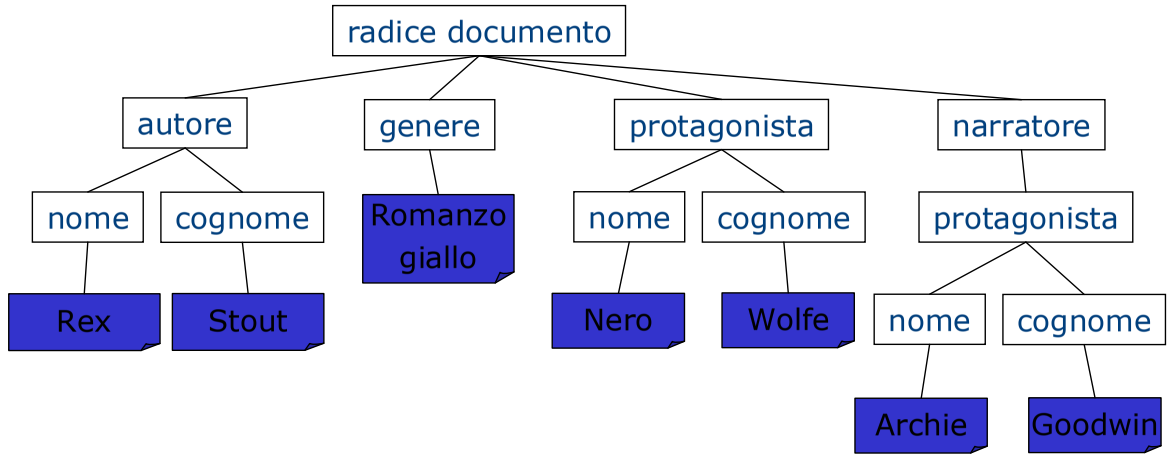
# XML

- Permettendo l'annidamento, un documento XML può essere rappresentato mediante una struttura ad albero. Ad esempio:

**<?xml version="1.0"?>**
**<autore><nome>**Rex**</nome><cognome>**Stout**</cognome></autore>**
è l'autore di molti **<genere>**romanzi gialli**</genere>**
aventi come protagonisti il famoso
**<protagonista attributo="famoso"><nome>**Nero**</nome>**
**<cognome>**Wolfe**</cognome></protagonista>**
e il suo aiutante e narratore
**<narratore>**
**<protagonista mansione="aiutante">**
**<nome>**Archie**</nome>**
**<cognome>**Goodwin**</cognome>**
**</protagonista>**
**</narratore>**

# XML

- Graficamente








# XML

- Un **documento XML è costituito da tre parti**, che possono essere contenute in tre file diversi:
  - **una Document Type Definition (DTD)**: è l'insieme delle regole che definisce la struttura dei dati, ovvero per la definizione degli elementi, degli attributi e delle loro correlazioni (la DTD definisce un vero linguaggio con un proprio vocabolario e regole sintattiche). Le DTD sono contenute alternativamente nel documento XML vero e proprio oppure in un file di testo con estensione dtd;
  - **un foglio di stile**: specifica le regole con cui un documento deve essere visualizzato da un applicativo (browser, word processor…);
  - **il documento XML vero e proprio: i dati**, organizzati secondo la struttura definita dalla DTD. I veri e propri documenti XML sono contenuti in file di solo testo con estensione xml.
- Come nel caso dell'HTML i dati possono alternativamente essere scritti in file oppure essere generati dinamicamente da applicativi quali i sistemi per la gestione di banche dati.

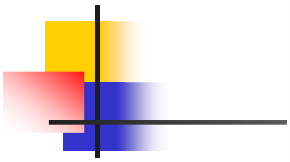
# XML

- La DTD contiene la definizione della struttura dei dati di uno o più documenti. Quando si parla di struttura dei dati di un documento si assume che il documento possa essere ricorsivamente scomposto in parti (per esempio un libro può essere scomposto in capitoli, che possono essere suddivisi in sezioni e così via)
- in certi casi i vari elementi devono rispettare un particolare ordine di precedenza (per esempio le voci di una rubrica telefonica potrebbero prevedere il nome di una persona come primo elemento, seguito dal numero di telefono e dall'indirizzo).
- La strutturazione dei documenti XML avviene attraverso la definizione di tutti gli elementi che possono denotare il documento e delle loro inter-relazioni
- in altri termini la DTD di un documento contiene le definizioni dei tag e della struttura attesa dei dati.





