



UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica

TESI DI LAUREA

WEB SEMANTICO: lo stato dell'arte

Relatore: Prof. *Michele Moro*

Laureando: *Davide Buongiorno*

ANNO ACCADEMICO 2008/2009

*alla mia famiglia e a quanti mi hanno
sostenuto in quest'impresa*

Indice

Sommario	7
Introduzione	9
1. World Wide Web	13
1.1 Introduzione	13
1.2 Tim Berners-Lee e Robert Cailliau	14
1.3 Storia	15
1.4 HTML e XHTML	16
1.5 HTTP e HTTPS	20
1.6 World Wide Web Consortium (W3C) e W3C Italia (W3C-IT)	22
2. Introduzione al Web Semantico	27
2.1 Dal web statico al web dinamico. Web Services	27
2.2 Dal web statico al Web Semantico	36
2.2.1 Motori di ricerca	38
2.2.2 Agente semantico	39
2.2.3 Introduzione all'architettura del Web Semantico	41
2.2.4 Metadati	43
2.2.5 Ontologie	45
2.2.6 Conclusione ipotizzata da Berners-Lee	48
2.2.7 Evoluzione dal Web 1.0 al Web 3.0	49
2.3 W3C Semantic Web Activity	50
3. Architettura a livelli del Web Semantico	53
3.1 Unicode, URI, IRI, XRI	53
3.2 XML, XML Schema	61
3.3 RDF, RDF/XML, RDF Schema	72
3.4 OWL, OWL2, SPARQL, SKOS, RIF, SWRL	95
3.5 POWDER	126

3.6 Logica, dimostrazioni, firme digitali e fiducia	133
4. Un'applicazione già operativa: GoPubMed	137
4.1 MEDLINE (PubMed), Gene Ontology, MeSH	138
4.2 GoPubMed	145
Conclusioni	157
Appendice	159
Indice delle abbreviazioni	159
Bibliografia	163
Ringraziamenti	169

Sommario

Il termine “Web Semantico” è stato introdotto dal suo inventore, il ricercatore Tim Berners-Lee, nel 1991, per indicare la trasformazione del World Wide Web in un ambiente dove i documenti pubblicati (quali pagine HTML, ecc.) sono associati a specifiche informazioni, i metadati, che ne specificano il contesto semantico.

Queste informazioni, aggiunte ai documenti, devono avere una forma tale da permettere l'interrogazione, l'interpretazione ed, in generale, l'elaborazione automatica, da parte dei computer, del contenuto dei documenti stessi.

L'informazione, che può anche essere intesa come insieme delle conoscenze, è spesso dispersa tra più fonti, che sono sparse nel Web. L'obiettivo principale del Web Semantico è fare in modo che le macchine riescano, autonomamente, ad estrarre e a dedurre nuova conoscenza.

Da allora ad oggi molto lavoro è stato fatto e tanto altro è tuttora in corso, sia da enti preposti, che da persone interessate all'argomento.

Non si è ancora arrivati alle capacità originariamente ipotizzate da Berners-Lee, in quanto la sua “visione” iniziale era troppo futuristica. Egli stesso, nel corso degli anni successivi, ha ridimensionato il proprio punto di vista.

Attualmente esistono sistemi in grado di rappresentare la conoscenza, intesa come la definizione di un dominio, delle sue specificazioni, delle sue proprietà e delle relazioni con altri domini, tramite le ontologie: questo può essere realizzato per domini abbastanza ristretti e ben specificati.

Per un prossimo futuro si ipotizza che questi “dati sui dati” riescano a rappresentare domini di conoscenza sempre più ampi che possano essere arricchiti dall'uomo ma, soprattutto, in maniera automatica e sempre più crescente dalle macchine.

Scopo di questa tesi è quello di fare il punto della situazione sul Web Semantico per permettere al lettore di capire a che punto è arrivata la ricerca, cosa si sia riuscito a realizzare effettivamente, quali sono i problemi incontrati e le idee per tentare di risolverli. Il lavoro è strutturato in 4 parti fondamentali:

- presentazione del Web attuale nei suoi vari aspetti;
- evidenziazione dei suoi limiti principali;
- esposizione delle tecnologie dell'architettura a livelli del Web Semantico;
- presentazione di un'applicazione, già operativa, che utilizza le tecniche del Web Semantico finora sviluppate.

Introduzione

Tim Berners-Lee, in un oramai famoso articolo apparso su “Scientific American” del Maggio 2001, immaginava la situazione in cui due fratelli, dovendo far fare alla madre una visita specialistica e delle cure presso un centro fisioterapico, per la prenotazione delle terapie facessero uso di “agenti” intelligenti. Questi agenti, in maniera automatica e grazie al Web Semantico, partendo dalle indicazioni del medico ricercavano la struttura ospedaliera più comoda e consona alla cure. Ricercavano poi, in maniera intelligente ed autonoma, gli orari delle sedute compatibili con gli impegni dei fratelli che, alla fine, dovevano dare solo il loro consenso alla prenotazione.

Questo è solo un esempio delle possibilità che saranno offerte dal Web Semantico, che deve essere visto ora come un'estensione del Web attuale.

“Lo cerco sul Web” è un'espressione che, al giorno d'oggi, è diventata abituale e che testimonia come il Web sia percepito dalla gente come un serbatoio di conoscenza.

Attraverso la rete, le persone possono trovare e fare molte cose: dalla consultazione dell'orario dei treni, alla prenotazione di un albergo, dalla visione della registrazione di un telegiornale, alle telefonate da pc a pc, dalla messaggistica alla chat on-line tra persone, dalla posta elettronica fino agli acquisti on-line, ecc. E, tutto questo, è in continua evoluzione.

Nei primi anni '90 il Web era composto solo da poche pagine testuali. Col passare del tempo, il testo è stato arricchito anche di contenuti multimediali: grafica più ricca, animazioni, audio e video. Il contenuto, inizialmente sviluppato ed implementato da pochi “tecnici”, e memorizzato in rete su poche migliaia di macchine, ha iniziato a crescere in quantità. Contemporaneamente si sono iniziate a migliorare le tecnologie che supportavano il tutto.

Man mano che nuovi standard venivano sviluppati e le potenzialità aumentavano è iniziata anche la partecipazione dell'utente che, anche se non in possesso di grosse conoscenze informatiche, ha iniziato ad arricchire il Web con i propri contenuti d'informazione. Questo ha dato una nuova linfa vitale al Web che è cresciuto in maniera esponenziale.

Questa crescita vertiginosa della informazione presente in rete ha creato alcuni problemi, uno dei più importanti consiste nella difficoltà da parte di un utente di reperire le informazioni precise di cui ha bisogno tra la moltitudine delle informazioni presenti nel Web.

Per gli esseri umani il processo di combinare informazioni, spesso incomplete, provenienti da fonti diverse e memorizzate in formati diversi (pagine web, database, fogli elettronici, ecc.) per ottenere una risposta adeguata alle proprie esigenze è ragionevolmente semplice, anche se talvolta noioso e/o ripetitivo.

Sarebbe desiderabile che le macchine potessero, automaticamente, combinare la conoscenza proveniente dalle diverse fonti ed, ancor meglio, da queste derivarne di nuova.

Per superare questi limiti del Web attuale, da qualche anno, i ricercatori stanno lavorando intensamente per la realizzazione del Semantic Web, che può essere anche definito come un'infrastruttura, basata su "metadati" per svolgere ragionamenti sul Web. Nel Web Semantico la conoscenza è rappresentata in maniera elaborabile dalla macchine e può essere utilizzata da componenti automatizzati, detti "agenti semantici".

I metadati sono informazioni, elaborabili in modo automatico, relative alle "risorse" presenti nel Web, che vengono identificate univocamente dagli "Uniform Resource Identifier".

La tecnologia di riferimento per la codifica, lo scambio ed il riutilizzo di questi metadati strutturati è la "Resource Description Framework", basata su un modello molto semplice di "statement", rappresentabili come triple.

Per esprimere le relazioni sulle associazioni, quindi per evitare che possano essere codificati degli statement sintatticamente corretti, ma privi di senso, è necessario un meccanismo per rappresentare "classi di oggetti". Da questa esigenza nasce l'"RDF Vocabulary Description Language", più noto come "RDF Schema".

Per poter effettuare dei ragionamenti, per definire le classi e per altre esigenze, l'"RDF Schema" da solo non è sufficiente ed occorre, quindi, un modo per rappresentare la conoscenza e le regole che permettano di dedurre ulteriore conoscenza: qui entrano in gioco le ontologie.

Il Web ha la caratteristica fondamentale di essere distribuito ed è inoltre necessario anche un linguaggio, che non solo consenta di esprimere dati e regole sui dati, ma anche che consenta di esportare queste conoscenze per renderle disponibili a qualsiasi applicazione. Il W3C, ente americano che sovrintende alla realizzazione del Web Semantico, ha definito per questa esigenza il "Web Ontology Language".

Altro obiettivo di questa tesi è anche quello di far comprendere al lettore come il Web Semantico, ed un approccio ontologico, consentano di superare gli schemi tradizionali di accesso all'informazione, permettendo di rappresentare, esportare e condividere la conoscenza ed assicurando l'interoperabilità semantica.

Nelle prossime pagine si focalizzeranno gli aspetti più interessanti del Web Semantico, che, attualmente, è diventato un ambito molto vasto ed in continua espansione.

Si farà una panoramica del Web attuale. Si illustrerà dove è nato, come si è sviluppato negli anni, quali sono i suoi standard attuali e quelli in fase di sviluppo.

Si presenteranno gli enti internazionali che ne curano lo sviluppo, la loro organizzazione interna ed il loro metodo di lavoro.

Il Web tradizionale, come tutte le cose, ha delle limitazioni e degli aspetti negativi. Tutto questo verrà analizzato e si farà vedere che vantaggi può portare l'introduzione della semantica nel Web. Questo, come accennato nelle righe precedenti, è fatto tramite degli standard che saranno presentati nel dettaglio tecnico: cosa sono, a cosa servono, che sintassi specifica hanno. A corredo, vengono presentati degli esempi sia teorici che pratici.

L'ambito puramente teorico del Web Semantico sviluppato tramite linguaggi, formati, specifiche, da qualche tempo ha iniziato ad essere applicato in maniera operativa. Questo, purtroppo, è limitato a campi ristretti di conoscenza e ciò viene fatto soprattutto da grosse aziende che, al loro interno, hanno gruppi di persone che si dedicano alla ricerca. In questa tesi viene presentata un'applicazione che illustra i vantaggi e gli svantaggi che si possono avere con l'uso del Web Semantico.

Capitolo 1

World Wide Web

1.1 Introduzione

Il “World Wide Web”, in sigla WWW o abbreviato in Web, è lo spazio di Internet destinato alla pubblicazione di contenuti multimediali, nonché uno strumento per implementare od utilizzare servizi, anche da parte degli stessi utenti di Internet.

Il WWW è nato nel giugno 1991, ad opera di Tim Berners-Lee e di Robert Cailliau, ricercatori, che pubblicarono il primo sito internet.

Scopo iniziale del WWW era quello di sviluppare un software in grado di far condividere tra ricercatori i documenti scientifici in formato elettronico, in una maniera indipendente dalla piattaforma informatica utilizzata dal ricercatore stesso.

Per far questo furono definiti uno standard e dei protocolli di trasmissione al fine di poter scambiare, sulle reti di calcolatori, i documenti prodotti. Inizialmente questi erano solo semplici pagine statiche che contenevano testo, grafici e collegamenti ad altre pagine. Nel 1993 questa nuova tecnologia fu resa disponibile al mondo intero, dando il via alla diffusione del WWW come sistema per inviare e ricevere dati. E' oggi uno dei servizi più utilizzati di Internet, assieme alla posta elettronica.

Gli standard attuali del WWW sono continuamente sviluppati, aggiornati e mantenuti dal World Wide Web Consortium (abbreviato in W3C), che verrà descritto nei prossimi paragrafi.

Oggigiorno il World Wide Web è in continua evoluzione.

Volendo fotografare la situazione attuale, esso appare come un insieme vastissimo di contenuti multimediali (composto da documenti testuali, immagini, audio, video, ecc.) ed un mezzo dove possono essere implementati servizi (quali download di software e dati, commercio elettronico, social network, ecc.).

Questi contenuti sono memorizzati in un'infinità di computer appartenenti alla rete Internet, sparsi nel mondo e detti “server web”, che rendono disponibile dello spazio detto “spazio web”. Questa caratteristica permette efficienza, in quanto i dati memorizzati non sono vincolati ad una particolare locazione fisica.

Questo spazio web è organizzato in “siti web” a loro volta suddivisi in “pagine web”. Le pagine web sono composte da testo e grafica e sono visualizzate sullo schermo del computer dell'utente tramite un programma detto “browser web” o semplicemente browser. Il passaggio tra una pagina ed un'altra, anche appartenente ad un sito diverso, è fatto tramite i “link” che sono parti di testo o grafica tali per cui, una volta che l'utente ci clicchi sopra con il mouse, visualizzano la nuova pagina a cui sono collegati. In realtà questa è una visione semplificata poiché, grazie ad altri “sistemi” di rappresentazione multimediale contenuti nella singola pagina si possono avere pagine molto “dinamiche” ed interattive.

Per accedere ad un sito web, si utilizza il suo “indirizzo web” (precisamente chiamato URL, Uniform Resource Locator), una sequenza di caratteri che permette

la rintracciabilità del sito nello spazio web.

Il WWW si presenta quindi come un'infinità di pagine, più o meno collegate tra loro, raccolte in siti. Non esiste un indice generale sui contenuti di queste pagine. L'utente che debba ricercare un argomento specifico o conosce l'indirizzo web specifico della pagina oppure deve affidare la ricerca ai "motori di ricerca", che non sono altro che siti web che permettono la ricerca dell'argomento tramite l'inserimento di "parole chiave" relative all'argomento ricercato.

Da alcuni anni esistono anche i cosiddetti "portali web", siti web da cui è possibile accedere direttamente ad informazioni che sono poste direttamente dai titolari dei siti web stessi. Un esempio sono i siti delle grandi catene commerciali, dove sono illustrati, specificati e messi in vendita i prodotti che sono venduti dalla catena stessa.

Oltre a quelli già accennati, altri servizi possono essere implementati sul web, anche da parte degli utenti stessi, se ne hanno capacità tecnica. I più conosciuti sono:

- web mail: che permettere di gestire la propria casella di posta elettronica direttamente via web;
- chat: che permette a più persone, situate in posti diversi, la comunicazione in tempo reale;
- streaming: che distribuisce in rete audio e video. Le più conosciute sono la Web TV (trasmissione di canali televisivi) e la Web Radio (trasmissione di canali radiofonici).

Il Web è implementato attraverso degli standard che, nei prossimi paragrafi di questa tesi, verranno approfonditi. I principali sono l'HTML, il "linguaggio" con cui sono scritte le pagine web, l'HTTP, il protocollo di rete per la trasmissione delle pagine stesse e l'URL, il sistema che permette il recupero delle pagine.

1.2 Tim Berners-Lee e Robert Cailliau



Tim Berners-Lee nasce a Londra l'8 Giugno 1955. Si laurea in fisica a 21 anni, presso l'università di Oxford.

Lavora per alcuni anni in aziende tecnologiche per poi approdare, per alcuni mesi, al CERN di Ginevra dove inizia a lavorare su quelli che poi diventeranno le basi del WWW. Nel 1981 lascia il CERN (Centro Europeo per la Ricerca Nucleare di Ginevra) per lavorare presso il John Poole's Image Computer System Ltd. Torna al CERN tre anni dopo.

Nel 1989 propone un progetto globale sugli ipertesti, che diverrà noto come World Wide Web, nome coniato dallo stesso Berners-Lee.

Egli ha sviluppato "httpd", il primo server per il Web e "WorldWideWeb", il primo browser per "navigare" nelle pagine web. Inoltre ha scritto la prima versione dell'HTML e le specifiche iniziali dell'HTTP e dell'URL.

Nel 1993 lascia il CERN e si trasferisce al Laboratory for Computer Science (LCS) del Massachusetts Institute of Technology (MIT) di Boston.

Nel 1994 fonda il World Wide Web Consortium (W3C).

Il 15 Aprile 2004 gli viene assegnato il premio “Millenium Technology”, per l'invenzione del World Wide Web.

Il 16 Luglio 2004 viene insignito dalla regina Elisabetta II d'Inghilterra del titolo di "Knight Commander" dell'Ordine dell'Impero Britannico.

Molti altri sono i riconoscimenti che egli ha ricevuto finora. Una lista completa è visibile nelle pagine a lui dedicate del sito della W3C.

Attualmente Berners-Lee è il direttore del World Wide Web Consortium, oltre a ricoprire numerosi altri incarichi in ambito universitario.

Nel Giugno 2009 il Primo Ministro Gordon Brown ha annunciato che Sir Tim lavorerà con l'UK Government per aiutare a rendere i dati più aperti ed accessibili sul Web, sviluppando il lavoro della “Power of Information Task Force”.

- . - . - . -



Robert Cailliau nasce il 26 Gennaio 1947 a Tongeren in Belgio. Si laurea nel 1969 come ingegnere civile ed ottiene un master in “Computer Science” presso l'Università del Michigan nel 1971.

Dal 1974 inizia a lavorare al CERN di Ginevra concentrandosi dal 1987 nell'ambito informatico. Nel 1989, indipendentemente da Berners-Lee, propose un sistema di ipertesti per accedere più velocemente alla documentazione del CERN. L'anno successivo iniziò ad interessarsi al progetto dello sviluppo del World Wide Web, assieme a Berners-Lee, con il quale svilupperà il World Wide Web.

Anch'egli riceve una onorificenza nazionale: nel 2004 è nominato dal Re Alberto II di Belgio “Comandante” nell'Ordine di Re Leopoldo.

Riceve anche alcune lauree honoris causa ed altri riconoscimenti ufficiali, taluni assieme a Berners-Lee.

Attualmente si occupa di promuovere l'uso del Web nell'industria e nella didattica.

1.3 Storia

La data di nascita ufficiale del World Wide Web è fissata nel 6 Agosto 1991, quando Tim Berners-Lee pubblicò il primo sito internet.

Già da alcuni anni era allo studio un software per la condivisione, tra i ricercatori del CERN, dei documenti prodotti: questo doveva servir loro per aumentare la collaborazione, sia in termini di comunicazione che di interscambio di materiale documentale.

Nel 1989 Berners-Lee, a quel tempo ricercatore, presentò una relazione intitolata "Information Management: a Proposal", ma questa fu allora sottovalutata dai suoi diretti superiori. Essa conteneva quelle che saranno le basi del WWW.

Assieme all'implementazione del software, in quegli anni iniziò anche la definizione di quegli standard che ancora oggi sono le parti fondamentali del WWW, che sono il linguaggio HTML ed il protocollo di rete HTTP, sulla base dei quali è stato realizzato il primo browser/editor ipertestuale per il WWW. Questi due argomenti saranno esplicitati nei paragrafi successivi di questa tesi.

Per un paio d'anni il WWW fu usato solo dai ricercatori. Il 30 Aprile 1993 il CERN decise di rendere questa tecnologia disponibile a tutti, rinunciando ad ogni diritto d'autore.

Nacque allora la cosiddetta "era del web", che si è sviluppata in maniera sempre più crescente.

Nel 1993 uscì il browser Mosaic, ad opera di Marc Andreessen, che combinava una buona capacità grafica e alcune tecnologie d'interfaccia multimediali. Lì nacque la Mosaic Communications che prese poi il nome di Netscape Communication: essa creò nel 1994, il primo browser commerciale: "Netscape Navigator".

Nel 1995 la Microsoft pubblicò il suo browser: "Internet Explorer", oggi ancora operativo e giunto alla versione 9.

Nel corso degli anni successivi, nell'ambito del web, si sono avute molte evoluzioni, difficili da riassumere in poche righe.

Inizialmente, quindi, il web era stato concepito per la visualizzazione di documenti ipertestuali statici creati in linguaggio HTML. Questo "stato" del web è stato definito, da alcuni studiosi, come "Web 1.0".

In seguito all'integrazione con i database e all'utilizzo di sistemi di gestione dei contenuti, il web si è arricchito di pagine e siti dinamici. Taluni definiscono questo come il "Web 1.5".