# XML



Esempio di DTD incorporato

```
<?xml version="1.0" standalone="yes"?>

<!DOCTYPE ricetta [
     <!ELEMENT ricetta (titolo,lista_ingredienti,preparazione)>
     <!ELEMENT titolo (#PCDATA)>
     <!ELEMENT lista_ingredienti (ingrediente+)>
     <!ELEMENT preparazione (passo+)>
     <!ELEMENT ingrediente (#PCDATA)>
     <!ELEMENT passo (#PCDATA)>
]>

<ricetta>
     <titolo> insalata </titolo>
     <lista_ingredienti>
          <ingrediente> indivia </ingrediente>
          <ingrediente> pomodoro </ingrediente>
          <ingrediente> cetriolo </ingrediente>
          <ingrediente> sedano </ingrediente>
          <ingrediente> olio </ingrediente>
          <ingrediente> aceto </ingrediente>
          <ingrediente> sale </ingrediente>
     </lista_ingredienti>
     <preparazione>
          <passo> lavare l'invidia, il sedano e il pomodoro</passo>
          <passo> pelare il cetriolo </passo>
          <passo> affettare cetriolo, pomodoro e sedano </passo>
          <passo> tagliare l'insalata</passo>
          <passo> mescolare le verdure in un'insalatiera </passo>
          <passo> versare olio e aceto, salare e mescolare</passo>
     </preparazione>
</ricetta>
```

# XML

- Un esempio di DTD completo:

```
<!ELEMENT organizer (rubrica, agenda)>

<!ELEMENT rubrica ((categoria, voce)*)>
<!ELEMENT categoria (#PCDATA)>
<!ELEMENT voce (((nome, cognome) | rag_sociale), tel*,
                indirizzo?)>
<!ATTLIST voce id_voce ID #REQUIRED>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT rag_sociale (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT indirizzo (#PCDATA)>

<!ELEMENT agenda (appuntamento*)>
<!ELEMENT appuntamento (#PCDATA)>
<!ATTLIST appuntamento giorno_sett (lun | mar | mer | gio | ven |
sab | dom) #IMPLIED>
<!ATTLIST appuntamento luogo CDATA #REQUIRED>
<!ATTLIST appuntamento chi_incontro IDREF #REQUIRED>
<!ATTLIST appuntamento quando CDATA #REQUIRED>
```

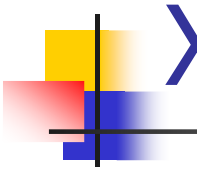




# XML

Supponendo che la DTD precedente sia memorizzata nel file "organizer.dtd" possiamo scrivere un documento XML che la utilizza:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE organizer SYSTEM
"http://www.di.unito.it/~baroglio/proveXML/organizer.dtd">

<organizer>

<!-- parte di rubrica -->
<rubrica>
    <categoria>amici</categoria>
    <voce id_voce="voce_matt">
        <nome> Matteo </nome>
        <cognome> Baldoni </cognome>
        <tel> 12345 </tel>
        <tel> 67890 </tel>
        <indirizzo> via pinco palla 21 </indirizzo>
    </voce>
</rubrica>

<!-- parte di agenda -->
<agenda>

    <appuntamento luogo="ristorante X"
                  chi_incontro="voce_matt" quando="4/6">
        Compleanno Matteo!!!
    </appuntamento>

</agenda>

</organizer>
```



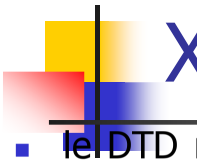
# XML

- Oltre agli elementi (tag) e ai loro attributi, una DTD può contenere la definizione di un certo numero di **entità**. Le entità corrispondono sostanzialmente a delle **macro**: consentono di attribuire a una sequenza di caratteri un nome convenzionale, che viene utilizzato (per comodità) al posto della sequenza stessa
- Le **notazioni** sono utilizzate tipicamente per indicare quale applicativo consente di trattare un certo tipo di documenti. Per esempio, la seguente specifica associa l'applicativo "netscape.exe" al tipo di documento jpeg.