Un salto concettuale avviene con quello che è indicato con il termine di "Web 2.0". Con questo termine negli ultimi anni, alcuni ricercatori, indicano uno stato di evoluzione del WWW, nel quale c'è uno spiccato livello di interazione sito-utente. Altri studiosi, scettici nell'uso di questa etichettatura, che ora propende verso il "Web 3.0", hanno iniziato ad utilizzare spesso il termine "Nuovo Web" per indicare dinamiche web future che comprendono, ma vanno oltre, il Web 2.0.

Il Web Semantico dovrebbe essere uno dei punti cardine del Web 3.0.

Nel prossimo capitolo si svilupperanno meglio queste tematiche.

1.4 HTML e XHTML

L'"Hyper Text Markup Language" è un linguaggio che viene utilizzato per descrivere la struttura dei documenti ipertestuali utilizzati nel World Wide Web.

E' stato sviluppato da Tim Berners-Lee ed è tuttora tenuto aggiornato dal World Wide Web Consortium. Dopo molte revisioni e migliorie, si è giunti nel Dicembre 1999 alla versione HTML 4.01.

Il 22 Gennaio 2008 il W3C ha rilasciato una nuova versione, la 5, che attualmente è in stato di bozza (in gergo "draft") e che ha l'obiettivo di assimilare le nuove

tendenze del web. L'HTML Working Group del W3C prevedeva di completare la definizione dell'HTML 5 non prima del 2010.

La maggior parte dei i siti web attuali sono scritti in codice HTML.

L'HTML (tradotto in: linguaggio di marcatura per ipertesti) è un linguaggio di markup, cioè un linguaggio che descrive il contenuto testuale, e non, di una pagina web. Questo codice viene letto dal browser dell'utente che, lo elabora, e genera la pagina che verrà visualizzata sullo schermo.

Il termine "markup", tradotto in marcatura, deriva dall'ambito tipografico dove si marcano con annotazioni le parti di un testo da evidenziare o da correggere.

Un linguaggio di markup descrive i meccanismi di rappresentazione (della struttura e per la rappresentazione vera e propria) del testo utilizzando convenzioni standardizzate, che possono essere utilizzate su più supporti.

La caratteristica fondamentale dell' HTML è che esso è stato concepito per definire il contenuto logico di un documento e non l'aspetto estetico finale; può succedere dunque che uno stesso documento venga visualizzato in modi diversi in due browser diversi (questo succedeva fino a qualche tempo fa anche con browser diffusi quali Internet Explorer e Mozilla Firefox), ma questo fatto ha garantito la massima diffusione di Internet.

Attualmente l'HTML si è evoluto: i documenti sono più ricchi sia dal punto di vista grafico che dal punto di vista dell'interazione con l'utente. Essi inoltre supportano le tecnologie di linguaggi quali Java, JavaScript o CSS, che ne potenziano le funzionalità.

I documenti HTML, riconoscibili dalle estensioni dei nomi in ".html" o ".htm", sono memorizzati su computer costantemente collegati alla rete Internet. Uno specifico software detto "web server" invia, tramite le regole del protocollo HTTP, i documenti ai browser degli utenti che li hanno richiesti. Ricevuto il documento, il browser lo visualizzerà quindi sul monitor.

Un'evoluzione dell'HTML è data dalle "pagine dinamiche", tramite le quali possono essere compiute operazioni più avanzate quali, ad esempio, la ricerca di un libro all'interno di una catalogo on-line di una biblioteca. In questi casi, il documento HTML viene generato da un codice eseguibile, residente sul server web, ed in grado di interagire con altre applicazioni residenti sul server stesso e che, alla fine, invia il risultato finale all'utente. Esempi di ciò sono documenti scritti in linguaggi quali PHP e ASP.

Nell'HTML, dal punto di vista sintattico, il componente principale è l'"elemento", che è la struttura base che deve indicare al browser le informazioni su come vanno formattate le pagine web.

Ogni elemento è a sua volta racchiuso in marcature dette "tag". Un tag è una sequenza di caratteri che è racchiusa tra due parentesi angolari: < e >. Il tag più importante è <a>, che descrive il collegamento ad un altro documento ipertestuale.

Una sezione di testo o di codice deve essere sempre racchiusa tra un "tag di apertura" e un "tag di chiusura", pena una segnalazione d'errore di tipo sintattico.

Un documento HTML è strutturato in tre parti fondamentali: il "Document Type Definition" (o DTD), l'"Header" ed il "Body".

Il DTD contiene le informazioni per il browser relative alla definizione del tipo di

documento HTML. L'Header (sezione d'intestazione del documento) contiene tutte quelle informazioni, non visualizzate dal browser, utili per le applicazioni esterne che accedono alla pagine, quali motori di ricerca, collegamenti a file esterni alla pagina, titolo della pagina ed altre informazioni per lo stile da applicare alla pagina stessa. Il Body (corpo del documento) contiene la parte che verrà visualizzata della pagina e tutte quelle indicazioni per l'aspetto grafico della pagina stessa, quali pulsanti, menù, ecc.

Anche i tag si dividono in due tipi: quelli dell'header e quelli del body, con scopi diversi. Non ci si addentra oltre in dettagli che esulano dagli scopi di questa tesi.

Si rammenta che le tutte le specifiche aggiornate ed ufficiali dell'HTML si trovano sul sito del W3C nelle pagine relative all'HTML.

L'HTML 5, all'inizio del suo sviluppo, si proponeva come una evoluzione della versione HTML 4.01. Era stato concepito per coesistere in modo complementare con l'XHTML 2, un altro linguaggio di markup, che verrà analizzato più avanti.

Le novità dell'HTML 5, rispetto all'HTML 4.01, sono l'introduzione di un nuovo insieme di interfacce di programmazione (API) che estendono le preesistenti e che forniscono funzionalità dedicate:

- alla grafica 2D;
- all'integrazione e al controllo di contenuti audio e video;
- all'archiviazione persistente dei dati sul client;
- all'esecuzione di web application in modalità off-line;
- alla modifica interattiva dei documenti da parte degli utenti;
- alla capacità di supportare anche i dispositivi mobili;

Le migliorie sono rivolte quindi a migliorare il disaccoppiamento tra struttura, che è definita dal markup, caratteristiche di resa quali tipo di carattere, colori, ecc., che sono definite dalle "direttive di stile", e contenuti di una pagina web, che sono definiti dal testo vero e proprio. Vengono rese più stringenti le regole per la strutturazione del testo in capitoli, paragrafi e sezioni.

Vengono introdotti elementi specifici per il controllo di contenuti multimediali, quali tag <video> e <audio> ed elementi di controllo per i menu di navigazione e per i moduli elettronici.

Inoltre HTML 5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricate dal browser, per consentire l'utilizzo di applicazioni basate su web (come per esempio le caselle di posta elettronica o altri servizi simili) anche in modalità off-line (cioè in assenza di collegamento ad Internet).

Infine vengono deprecate (cioè, si fa in modo che una cosa non venga utilizzata od è fortemente sconsigliata nell'uso) od eliminati alcuni elementi che hanno dimostrato scarso o nessun utilizzo effettivo; tutto questo al fine di fare pulizia ed eliminare le cose vecchie ed ormai superate.

Attualmente HTML 5 è in pieno sviluppo. Dalla W3C vengono periodicamente pubblicate le "W3C technical reports" (relazioni tecniche) sullo stato di avanzamento del lavoro, che è curato dal W3C HTML Working Group.

- . - . - . -

L' "eXtensible Hyper Text Markup Language", abbreviato in XHTML, è un linguaggio di marcatura che associa alcune delle proprietà dell'XML con le

caratteristiche dell'HTML. Un documento XHTML è un documento HTML scritto conformemente allo standard XML. L'XML, che verrà analizzato in dettaglio nei prossimi capitoli, è un linguaggio che permette di creare tag personalizzati, permettendo agli utenti di aggiungere una struttura ai loro documenti.

L'XHTML (tradotto in: linguaggio di marcatura di ipertesti estensibile) prevede delle restrizioni nell'uso dei tag del HTML: esso esige una sintassi più corretta relativamente alla parte logica della pagina e obbliga l'uso dei "Cascading Style Sheets", abbreviati in CSS (fogli di stile a cascata), per la grafica.

Questo nuovo linguaggio, ufficializzato nel Gennaio 2000, è nato dall'esigenza di avere una sintassi definita meglio rispetto a quella dell'HTML 4.01: questo è dovuto al fatto che alcuni nuovi dispositivi non hanno tutte quelle risorse hardware e software in grado di interpretare correttamente le specifiche dell'HTML.

Dal punto di vista tecnico, non è altro che una riformulazione dell'HTML 4.01 in XML 1.0.

Esistono tre versioni di XHTML 1.0, ognuna con particolarità proprie:

- XHTML 1.0 Transitional: è una versione nata per favorire il passaggio da vecchie versioni di HTML, quali la 3.2 all'XHTML. Sostiene anche sintassi di tag HTML che non sono validi in XHTML (dette definizioni deprecate);
- XHTML 1.0 Strict: è la versione "di manica stretta" dell'XHTML. Utilizza solo codice che rispetta rigorosamente tutte le regole imposte dallo standard XHTML;
- XHTML 1.0 Frameset: è una versione in abbandono, utilizzata per essere compatibile con la suddivisione delle finestre in sottofinestre dette frame.

Esistono altre versioni particolari del XHTML. Le più note sono:

- XHTML Basic: è una versione semplificata dell'XHTML 1.0, adatta per dispositivi che non supportano tutte i tag e i comandi XHTML;
- XHTML Mobile Profile: sviluppato dalla Nokia. Ha funzionalità adatte all'uso nei telefoni cellulari e/o palmari.

La versione più recente dell'XHTML è la versione 1.1, che non è altro che una versione particolare dell' XHTML 1.0 Strict.

Merita attenzione uno strumento sviluppato dal World Wide Web Consortium per il controllo di qualità di un documento XHTML: il "Markup Validation Service".

Viene definito "well formed" o "documento valido" un documento XHTML che risponde a tutte le specifiche imposte dallo standard: questo al fine di rendere il documento visualizzabile, dal punto di vista teorico, con qualsiasi browser. Il Markup Validation Service è, in pratica, un sito web gestito dal W3C, che permette di eseguire un controllo, detto "validazione", sul contenuto sintattico di un documento XHTML. Questo può essere utile ai programmatori e/o ai webmaster per correggere eventuali errori contenuti nel documento stesso.

Se il documento supera il test, il validatore restituisce all'utente un'icona che può essere inserita nel documento, al fine di affermarne la conformità allo standard.

Il W3C permette anche la validazione dei documenti HTML, dei collegamenti ipertestuali tra documenti, dei fogli di stile CSS, dei feed RSS ^[1] e di altri standard implementati dalla W3C.

XHTML 2 è un linguaggio di progettazione per il web, fino a qualche mese fa ancora in fase di specifica presso il World Wide Web Consortium. Nelle intenzioni degli sviluppatori, doveva rappresentare l'evoluzione dell'XHTML 1.1.

Negli studi, esso doveva caratterizzarsi principalmente per una struttura modulare, in cui le regole per l'impostazione dei vari elementi di una pagina web (suddivisione del testo, moduli elettronici, dati tabellari, ecc.) andranno descritte in moduli separati, in modo da poter evolvere indipendentemente tra loro.

Rispetto alla precedente versione di XHTML, venivano rese più forti le caratteristiche XML per le regole sintattiche. Inoltre venivano approfondite ed estese le possibilità di controllo sui moduli elettronici (form), introducendo strutture più complesse ed elementi di elaborazione.

La definizione di questa versione è proceduta in parallelo a quella di HTML 5, che viene considerato dal World Wide Web Consortium come uno standard di markup complementare rispetto a XHTML 2.

Il 2 luglio 2009 il World Wide Web Consortium ha decretato la cessazione dello sviluppo di XHTML 2, a causa di problemi di compatibilità: quello che si stava ottenendo era un markup completamente nuovo e non un'effettiva evoluzione dell'XHTML 1.1. Tutti coloro che sviluppavano l'XHTML 2 ora stanno indirizzando le risorse a favore dell'HTML 5.

1.5 HTTP e HTTPS

L'“HyperText Transfer Protocol” (acronimo di: protocollo di trasferimento di un ipertesto) è il sistema principale per la trasmissione di informazioni sul web.

E' stato ideato e studiato da Tim Berners-Lee. La prima versione teorica, l'HTTP 0.9, risale al 1989; la prima versione disponibile, implementata sempre da Berners-Lee, è l'HTTP 1.0 del 1991. Successivamente, si apportarono delle modifiche alla versione 1.0 al fine di aumentarne il livello di sicurezza e di implementare l'ospitabilità di più siti web nello stesso server (virtual host). Si arrivò alla versione HTTP 1.1 del 1997, aggiornata poi nel 1999. Attualmente è ancora quella operativa. Poiché sia le estensioni dell'http che l'HTTP 1.1 sono attualmente specificazioni stabili, la W3C ha chiuso l'ambito di ricerca “HTTP Activity”.

Il funzionamento dell'HTTP si basa sul meccanismo client/server, cioè su richiesta/risposta. Il client, che in questo caso è il browser dell'utente, esegue la richiesta tramite un “messaggio richiesta”. Il server, che risponde con un “messaggio risposta”, è il web server.

Questo protocollo di trasmissione ha una proprietà importante: la connessione tra server e client è chiusa una volta che la richiesta sia stata soddisfatta. Questa proprietà è comoda per il web, specialmente nel passaggio da una pagina ad un'altra, pagina quest'ultima che può essere memorizzata su un server diverso. Tuttavia, ha il problema che non si riesce a definire uno “stato” sull'utente: per risolvere ciò, si utilizzano i “cookie”. Tradotti letteralmente in “biscotti”, questi altro non sono che frammenti di testo inviati da un server ad un client/browser e poi rimandati indietro dal client al server, senza modifiche, ogni volta che il client accede allo stesso server. Vengono usati per eseguire autenticazioni e tracking (traccature) di sessioni e/o per memorizzare informazioni specifiche riguardanti gli utenti che accedono al server.

Nell'HTTP il trasferimento delle informazioni tra il mittente ed il destinatario, avviene senza porre in relazione i dati tra le sessioni precedenti e le successive. Questo, in termini pratici, significa minore quantità di dati da trasferire da server a client e dunque permette una maggior velocità.

Il traffico HTTP avviene, in ricezione e trasmissione, mediante il protocollo TCP (Transmission Control Protocol) che, in questo ambito, non verrà approfondito.

L'utilizzo del HTTP all'interno di un browser viene visualizzato dal prefisso "http" posto all'inizio dell'indirizzo del sito che l'utente sta consultando. Ad esempio, <http://www.unipd.it>, per il sito dell'Università di Padova.

Tutto il traffico HTTP ha, inoltre, la caratteristica di essere anonimo ed in chiaro. Questo ha creato la necessità di avere delle caratteristiche di sicurezza quali: una cifratura, una verifica dell'integrità del traffico stesso e un'autenticazione di utente/client e server.

Sono state standardizzate due versioni sicure del protocollo HTTP:

- SHTTP: che scambia in chiaro le richieste tra server e client, crittografando solo il contenuto (body) della pagina. Questa tecnica è oramai in disuso ed è stata soppiantata dalla HTTPS;
- HTTPS: sviluppata dalla Netscape Communication Corporation, usa criteri più sofisticati per impedire l'intercettazione indesiderata. Nelle righe successive, viene trattata in maniera più esauriente.

- . - . - . -

L'HTTPS, acronimo di "HyperText Transfer Protocol over Secure Socket Layer", è una aggiunta al protocollo HTTP di tecniche di sicurezza, per la riservatezza dei dati durante la loro trasmissione nel web.

Si ottiene ciò applicando dei protocolli di crittografia asimmetrica al protocollo HTTP. Questi meccanismi di crittografia/autenticazione sono noti con i nomi di "Secure Sockets Layer (SSL)" e di "Transport Layer Security (TLS)". Non ci si addentrerà in dettagli tecnici. A differenza dell'HTTP i dati transitano attraverso una porta diversa del PC: al posto della porta 80 si utilizza la 443; per l'utilizzatore finale nulla cambia dal punto di vista operativo.

Dal punto di vista pratico, nella comunicazione tra server e client, avviene inizialmente uno scambio di "certificati digitali", in modo tale che il server ed il client siano univocamente determinati e che essi siano i soli a conoscere quello che sarà il contenuto della comunicazione. Questo scambio crea un canale criptato, dove saranno scambiati i dati effettivi, senza la possibilità che un terzo incombodo, chiamato in gergo "man in the middle", possa intercettarli.

L'operazione di criptaggio avviene quando il proprietario di un sito web acquista un certificato da un'autorità di certificazione, come Verisign, o lo genera in proprio. Questo certificato non è altro che un codice di notevole complessità creato ad hoc per uno specifico utente e per uno specifico sito, in una specifica sessione. Oltre il codice vengono acquisite anche informazioni aggiuntive relative alle entità interessate, come ad esempio il nome del server che ospita il sito. Un certificato digitale può essere visto anche come un documento che crea un'associazione tra una persona ed una chiave, qual è nome utente e password.

Un uso del protocollo HTTPS può essere fatto anche all'interno di una rete aziendale (rete intranet) per l'utilizzo della rete stessa da parte dei dipendenti dell'azienda. Altresì l'HTTPS può essere adoperato da un responsabile del web server per restringere gli accessi al server ai soli utenti/clienti abilitati.

L'HTTPS è utilizzato soprattutto nel trasferimento di dati "sensibili", quali posta elettronica (webmail), pagamenti elettronici (e-commerce, home-banking) e nella gestione di archivi di dati che necessitano di protezione e privacy.

L'utilizzo del HTTPS all'interno di un browser viene evidenziato dal prefisso "https" posto sempre all'inizio dell'indirizzo del sito. Ad esempio, la posta elettronica del Dipartimento di Informatica dell'Università di Padova, consultata via web, ha indirizzo: <https://mail.dei.unipd.it>.

Il browser in presenza di una connessione HTTPS, in taluni casi, può visualizzare una finestra con tutti i dati caratteristici della connessione e chiedere all'utente l'accettazione del certificato. Tuttavia, come tutti i sistemi di sicurezza, anche l'HTTPS non può dirsi sicuro al 100%.

Per tutte le specifiche tecniche aggiornate, sia dell'HTTP che dell'HTTPS si deve fare riferimento al sito internet ufficiale del W3C, riportato in bibliografia.

1.6 World Wide Web Consortium (W3C) e W3C Italia (W3C-IT)



Figura 1.1 – Logo W3C

Il "World Wide Web Consortium" (in sigla W3C)^[1.3] è un'associazione fondata nell'Ottobre del 1994 da Tim Berners-Lee, presso il Massachusetts Institute of Technology (MIT), la più importante università di ricerca del mondo, che ha sede a Cambridge negli USA.

Questa associazione ha lo scopo di migliorare gli attuali protocolli e linguaggi per il WWW e di sviluppare tutte le potenzialità del Web. Attualmente (Ottobre 2009) conta 343 membri ed uno staff di 70 persone, sparse nel mondo, che lavora a tempo pieno. 16 sono i membri italiani che lavorano presso il W3C.

Nel corso degli anni, altri istituti si sono associati a quello che, ora, viene chiamato il "Consortio". Tra questi molte importanti aziende informatiche, telefoniche e grosse società, appartenenti a vari settori, interessate allo sviluppo del Web. Inoltre fanno parte del W3C università, istituti di ricerca ed associazioni. Tuttavia, il Consortio non è un organismo di standardizzazione, quale invece l'ISO.

Gli obiettivi primari del W3C sono i seguenti:

- tenere aggiornate le specifiche relative ai linguaggi e gli standard del Web;
- agevolare l'accesso al Web stesso, con un occhio di riguardo alle persone diversamente abili;

- controllare l'assoluta libertà del Web, evitando che essa venga limitata da interessi di qualsiasi genere.

Motto del W3C è infatti: "Leading the Web to Its Full Potential...", cioè "condurre il Web alla sua piena potenzialità...".