```
<!NOTATION jpeg SYSTEM "netscape.exe">
```







# XML Schema

- I DTD non sono l'unico modo per definire la struttura di un documento XML. Un'alternativa è data dagli **XML Schema**. Le principali differenze rispetto alle DTD sono che da un lato è possibile :
  - esprimere vincoli di tipo,
  - esprimere vincoli di cardinalità.
- Ad esempio elemento peso definito come <!ELEMENT peso (#PCDATA)>. In un documento, sono ammessi utilizzi come <peso>15</peso>, <peso>Anna</peso>, <peso>una giornata in barca</peso>. Intuitivamente potremmo pensare che si tratti di una misura, quindi che il primo uso sia l'unico corretto però è impossibile che il parser controlli che peso racchiuda solo dei numeri, quindi tutti e tre gli usi sono formalmente corretti. La seconda differenza riguarda i vincoli di cardinalità: abbiamo visto che nelle regole che specificano la struttura degli elementi possono essere utilizzati gli operatori di ricorrenza * e +. Non possiamo però esprimere strutture del tipo "l'elemento A è costituito da una lista di almeno 2 ma non più di 5 elementi B".

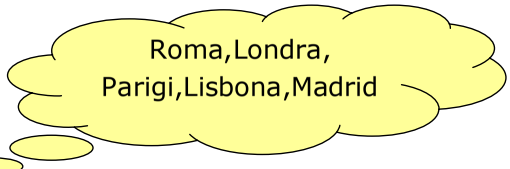
# XML Schema

- DTD (Document Type Definition)
  - Specifica annidamenti e combinazioni permesse di elementi, attributi ecc.
  - Un doc XML conforme a DTD = valido
- XML Schema
  - Definisce lo schema sintattico con la stessa sintassi XML
  - Fa uso di namespace e definizione di tipi
  - Un doc XML conforme a XML Schema = schema valido

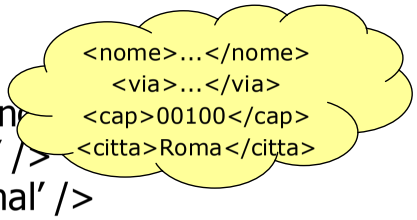




# XML Schema




- Definizioni di tipo
  - Semplici
    <simpleType name='stringlist' base='string' derivedBy='list' />
  - Complessi
    <complexType name='indirizzo'>
      <sequence>
        <element name='nome' type='strin
        <element name='via' type='string' />
        <element name='cap' type='decimal' />
        <element name='citta' type='string' />
      </sequence>
    </complexType>
- Dichiarazioni di elementi e attributi
  <element name='elenco' type='stringlist'>



# XML Schema

- Schema

```
<schema xmlns='http://www.w3.org/1999/XMLSchema'
    targetNamespace='http://lazoo.org'>
  <simpleType name='comment' base='string'>
    <maxLength value='1024'>
  </simpleType>
  <element name='about' type='comment'>
  <element name='autore' type='string'>
</schema>
```

- Istanza schema-valida

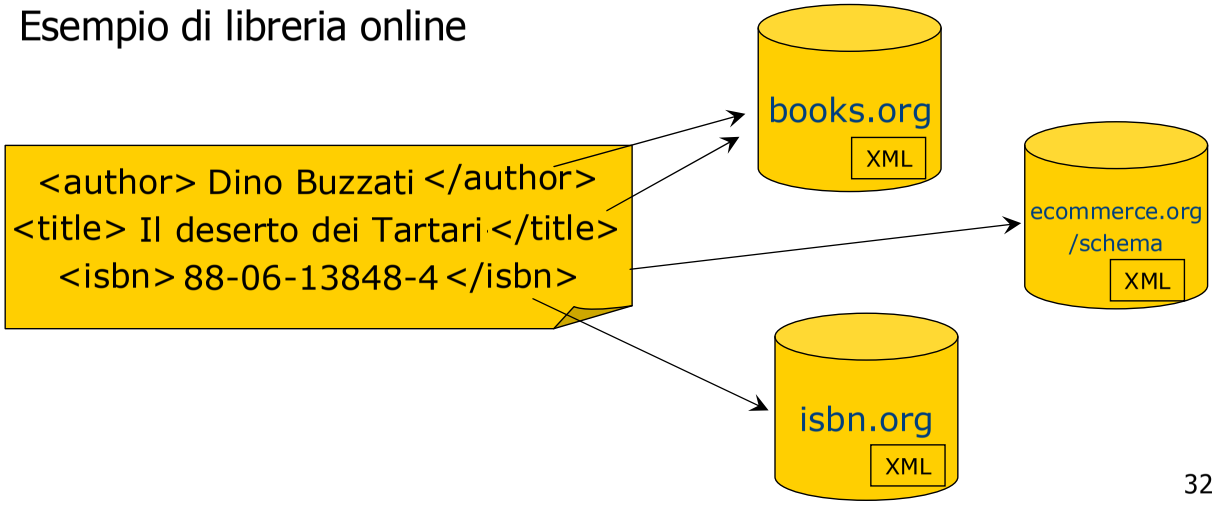
```
<?xml version='1.0' xmlns='http://lazoo.org' ?>
<autore>Luigi Pirandello</autore>
<about>romanziere e commediografo</about>
```

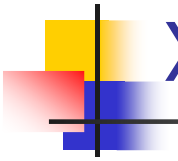




# XML NameSpaces

- Namespace = collezione di nomi identificati da URI (Uniform Resource Identifier)
- Associati a elementi e attributi di documenti XML
- Permettono la visibilità al di fuori del documento
- Esempio di libreria online



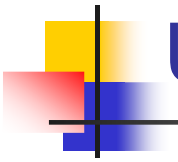


# XML NameSpaces

```
<?xml version='1.0'
    xmlns:books='http://www.books.org/'
    xmlns:store='http://ecommerce.org/schema'
    xmlns:isbn='http://www.isbn.org/' ?>
<store:bookstore>
    <books:book>
        <books:title>Il deserto dei Tartari</books:title>
        <books:author>Dino Buzzati</books:author>
        <isbn:number>88-06-13848-4</isbn:number>
    </books:book>
    <books:book>
        <books:title>L'isola di Arturo</books:title>
        <books:author>Elsa Morante</books:author>
        <isbn:number>88-06-13838-3</isbn:number>
    </books:book>
</store:bookstore>
```

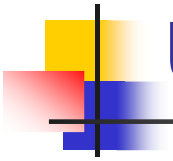






# Utilizzo di XML

- E' stato introdotto il linguaggio XML, ponendo particolare attenzione alla costruzione di DTD. Una domanda che potrebbe sorgere spontanea è: ora che abbiamo strutturato l'informazione relativa ad un dominio, che cosa ce ne facciamo?
- Alcuni utilizzi di XML:
  - Come sintassi di serializzazione per altri linguaggi
  - Come markup per risorse (documenti, pagine web, configurazioni ecc.)
  - Come formato uniforme per lo scambio dei dati, anche al fine di accedere alle informazioni in maniera strutturata (stile database)



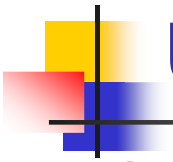
# Utilizzo di XML

- Come sintassi di serializzazione per altri linguaggi
  - Esempio: Java → XML

```
class Hello
{
    static public void main(String[] args)
    {
      System.out.println("Hello world!");
    }
}
```







# Utilizzo di XML

- Come sintassi di serializzazione per altri linguaggi
  - Esempio: Java → XML

```
<?xml version="1.0" ?>
<class name="Hello">
    <method name="main" static="true" scope="public"
      returnType="void" input="String[]">
        System.out.println("Hello world!");
    </method>
</class>
```

# Utilizzo di XML

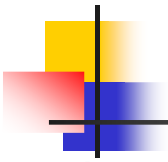
- Come markup per risorse (documenti, pagine web, configurazioni)
  - Es. classificazione delle istanze di un documento:

    Tra i precursori del genere giallo troviamo

    **<autore>**Edgar Allan Poe**</autore>**, con **<romanzo genere="giallo">**I delitti della Rue Morgue**</romanzo>**, e **<autore>**Arthur Conan Doyle**</autore>** con il suo **<personaggio>**Sherlock Holmes**</personaggio>**.
  - Altro esempio quello degli editor di testo: diversi editor utilizzano (in modo trasparente all'utente) il linguaggio XML per mantenere le informazioni relative ai documenti costruiti. Questo significa che il testo scritto viene man mano opportunamente "taggato" in XML dall'editor.
  - Ai vari tag sono inoltre associate regole di visualizzazione (o presentazione) e l'editor le usa per far sì che ad esempio un testo dichiarato in grassetto, oltre ad essere descritto in XML, appaia agli occhi dell'utente in grassetto.







# Utilizzo di XML

- Come sintassi di serializzazione per altri linguaggi
- Come markup per risorse (documenti, pagine web, configurazioni ecc.)
- Come formato uniforme per lo scambio dei dati
  - Es. SOAP (Simple Object Access Protocol)

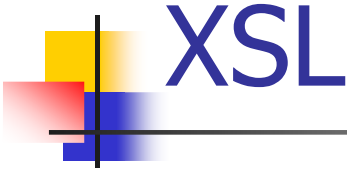
```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.stock.org/stock">
  <m:GetStockPrice>
        <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