I principi strategici del W3C sono riassumibili in 7 punti:

- 1) **Accesso Universale:** Il W3C definisce il Web come "l'universo delle informazioni accessibili in rete". L'obiettivo è quello di rendere queste opportunità fruibili a tutti, indipendentemente da eventuali limitazioni determinate da hardware, software, supporto di rete a disposizione, lingua madre, cultura, collocazione geografica, capacità fisiche e mentali. Alcune attività in opera sono: Internationalization Activity, Device Independence Activity, Voice Browser Activity e Web Accessibility Initiative.
- 2) **Web Semantico:** con l'avvento di questa tipologia di Web, le persone saranno capaci di esprimersi in una maniera tale che i computer potranno interpretare e scambiare informazioni, sia con le persone, che tra computer stessi. Così facendo, i computer avranno la capacità di risolvere alcuni problemi quale, ad esempio, la ricerca nel Web di uno preciso argomento richiesto da un utente. Il linguaggio base del W3C per questo scopo attualmente sono RDF, XML, XML Schema e XML Signatures. Nei capitoli seguenti verranno analizzati più nello specifico.
- 3) **Fiducia:** l'obiettivo di questo punto è la progettazione di un "Web of Trust", cioè di un Web che garantisca riservatezza e fiducia. Chi pubblicherà un documento dovrà "firmare" ciò che ha messo in rete e dovrà meritarsi la fiducia di chi andrà ad utilizzare quel documento. Fiducia intesa anche nel senso che il documento non deve contenere falsità. Nei capitoli seguenti verrà approfondito anche questo punto.
- 4) **Interoperabilità:** fino a qualche anno fa molti utenti utilizzavano software che funzionava solo con altri software dello stesso produttore. Oggi invece si possono far lavorare assieme anche software di case produttrici diverse. Gli utenti hanno anche altre esigenze quali, ad esempio, la visualizzazione di pagine web su dispositivi diversi, quali pocket PC, telefoni cellulari, ecc. Il Consorzio è impegnato nel promuovere l'interoperabilità tra piattaforme diverse, tramite la progettazione e la promozione di linguaggi e di standard "aperti", che permettono a tutti di utilizzarli al meglio e di proporre modifiche.
- 5) **Capacità evolutiva:** il W3C si adopera, inoltre, anche per costruire un Web che possa facilmente evolvere in un Web ancora migliore. I principi che stanno alla base di questa evoluzione sono quelli della semplicità, modularità, compatibilità ed estensibilità.
- 6) **Decentralizzazione:** la decentralizzazione è un principio fondamentale dei moderni sistemi distribuiti. Il progetto del Consorzio è quello di limitare il numero delle risorse Web centralizzate, al fine di ridurre la vulnerabilità del Web nel suo complesso.
- 7) **Multimedia:** il lavoro del W3C spazia anche nell'ambito della multimedialità. Alcuni componenti dello staff stanno sviluppando un "Cooler Web", tradotto in "Web più eccitante", grazie a linguaggi come SVG (Scalable Vector Graphics) e SMIL (Synchronized Multimedia Integration Language) che aumentano e migliorano le capacità multimediali e di interattività del Web.

Il World Wide Web Consortium è organizzato in tre gruppi di persone:

- l'Advisory Comitee: è composto da una persona per ogni società facente parte del consorzio. E' la mente dell'organizzazione. Il suo ruolo principale è quello di rivedere i piani del W3C e di effettuare revisioni delle tecnologie esistenti.
- il W3C Team: è formato da professionisti pagati e da membri delle società associate.
Sono le cosiddette "braccia" dell'organizzazione. E' suddiviso ulteriormente per gruppi (Active Group), ognuno dei quali è focalizzato nello studio ed implementazione di un argomento specifico. Ad esempio, il Semantic Web Coordination Group.
- L'Advisory Board: formata da nove membri, in carica due anni. Ha il ruolo di consigliere e si occupa dell'ambito legale del Consorzio. E' presieduto da una persona chiamata Chairman.

A capo del Consorzio c'è un Director: attualmente è lo stesso Tim Berners-Lee.

Le attività del W3C sono divise in 4 aree dette "domini":

- l'Architecture Domain gestisce la tecnologia che è alla base del Web;
- l'Interaction Domain semplifica l'interazione uomo-informazioni e la maniera di connettersi al Web
- il Technology and Society Domain adatta l'infrastruttura tecnologica agli interessi sociali, legali e pubblici;
- il Web Accessibility Initiative (WAI) garantisce l'accessibilità di tutti al Web.

Un'attenzione particolare meritano le cosiddette "raccomandazioni" (Recommendations). Esse non sono altro che il risultato di finale, in pratica un documento che viene pubblicato, del processo di definizione di uno standard da parte del W3C. Questa definizione, passa attraverso 4 stadi: il Working Draft (la bozza), il Last Call (l'ultimo appello), la Proposed Recommendation (la proposta di raccomandazione) ed infine la Candidate Recommendation (la raccomandazione candidata). Una volta pubblicata, una raccomandazione può essere corretta tramite documenti di "Errata" oppure ritirata, modificata e pubblicata nuovamente. Esistono anche le "Note Informative", pubblicazioni ma da considerarsi non come standard.

Finora il W3C ha proposto, discusso, definito ed ufficializzato oltre 50 standard industriali. Alcuni di questi sono già stati menzionati. Altri, relativi al Web Semantico^[1.4], saranno sviluppati nei prossimi capitoli. Per una lista completa si può fare riferimento al sito della W3C, riportato in bibliografia.

Molti di questi standard sono stati adottati in ambito internazionale. Altri, tuttavia, non sono riusciti ad imporsi, per vari motivi. Ad esempio, XML è utilizzato moltissimo, mentre il passaggio dei documenti da HTML a XHTML va a rilento.

- . - . - . -



Figura 1.2 – Logo W3C - IT

L'Ufficio W3C Italiano (W3C-IT)^[1.5] è il punto di contatto nazionale per le attività W3C in Italia.

E' attivo oramai da dieci anni. Ha sede presso l'Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" (ISTI) del C.N.R. a Pisa.

Si occupa anche di organizzare convegni e seminari. Si segnalano i due più recenti tenuti nell'Ottobre 2009:

- In occasione del decennale della sua costituzione, l'Ufficio ha organizzato un incontro sul tema: “La dimensione sociale del Web”. Esperti del W3C, del mondo della ricerca e delle imprese hanno illustrato come le tecnologie del W3C intendono affrontare i problemi sollevati dall'esplosione del “social web”, analizzato fenomeni emergenti come il crowdsourcing^[2] e discusso di come possono essere mutate le prospettive e le opportunità di sviluppo negli ultimi dieci anni e di cosa si può immaginare per il futuro;
- “Internet Governance Forum Italia 2009”. Il forum, rappresenta una delle più importanti occasioni d'incontro dell'intera comunità Internet italiana e risponde all'invito del Parlamento Europeo di promuovere iniziative nazionali analoghe agli appuntamenti globali.



Figura 1.3 – Logo IGFI 2009

In definitiva, il W3C può essere considerato come un ambiente che favorisce il raggiungimento di compromessi utili per la crescita del Web.

NOTE CAPITOLO 1

1. **Feed RSS:** il termine (o feed web) indica un'unità di informazioni formattata secondo determinate specifiche per rendere interoperabile ed interscambiabile il contenuto fra diverse applicazioni o piattaforme. E' usato per fornire agli utenti una serie di contenuti aggiornati di frequente dai distributori, previa iscrizione. Queste informazioni possono essere visualizzate dall'utente nella stessa finestra, senza dover accedere ogni volta al sito principale.

2. **Crowdsourcing:** il termine (da *crowd* cioè gente comune + *outsourcing* cioè esternalizzare una parte delle proprie attività) definisce un modello di business nel quale un'azienda o un'istituzione richiede lo sviluppo di un progetto, di un servizio o di un prodotto ad un insieme distribuito di persone, non già organizzate in un team. Tale processo avviene attraverso degli strumenti web o dei portali su internet.

Capitolo 2

Introduzione al Web Semantico

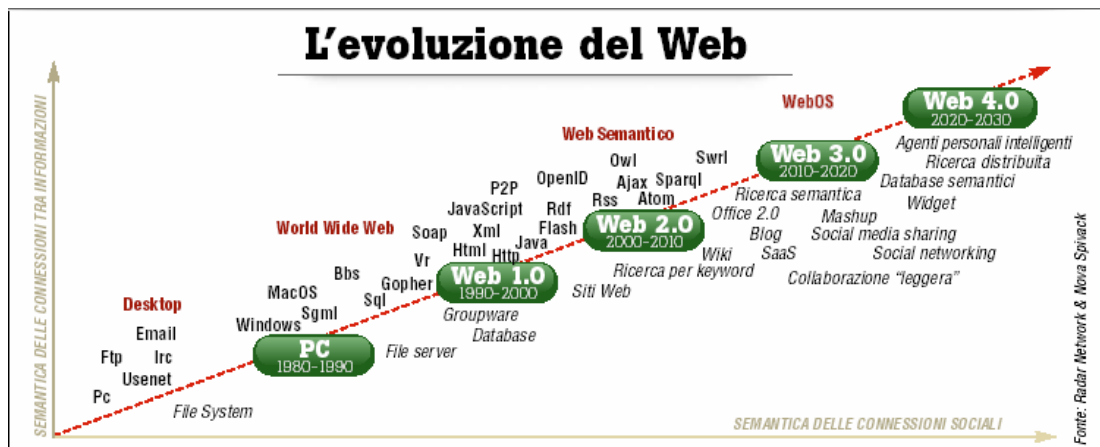


Figura 2.1 - L'evoluzione del Web

Dalle prime pagine web, pubblicate nella prima metà degli anni '90 e scritte con le prime versioni dell'HTML, a quelle pubblicate ora, molta strada è stata fatta.

Nelle prossime pagine verranno descritte le principali evoluzioni tecniche che, nel corso degli anni, hanno portato a quello che è il Web attuale.

Nel primo paragrafo verrà posta l'attenzione sugli sviluppi subiti dai linguaggi che gravitano attorno al Web. Una trattazione completa e approfondita di tutto ciò, tuttavia, esula da questa tesi, dato che lo scopo principale è quello di far capire al lettore a che punto è arrivato lo sviluppo tecnico del Web. Verranno poi presentati i Web Services, che si sono evoluti nel corso degli anni e ai quali si stanno applicando le teorie semantiche. Nel secondo paragrafo, invece, ci si addenterà in un altro filone dell'evoluzione del Web, il Web Semantico, che è il vero scopo del lavoro di questa tesi. Nel terzo paragrafo, infine, si illustrerà la W3C Semantic Web Activity, ovvero chi sono e come sono organizzati coloro che si occupano dello sviluppo del Web Semantico all'interno del W3C.

2.1 Dal web statico al web dinamico. Web Services

Le prime pagine web pubblicate in rete avevano la caratteristica principale della "staticità": esse venivano riprodotte dal browser dell'utente (indicato in seguito anche come client) esattamente come erano state create dall'autore. All'utente, che ne usufruiva, era consentito solo la lettura, la stampa, la possibilità di copiarne delle parti ed il salvataggio. Un aggiornamento del loro contenuto era possibile solo da parte di coloro che le avevano create e che avevano accesso al server dove erano memorizzate. Per fare queste modifiche, talvolta, era necessaria una conoscenza

tecnica particolare e l'uso di applicativi software specifici, cose che erano fattibili solamente da poche persone.

Questa limitazione iniziale fu, in seguito, risolta dagli sviluppatori del Web con l'introduzione delle "pagine HTML dinamiche", il cui contenuto si adatta a ciò che deve essere rappresentato, e cioè alle effettive richieste ed esigenze dell'utente. Si consideri, ad esempio, la consultazione di un archivio (database) che è disponibile on-line: l'utente specifica in una pagina web i parametri della ricerca che intende fare sui dati contenuti nel database: imposta quindi quella che sarà la query. Una volta che la ricerca è stata completata, nel documento ottenuto come risposta, che altro non è che la pagina che verrà visualizzata, vengono inseriti solo i dati di quelle entità che soddisfano i parametri della ricerca. La pagina web varia quindi al variare dei risultati della query: di qui l'attributo di "dinamiche" alle pagine.

Una delle prime implementazioni web "dinamiche" è stata la "Common Gateway Interface" (in sigla CGI): attraverso questa tecnologia si può richiedere ad un server web di interfacciarsi e di utilizzare un'applicazione esterna. Nell'esempio di prima, il server web utilizza uno specifico programma per eseguire la query e, una volta ottenuti i risultati, li manda al browser dell'utente che, a sua volta, li visualizza come una normale pagina web statica. L'utente, dal suo punto di vista, non si accorge di tutto ciò.

In termini tecnici, ogni volta che un client richiede al server web un indirizzo internet, se questo corrisponde ad un documento scritto in puro HTML, gli viene restituito il documento statico (la pagina); se invece l'indirizzo corrisponde ad un programma CGI, il server lo esegue in tempo reale, generando dinamicamente le informazioni che saranno inviate al browser per essere pubblicate. La tecnologia CGI ha, tuttavia, dei problemi: il programma deve essere eseguito ogni volta che viene richiesto dall'utente e non viene tenuta traccia di altre sessioni precedenti.

Il programma (o uno script) CGI può essere scritto in qualsiasi linguaggio di programmazione (C/C++, Perl, PHP, Visual Basic, Tcl/Tk, AppleScript, Suneido, ecc.): la scelta di ciò si basa sul sistema su sarà fatto girare. I più comunemente utilizzati sono PHP e ASP.

- • -

Un altro avanzamento tecnologico si è ottenuto con l'utilizzo delle "ASP", sviluppate dalla Microsoft.

Le ASP (Active Server Pages) sono pagine web contenenti, oltre al puro codice HTML, degli script che verranno eseguiti dal server per generare, durante l'esecuzione, il codice HTML da inviare al browser dell'utente. Così facendo è possibile mostrare contenuti dinamici (ad esempio estratti da database che risiedono sul server web) e modificarne l'aspetto secondo le regole programmate negli script, il tutto senza dover inviare il codice del programma all'utente finale, al quale va inviato solo il risultato, con un notevole risparmio di tempi e di banda: per questo motivo le ASP vengono generalmente definite "pagine web dinamiche".

I linguaggi utilizzati, per l'ambiente ASP, sono VBScript e Jscript. E' grazie a questi linguaggi che il sistema dinamico può comunicare, dal lato server, con tutti gli oggetti presenti sul sistema: le possibilità offerte dal sistema sono fortemente orientate verso l'interfacciamento con un corrispondente database, rendendo così possibile lo sviluppo di siti dinamici basati sulle informazioni contenute nel database

stesso. È possibile interfacciare le pagine ASP con diversi tipi di database, quali ad esempio Access, SQL Server, MySQL, Oracle, Firebird, Sybase, ecc.

Una caratteristica, molto apprezzata dai programmatori che utilizzano l'interprete ASP, è la semplicità e la comprensibilità della sintassi di programmazione; l'interprete ASP, tuttavia, presenta alcuni limiti, specialmente nelle prestazioni.

ASP, oramai, è stato ufficialmente abbandonato, anche se continua ancora ad essere supportato.

- • -

Principale concorrente di ASP è il "PHP", che funziona in modo molto simile al precedente, ma con una sintassi del tutto diversa.

Il PHP (acronimo ricorsivo di "Hypertext Preprocessor" e tradotto in "preprocessore di ipertesti") è un linguaggio di scripting per la realizzazione di pagine web dinamiche. Attualmente è utilizzato principalmente per sviluppare applicazioni web eseguite lato server, ma può essere usato anche per eseguire script tramite interfaccia testuale o sviluppare applicazioni stand-alone.

E' nato nel 1994 con il nome originario di "Personal Home Page" e conteneva, inizialmente, una raccolta di script CGI per la gestione delle pagine web. Negli anni successivi venne esteso con l'aggiunta di funzioni per l'integrazione con il DBMS mSQL. Queste estensioni permettevano inoltre di integrare pezzi di codice PHP nel linguaggio HTML per la realizzazione della dinamicità delle pagine web. Negli anni successivi il PHP fu sviluppato al punto da iniziare a competere con l'ASP.

La popolarità e l'uso del linguaggio PHP sono in costante crescita, soprattutto grazie alla sua flessibilità: nel Giugno 2001 oltre un milione di siti erano, almeno in parte, scritti in PHP. Nel gennaio 2005 è stato insignito del titolo di "Programming Language of 2004" dal TIOBE Programming Community Index^[2.1], classifica che valuta la popolarità dei linguaggi di programmazione sulla base delle informazioni raccolte dai motori di ricerca. Dal punto di vista tecnico, esso è multiplatforma ed ha licenza open-source parzialmente libera.

Il PHP ora è in grado di interfacciarsi ad innumerevoli database, tra cui si citano MySQL, PostgreSQL, Oracle, Firebird, IBM DB2, Microsoft SQL Server; esso, inoltre, supporta numerosi altri tipi di tecnologia, come XML, SOAP, IMAP, FTP. Si integra bene anche con altri linguaggi/piattaforme quali Java e .NET e si può dire che esista un wrapper (involucro) per ogni libreria esistente, come CURL, GD, Gettext, GMP, Ming, OpenSSL, ecc. Esso fornisce un'interfaccia di programmazione specifica per interagire con Apache (attualmente la più diffusa piattaforma server web), nonostante funzioni naturalmente anche con altri numerosi server web. È, inoltre, ottimamente integrato con il database MySQL, per il quale possiede più di una API (interfaccia di programmazione). Per questo motivo esiste un'enorme quantità di script e librerie in PHP, che sono disponibili liberamente su Internet.

- • -

Come già visto nel capitolo precedente, un ulteriore sviluppo del Web è stato fatto con l'evoluzione del linguaggio HTML, che ha potenziato e migliorato i contenuti delle pagine web, come pure le capacità di visualizzazione dei browser stessi. Lo sviluppo delle specifiche dello standard HTML è sempre allo studio da parte dei ricercatori del W3C.

Un altro insieme di tecnologie che permettono di cambiare in modo dinamico la rappresentazione ed il contenuto di un documento, oltre ad aumentare l'interattività dell'utente sulla pagina è il "DHTML".

Il DHTML, acronimo di Dynamic HTML, e conosciuto anche come "HTML dinamico", non è un vero e proprio linguaggio, ma è una sorta di contenitore di script a cavallo tra il JavaScript, l'HTML e il CSS. Gli elementi, gli attributi e gli stili del DHTML sono basati sull'HTML esistente e sulle specifiche classiche del W3C.

Le caratteristiche più interessanti di questa tecnologia sono: la dinamicità degli stili e dei contenuti, il posizionamento e le animazioni sugli elementi. C'è inoltre la possibilità di fare il download di font che non sono presenti su di una macchina, il data binding (una tecnica che lega due sorgenti di dati/informazioni insieme e che le mantiene sincronizzate) ed un accesso facilitato al DOM (Document Object Model). Ad esempio, tramite il DHTML è possibile fare cambiare il CSS (foglio di stile) di un oggetto in modo dinamico.

Non ci si sofferma ulteriormente su questo ambito, in quanto già visto nel capitolo precedente e non facente parte degli scopi principali di questa tesi.

- • -

Un'altra tecnologia che ha dato linfa vitale al Web è stata la possibilità d'interpretazione, da parte dei browser, dei linguaggi di scripting quali il JavaScript.

Un linguaggio di scripting è un linguaggio di programmazione interpretato (cioè che non viene compilato) che è destinato, in genere, a compiti di automazione del sistema (batch) o delle applicazioni (macro) o ad essere usato all'interno delle pagine web.

La differenza rispetto ai linguaggi interpretati veri e propri è data più che altro da ragioni "storiche". I primi linguaggi di scripting erano molto rudimentali, permettevano poche e semplici operazioni e non erano adatti alla scrittura di veri programmi. Oggi, che i linguaggi di scripting hanno una potenza equivalente agli altri linguaggi, la distinzione resta solo in base all'uso che si fa del linguaggio.