# XLL

- L'XML fornisce un complesso meccanismo per la gestione dei link, definiti dallo standard XML Linking Language (XLL), tramite cui:
- specificare quando un link deve attivarsi (manualmente o meno);
  - specificare cosa effettuare a seguito dell'attivazione (aprire una nuova finestra, sostituire la precedente, o sostituire l'elemento iniziale;
  - collegare l'elemento iniziale a più risorse (link semplici o estesi), specificando la regola per ogni singolo locator;
  - definire una gerarchia di documenti tramite l'X-pointer, così da poter supportare link assoluti (es puntare alla radice, puntare al terzo figlio), o relativi alla gerarchia (es puntare al precedente documento), eventualmente completandoli con string matching, ad esempio:
  - ROOT.CHILD(3,CHAP).STRING(7,"Napoleone")
    trova la settima occorrenza della stringa "Napoleone" all'interno del terzo capitolo del documento puntato.







# XSL

- L'XML fornisce un complesso meccanismo per la rappresentazione dei documenti, l'eXtensible Stylesheet Language (XSL).
- l'XSL si basa sulle template rules, regole che consentono di intercettare una sequenza richiesta di tag all'interno del documento XML da formattare e di associarvi un opportuno oggetto di formattazione.
- L'XSL processor prende in ingresso il documento XML da rappresentare (source tree), vi applica le template rules e genera il documento con i soli oggetti di formattazione (output tree). Esso viene dato ad un formatter in cui viene dettagliatamente descritto come rappresentare ogni oggetto di formattazione. Il formatter deve essere incluso nei browser.

# RDF

- **Resource Description Framework** (RDF) è una particolare *applicazione XML* (standardizzata dal W3C) deputata alla gestione delle **relazioni tra informazioni** ispirandosi ai principi della *logica dei predicati* (o logica predicativa del primo ordine) e ricorrendo agli strumenti tipici del Web (ad es. URI) e dell'XML (namespace).
- In estrema sintesi, secondo la logica dei predicati le informazioni sono esprimibili con **asserzioni** (statement) costituite da triple formate da **soggetto**, **predicato** e **valore** (noti anche come subject, verb e object, rispettivamente).
- RDF lavora quindi con tre tipi di elementi fondamentali:
  - **Risorse**: entità riferite attraverso URI
  - **Proprietà**: relazioni binarie tra risorse e/o valori atomici di tipo primitivo
  - **Affermazioni**: specificano il valore di una certa proprietà relativa a una risorsa





# RDF

- Esempio:

affermazione

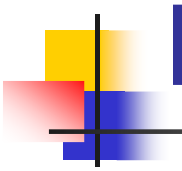
Autore(http://www.mywebsite.com)=JohnDoe

proprietà — risorsa soggetto — risorsa oggetto

Si può esprimere come tripla (soggetto, predicato, oggetto):
**(http://www.mywebsite.com, Autore, JohnDoe)**

- Altro esempio:

autoreDi("http://www.pinco.com","http://www.books.org/ISBN00515")
haPrezzo("http://www.books.org/ISBN00515", "10 euro")



# RDF

- Le triple RDF sono rappresentabili in diversi modi, ma quello piu' diffuso e' tramite XML, ad esempio la frase "Il Signor Napolitano vive a Roma ed ha Codice Fiscale NPLGRO20T09E625V", ha la formalizzazione RDF equivalente:

```
<?xml version='1.0'?>
  <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:wikipedia="http://it.wikipedia.org/wiki/"
        xmlns:wikidizionario="http://it.wiktionary.org/wiki/">
<rdf:Description
        rdf:about="http://www.quirinale.it/presidente/napolita.htm">
        <wikidizionario:vivere rdf:resource="http://www.comune.roma.it"/>
        <wikipedia:codice_fiscale>NPLGRO20T09E625V
                </wikipedia:codice_fiscale>
</rdf:Description>
</rdf:RDF>
```