La caratteristica principale dei linguaggi di scripting sta nel fatto che il programmatore, generalmente, si può disinteressare delle risorse di sistema, che il programma finito dovrà utilizzare, demandando il tutto al sistema stesso. Per risorse si intendono, per esempio, la gestione della allocazione e deallocazione della memoria, la conversione tra tipi, l'inizializzazione e la chiusura dell'applicazione, ecc. Così facendo si evitano molti problemi tipici della programmazione tradizionale, che risulta essere soggetta ad errori non facilmente individuabili e che, inoltre, costringe il programmatore ad occuparsi di problematiche non strettamente connesse con l'obiettivo che il software deve raggiungere. L'utilizzo di un linguaggio di scripting permette quindi al programmatore di concentrarsi direttamente sulla soluzione del problema.

Esempi di linguaggi di scripting sono Perl, JavaScript, VBScript, Shell scripting (Unix), PHP, Python e Ruby.

Uno dei linguaggi più comunemente utilizzati nei siti web, per l'arricchimento delle pagine, è il JavaScript, orientato agli oggetti, oramai giunto alla versione 2.1 del 2009 e classificato come uno standard ISO. E' prodotto dalla Netscape Communication. Esiste anche il linguaggio JScript, compatibile con JavaScript, prodotto dalla Microsoft, sviluppato per contrastare il successo di quest'ultimo.

Il linguaggio JavaScript non deve essere confuso, avendo il nome molto simile, con il linguaggio Java, che è prodotto dalla Sun Microsystem. I due linguaggi hanno una sintassi simile, che deriva dal linguaggio C, ma una semantica diversa, soprattutto negli “object model”, che sono ampiamente incompatibili.

JavaScript, come già anticipato, ha la caratteristica principale di essere un linguaggio interpretato: il codice non è compilato, ma c'è un interprete che esegue riga per riga, durante l'esecuzione, quanto scritto nello script. L'interprete in JavaScript, dal lato client, è incluso direttamente nel browser dell'utente. Il JavaScript ha, inoltre, la caratteristica di un normale linguaggio interpretato con una sintassi analoga a quella di un linguaggio compilato, quindi con la possibilità di utilizzare funzionalità tipiche dei linguaggi di programmazione ad alto livello. E' in grado anche di definire strutture complesse molto simili a quelle dei linguaggi orientati agli oggetti.

Il codice JavaScript viene eseguito sul client (cioè dal browser dell'utente): questo ha il vantaggio di non sovraccaricare il server con l'esecuzione di script complessi, ma ha lo svantaggio che, nel caso di script che hanno una grosse mole di dati, rende lento lo scaricamento. Un ulteriore limite di questa tecnologia si avverte nel caso di utilizzo di dati memorizzati in un database: per questo accesso è necessario un ulteriore linguaggio che effettua la transizione, restituendo i risultati in variabili JavaScript. Per ovviare a questo problema è stato introdotto l'AJAX, che sarà illustrato nel prossimo paragrafo.

Il JavaScript è un linguaggio di programmazione che non può essere espanso, in quanto utilizza parole chiave riservate e dato che viene eseguito direttamente dal codice sorgente. Esso non ha costrutti propri di input/output: deve quindi basarsi su di un “programma ospite”, nel quale è integrato. Molti sono i programmi e/o le applicazioni che possono integrare al loro interno un interprete JavaScript. Gli esempi più conosciuti vengono dal Web, nel quale esso è integrato direttamente nel browser dell'utente: l'interfacciamento è fatto tramite i DOM (Document Object Model). In questo caso il JavaScript è utilizzato per la scrittura di piccole funzioni integrate nelle pagine HTML al fine di eseguire azioni che non potrebbero essere svolte con l'HTML statico: un esempio di ciò è il controllo dei dati inseriti in un campo di input (form) di una pagina. Esso, quindi, viene utilizzato per la gestione di ogni aspetto dello scripting di un browser web: esempio di ciò è la creazione delle finestre di pop-up (spesso a scopo pubblicitario) che appaiono nello schermo dell'utente durante la navigazione in alcuni siti web.

Altro esempio di utilizzo di JavaScript sono i “Bookmarklet”. Un Bookmarklet è un piccolo programma JavaScript che può essere memorizzato come un normale URL all'interno dei segnalibri (bookmark) nei browser più popolari, o all'interno degli collegamenti ipertestuali di una pagina web. I bookmarklet sono chiamati anche Favlets (o Favelets) per via del fatto che il browser Internet Explorer utilizza il termine "Favorites" (preferiti) per indicare appunto i segnalibri.

Il browser dell'utente che analizza l'URL, al fine di caricare la pagina, nell'analizzare la prima parte dell'URL stesso si accorge, tramite il codice `javascript:`, che non deve utilizzare il resto dell'URL per caricare la pagina, ma che deve eseguire la stringa come un programma ed usare la stringa risultante come link alla nuova pagina da aprire. Lo script eseguito ha accesso alla pagina attuale, può leggerla e modificarla. Se l'operazione ritorna un tipo indefinito piuttosto che una stringa, non

viene caricata nessuna nuova pagina e il solo effetto che si ottiene è quello di eseguire il codice sulla pagina attuale.

Anche in ambiti esterni al Web, gli interpreti JavaScript sono integrati in alcune applicazioni molto note. Si segnalano l'Adobe Acrobat e l'Acrobat Reader nella gestione dei file di tipo PDF.

- • -

L'AJAX, acronimo di "Asynchronous JavaScript and XML", è una tecnica di sviluppo per la realizzazione di applicazioni web interattive, chiamate in gergo tecnico "Rich Internet Application". E' inoltre multi-piattaforma, visto che è utilizzabile su molti sistemi operativi, architetture informatiche e su diversi browser. Ne esistono numerose implementazioni libere (open-source), sia in formato di librerie che di strutture (framework).

Lo sviluppo di applicazioni HTML tramite l'AJAX si basa su di uno scambio di dati, svolto in background, fra il browser del client ed il server, scambio che consente l'aggiornamento dinamico di una pagina web senza l'esplicito ricaricamento da parte del browser. L'AJAX è asincrono, nel senso che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente. Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript.

E' da far notare che, nonostante il nome di AJAX, l'uso di JavaScript e di XML non è obbligatorio, come non è necessario che le richieste di caricamento debbano essere necessariamente asincrone.

La tecnica AJAX utilizza una combinazione di vari componenti:

- HTML (o XHTML) e CSS, per il markup e lo stile;
- DOM, elaborato attraverso un linguaggio quale JavaScript o JScript per mostrare le informazioni ed interagirvi;
- l'oggetto, chiamato "XMLHttpRequest", per l'interscambio asincrono dei dati tra il browser dell'utente e il web server;
- XML, usato, generalmente, come formato di scambio dei dati anche se, di fatto, qualunque formato può essere utilizzato, incluso il testo semplice e l'HTML preformattato.

Come il DHTML, che è stato presentato prima, anche l'AJAX non è una tecnologia individuale, ma piuttosto è un certo numero di tecnologie utilizzate insieme.

Le applicazioni web che usano l'AJAX richiedono dei browser che siano in grado di supportare le tecnologie necessarie, quali quelle che sono state illustrate qui sopra. Esempi di browser che includono queste funzionalità sono: Mozilla Firefox, Konqueror, Safari, Internet Explorer, Chrome ed Opera.

- • - • -

Un'altra strada seguita dagli sviluppatori per aumentare l'interattività e la dinamicità del Web è stata quella di potenziare i Web Server, integrandoli con linguaggi quali JSP, PHP, ASP. Questi linguaggi trasformano i web server in quelli che ora sono chiamati "Application Server".

L'utilizzo del Web come piattaforma applicativa ha, quindi, trovato un suo sviluppo nei "Web Service", (anche detti webservice) il cui scopo è limitare al massimo le implementazioni, dando la possibilità di accedere a servizi software che sono

disponibili in rete. Questi servizi sono utilizzati solamente per lo scopo dell'utente e possono essere pagati solamente per il loro uso effettivo. In gergo informatico sono indicati con termini del tipo: "on demand software", "pay per use", ecc.

Un Web Service (tradotto in servizio sul Web e sintetizzato in WS, dove non sia fonte di confusione con l'abbreviazione in italiano di Web Semantico) è, nella definizione data dalla W3C, "un sistema software progettato per supportare l'interoperabilità tra diversi elaboratori su una rete". La sua caratteristica fondamentale è data dalla presenza di una interfaccia software, utilizzando la quale gli altri sistemi possono interagire con il web service stesso, in maniera automatica, per attivare quelle operazioni che il web service rende disponibili e che sono presentate nell'interfaccia stessa.

L'uso nei web service di standard "aperti" rende possibile l'interoperabilità tra diversi software e tra diversi hardware. Così facendo, le applicazioni software, scritte in diversi linguaggi di programmazione ed implementate su diverse piattaforme hardware, possono essere utilizzate per lo scambio di informazioni e per l'effettuazione di operazioni complesse, sia su reti aziendali che sulla rete Internet. Un esempio di questi standard utilizzati è la formattazione, mediante tag XML, di tutti i dati scambiati.

L'interfaccia, che il web service rende pubblica, è descritta tramite il "Web Services Description Language" (WSDL), un linguaggio, basato su XML, che è usato per la creazione di "documenti" descrittivi delle modalità di uso ed interfacciamento del WS: questa interfaccia è descritta in un formato "machine-processable", cioè capibile ed utilizzabile dai sistemi, senza bisogno dell'uomo.

Altri sistemi possono quindi interagire con il web service nella maniera stabilita dalla sua descrizione usando dei "messaggi SOAP (Simple Object Access Protocol)", che sono trasferiti tra il richiedente ed il fornitore del servizio usando il protocollo HTTP e con una formattazione dei dati di tipo XML. Nell'uso comune, il termine WS si riferisce spesso anche a client e a server che comunicano tra loro attraverso il protocollo HTTP.

Attualmente, i web service sono solamente interfacce di programmazione di applicazioni (API), dette anche "web APIs", che possono essere accedute attraverso una rete ed eseguite direttamente nel sistema remoto (host), dove sono ospitati i servizi richiesti. Quando sono usate nel contesto del web, le web API sono, tipicamente, un insieme definito di messaggi di richiesta assieme ad una definizione della struttura del messaggio di risposta che, solitamente, viene espressa o in formato XML o in formato JSON (JavaScript Object Notation).

I Web Service possono essere suddivisi in due categorie: i "Big Web Services" e i "RESTful Web Services".

I Big Web Services usano messaggi formattati in XML e che utilizzano lo standard SOAP; sono diventati popolari grazie al loro uso in ambito aziendale. In tali sistemi, la descrizione delle operazioni offerte dal servizio è scritta in WSDL.

I Web Services *REpresentational State Transfer (RESTful)*, ultimamente, stanno riguadagnando popolarità, particolarmente nelle aziende/organizzazioni che sono attive su Internet. Utilizzando i metodi PUT, GET and DELETE HTTP, insieme a POST, questi WS sono riusciti ad integrare meglio i loro servizi con l'HTTP e con i browser, più che i WS i cui servizi sono basati sul SOAP. I REST WS, quindi, non richiedono messaggi formattati XML o definizioni in WSDL.

Un web API è, in definitiva, uno sviluppo dei servizi web, dove l'enfasi è stata spostata dai servizi basati sul Simple Object Access Protocol (SOAP) verso comunicazioni più dirette di tipo Representational State Transfer (REST). Queste permettono la combinazione di servizi web multipli in una nuova applicazione conosciuta come “mashups”^[1].

I WS sono, come accennato, un insieme di strumenti che possono essere utilizzati in vario modo. Le modalità principali sono “RPC” (Remote Procedure Calls), “SOA” (Service-Oriented Architecture) e “REST” (REpresentational State Transfer):

- i servizi web di tipo RPC presentano un'interfaccia di chiamata delle funzioni distribuite che, nel corso degli anni, è divenuta familiare a molti programmatori. Questo servizio si basa sul WSDL. I primi WS erano basati sull'RPC e perciò questa tipologia è stata largamente sviluppata e supportata. Tuttavia l'RPC è talvolta criticato in quanto è implementato direttamente nel linguaggio specifico delle chiamate delle funzioni o dei metodi. Alcuni programmatori/distributori ritengono che, oramai, questo approccio sia arrivato ad un punto morto e, quindi, fanno pressioni affinché venga abbandonato;
- i WS possono anche essere utilizzati per implementare un'architettura in accordo con i concetti del SOA, dove l'unità base della comunicazione è un messaggio, piuttosto che un'operazione. Questi services sono spesso indicati come servizi “orientati ai messaggi”. I web services di tipo SOA sono supportati dalle principali case software. Al contrario dei WS RPC, qui un accoppiamento client-server meno rigido è più fattibile, dato che in questo tipo di WS l'attenzione è sul “contratto” che il WSDL fornisce, piuttosto che sui dettagli alla base dell'implementazione;
- il REpresentational State Transfer (REST) cerca di descrivere le architetture che usano l'HTTP, o protocolli simili, costringendo l'interfaccia ad essere un insieme di operazioni ben conosciute e standard. Qui l'attenzione si concentra sull'interazione tra le risorse, piuttosto che sui messaggi o sulle operazioni. Un'architettura basata sul REST può usare il WSDL per descrivere i messaggi di tipo SOAP; tuttavia, questa architettura può essere implementata semplicemente o come un'astrazione del SOAP oppure può essere creata senza l'utilizzo del SOAP.

I WS possono essere sviluppati fondamentalmente in due modi: 1) usando il metodo “bottom-up”, cioè scrivendo prima le classi utilizzate in un linguaggio di programmazione e poi usando uno strumento di generazione WSDL per esporre i suoi metodi come un servizio web; 2) usando il metodo “top-down”, cioè scrivendo prima il documento in WSDL e poi usando uno strumento generatore di codice per produrre lo scheletro delle classi, che saranno completate più tardi. Quest'ultimo metodo è più complicato rispetto al bottom-up, ma produce schemi più chiari.

L'uso dei WS, come tutte le cose, presenta dei vantaggi e degli svantaggi.

Uno dei vantaggi è dato dalla centralizzazione e dalla localizzazione dei WS stessi in un “registro” comune, che permette un reperimento veloce dei servizi offerti che sono presenti in rete. Un altro vantaggio viene dal fatto che i WS possono riutilizzare infrastrutture ed applicazioni già sviluppate o combinare assieme i servizi per crearne altri più potenti. Altro lato positivo dei WS è il “disaccoppiamento” che c'è tra l'utente ed il web service stesso: modifiche da una o entrambe le parti possono essere fatte in maniera trasparente all'altra parte; questo fatto implica una buona flessibilità.

Svantaggi nell'uso dei WS, rispetto ad altri sistemi di “distributed computing”, vengono dalle minori performance che si ottengono, dovute all'uso dell'XML come formato dei messaggi, al SOAP che li racchiude e all'HTTP che li trasporta. Inoltre, anche l'ambito della sicurezza non è ancora ad un livello soddisfacente, dato che i WS per essere operativi talvolta bypassano le protezioni offerte dai firewall: si sta perfezionando sempre più questa tecnologia.

Quando molteplici WS sono in esecuzione, ciascun sub-service (o singolo servizio) può essere considerato autonomo: l'utente non ha il controllo su questi servizi. I WS, inoltre non sono affidabili: i fornitori del servizio possono rimuovere, cambiare o aggiornare i loro servizi senza che sia data comunicazione all'utente. L'affidabilità e la capacità del sistema di continuare a funzionare anche in presenza di guasti non sono ancora ben supportati: errori ed malfunzionamenti possono accadere durante l'utilizzo. La gestione dell'eccezione, nel contesto dei web services, è un ambito di ricerca ancora aperto.

Dal punto di vista operativo, esistono degli strumenti per i WS che facilitano la creazione di nuove interfacce: questi mezzi possono essere visti come un vantaggio per i programmatori e per gli utenti. Questo però può creare dei problemi: ad esempio, un cambiamento, anche secondario, sul server può dare come risultato un diverso WSDL e una diversa interfaccia di servizio, che potrebbero non essere più interpretabili ed utilizzabili dal lato client.

Alcune critiche vengono fatte ai WS di tipo non-REST, in quanto questi ultimi risultano, talvolta, molto complessi e basati su formati proprietari, piuttosto che essere implementati in formato open-source: esistono comunque anche implementazioni di quest'ultimo tipo.

Il consorzio W3C e il consorzio OASIS (Organization for the Advancement of Structured Information Standard)^[2.2] sono, attualmente, tra i principali responsabili dello sviluppo e dell'implementazione degli standard dei WS.

L'OASIS ha standardizzato molte estensioni dei WS, inclusi la Web Service Resource Framework e il WSDM. Il WSDM (Web Services Distributed Management) è un web service standard per la gestione ed il monitoraggio dello stato di altri service.

Il W3C Web Services Activity^[2.3] sta, già da tempo, progettando l'infrastruttura, definendo l'architettura e creando il “nucleo” tecnologico per i WS.

In questi anni, i vari WS Working Group hanno sviluppato e pubblicato le raccomandazioni:

- XML Protocol Working Group: SOAP Version 1.2, XML-binary Optimized Packaging, SOAP MTON (Message Transmission Optimization Mechanism), Resource Representation SOAP Header Block.
- Web Services Description Working Group: Web Services Description Language (WSDL) Version 2.0.
- Web Services Addressing Working Group: Web Services Addressing 1.0.
- Semantic Annotations for WSDL Working Group: Semantic Annotations for WSDL and XML Schema.
- Web Services Policy Working Group: Web Services Policy Version 1.5.

Il SOAP JMS Binding Working Group ha pubblicato una raccomandazione candida del SOAP over Java Message Service 1.0.

Negli ultimi tempi il Web Services Resource Access Working Group ha pubblicato alcune Working Drafts (bozze) relative a:

- Web Services Transfer (WS-Transfer);
- Web Services Resource Transfer (WS-RT);
- Web Services Metadata Exchange (WS-MetadataExchange);
- Web Services Eventing (WS-Eventing);
- Web Services Enumeration (WS-Enumeration).

Il 10 Luglio 2009 sono stati chiusi i gruppi di lavoro: “XML Schema Patterns for Databinding Working Group”, il “Web Services Choreography Working Group” e l’”XML Protocol Working Group”.

Oltre a quelli illustrati finora, esistono altri approcci, nell’ambito dei WS, per risolvere l’insieme dei problemi vecchi e nuovi. Alcuni di essi sono: Object Management Group's (OMG), Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) e Sun Microsystems's Java/Remote Method Invocation (RMI).

2.2 Dal web statico al Web Semantico

Nel paragrafo precedente sono stati introdotti gli sviluppi tecnologici delle pagine web, che sono passate dalla staticità alla dinamicità. Per quanto siano nette le differenze esistenti tra il Web attuale e quello dei primi anni, si fa notare che la struttura di base è rimasta fundamentalmente la stessa: una rete di risorse di informazioni che è basata sull’infrastruttura di Internet.

Si analizzano ora gli sviluppi “concettuali” che portano al Web Semantico.

Con il termine WS, trascritto dalla definizione presente nell’enciclopedia on-line Wikipedia^[2,4], si intende “la trasformazione del Word Wide Web in un ambiente dove i documenti pubblicati sono associati a metadati che ne specificano il contesto semantico, in un formato adatto all’interrogazione, all’interpretazione e, più in generale, all’elaborazione automatica”.

Fino a qualche anno fa, lo scenario del World Wide Web è stato quello di un insieme di testi collegati tra loro, documenti questi che erano stati creati ad uso e consumo dei soli esseri umani, con la caratteristica fondamentale “dell’universalità”.

Con l’interpretazione del contenuto dei documenti, che il WS prospetta, saranno possibili nuove potenzialità e funzionalità, soprattutto nell’ambito della ricerca di informazioni.

Inoltre, sarà possibile costruire reti di relazioni e connessioni tra pagine web secondo logiche più elaborate del semplice link ipertestuale, che caratterizzava le prime pagine on-line.

I link ipertestuali, che sono tuttora alla base del Web, permettono che “qualunque cosa possa essere collegata a qualunque altra cosa da chiunque”.

Si possono distinguere i collegamenti fundamentalmente in due classi:

- collegamenti sintattici: in questo tipo di link le pagine web sono collegate sintatticamente mediante degli indici (indirizzi) che localizzano, in modo univoco, una pagina/risorsa attraverso un URL (Uniform Resource Location). Questo tipo di collegamenti ha tuttavia il problema dell’updating, che può

capitare in caso di cancellazione o di spostamento della pagina stessa. C'è inoltre il problema dell'assenza di significato del collegamento stesso;

- collegamenti semantici: questi ultimi invece descrivono il “significato” di un collegamento. Essi però sono talvolta generici e spesso vaghi in quanto, non sempre, descrivono dove porta il collegamento ipertestuale e, soprattutto, che significato preciso viene dato al collegamento stesso.