# RDF Schema

- RDF consente di definire un semplice modello dei dati per descrivere proprietà di e relazioni fra risorse. In RDF però non esistono livelli di astrazione: risorse e relazioni sono organizzate in un grafo piatto. In altri termini non è possibile definire tipi (o classi) di risorse con loro proprietà specifiche.
- Vista l'utilità di poter definire classi di risorse, RDF è stato arricchito con un semplice sistema di tipi detto **RDF Schema**. Il sistema di tipi RDF Schema ricorda i sistemi di tipi dei linguaggi di programmazione object-oriented (tipo Java). Una risorsa può, per esempio, essere definita come istanza di una classe (o di più classi) e le classi possono essere organizzate in modo gerarchico. RDF Schema utilizza il modello RDF stesso per definire il sistema di tipi RDF, fornendo un insieme di **risorse e proprietà predefinite** che possono essere utilizzate per definire classi e proprietà a livello utente. L'insieme delle risorse e delle proprietà di base delle risorse è detto vocabolario dell'RDF Schema.

# RDF Schema

- Esempio:

```
<rdf:Description rdf:ID="Mammifero">
   <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Class"/>
   <rdf:subClassOf rdf:resource="http://www.w3c.org/2000/01/rdf-
schema#Resource"/>
</rdf:Description>

<rdf:Description rdf:ID="Orso">
    <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Class"/>
    <rdf:subClassOf rdf:resource="#Mammifero"/>
</rdf:Description>
```

- tutte le classi sono sottoclasse di qualcosa. Nel caso in cui definiamo una classe che apparentemente non è sottoclasse di nulla, occorre comunque specificare che è sottoclasse di Resource. Nell'esempio "Mammifero" è la classe base (sottoclasse di rdf:Resource) mentre "Orso" è sottoclasse di "Mammifero". Poiché una classe può avere molte sovra-classi, <rdf:subClassOf … /> può comparire molte volte all'interno di una stessa descrizione.





# Utilizzo di RDF

- Condizione necessaria per il buon utilizzo di RDF è la **disponibilità on-line di riferimenti di qualità** alle URI utilizzate/referenziate. In particolare, è importante che le risorse siano **note, condivise e stabili nel tempo**. Ad es., il riferimento utilizzato per il Presidente Napolitano non è dei migliori perché valido solo durante il mandato, dopodiché sarà spostato in *http://www.quirinale.it/ex_presidenti/Napolitano/napolita.htm* dove già si trovano quelle dei suoi predecessori. Dopo questa data l'asserzione [RDF] continuerà a valere, ma si perderà il contributo informativo della pagina web, utile per una completa interpretazione.
- Meglio utilizzare la biografia in wikipedia oppure il Codice Fiscale con *http://www.agenziaentrate.it/servizi/CF#NPLGRO20T09E625V* Sebbene questa URI oggi non referenzi alcunché sul Web, la si potrebbe comunque utilizzare allo scopo perché RDF non presuppone alcuna verifica sulla effettiva disponibilità della risorsa.

# Esempio di RDF

- Production propertyElt :

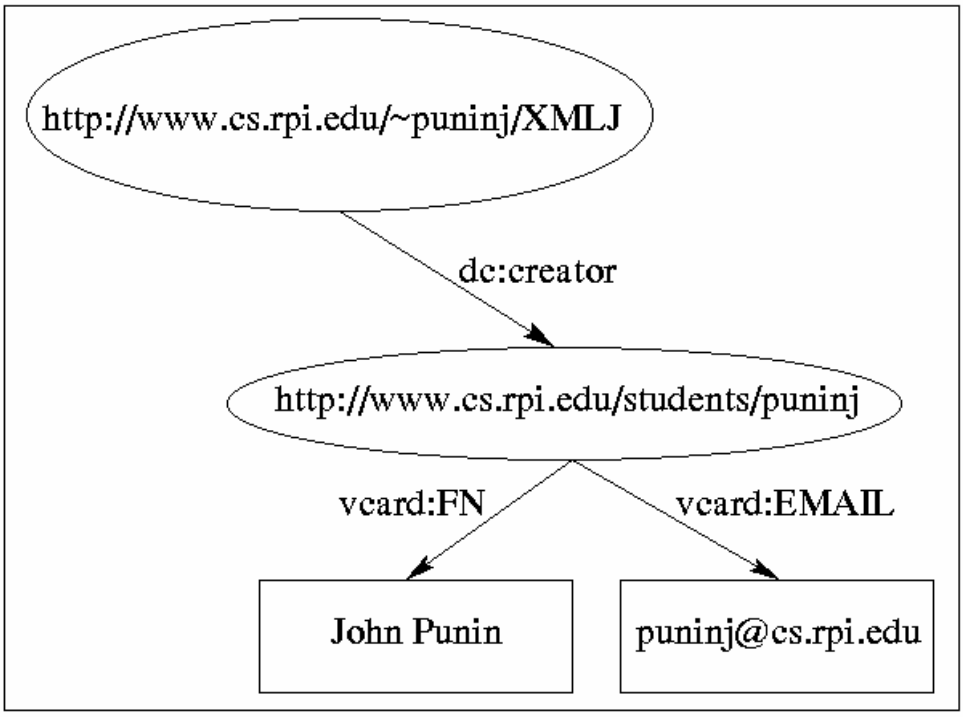
```
<PropertyName> Value </PropertyName> or
<PropertyName rdf:resource="URI" />
```

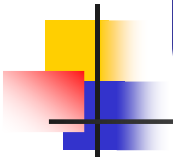
```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:vcard="http://imc.org/vCard/3.0#">
  <rdf:Description about="http://www.cs.rpi.edu/~puninj/XMLJ/">
    <dc:creator rdf:resource="http://www.cs.rpi.edu/students/puninj"/>
  </rdf:Description>

  <rdf:Description about="http://www.cs.rpi.edu/students/puninj">
      <vcard:FN>John Punin</vcard:FN>
      <vcard:EMAIL>puninj@cs.rpi.edu</vcard:EMAIL>
  </rdf:Description>
</rdf:RDF>
```





# Esempio di RDF





# Utilizzo di RDF

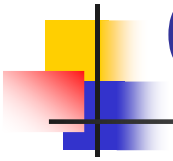
- Una delle applicazioni RDF più diffusa è l'RSS (RDF Site Summary), una modalità di rappresentazione di informazioni contenute nei siti web. L'esigenza nasce dal volere
  - accedere a informazioni **disseminate in piu' siti**, quindi da integrare
  - accedere ad informazioni **variabili nel tempo**, quindi si deve mantenere aggiornato quanto desiderato;
- classico esempio, le "news" di siti informativi, variabili e distribuite
- RSS 1.0, basato su RDF (versioni precedenti di RSS non utilizzavano RDF), è un vocabolario RDF che permette la descrizione di informazioni memorizzate su siti web
- RSS è solo uno strumento descrittivo; come sempre, occorre che una qualche applicazione ne faccia uso per ottenere i risultati voluti (integrazione di informazioni variabili nel tempo).





# Utilizzo di RDF

- Esistono tre categorie di software che gestiscono l'RSS 1.0:
  - **On-line aggregators**, come Meerkat e NewsIsFree, che effettuano l'integrazione di diversi item disseminati su vari siti, proponendola all'utente e permettendo di effettuare ricerche su tale aggregazione; in questo modo si possono avere le ultime informazioni su, ad esempio, la politica in medio oriente, senza dovere cercare in tanti siti (a patto ovviamente che i siti supportino l'RSS)
  - **Desktop Readers**, che permettono agli utenti di sottoscriversi a diversi feed RSS, tenendoli periodicamente aggiornati sul desktop del client
  - **Scripts**, software che permettono di includere feed RSS di altri siti nel proprio (scopo originale di RSS)



# Oltre l'RDF…

- RDF Schema è uno strumento che consente (in modo molto basilare) di definire vocabolari RDF, tuttavia sarebbe auspicabile arricchire questo linguaggio o, in alternativa, definire nuovi **linguaggi che forniscono qualche capacità in più**. Alcune di queste capacità sono state individuate ed esistono già linguaggi (es. DAML+OIL) che le forniscono; esse sono:
  1. vincoli di cardinalità (es. una persona ha una sola madre e un solo padre)
  2. possibilità di indicare che due classi definite in schemi differenti rappresentano lo stesso concetto
  3. possibilità di indicare che due istanze, definite separatamente, rappresentano lo stesso individuo
  4. possibilità di indicare che una proprietà è transitiva
  5. la possibilità di definire una nuova classe come combinazione di più classi esistenti.
- La definizione e la realizzazione di queste (e altre) capacità sono lo scopo dei gruppi di lavoro sui linguaggi per la definizione di ontologie.