All'interno del Web, l'utente generico che è alla ricerca di informazioni, si orienta grazie a “parole chiave” o alla propria “esperienza” di navigazione.

L'utilizzo di parole, od espressioni, chiave è fortemente legato al codice: questo sistema è alla base del lavoro dei “motori di ricerca”, che saranno trattati nei prossimi paragrafi.

L'esperienza di navigazione, invece, si costruisce col tempo ed è un modo importante di reperire informazioni in rete. L'utente impara che determinati contenuti si trovano in determinati portali e che già l'aspetto di un sito può indicare qualcosa sul genere di informazioni contenute nel sito stesso. Tutte queste capacità, fino a qualche anno fa, non potevano essere svolte da nessuna applicazione dato che, in definitiva, questa non era in grado di interpretare il contenuto delle pagine. Ora, nell'ambito dello sviluppo del WS e dei motori di ricerca, sono allo studio delle applicazioni che “tengono traccia” delle ricerche svolte dall'utente in precedenza e che, in base ad esse, “fanno delle scelte” e “pilotano” quella che è la ricerca attuale fatta dall'utente. L'ambito della ricerca di informazioni è solamente una parte di quell'argomento che generalmente viene indicato con il nome di “rappresentazione della conoscenza” (o knowledge representation). Questa tematica, che è ampiamente studiata e sviluppata nel campo dell'Intelligenza Artificiale, si occupa di trovare dei sistemi e/o dei meccanismi che consentano di organizzare, memorizzare, gestire e rendere fruibile la “conoscenza” che l'uomo ha del mondo che lo circonda.

Il termine “Web Semantico”, come già anticipato nell'introduzione, è stato proposto per la prima volta da Tim Berners-Lee, in un suo articolo^[1.1] apparso su Scientific American del Maggio 2001. Egli scriveva: “ Il Web Semantico è un'estensione del Web attuale, in cui le informazioni hanno un ben preciso significato e in cui computer e utenti lavorano in cooperazione”.

In un altro precedente intervento^[2.5] del 1999, egli aveva già presentato così la sua visione del Web, come una realtà diversa da quella attuale, non una sua alternativa, ma una sua evoluzione:

“Ho fatto un sogno riguardante il Web [...] ed è un sogno diviso in due parti. Nella prima parte, il Web diventa un mezzo, di gran lunga più potente, per favorire la collaborazione tra i popoli. Ho sempre immaginato lo spazio dell'informazione come una cosa a cui tutti abbiano accesso immediato e intuitivo, non solo per navigare, ma anche per creare. Nella seconda parte del sogno, la collaborazione si allarga ai computer. Le macchine diventano capaci di analizzare tutti i dati sul Web, il contenuto, i link e le transazioni tra persone e computer. [...] i meccanismi quotidiani di commercio, burocrazia e vita saranno gestiti da macchine che parleranno a macchine, lasciando che gli uomini pensino soltanto a fornire l'ispirazione e l'intuito. Questo si concretizzerà introducendo una serie di progressi tecnici e di adeguamenti sociali attualmente in fase di sviluppo. [...] il Web sarà un luogo in cui l'improvvisazione dell'essere umano e il ragionamento della macchina coesisteranno in una miscela ideale e potente”.

Da allora egli ha associato questo termine all'idea di un Web nel quale agiscano “agenti intelligenti”, che altro non sono che applicazioni in grado di comprendere il significato dei documenti presenti nel Web.

Questi agenti, inoltre, dovrebbero guidare l'utente verso l'informazione ricercata o sostituirsi all'utente stesso nello svolgimento di alcune operazioni. Una di queste è quella di comprendere il significato dei testi contenuti nelle pagine web, per poi creare percorsi all'interno del Web stesso, in base alle richieste dell'utente, con la possibilità di spostarsi di sito in sito tramite i collegamenti logici dell'informazione. E' qui che entrano in gioco, oltre ai collegamenti sintattici, i collegamenti semantici. Questi agenti hanno anche il compito di verificare l'attendibilità di una informazione reperita, tramite ricerche incrociate o con altri metodi, che saranno illustrati nei capitoli seguenti.

Così facendo, si può “automatizzare la ricerca delle pagine”. Per far questo, è necessario che, durante la creazione delle pagine, le informazioni siano definite ed inserite secondo precise regole semantiche. Di qui l'aggettivo di “Semantico” che viene dato al Web la cui evoluzione segue questa nuova strada.

In prima analisi, il Web Semantico, che viene qui abbreviato in WS (in qualche pubblicazione è indicato anche con WebSem, WSem o SemWeb) è, quindi, un nuovo modo di concepire i documenti presenti nel World Wide Web.

A tuttora, le nuove possibilità e potenzialità che possono essere offerte dall'implementazione del WS sono talmente tante e tali che si parla di “visione del Web Semantico”.

Il W3C, si ricorda, ha già da alcuni anni inserito il Web Semantico tra i suoi obiettivi strategici primari di medio e lungo termine.

Nei prossimi paragrafi saranno analizzati i punti cardine che hanno portato all'evoluzione del Web verso il Web Semantico e sarà illustrata anche l'attività svolta in questo ambito dalla ricerca.

2.2.1 Motori di ricerca

Uno degli strumenti più diffusi ed utilizzati dagli utenti per cercare risorse informative sul Web è rappresentato dai motori di ricerca. Questi strumenti, nel corso degli anni, hanno subito molte migliorie e potenziamenti. Si può descrivere in breve il loro funzionamento di base per punti:

- l'utente, tramite un form presente in una pagina web, interagisce con il motore di ricerca inviando una interrogazione, specificata tramite parole chiave;
- il motore di ricerca utilizza le parole dell'interrogazione per cercare nei file indice (già precostituiti precedentemente) le pagine web che contengono o che sono marcate con quelle parole;
- queste pagine vengono quindi ordinate per pertinenza, utilizzando vari criteri che si basano o sul contenuto testuale della pagina stessa o nelle informazioni rappresentate dai link che puntano alla pagina;
- il risultato di questa interrogazione è mostrato, in maniera condensata, all'utente in una, o più, pagine web;

- l'utente dopo aver letto questo documento riassuntivo, clicca con il mouse nel collegamento alla pagina che ritiene soddisfi meglio le sue esigenze: la pagina viene caricata e visualizzata dal browser dell'utente.

I motori di ricerca hanno, purtroppo, dei limiti che si manifestano, talvolta in maniera evidente, andando ad analizzare i risultati che vengono restituiti da una ricerca.

Il primo limite è dato dall'esistenza di ciò che viene chiamato "web nascosto". Esso è composto da quella grande mole di informazioni che sono effettivamente presenti nel Web, ma che il motore di ricerca non è riuscito a trovare o che ha messo in fondo alla lista. E' stato stimato a grandi linee, vista l'impossibilità di farlo direttamente, che il web nascosto sia pari all'80% delle risorse disponibili. Cause di questa non rintracciabilità sono dovute ai documenti il cui contenuto non è stato indicizzato, ad informazioni presenti in pagine secondarie, immagini e materiale audio/video a cui non si riesce ad attribuire una chiave, ad archivi che sono memorizzati in formato compresso, ad informazioni presenti in basi di dati o a pagine i cui contenuti cambiano in tempo reale.

Altro limite è dato invece dalla troppa quantità di informazioni che vengono visualizzate quando si ricerca un'informazione molto comune o generica: si ottengono lunghissime liste di pagine, nelle quali l'utente spesso è disorientato.

Altri grandi limiti del motore di ricerca sono la mancanza di risultati e la bassa pertinenza dei risultati proposti rispetto a quanto ricercato dall'utente. Sono da segnalare anche i problemi di vocabolario: la ricchezza e, talvolta, l'ambiguità del linguaggio naturale possono essere mal gestiti dal motore di ricerca. Questo ambito è tuttora in fase di ricerca e di sviluppo.

La realizzazione completa dell'architettura del Web Semantico consentirà anche di potenziare enormemente le capacità operative dei motori di ricerca. Con il WS si può aggiungere alle pagine web un senso compiuto, un significato che va oltre le parole scritte, una "personalità" che può aiutare ogni motore di ricerca ad individuare ciò che l'utente sta cercando, semplicemente perché di fatto "lo è", scartando tutto ciò che esula dalla ricerca.

Tutto questo è ottenuto, non grazie a sistemi di intelligenza artificiale, ma grazie ad una "marcatura" dei documenti, di un linguaggio gestibile da tutte le applicazioni e dall'introduzione di vocabolari specifici, ossia insiemi di frasi alle quali possano associarsi relazioni stabilite tra elementi marcati.

Quindi, il WS per funzionare ha bisogno di informazioni strutturate e di regole di deduzione per gestirle, in modo da poter accostare quelle informazioni a quello che un utente chiede tramite una interrogazione.

2.2.2 Agente semantico

Un ambito in cui si sta svolgendo molto lavoro per estendere le possibilità del Web Semantico è quello degli "agenti semantici", programmi in grado di esplorare ed interagire in maniera automatica con i sistemi informatici ed il cui ruolo è quello di fornire capacità di inferenza.

Sono stati ipotizzati da Tim Berners-Lee nel famoso articolo^[1.1] intitolato "The Semantic Web" del 2001. In questo articolo viene presentato un esempio di cosa saranno e cosa potranno fare questi agenti in un prossimo futuro. Nell'esempio si racconta di due fratelli, Lucy e Pete, che devono far fare delle sedute di fisioterapia

alla madre, secondo le indicazioni mediche fornite dallo specialista. Per far ciò essi si affidano a degli “agenti web semantici”. Questo agente, che fa parte integrante del browser dei fratelli, si collega in maniera automatica con l’agente del medico per avere i dati relativi al tipo di cure da far eseguire al centro fisioterapico. Nello stesso tempo, cerca e consulta gli agenti dei centri fisioterapici della città per trovarne uno in grado di fornire le prestazioni richieste e per combinare tra loro le disponibilità di orari di apertura del centro con i tempi liberi dei fratelli, segnati questi ultimi nelle rispettive agende. In un tempo ragionevole l’agente di Lucy, o quello di Pete, propone una soluzione, che dovrà essere valutata dai fratelli. In caso positivo, cioè se la soluzione proposta è accettata, l’agente prenota le sedute; in caso negativo, l’agente rielabora da solo il tutto proponendo una soluzione alternativa.

Tutte queste operazioni, svolte dagli agenti, non possono essere svolte nell’ambito del Web tradizionale, come utilizzato finora, ma dalla sua evoluzione futura rappresentata dal WS. Tutto questo è dovuto al fatto che la maggior parte del Web è stata progettata per un uso da parte degli umani e non dalle macchine, che non hanno un sistema proprio per trattare la semantica. Secondo la visione di Berners-Lee non serve applicare l’intelligenza artificiale a ciò, ma basta solo codificare l’aspetto semantico direttamente nelle pagine web. Affinché il WS funzioni i computer devono avere accesso a serie strutturate di informazioni e a regole di deduzione che permettano un ragionamento automatizzato.

Berners-Lee illustra inoltre “che la vera potenza del WS si realizzerà quando verranno creati molti programmi che raccolgano il contenuto Web da diverse fonti, elaborino le informazioni e scambino i risultati con altri programmi. La potenza di questi agenti software crescerà esponenzialmente via via che saranno disponibili sempre più pagine web leggibili da macchine e servizi automatizzati. [...] anche agenti che non erano progettati per lavorare assieme possono scambiarsi dati, se questi sono dotati di una semantica.”

Un’altra capacità ipotizzata per questi agenti è quella dello scambio di “prove”, scritte in un linguaggio unificato, tra agenti, al fine di controllare la “veridicità” dell’informazioni scambiate; altra caratteristica importante è che questi agenti potranno sviluppare, nel tempo, nuove capacità di ragionamento quando scoprono nuove ontologie, queste ultime utilizzate per definire il significato semantico dei dati, che permettono all’agente di capire i contenuti, di interagire con i siti e di usare i servizi automatizzati.

In definitiva, gli agenti semantici, (o agenti software per il Web Semantico), sono dei programmi che raccolgono dati, elaborano e scambiano risultati con altri programmi. Da un altro punto di vista, questi agenti sono uno dei tre componenti su cui si basa il WS, assieme alla rappresentazione della conoscenza e all’utilizzo delle ontologie. Questi tre componenti sono infatti legati da un rapporto di dipendenza reciproco, che li sostiene ed aumenta le capacità di ognuno di essi: infatti se gli agenti semantici non hanno motivo di esistere senza una rappresentazione della conoscenza e senza l’utilizzo delle ontologie, questi ultimi, senza gli agenti, non porterebbero ad alcuna reale evoluzione della conoscenza.

Scrivono Paolo Bouquet e Roberta Ferrario in una introduzione^[2,6]: “gli agenti semantici devono essere in grado di rappresentare gli obiettivi di un certo utente, di mettere in atto una sequenza di azioni che possa soddisfare tali obiettivi ed, eventualmente, cooperare con gli altri agenti per ottenere tale risultato”.

Realizzando questa cooperazione, sulla base della comprensione del significato dei dati oggetti di scambio, si realizza quella interoperabilità semantica che permetterà a due agenti di capirsi nonostante abbiano a disposizione due diverse rappresentazioni di un determinato dato che si stanno scambiando e, quindi, due diverse rappresentazioni della conoscenza.

2.2.3 Introduzione all'architettura del Web Semantico

Vediamo ora quali trasformazioni e quali tecnologie stanno alla base della realizzazione del WS.

Questa nuova realizzazione non richiede alle macchine di essere dotate di una forma di intelligenza paragonabile a quella umana, bensì di essere fornite di un'abilità: esse devono essere in grado di risolvere problemi ben definiti, tramite operazioni ben definite, su dati ben definiti. Nel WS non è richiesto ai computer di capire il linguaggio umano e la sua logica, ma la chiave di tutto è nel richiedere all'uomo di sforzarsi nel progettare e realizzare "in maniera adeguata" le risorse web.

La maggior parte del Web attuale ha la caratteristica di essere "machine-readable", cioè di essere leggibile dalle macchine, ma non "machine-understandable", cioè non comprensibile dalle macchine. Con l'introduzione del termine "semantico", cioè "che ha a che fare con il significato", il Web assumerà la caratteristica di "machine-processable", cioè elaborabile dalle macchine.

Il WS può diventare dunque un ambiente in cui si può pubblicare e rintracciare informazioni, che sono memorizzate in un formato tale da essere adatto all'interrogazione, all'interpretazione e all'elaborazione "automatica". Inoltre vi sarà la presenza di strutture di collegamento più espressive del semplice link ipertestuale: i collegamenti semantici.

Ritornando all'esempio della visita specialistica della madre di Pete e Lucy, fatto da Berners-Lee e illustrato nelle righe precedenti, il lettore si chiede come sia possibile realizzare tutto ciò. Per dare risposta a questa domanda, nelle prossime pagine, si scalerà, gradino per gradino, la piramide tecnologica che costituisce l'architettura che sta alla base del web semantico. Questa architettura è stata proposta da Tim Berners-Lee ed è ora utilizzata come base per rappresentare un'immagine dello stato delle tecnologie attuali e di quelle emergenti.

Il problema della rappresentazione della conoscenza assume un ruolo centrale nello sviluppo del WS. Esso viene affrontato e suddiviso attraverso un'architettura che è stratificata su tre livelli principali:

- i dati: che sono definiti in modo strutturato tramite XML;
- i metadati: "informazioni sui dati" che sono gestite tramite RDF;
- le ontologie: una "rappresentazione semantica" di dati e metadati tramite specifici linguaggi.

Questi livelli principali sono affiancati da una struttura tecnologica che è rappresentata nella Figura 2.2. L'immagine mostra: 1) le diverse risorse tecnologiche che sono implicate in questa architettura (indicate con i rettangoli colorati); 2) un altro modo di partizionare in livelli la piramide (in Documento - Dati - Regole/ragionamento); 3) una suddivisione temporale, da non considerarsi così netta, dello stato di avanzamento dello sviluppo del WS.

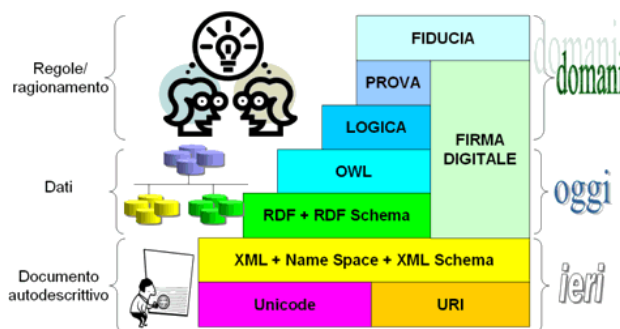


Figura 2.2 – L’architettura del Web Semantico

Alcune di queste risorse tecnologiche sono già disponibili (Unicode, URI, XML, RDF), altre sono in fase di sviluppo (OWL), le altre rappresentano il futuro (Logica, Prova, Fiducia). Trasversali rispetto a più livelli risultano le tecnologie legate alla Firma digitale.

Analizziamo, in breve, i vari componenti (i rettangoli) della piramide, partendo dalla base:

Unicode: è un sistema di codifica che assegna una combinazione di bit ad ogni carattere in maniera indipendente dal programma, dalla piattaforma e dalla lingua;

URI: (Uniform Resource Identifier) è una stringa che identifica in maniera univoca una risorsa nel Web;

XML, XML Schema e Namespace: XML (eXtensible Markup Language) è un metalinguaggio che fornisce un insieme di regole sintattiche, dette specifiche, per modellare la struttura dei documenti e dei dati. XML Schema fornisce un metodo per comporre vocabolari XML. Un Namespace è un insieme di nomi di elementi e/o attributi identificati in modo univoco da un identificatore;

RDF e RDF Schema: RDF (Resource Description Framework) fornisce un insieme di regole per definire informazioni descrittive sugli elementi che costituiscono un documento web. Queste asserzioni sono realizzate tramite triple (soggetto, predicato ed oggetto) che legano gli elementi in una relazione binaria. RDF Schema fornisce un metodo per combinare tra loro queste descrizioni in un singolo vocabolario.

Il modo per sviluppare vocabolari specifici per un dato dominio di conoscenza è rappresentato dalle “ontologie”.

OWL: (Ontology Web Language) è un linguaggio di markup utilizzato per rappresentare esplicitamente significato e semantica di termini con vocabolari e relazioni tra i termini. Tale rappresentazione dei termini e delle relazioni tra di essi è un’ontologia. L’obiettivo del WS è di permettere alle applicazioni software di elaborare il contenuto delle informazioni dei documenti scritti in OWL.

Quelle introdotte finora sono le tecnologie che stanno alla base del processo di rappresentazione della conoscenza. I gradini più in alto nella piramide sono occupati da tecnologie che sono tuttora in evoluzione.

Logica: si vuole costruire un linguaggio logico che riesca a realizzare le inferenze (conclusioni ottenute tramite procedimenti deduttivi partendo da alcune premesse) tra i molteplici dati, estratti in maniera automatica, al fine di dare all'utente le effettive informazioni ricercate;

Prova: si vuole che queste informazioni, ottenute come conclusioni della logica, siano valide;

Fiducia: si desidera inoltre che queste informazioni provengano da fonti o utenti che siano ritenuti, senza dubbio, attendibili. L'obiettivo finale è quello del "Web of Trust", cioè di un Web che offra riservatezza e che ispiri sempre più fiducia da parte di chi ne usufruisce;

Firma digitale: questo ambito garantisce, tramite un sistema di crittografia, l'autenticità delle varie asserzioni e permette di scoprirne la loro provenienza. Può essere allegata direttamente ai documenti web stessi. In altre parole, chi pubblica materiale nel Web se ne assume la responsabilità.

Questa architettura, descritta qui in breve, verrà ampiamente trattata ed estesa nel prossimo capitolo nel quale, inoltre, sarà descritto anche lo stato di sviluppo della ricerca attuale, relativamente ad ogni singolo componente.

2.2.4 Metadati

I metadati, che sono alla base del Web Semantico, sono letteralmente dei "dati su (altri) dati". Essi non sono altro che informazioni relative ai dati, tramite le quali è possibile ricavare delle ulteriori informazioni sulla risorsa a cui queste sono associate. Un esempio tipico di metadati è la scheda del catalogo di una biblioteca, che contiene le informazioni sul contenuto, autore, genere, casa editrice, anno di pubblicazione, sulla collocazione del libro all'interno della biblioteca, ecc.

La funzione principale di un sistema di metadati, nell'ambito del Web, è quella di consentire il raggiungimento dei seguenti obiettivi:

- Ricerca: individuazione dell'esistenza di un determinato documento;
- Localizzazione: rintracciare dove si trova una particolare occorrenza di quel documento;
- Selezione: di un determinato documento in base a valutazioni e/o filtri;
- Interoperabilità semantica: che permette la ricerca in ambiti disciplinari diversi, grazie all'uso di descrizioni equivalenti;
- Gestione risorse: gestione di raccolte di documenti tramite cataloghi e banche dati;
- Disponibilità: avere informazioni sull'effettiva disponibilità del documento.

I "campi" di un metadato sono costituiti dalle informazioni che descrivono le risorse informative a cui si applicano i metadati, al fine di renderle più visibili e più facilmente accessibili: ad esempio l'autore di un libro. I metadati, infatti, permettono il recupero dei documenti primari indicizzati, attraverso le stringhe (o campi) descrittive contenute in record (o "schede"): in queste ultime vengono rappresentate le caratteristiche più significative o le proprietà principali dei dati in questione,

cosicché la loro essenza possa essere racchiusa in un'unica concisa descrizione che, in modo conciso e standardizzato, fornisce un mezzo per il recupero dei dati stessi.

I sistemi di metadati strutturati in modo gerarchico sono più noti come “schemi” od ontologie, che verranno esplicitate nel prossimo paragrafo.

Attualmente il termine metadato è utilizzato quasi esclusivamente in riferimento al contesto dell'informazione digitale in rete: i metadati sono intesi come un'amplificazione delle tradizionali pratiche di catalogazione bibliografica in un ambiente elettronico.

I metadati possono essere divisi in tre gruppi: metadati descrittivi, metadati amministrativi e gestionali e metadati strutturati.

- I metadati descrittivi servono per identificare e recuperare gli oggetti digitali: sono composti da descrizioni dei documenti originali (fonte), o dei documenti già nati in formato digitale; essi risiedono in basi di dati di Information Retrieval, collocate all'esterno dell'archivio digitale e collegati a questo ultimo tramite appositi link.
- I metadati amministrativi e gestionali evidenziano le modalità di archiviazione e manutenzione degli oggetti digitali nel sistema di gestione dell'archivio; nel mondo digitale, essi documentano i processi tecnici che sono associati alla conservazione permanente, forniscono informazioni sulle condizioni ed i diritti di accesso agli oggetti, certificano l'autenticità e l'integrità del contenuto ed individuano in maniera univoca il documento stesso.
- I metadati strutturali collegano le varie componenti delle risorse per un utilizzo completo; inoltre forniscono i dati di identificazione e localizzazione del documento (codice identificativo, indirizzo del file sul server, ecc.). Inoltre, possono essere utilizzati per consentire un impiego funzionale dei documenti in un determinato sistema informativo: ad esempio possono identificare l'autore, il formato del documento, i legami con altri documenti, definire gli accessi alla lettura, ecc.

Nel Web, sono presenti contenuti/risorse che, in alcuni casi, sono destrutturati e privi di metadati, oppure che sono descritti con schemi di metadati diversi tra loro. Questi problemi hanno dato il via ad una discussione sui metadati che è diventata una discussione sugli schemi standard condivisi; questo al fine di consentire che una indicizzazione con schema di metadati omogeneo permetta l'interoperabilità tra tipi di risorse diversi e tra diversi sistemi informativi. Ciò è possibile con l'accordo sull'uso di un sistema di indicizzazione standard e quindi sullo stesso schema di metadati. In questi anni sono stati proposti diversi schemi di metadati: uno dei più diffusi è il Dublin Core^[2.7], un sistema di metadati di tipo descrittivo costituito da un insieme minimale di elementi per descrivere materiale digitale accessibile via rete. Il set minimo è costituito da 15 elementi: Titolo (Title); Creatore (Creator); Soggetto (Subject); Descrizione (Description); Editore (Editor); Autore di contributo subordinato (Contributor); Data (Date); Tipo (Type); Formato (Format); Identificatore (Identifier); Fonte (Source); Lingua (Language); Relazione (Relation); Copertura (Coverage); Gestione dei diritti (Rights).

2.2.5 Ontologie

Tim Berners-Lee, nei suoi scritti ed interventi, ha sottolineato più volte che uno degli elementi fondamentali del web semantico sarà la compresenza di più ontologie.

Le ontologie sono, come già accennato prima, il terzo componente alla base del WS; esse sono raccolte di informazioni o documenti, scritti in un linguaggio quale RDF, che definiscono formalmente le relazioni tra i termini o gli oggetti di un determinato dominio. Tim Berners-Lee nel suo articolo^[1.1], più volte citato, evidenziava già allora un problema che può capitare nella gestione della conoscenza: egli supponeva che vi fosse la necessità di confrontare tra loro le informazioni contenute in due database che, però, utilizzavano identificatori diversi per uno stesso concetto. Si pensi, per esempio, all'indirizzo della residenza di una persona: ebbene un database può memorizzare i dati quali via, numero civico, CAP, città, ecc. in campi separati, mentre un altro database può memorizzare il tutto in un'unica stringa. Una semplice elaborazione, quale il confronto tra questi due indirizzi, può far nascere il problema quando si raffrontino i valori dei singoli dati. Le ontologie, ipotizza ancora Berners-Lee, dovrebbero porre rimedio a questo problema.

Il tipo più semplice di ontologia, utilizzato nel Web, è composto da due parti: una tassonomia ed un insieme di regole. La tassonomia definisce le classi di oggetti e le loro relazioni: si possono esprimere un gran numero di relazioni tra gli enti assegnando proprietà alle classi e permettendo alle classi di ereditare tali proprietà. Le regole ontologiche sono di tipo deduttivo: ad esempio, "se A implica B e B implica C, allora A implica C".

L'utilizzo di pagine "ontologiche" in rete permetterebbe di risolvere alcuni problemi di terminologia: ad esempio, il significato di taluni termini all'interno di una pagina web può essere definito da puntatori che rimandano alla pagina ontologica, nella quale questo significato è definito con precisione.

Un'ulteriore applicazione delle ontologie può essere fatta nelle ricerche di informazioni in rete. La ricerca, tramite un semplice motore di ricerca, di un termine quale ad esempio "Rossi" fa risultare un'enorme quantità di dati (nel motore di ricerca Google sono oltre 12 milioni) che spaziano dall'aggettivo rossi al cantante Vasco Rossi, dal pilota Valentino Rossi ad un ipotetico sig. Rossi, quello utilizzato negli esempi, al giocatore di calcio De Rossi, e via così. Se le pagine web contenessero puntatori alle ontologie, la ricerca potrebbe essere semplificata: specificando il "contesto" dove è utilizzata la parola "rossi", i risultati possono essere più precisi. Nei prossimi capitoli, si mostreranno i miglioramenti che gli sviluppatori hanno apportato ai motori di ricerca per le ricerche in rete.

Sottolinea ancora Berners-Lee che "la vera potenza delle ontologie si otterrà quando i computer le utilizzeranno in maniera sofisticata per correlare le informazioni contenute in una pagina con le relative strutture di conoscenza e le regole di deduzione"^[1.1]. Così facendo si potrebbero reperire informazioni che non sono contenute in una singola pagina, ma che sono contenute in più pagine anche non collegate sintatticamente tra loro.

In informatica, una ontologia è "una rappresentazione formale di un insieme di concetti all'interno di un dominio e delle relazioni tra questi concetti". L'ontologia é usata per ragionare sulle proprietà di quel dominio e può essere utilizzata per definire il dominio stesso. Un dominio è un tipo di oggetti e/o concetti che esistono, con le

loro proprietà e relazioni. Dal punto di vista teorico, l'ontologia può essere espressa come una “rappresentazione formale, esplicita e condivisa di una concettualizzazione di un dominio d'interesse”.

Le ontologie sono utilizzate in svariati campi oltre all'Intelligenza Artificiale, ambito dove sono stati fatti i primi studi, campi che vanno dal Web Semantico, all'Ingegneria del Software, alle “architetture dell'informazione” come una forma di rappresentazione della conoscenza sul mondo o una parte di esso. In Informatica, i programmi possono utilizzare un'ontologia per una grande varietà di scopi, tra cui il ragionamento induttivo, la classificazione e per l'utilizzo di tecniche di problem-solving, oltre che per facilitare la comunicazione e lo scambio di dati fra sistemi diversi. Le ontologie informatiche sono legate a vocabolari “controllati”, in base ai quali tutto il resto deve essere descritto. Generalmente essa è, quindi, una struttura dati gerarchica, che contiene tutte le entità rilevanti, le relazioni esistenti tra di esse, le regole, gli assiomi ed i vincoli specifici del dominio.

Il termine ontologia deriva dalla Filosofia, dove è relativo allo studio della natura della realtà, cioè “di tutto quello che è o che esiste”. Quello che c'è in comune con l'informatica è la rappresentazione di entità, idee, eventi, insieme con le loro proprietà e relazioni, come un sistema di categorie.

I primi studi sulle ontologie risalgono alla seconda metà del XX secolo; dal 2001 il termine è diventato popolare grazie all'intensa attività e alla forte crescita dei ricercatori impegnati nel Web Semantico. Le ontologie moderne condividono con quelle più vecchie molte strutture simili, a dispetto del linguaggio in cui sono espresse.

I componenti più comuni contenuti in un'ontologia moderna sono:

- **Individui:** sono le istanze o gli oggetti (detti anche “livello base”);
- **Classi:** sono gli insiemi, le collezioni, i concetti, i tipi di oggetto, i generi di cose;
- **Attributi:** sono gli aspetti, le proprietà, le caratteristiche, i caratteri o i parametri che l'oggetto o la classe può avere;
- **Relazioni:** sono i modi in cui le classi e gli individui sono collegati tra loro;
- **Termini funzione:** sono strutture complesse formate da certe relazioni che possono essere usate al posto di un termine individuale in una dichiarazione;
- **Restrizioni:** sono composte da descrizioni, formalmente stabilite, di quello che deve essere vero e permesso per alcune asserzioni, al fine di poter essere accettate in ingresso;
- **Regole:** sono espressioni, scritte nella forma IF-THEN, che descrivono l'inferenza logica che deve essere ottenuta da una espressione in una forma particolare;
- **Assiomi:** sono le affermazioni e le regole espresse in una forma logica tale che, messe insieme, comprendono tutta la realtà che l'ontologia descrive nel suo dominio di applicazione. La definizione differisce da quello di “assioma” proprio della grammatica e della logica formale;
- **Eventi:** il cambiamento di una attributo o di una relazione.

Le ontologie sono genericamente strutture gerarchiche e sono solitamente divise in due tipi: le ontologie “di dominio” e le ontologie “fondamentali”.

Un'ontologia di dominio (detta anche domain-specific ontology) modella un dominio o una parte del mondo: essa rappresenta il significato specifico dei termini e come questi sono utilizzati in quel dominio.

Un'ontologia fondamentale (detta anche fondazionale o superiore o upper ontology) è un modello degli oggetti comuni che sono, di solito, utilizzati trasversalmente in un ampio campo di domini ontologici. Essa contiene un glossario, nei termini del quale, gli oggetti in un insieme di domini possono essere descritti. Esistono già diverse ontologie fondamentali standardizzate e pronte per l'uso: una delle più conosciute è la Dublin Core, una ontologia semplice per i documenti e le pubblicazioni.

Dato che le ontologie di dominio rappresentano i concetti in una maniera molto specifica e, spesso, in un modo eterogeneo, c'è frequentemente il problema della compatibilità fra di esse. Man mano che i sistemi che si basano su un dominio ontologico si espandono, capita spesso la necessità di fondere più domini ontologici in uno, generalmente più espressivo: questa è la sfida che si presenta ai progettisti delle ontologie. Infatti, possono esistere diverse ontologie che sono rappresentative dello stesso dominio: questo è dovuto alla diversa percezione del dominio di chi ha costruito l'ontologia, differenza basata su background culturale, educazione, ideologie o solo perché ciò è stato rappresentato con un diverso linguaggio. Al momento, il lavoro di fondere ontologie, che non sono state sviluppate da una ontologia di base comune, è un lavoro principalmente manuale e perciò lungo e costoso. Invece, la fusione di ontologie che utilizzano la stessa ontologia fondamentale, per fornire un insieme di elementi base con i quali specificare il significato degli elementi del dominio ontologico, può essere fatta in maniera automatica.

L'Ingegneria "ontologica" (detta anche ontologia costruttiva) è un sottocampo della ingegneria della conoscenza che studia i metodi e le metodologie per costruire ontologie. Essa elabora i processi di sviluppo delle ontologie, i loro cicli di vita e gli strumenti più adatti per supportarle, quali i linguaggi. Ha inoltre l'obiettivo di rendere esplicita la "conoscenza" che sarà poi utilizzata all'interno delle applicazioni informatiche. Altro obiettivo è quello di indicare una soluzione ai problemi di interoperabilità, causati da ostacoli semantici quali ad esempio definizioni diverse di termini e di classi. L'ingegneria ontologica fornisce anche delle risorse che possono essere utili allo sviluppo di nuove ontologie per uno specifico dominio.

Le ontologie sono comunemente codificate usando gli "ontology languages". Esistono tuttora molti linguaggi per le ontologie, sia proprietari che basati sugli standard. Si citano: Common Algebraic Specification Language, Common Logic, Cyc, DOGMA (Developing Ontology-Grounded Methods and Applications), Gellish, IDEF5, KIF, RIF (Rule Interchange Format), SADL, OBO, ecc. Uno di quelli basati su standard, che verrà analizzato nel prossimo capitolo, è l'OWL acronimo di "Ontology Web Language"; l'OWL è un linguaggio per fare dichiarazioni ontologiche, sviluppato come proseguo dal RDF e dal RDF Schema, così come da altri precedenti progetti di linguaggio ontologico tra i quali si cita OIL (Ontology Inference Layer), DAM (DARPA Agent Markup Language) e DAM+OIL.

L'OWL è stato studiato per un utilizzo nel World Wide Web: tutti i suoi elementi sono definiti in RDF come risorse ed identificati tramite URI. Nel corso degli anni, altri linguaggi ontologici sono stati sviluppati per un utilizzo in ambiti diversi dall'informatica, quali medici/biologici, linguistici, architettonici, ecc. Il loro approfondimento esula dallo scopo di questa tesi. Molto materiale a riguardo di essi può

essere trovato in rete per chi volesse approfondire l'argomento. L'OWL è attualmente lo standard per le ontologie negli ambienti Web.

Lo sviluppo di ontologie per il Web ha portato alla comparsa di servizi, includenti crawler, che forniscono liste o archivi di ontologie well-formed, cioè “fatte bene” e facilmente reperibili in rete. Un crawler (detto anche spider o robot), è un software che analizza i contenuti di una rete (o di un database) in un modo metodico e automatizzato, in genere per conto di un motore di ricerca.

Un esempio di questi servizi è “Swoogle”^[2.8], che è un archivio ed un motore di ricerca di tutte le risorse RDF disponibili in rete, ontologie comprese. Un altro esempio è “Ontaria”^[2.9], una archivio di ricerca e di navigazione di dati web semantici, focalizzato su vocabolari RDF con ontologie scritte in OWL.

Finora nessuna ontologia superiore è stata riconosciuta come uno standard *de facto*: infatti la creazione di una ontologia fondamentale e totale è un'impresa ardua, al contrario dello sviluppo di ontologie limitate a domini ben precisi. Diverse organizzazioni stanno lavorando alla definizione di ontologie standard per domini di applicazione specifici: ad esempio il “Process Specification Language, PSL” creato dal National Institute for Standards and Technology, NITS.^[2.10]

2.2.6 Conclusione ipotizzata da Berners-Lee

A questo punto il lettore si può chiedere come queste tecnologie viste possano cooperare, affinché il Web possa effettivamente fornire i servizi ipotizzati. Tim Berners-Lee proponeva che alla base di tutto ciò deve esserci una diversa e più attenta filosofia di progettazione delle risorse web, che devono essere basate sull'uso dell'XML, che devono rispettare gli standard definiti e che devono avere con sé una descrizione delle proprie caratteristiche, rappresentate queste ultime per mezzo del RDF e dei metadati. Tramite tutto questo gli agenti intelligenti possono operare per compiere le proprie azioni e per prendere le decisioni. Questi agenti potranno muoversi nello spazio Web sfruttando il sistema di rappresentazione della conoscenza disponibile (le ontologie). Le decisioni degli agenti saranno consentite grazie all'utilizzo di linguaggi di inferenza logica, che terranno conto anche del grado di fiducia attribuito alle risorse, con i loro autori, dagli utenti stessi.

Già in quegli anni si capì che il Web non si sarebbe trasformato in Web Semantico in poco tempo. I primi strumenti disponibili erano primitivi e il lavoro da fare era molto oneroso.

Le critiche principali sulla effettiva validità del progetto Web Semantico erano legate al fattore tempo (quanto tempo occorre per mappare tutto il Web nelle ontologie?) ed, inoltre, ci si chiedeva come poter superare le difficoltà di comunicazione tra le diverse ontologie.

Pensare, oggi, al Web come ad una infrastruttura regolata nel suo complesso da una struttura semantica, è ancora una prospettiva incerta, quantomeno nel breve periodo. Il discorso cambia se si considerano schemi semantici utilizzati all'interno di architetture legate ad un dominio piccolo e ben preciso e quindi a comunità ristrette di utenti che sono orientate ad ottenere uno scopo proprio.

Nel prossimi capitoli, si analizzeranno i progressi svolti dalla ricerca nell'ambito del WS, e verrà approfondita un'applicazione attualmente in studio e/o in uso; questo servirà per fare il punto della situazione attuale o, detto in un altro modo, per conoscere "lo stato dell'arte".

2.2.7 Evoluzione dal Web 1.0 al Web 3.0

In questo paragrafo si illustra un altro modo di analizzare l'evoluzione del Web che porta al Web Semantico. L'analisi è fatta per "versioni numerate", come viene fatto per le versioni degli applicativi software. Questa suddivisione, si anticipa subito, non è però netta, in quanto non è possibile fare una "fotografia" che racchiuda in sé tutte le caratteristiche proprie di ogni generazione cosa che, invece, è possibile con un applicativo software, dato che una versione differisce da un'altra per le caratteristiche tecniche aggiunge o modificate.

Tim Berners-Lee, nel Maggio del 2006, affermava: " Le persone si chiedono cosa sia il Web 3.0. Penso che, quando si sarà ottenuta una sovrapposizione della Grafica Vettoriale Scalabile – oggi tutto appare poco nitido, con pieghe ed increspature – nel Web 2.0 e l'accesso ad un Web semantico integrato attraverso un grosso quantitativo di dati, si potrà ottenere l'accesso ad un'incredibile risorsa di dati i quali generano il Web 3.0"^[2,11].

Il termine "Web 3.0" è apparso per la prima volta agli inizi del 2006, in un articolo di Jeffrey Zeldman^[2], nel quale egli criticava il Web 2.0 ed alcune sue tecnologie. Il termine Web 3.0 ha, tuttora, diversi significati che tendono a descrivere l'evoluzione dell'utilizzo del Web e le interazioni tra i numerosi percorsi di avanzamento che sono possibili. Tra questi significati, si evidenziano:

- la trasformazione del Web in un database, per consentire l'utilizzo dei contenuti da parte di molteplici applicazioni e quindi non solo dai browser;
- un miglior sfruttamento di quelle tecnologie che sono basate sull'intelligenza artificiale, al fine di rendere quasi "umana" l'interazione con il Web;
- il Web Semantico;

Nell'ultimo anno sono apparse le prime applicazioni web "intelligenti", in grado di comprendere alcuni dati pubblicati on-line. Questo nuovo modo di gestire le informazioni rende operativa l'evoluzione del Web, a distanza di vent'anni dalla nascita. Sul nome da dare a queste nuove tecnologie non c'è tuttora accordo: taluni informatici lo chiamano già "Web Semantico", altri il "Web dei dati", o chi ancora "Web 3.0". Per dare un'idea della poca chiarezza che regna, basti pensare che l'enciclopedia on-line Wikipedia, nella sua versione inglese, a seguito dei numerosi dibattiti occorsi, ha cancellato la voce "Web 3.0" dalle proprie definizioni.

Nei primi dieci anni della sua vita (indicativamente 1993-1999), il Web è stato un mezzo di sola "lettura", in pratica una grande biblioteca da cui attingere informazioni. Questo stato dell'evoluzione viene anche identificato con l'etichetta od il termine "Web 1.0". Il Web di quegli anni era composto prevalentemente dai siti web statici, senza nessuna possibilità di interazione con l'utente eccetto la normale navigazione tra le pagine, l'uso della posta elettronica e dei primi motori di ricerca.

La seconda decade (approssimativamente 2000-2008), ha visto la diffusione di quei servizi che lo hanno trasformato in un mezzo di “scrittura”. Gli utenti del Web hanno iniziato ad essere loro gli autori dei contenuti, tramite forum, blog, chat, sistemi quali “Youtube” e social network quali “Facebook”, per citare quelli più noti. Questo approccio è stato battezzato col nome di “Web della partecipazione” o “Web 2.0”, in quanto vi è un notevole livello di interazione sito-utente.

Tuttavia, ci sono gruppi di ricercatori che sono scettici sull’utilizzo di questa etichettatura numerica, specialmente quando essa è utilizzata per indicare, in maniera univoca, una complessa e continua innovazione del Web. In molti dei loro studi si nota sempre di più l’utilizzo del termine “Nuovo Web” per indicare anche dinamiche future che comprendono, ma vanno oltre il Web 2.0.

Un primo passo verso un Web 3.0 è stato fatto con l’emergere dei “Data Web”, dato che gli archivi di dati strutturati sono ora pubblicati sul Web in formati utilizzabili ed interrogabili anche da remoto. La recente crescita della tecnologia SPARQL, che sarà analizzata nel prossimo capitolo, fornisce un linguaggio di query standardizzato e le interfacce di programmazione delle applicazioni (API) per la ricerca attraverso database RDF distribuiti nel Web. I Data Web permettono quindi un nuovo modo di interazione e di interoperabilità delle applicazioni, rendendo i dati disponibili e collegabili come fossero pagine web. Questi Data Web sono il primo passo verso il WS: i Data Web strutturano l’informazione disponibile; il WS darà poi la possibilità di accedere a questi ultima.

Da qualche tempo a questa parte, come accennato prima, le informazioni disponibili on-line hanno cominciato ad essere comprensibili anche da parte dei calcolatori, in una maniera tale da poter essere collegate tra loro e riutilizzate tramite strumenti automatici. Questo è reso possibile tramite tecnologie oramai comunemente chiamate “semantiche” o utilizzando in maniera impropria il termine “Web 3.0”. Il Web 3.0 potrebbe, quindi, costituire la realizzazione e l’estensione del concetto di Web Semantico.

Nel quarto capitolo si analizzerà una tecnologia che utilizza le potenzialità del WS. Nel quinto capitolo sarà presentata un’applicazione dove questa tecnologia è resa operativa.

2.2 W3C Semantic Web Activity



Figura 2.3 - Loghi W3C Semantic Web Activity

Il World Wide Web Consortium (W3C), come è già stato visto, oltre allo sviluppo ed alla definizione della maggior parte degli standard del Web, si occupa anche della ricerca e dell’evoluzione di vari settori specifici del Web stesso. Uno di questi è

quello relativo al Web Semantico ed è riassunto con il termine di “W3C Semantic Web Activity”. Questa attività è divisa in vari gruppi di lavoro, chiamati “Active Groups”, ognuno dei quali è composto da varie persone ed è incaricato di svolgere dei precisi compiti.

Gli Active Groups, relativi al Web Semantico sono:

- Semantic Web Coordination Group: fornisce un forum per gestire le interrelazioni e le interdipendenze tra i vari gruppi che stanno focalizzando le loro attenzioni sugli standard e le tecnologie relative al WS. E' stato creato, inoltre, per coordinare, per rendere più facili e per aiutare ad elaborare i lavori svolti dagli altri gruppi collegati, tutto questo al fine di evitare duplicazioni e/o frammentazioni del lavoro, che possono essere causati dall'uso di tecnologie e di standard non compatibili;
- Rule Interchange Format Working Group: è impegnato nel realizzare delle regole centrali di linguaggio più le estensioni le quali, messe insieme, permettono alle regole di essere tradotte in regole di linguaggio e poi di essere trasferite in regole di sistema. Questo gruppo dovrà bilanciare la necessità di comunità differenti, specificando estensioni per le quali si può esprimere chiaramente un progetto unanime e le quali sono sufficientemente motivate da casi pratici;
- OWL Working Group: questo gruppo deve presentare delle raccomandazioni W3C che perfezionino ed estendano la versione del 2004 dell'Ontology Web Language. La nuova versione è chiamata OWL2;
- RDB2RDF Working Group: costoro devono standardizzare un linguaggio per mappare i dati relazionali e gli schemi di database relazionali nei linguaggi RDF ed OWL. Tutto questo lavoro viene chiamato “RDB2RDF (Relational Data Base to RDF) Mapping Language” in sigla R2RML;
- SPARQL Working Group: questo gruppo ha sviluppato le raccomandazioni sullo SPARQL (un acronimo ricorsivo che sta per SPARQL Protocol and RDF Query Language), che sono state pubblicate nel 2008. Attualmente si occupa di aggiornare queste specifiche;
- Semantic Web Deployment Working Group: esso deve fornire la guida, tramite i rapporti tecnici detti “W3C Technical Reports”, sulle questioni dello sviluppo teorico e pratico dell'RDF pubblicando vocabolari, manuali sull'OWL ed integrando l'RDF con documenti HTML. E' inoltre responsabile dello sviluppo delle specifiche dell'RDFa (Resource Description Framework – in - attributes) e degli SKOS (Simple Knowledge Organization Systems);
- Semantic Web Interest Group: è un forum per i membri, e non, del W3C dove sono discusse le applicazioni innovative del Web Semantico. Questo gruppo promuove discussioni sugli argomenti relativi al lavoro futuro da fare, questo al fine di rendere le nuove tecnologie capaci di integrare il WS. Inoltre, questo gruppo relaziona questo lavoro verso le altre attività del W3C che si estendono dai contesti sociali a quelli legali in cui il Web si trova;
- Semantic Web Health Care and Life Sciences Interest Group: questo gruppo migliora la collaborazione, la ricerca, lo sviluppo e l'adozione delle innovazioni nelle società/associazioni a carattere scientifico che sono impegnate nella cura della salute.

In passato esistevano altri gruppi, ora chiamati “Past Group”, che sono stati chiusi, quali ad esempio il “Semantic Web Education and Outreach Interest Group”, o altri gruppi detti “Completed Group”, che hanno terminato il loro lavoro e che non terranno più incontri regolari.

Lo staff del W3C, che si dedica al Semantic Web Activity, è composto da Harry Halpin, Sandro Hawke, Eric Prud’hommeaux, Dave Raggett e Ralph Swick, tutti sotto la supervisione del leader Ivan Herman.

All’interno del sito ufficiale del Consorzio W3C c’è una sezione, chiamata “Semantic Web Activity”^[2.12] composta da pagine internet, in lingua inglese, contenenti le pubblicazioni, gli articoli, le interviste e le presentazioni fatte dai membri del gruppo, ordinate in maniera temporale: talune, in formato testuale e/o audio, possono essere scaricate per una lettura/ascolto.

Alcune parti di questa tesi hanno preso spunto da lì, in quanto queste pagine sono tra le più recenti ed aggiornate nell’ambito del Web Semantico.

All’interno di queste pagine ci sono anche i link per accedere ad altre pagine dove è testimoniato come le tecnologie del WS sono utilizzate nelle aziende e nelle istituzioni. Queste sono divise in due tipi: “Case Studies” (studi del caso) e “Use Cases” (casi utilizzati). I Case Studies descrivono i sistemi che sono stati studiati e che sono stati sviluppati all’interno di una organizzazione. Gli Use Cases sono i sistemi che un’organizzazione ha costruito come un prototipo di sistema ma che, attualmente, non sono usati nell’ambito lavorativo.

Tutti gli use cases e i case studies sono stati presentati da istituzioni di un certo rilievo e non prodotti direttamente dal W3C.

Sempre nelle pagine del sito della W3C SW Activity è presente una lista aggiornata dei libri pubblicati relativi al WS. Inoltre c’è la possibilità di scaricare i loghi, le immagini ed i bottoni con la grafica del WS. E’ presente anche un accesso alle ultime news pubblicate; ci sono anche i link ai validatori che sono on-line e che sono relativi al solo Web Semantico.

NOTE CAPITOLO 2

1. **Mashup**: in ambito informatico, un mashup (tradotto in poltiglia) è una pagina web page, un sito o un’applicazione che mette assieme i dati o le funzionalità da due o più sorgenti esterne per creare un servizio completamente nuovo. Il contenuto dei mashup è normalmente preso da terzi via API, tramite feed (es. RSS e Atom) o Javascript. Un esempio di ciò potrebbe essere un programma che, acquisendo da un sito web una lista di appartamenti, ne mostra l’ubicazione utilizzando il servizio “Google Maps” per evidenziare il luogo in cui gli stessi appartamenti sono localizzati. I mashup sono semplici da progettare e, dato che richiedono minime conoscenze tecniche, possono anche essere creati da utenti inesperti.

2. **Jeffrey Zeldman** è un docente universitario. Il suo ultimo libro si intitola: “*Designing With Web Standards*”. Zeldman è cofondatore del Web Standards Project (WaSP), un gruppo di professionisti progettisti di siti web, con lo scopo di diffondere ed incoraggiare l’uso degli standard promossi dal World Wide Web Consortium (W3C). Egli è considerato uno dei pionieri della progettazione di componenti visuali; ha adottato e promosso l’uso di standard di base ed una soluzione, attraverso browser, ai problemi di web design. In particolare, egli ha tentato di distruggere il mito che i siti web accessibili debbano essere brutti o apparentemente mal progettati. I suoi libri e siti web hanno aiutato a promuovere un generale miglioramento nella tecnica, progettazione visiva, utilizzabilità ed accessibilità dei siti web attraverso l’uso di codice XHTML e dei CSS.

Capitolo 3

Architettura a livelli del Web Semantico

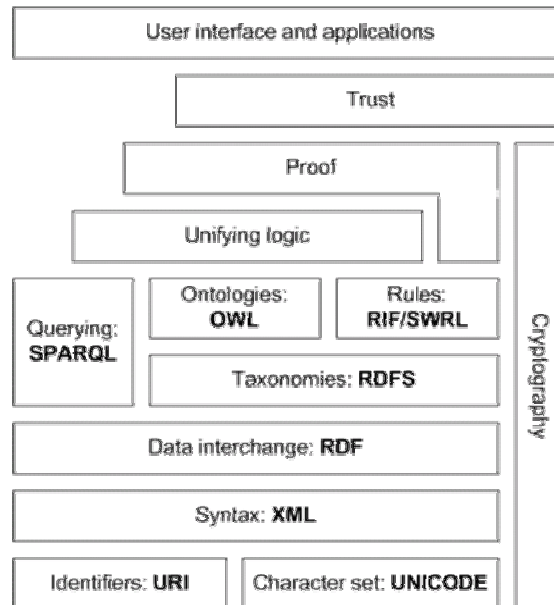


Figura 3.1 – La piramide tecnologica del WS

In questo capitolo si analizzano, uno per uno, i singoli elementi che compongono la piramide delle tecnologie che supportano il Web Semantico. Questi elementi sono già stati introdotti nel capitolo precedente: si entrerà ora nei dettagli tecnici più specifici. Si inizia dai blocchi posti più in basso, ambiti questi dove le tecnologie sono già consolidate con standard. Si salirà poi verso i livelli più alti, che sono composti da quelle tecnologie ed applicazioni in cui la ricerca è in ancora in corso.

3.1 Unicode, URI, IRI, XRI

L' "Unicode" è uno standard informatico che permette ai computer di rappresentare in maniera consistente e di manipolare i testi espressi nella maggior parte delle lingue del mondo. Questo standard altro non è che un sistema di codifica che assegna ad ogni carattere usato per la scrittura dei testi non il segno grafico (glyph=glifo) ma un numero univoco, detto "codepoint"; tutto questo è fatto in una maniera che è indipendente dalla lingua, dalla piattaforma informatica e dal software utilizzati. In questo standard sono codificati i caratteri che sono utilizzati in quasi tutte le lingue vive ed in alcune di quelle morte, oltre ai simboli matematici e chimici, ai segni cartografici, ad alcuni ideogrammi, ai simboli musicali, ecc. L'Unicode comprende quasi tutti i sistemi di scrittura attualmente utilizzati, fra i quali si citano, come esempio, gli alfabeti arabo, Braille, cirillico, greco, cinese Han (ideogrammi Hanzi e Hanja), fonetico internazionale, latino (base ed esteso), tibetano, ecc. In aggiunta a quelli citati, sono disponibili i glifi appartenenti a molte lingue morte quali gli

alfabeti cuneiforme, l'antico italico (etrusco, osco e umbro), l'antico persiano, il fenicio, ecc.

Lo standard consiste in un repertorio di oltre 107.000 caratteri che coprono 90 lingue; esso comprende inoltre tutta una serie di proprietà dei caratteri, quali minuscole e maiuscole, ed una serie di regole per la normalizzazione, la decomposizione, il confronto, il rendering (rappresentazione) e per la scrittura nei due versi (es. da destra verso sinistra per linguaggi quali l'arabo).

Lo standard Unicode è costantemente compilato ed aggiornato dall' "Unicode Consortium", una organizzazione no-profit, che ha sede in California e che è in partnership con aziende internazionali interessate all'interoperabilità informatica dei testi scritti in lingue diverse, quali ad esempio: Adobe System, Apple, Google, IBM, Microsoft, Oracle Corporation, Sun Microsystem e Yahoo. Questo standard è sviluppato in stretta collaborazione con l'ISO "International Organization for Standardization"^[3.1] e condivide il repertorio dei caratteri con la norma ISO/IEC 10646 detta "Universal Character Set". Rispetto a questo ultimo insieme, lo standard Unicode è più ricco di informazioni tecniche per l'utilizzatore.

L'Unicode è presentato nel formato di libro con il titolo "The Unicode Standard". La prima versione risale al 1991 e da allora lo sviluppo di questo standard è proseguito di anno in anno. La versione principale è la 5.0 presentata nel 2007; c'è una versione più recente, la 5.2.0 aggiornata ad ottobre 2009, con alcune migliorie che è scaricabile dal sito internet del consorzio^[3.2]. Negli ultimi anni una nuova versione è stata rilasciata praticamente ad ogni inizio anno, a seguito di una media di oltre 1.000 richieste di modifiche all'anno da parte dell'utenza.

Scopo dei membri del Consorzio Unicode è quello sostituire, in maniera definitiva, alcuni degli schemi di codifica dei caratteri attualmente esistenti, che oramai sono diventati limitati in dimensione ed utilizzo e che sono incompatibili con gli ambienti multilingue, con l'Unicode ed i suoi schemi standard detti "Unicode Transformation Format", abbreviato in UTF.

Attualmente, questo standard non rappresenta tutte le lingue utilizzate nel mondo: ciò è dovuto al fatto che alcune di queste, ad esempio Tengwar, sono recenti e non sono realmente in uso. Essendo l'Unicode ancora in evoluzione, i suoi sviluppatori si sono prefissati l'obiettivo di riuscire a gestire tutti i caratteri rappresentabili, garantendone la compatibilità e la non sovrapposizione con le codifiche di caratteri già definiti. L'Unicode è attualmente molto utilizzato e supportato dai moderni standard della programmazione e del markup (ad esempio XML, Java, Microsoft .NET Framework) e da svariati sistemi operativi.

Dal punto di vista storico, l'Unicode è stato "pensato" nel 1987 da Becker della Xerox assieme a Collins e Davis della Apple. E' stato presentato come bozza nel 1988 con una codifica a 16 bit (= 4 cifre esadecimali) che permetteva di rappresentare $2^{16}=65.536$ caratteri diversi. Nel 1991 nacque il Consorzio Unicode che presentò, nel 1996, la versione 2.0. Attualmente lo standard Unicode si è adattato alla norma ISO/IEC 10646 e può essere codificato fino a 21 bit, permettendo di rappresentare fino ad oltre due milioni di caratteri. Al momento sono circa un milione i caratteri codificati, sufficienti per coprire il patrimonio letterale, linguistico e dei segni grafici utilizzati dall'umanità compresi, ad esempio, i geroglifici egiziani.

Dal punto di vista tecnico, l'Unicode definisce uno spazio di codici di 1.114.112 codepoints nell'intervallo (nel sistema numerico esadecimale) $0_{\text{hex}} - 10\text{FFFF}_{\text{hex}}$. Il codice che viene assegnato al singolo carattere è rappresentato con U+, seguito dalle quattro (o sei) cifre esadecimali che lo individuano. Ad esempio: $U+0058_{\text{hex}}$ corrisponde al carattere "X". In questo modo l'Unicode riesce a rappresentare un carattere in maniera astratta, lasciandone la visualizzazione (dimensione, font, stile, ecc.) al software in uso, quale può essere un word-processor o un browser web.

L'Unicode ha la struttura dei codici sviluppata su 17 "plane" (piani), ciascuno dei quali ha 65.536 posizioni. Finora il consorzio ha assegnato solo i primi e gli ultimi tre piani. Il piano più conosciuto è il primo, detto BMP (Basic Multilingual Plane), che contiene le lingue più utilizzate. All'interno di ogni piano i caratteri sono distribuiti nei cosiddetti "blocks" (blocchi) dei caratteri che sono attinenti tra loro.

L'Unicode ha la caratteristica di avere dei campi di codici "non utilizzati", che possono essere impiegati in maniera autonoma all'interno di particolari applicazioni.

Diversi meccanismi sono stati specificati per implementare l'Unicode: la scelta sarà fatta dall'utilizzatore/programmatore a seconda dello spazio di memoria disponibile, della compatibilità con il codice sorgente e dall'interoperabilità con gli altri sistemi.

L'Unicode definisce due modi di mappatura: la codifica "Unicode Transformation Format" (UTF) e la codifica "Universal Character Set" (UCS).

Una mappa codifica un intervallo di Unicode codepoints in una sequenza di valori, in alcuni intervalli fissi, detti "codevalue". I numeri nel nome della codifica indicano il numero dei bit in un codevalue per la codifica UTF od il numero di byte in un codevalue per la codifica UCS, cioè $UFT-x$ x =numero di bit e $UCS-y$ y =numero di byte. L'UTF-8 e l'UTF-16 sono le codifiche più utilizzate; L'UCS-2, oramai obsoleta, è un sottoinsieme dell'UTF-16. L'UTF-8 (detta *byte*) usa 1 byte per tutti i caratteri ASCII, ha gli stessi codevalue come la codifica ASCII e si espande fino a 4 byte per gli altri caratteri. L'UTF-16 (detta *word*) estende l'UCS-2 usando 4 byte per codificare i caratteri mancanti dell'UCS-2. Esistono anche le codifiche UTF-1, UTF-7, UTF-32 (detta *double word*), UTF-EBCDIC, UCS-4 ed altre meno utilizzate o già abbandonate. L'UCS-4 e l'UTF-32 sono equivalenti dal punto di vista funzionale.

Dal punto di vista operativo, tutto questo repertorio di codici numerici è serializzato mediante diversi "schemi di ricodifica", che consentono l'uso di codici più compatti per i caratteri usati più di frequente.

L'UTF-8, che utilizza da uno a quattro byte per code point, diventa più compatto per gli alfabeti latini ed è compatibile con lo standard ASCII. Questa codifica è lo standard *de facto* per l'interscambio di testo Unicode. E'anche utilizzata in alcune recenti versioni di Linux, come un diretto sostituto delle codifiche oramai divenute obsolete nel trattamento e nell'archiviazione dei testi. Le ultime versioni di Windows, dall'NT al Vista, usano la codifica UTF-16 come unica codifica interna dei caratteri.

Si consideri ora l'operazione d'inserimento di caratteri in un testo: dato che le tastiere non hanno i tasti, o non permettono semplici combinazioni di tasti, per tutti i caratteri, la maggior parte dei sistemi fornisce dei metodi per l'inserimento dell'intero repertorio Unicode. Un modo è quello di inserire i caratteri tramite la rappresentazione esadecimale del loro codepoint. Un altro sistema più semplice è quello in cui, in una finestra dell'applicazione di videoscrittura, viene visualizzata

una tabella con tutti i caratteri disponibili per un determinato linguaggio: l'utente deve solamente cliccare col mouse sul carattere di cui necessita. L'applicazione lo considererà come un qualsiasi altro carattere digitato alla tastiera.

Lo standard Unicode è presente anche nell'ambito della posta elettronica. Lo standard MIME definisce due diversi meccanismi per inserire caratteri non appartenenti allo standard ASCII in una e-mail. Il MIME, "Multipurpose Internet Mail Extensions", è uno standard che definisce il formato delle e-mail. Non volendo entrare in dettagli troppo tecnici, si fa presente che lo standard utilizzato è l'UTF-8. La sua introduzione nel corso del tempo è stata molto lenta, visto che alcuni testi est-asiatici sono ancora codificati in codifiche tipo la ISO-2022. Anche alcuni dispositivi, quali cellulari, non si sono ancora uniformati all'Unicode: le case costruttrici stanno provvedendo al supporto necessario. L'Unicode è invece già ben supportato dai principali mail provider, quali Yahoo, Google (Gmail), ecc.

Lo standard Unicode è utilizzato anche nell'ambito del Web. Tutte le raccomandazioni del W3C, ad iniziare dall'HTML 4.0, hanno indicato l'Unicode come insieme dei caratteri da utilizzarsi. I browser web già da alcuni anni supportano l'Unicode, soprattutto nella codifica UTF-8. Attualmente, l'Unicode è alla base del Web come standard internazionale.

Per un approfondimento dell'Unicode si rimanda al già citato libro "The Unicode Standard" o al sito internet del Consorzio Unicode, riportato nella bibliografia.

Una raccolta di tutte le tabelle Unicode, divise per alfabeti e simboli, si può trovare ancora in rete all'indirizzo <http://www.unicodetables.com/> o nella sezione charts (prospetti) del sito del consorzio all'indirizzo <http://unicode.org/charts/>.

Nelle righe seguenti si richiamano ed approfondiscono i cosiddetti "resource identifier" utilizzati, come già anticipa il termine, per identificare le risorse.

- . - . - . -

Un "Uniform Resource Identifier" (URI) è una stringa di caratteri che viene utilizzata per identificare univocamente una "risorsa" generica su Internet. Tale identificazione permette l'interazione con le rappresentazioni della risorsa attraverso una rete (tipicamente il WWW) usando i protocolli specifici. Gli "scheme" (schemi) specificano la sintassi precisa ed il protocollo che è associato a ciascun URI. Una risorsa, nel nostro caso, può essere ad esempio un sito web, un indirizzo di posta elettronica, un documento/file, un servizio disponibile in rete, ecc.

Un URI può essere classificato come "URL" o "URN" od entrambi:

- un URL (Uniform Resource Locator) è un URI che, oltre ad identificare una risorsa, fornisce dei mezzi per agire su di essa o per ottenerne una rappresentazione; inoltre, esso descrive il suo meccanismo di accesso primario e la sua "location" (ubicazione) in una rete.
- un URN (Uniform Resource Name) è un URI che identifica una risorsa mediante un "name" (nome) in un particolare dominio di nomi che viene detto "namespace". Un URN può essere utilizzato per descrivere una risorsa senza lasciare intendere la sua ubicazione o su come ottenerne una rappresentazione.

Per fare un esempio, mentre un URN può essere paragonato al nome di una specifica persona, l'URL può essere pensato come l'indirizzo della residenza di quella persona. Quindi, mentre l'URN identifica univocamente la persona, l'URL fornisce un mezzo per trovarla. Così facendo, gli URL e gli URN hanno scopi complementari anche se entrambi rientrano nell'ambito generale della "resource identification".

Un altro esempio tipico di URI, che può essere suddiviso in URN e in URL, viene dal sistema di classificazione dei libri "ISBN" (International Standard Book Number). In questo sistema internazionale ogni libro è classificato tramite un codice univoco: ad esempio ISBN 0470396792 corrisponde al libro "Semantic Web for Dummies" di J.T. Pollock: questo è l'URN di quel libro (urn:isbn: 0-470-39679-2). Al fine di poterlo reperire per leggerlo, si deve utilizzare l'URL che indica dove si può trovarne una copia disponibile. Un URL, di questo libro, è l'indirizzo internet:

<http://www.amazon.com/gp/product/0470396792>, che indica dove c'è una copia memorizzata in formato elettronico che, in questo caso, può essere acquistata on-line; questo URL, inoltre, dal punto di vista tecnico, fa capire che una rappresentazione della risorsa (il codice HTML della pagina per effettuare l'acquisto) è ottenibile tramite il protocollo HTTP da un host di rete chiamato www.amazon.com.

Il formato URI segue schemi prefissati, il cui registro principale è conservato presso lo IANA (Internet Assigned Numbers Authority)^[3.3], un organismo che ha responsabilità dell'assegnazione, ad esempio, degli indirizzi IP e delle altre "internet protocol resources". La sintassi di un URI dipende dallo quindi dallo schema. Le specifiche generiche della sintassi degli URI sono contenute nella RFC 3986^[3.4], (Request for Comments) pubblicata da Tim Berners-Lee nel 2005 nel sito della Internet Engineering Task Force (IETF)^[3.5], una comunità internazionale, aperta a tutti, composta da progettisti, ricercatori ed operatori interessati all'evoluzione dell'architettura di Internet e alla semplificazione delle operazioni su Internet stesso. Questa RFC 3986, che stabilisce la sintassi degli URI come un protocollo ufficiale di Internet, è la versione attuale di molte altre RFC precedenti. La prima di queste è la RFC 1630, pubblicata nel 1994, in cui sono raccolte le evoluzioni della sintassi degli URI, nate dalla necessità di uniformare le differenti visioni degli URI da parte del W3C e del IETF. Nella RFC 1630 si definiva l'URI come "Universal Resource Identifier", assieme alla prime definizioni e regole sintattiche; nel 1998, con la RFC 2396, il termine "Universal" divenne "Uniform" e furono migliorate ed espanse alcune specifiche tecniche. Una lista completa di tutte le RFC, con i relativi testi, è contenuta nel già citato sito della IETF.

L'idea iniziale di un URL fu implicitamente proposta, nel 1990, da Berners-Lee come una breve stringa rappresentante la risorsa che era "puntata" dall'hyperlink (collegamento ipertestuale) e chiamata "hypertext name".

Oggi, le pubblicazioni tecniche, quali gli standard prodotti dalla W3C e dalla IETF, non utilizzano il termine URL, dato che raramente si presenta la necessità di distinguere tra URI e URL. Invece, in ambiti non tecnici e nel software specifico per il Web, il termine URL è molto usato, come pure il termine "web address", che non ha una definizione formale, ma che si trova spesso in questo tipo di pubblicazioni come sinonimo di URL o di URI, sebbene esso si riferisca generalmente solo agli schemi URL di tipo "http" o "https".

Come già accennato, la sintassi di un Uniform Resource Identifier dipende dallo schema. Gli URI assoluti hanno una sintassi del tipo:

`<scheme>:<scheme-specific-part>`

dove `<scheme>` è il nome dello schema utilizzato e `<scheme-specific-part>` è una stringa la cui interpretazione dipende dallo schema e la cui sintassi non è la stessa per tutti i tipi di URI.

Esempi di URI, oramai conosciuti dalla maggior parte delle persone, sono:

- `http://www.dei.unipd.it` – schema per servizi HTTP;
- `mailto:buongior@dei.unipd.it` – schema per indirizzi di posta elettronica;

La sintassi degli URI impone talvolta delle restrizioni sulla `<scheme-specific-part>`.

Informazioni aggiuntive agli URI possono essere aggiunte tramite la “percent-encoding”, che viene presentata nelle prossime righe. Ad esempio, nello specificare un URL in uno schema HTTP, i caratteri che non appartengono alla codifica ASCII devono essere percent-encoded.

- • -

La percent-encoding (codifica percentuale), conosciuta anche come “URL encoding”, è un meccanismo usato per codificare, sotto certe circostanze, informazioni in un URI. Nonostante il nome, essa è di fatto utilizzata generalmente all’interno dell’insieme principale degli URI.

I caratteri utilizzati in un URI sono classificati in “reserved” (riservati) o “unreserved” (non riservati). I caratteri riservati sono quei caratteri che, talvolta, hanno un significato particolare all’interno di una specifica applicazione, al contrario di quelli non riservati. Utilizzando la percent-encoding, i caratteri che altrimenti non sarebbero permessi vengono rappresentati utilizzando i caratteri consentiti.

L’insieme dei caratteri riservati o non riservati e l’insieme delle circostanze sotto le quali certi caratteri hanno un significato particolare sono cambiati leggermente con ciascuna delle revisioni delle specifiche che regolano gli URI e gli schemi degli URI. Quando un carattere appartenente ad un insieme riservato (un carattere riservato) ha un significato speciale (uno scopo riservato) in un certo contesto, e lo schema URI dice che è necessario usare proprio quel carattere per alcuni altri scopi, allora il carattere deve essere percent-encoded. Fare la percent-encoding di un carattere riservato significa, quindi, convertire quel carattere nel suo corrispondente valore (in un byte) in codice ASCII e rappresentare poi quel valore come una coppia di cifre esadecimali. Le cifre, precedute dal segno di percentuale “%”, sono utilizzate al posto del carattere riservato. I caratteri non ASCII sono tipicamente convertiti nella loro sequenza (un byte) in codice UTF-8 e poi ciascun valore è rappresentato come detto sopra.

Ad esempio, il carattere riservato “/” se usato nel componente “path” di un URI ha il significato di essere un divisore tra segmenti di “path”. Se, in accordo con un determinato schema URI, “/” è necessario che sia all’interno di un segmento di “path”, allora deve essere utilizzata la codifica “%2F” o “%2f” al posto di “/”.

I caratteri non riservati possono anch’essi essere percent-encoded nella stessa maniera di quelli riservati.

- • -

Il Web Semantico usa gli schemi URI HTTP per identificare sia i documenti che i concetti nel mondo reale: questo ha talvolta causato confusione su come distinguere i due. Nel Marzo del 2008 il W3C ha pubblicato il documento "Interest Group Note Cool URI's for the Semantic Web" ^[3,6], che spiega in maniera precisa e dettagliata l'uso di questi ultimi.

Si accenna, per completezza, anche allo URI reference.

Un "URI reference" (riferimento, rimando) è un tipo di stringa che rappresenta un URI e, subito dopo, la risorsa identificata da quell'URI. Nell'uso informale spesso non si mantiene la distinzione tra URI e URI reference ma, ad esempio, nei file dei protocolli dei dati tale ambiguità non è permessa.

Un URI reference può avere la forma completa di un URI o solamente la porzione `<scheme-specific>`, od anche alcune componenti che seguono quest'ultima, compresa la stringa vuota. Un frammento identificatore opzionale, preceduto dal carattere "#", può essere presente alla fine dell'URI reference. La parte del riferimento che precede il simbolo "#" identifica indirettamente una risorsa, mentre il frammento identificatore identifica alcune parti specifiche di quella risorsa.

Un esempio di URI reference è:

```
http://en.wikipedia.org/wiki/URI#Examples_of_URI_references
```

dove "http" specifica il nome dello "schema", "en.wikipedia.org" è la "authority", "/wiki/URI" è il "percorso" che punta alla voce contenuta in Wikipedia relativa agli URI, e "#Examples_of_URI_references" è il "frammento" che punta alla parte di testo relativa agli esempi.

Un'altra forma dell'URI reference è quella più nota dell'URI; ad esempio:

```
http://www.unipd.it .
```

Al fine di recuperare un URI da un URI reference, esistono dei software che convertono l'URI reference in un URI assoluto fondendolo con un URI di base, secondo uno specifico algoritmo. L'URI di base solitamente identifica il documento che contiene l'URI reference.

Attualmente, i linguaggi di mark-up dei documenti web usano frequentemente gli URI reference per collegarsi ad altre risorse, quali documenti esterni o specifiche porzioni dello stesso documento.

- • - • -

Un "Internationalized Resource Identifier" (IRI), è una forma generale di Uniform Resource Identifier ed è, attualmente, uno standard proposto dallo IETF.

Un IRI, a differenza di un URI, è costituito da una sequenza di caratteri appartenenti all' Universal Character Set (UCS) (Unicode/ISO 10646): ciò significa che al suo interno possono occorrere caratteri non appartenenti all'insieme ASCII, compresi caratteri dell'alfabeto Cinese, Giapponese (Kanji), Coreano, ecc.

Un IRI può essere convertito in URI seguendo delle precise regole. La sintassi precisa è definita nel documento RFC 3987^[3.7], disponibile presso la IETF.

L'utilizzo del IRI presenta dei vantaggi e degli svantaggi rispetto all'uso degli URI. Un vantaggio deriva dalla possibilità di vedere gli URI visualizzati in diverse lingue: ciò è una facilitazione per gli utenti che non hanno familiarità con l'alfabeto latino (A-Z).

Un svantaggio si ha quando si mescolano assieme IRI e URI di tipo ASCII: così facendo si può rendere più facile subire attacchi di pirateria informatica di tipo "phishing" (cioè, si ha una replica illegale di una pagina web già esistente per ottenere dati privati o la parola d'ordine per accedervi come utente), i quali possono ingannare l'utente facendolo credere di essere su di un sito, quando realmente non lo è. Per esempio, sostituendo la "a" in www.ebay.com con un carattere che assomiglia ad una "a", si può far sì che l'IRI punti ad un sito "maligno".

Un altro esempio di svantaggio si ha quando degli utenti, che utilizzano tastiere con lingua diversa, cercano di accedere a delle risorse i cui identificatori comprendono caratteri che la loro tastiera non può generare. Tutto questo può portare ad IRI effettivamente molto meno universali degli URI.

Non si approfondisce ulteriormente il discorso sugli IRI, in quanto esula dal lavoro di questa tesi. Per chi volesse approfondire l'argomento, molto materiale può essere trovato in rete, anche nel sito della W3C nella sezione Internationalization (I18n) Activity.^[3.8]

Per la completezza della trattazione degli resource identifier si illustra, brevemente, anche lo standard XRI.

- . - . - . -

L' "eXtensible Resource Identifier" (XRI) è uno schema ed un protocollo per la definizione di identificatori astratti, compatibile con l'Uniform Resource Identifier e con l'Internationalized Resource Identifier. È sviluppato dallo "XRI Technical Committee"^[3.9] presso il consorzio OASIS.

L'obiettivo degli sviluppatori dello XRI è quello di fornire una sintassi standard e di sviluppare formati per gli identificatori astratti e strutturati, quali domini, locazioni, applicazioni e per il trasporto indipendente, in modo tale che questi possano essere distribuiti attraverso un qualsiasi numero di domini, di directory e di protocolli di comunicazione.

Lo XRI, nella versione 2.0, tra varie difficoltà è fallito nel divenire un standard OASIS. Ciò è dovuto all'intervento del W3C Technical Architecture Group il quale ha fatto una dichiarazione contro l'utilizzo degli XRI e nel proseguo delle specifiche XRI. Il cuore della disputa riguardava se o meno l'ampia interoperabilità degli URI HTTP era in grado di soddisfare il ruolo di astrazione degli identificatori strutturati.

I progettisti dell'XRI credevano che, a causa dell'aumento dell'XML, dei Web Services e di altri modi di adattare il Web all'automatizzazione, le comunicazioni di tipo machine-machine fossero, in maniera sempre più crescente, importanti per identificare una risorsa in maniera indipendente dalla locazione, dal percorso fisico sulla rete o dal protocollo utilizzato; tutto questo al fine di:

- creare identificatori strutturati che descrivono in maniera propria i "tag" che possono essere compresi nei vari domini;

- mantenere un collegamento persistente alla risorsa senza preoccuparsi se la sua “location” nella rete cambia;
- delegare la gestione degli identificatori non solo nella parte di segmento dell’authority (cioè il primo segmento che segue il nome dello schema “xxx://”), ma ovunque nel path dell’identificatore;
- mappare gli identificatori utilizzati per identificare una risorsa in un dominio in altri “sinonimi” usati per identificare la stessa risorsa nello stesso dominio, o in altri domini.

Questo lavoro ha portato, dall’inizio del 2003, alla pubblicazione di un protocollo basato sull’HTTP e sui documenti XML chiamato “eXtensible Resource Descriptor Sequence” (XRDS). Esso è un formato, basato sull’XML, utilizzato per scoprire e presentare metadati su di una risorsa: nello specifico si usa per ritrovare servizi associati con una risorsa: processo questo noto come “service discovery”. Ad esempio, un sito web che offre un OpenID login può determinare l’OpenID identifier dell’utente su un documento XRDS per trovare la “location” del fornitore del servizio dell’utente.

Per un approfondimento di tale argomento, Internet si rivela essere una grande fonte di informazioni. Un sito fondamentale è quello dell’OASIS, nella parte relativa al lavoro svolto dal Comitato Tecnico dell’XRI. ^[3.9]

- • - • - • -

Fino a questo punto sono stati illustrati gli standard utilizzati dal Web semantico per rappresentare e localizzare le risorse di interesse dell’utente. Si passa ora al secondo livello della piramide tecnologica, rappresentato dai linguaggi di mark-up, necessari per fornire un insieme di specifiche per modellare la struttura dei documenti, dei dati e, più in generale, delle risorse.

3.2 XML, XML Schema

L’XML, acronimo di “eXtensible Markup Language”, è un insieme di regole per codificare i documenti in maniera elettronica. E’ definito dalla specifica XML 1.0 prodotta dal consorzio W3C e da altre specifiche relative; sono tutti standard aperti. Essendo una delle tecnologie fondamentali del Web semantico, verrà ampiamente sviluppato nelle righe che seguono.

Lo scopo dei progettisti dell’XML è stato quello di enfatizzare la semplificazione, la generalizzazione e l’utilizzabilità attraverso Internet. L’XML è un formato per i dati testuali con un buon supporto, tramite l’Unicode, di tutti i linguaggi utilizzati nel mondo. Sebbene il progetto dell’XML si focalizzi sui documenti, esso è anche largamente utilizzato per la rappresentazione di strutture dati arbitrarie, ad esempio nei web services. Esistono, a tal fine, moltissime interfacce di programmazione che i sviluppatori di software possono utilizzare per accedere ai dati XML oltre a svariati schemi di sistema progettati in aiuto alla definizione dei linguaggi basati sull’XML.

Attualmente sono centinaia i linguaggi basati sull’XML che sono stati sviluppati, qual è ad esempio l’XHML, introdotto nei capitoli precedenti. Inoltre i formati basati sull’XML sono diventati il “default” per molti strumenti per la produzione di

documenti per gli uffici, quali Microsoft Office (Office Open XML), OpenOffice.org (OpenDocument) e Apple's iWork.

Come già anticipato tutte le specifiche dell'XML sono curate dal consorzio W3C e pubblicate nei Technical Report (TR) come recommendation ^[3.10]. La trattazione completa di queste specifiche sarebbe lunga ed impegnativa ed esulerebbe dal lavoro di questa tesi. Tuttavia, per la comprensione dei paragrafi successivi di questo lavoro, è necessario introdurre i principali key constructor (costrutti chiave) che compongono il linguaggio XML:

- Character: per definizione, un documento XML è una stringa di caratteri. La maggior parte di quelli che sono codificati nell'Unicode possono essere utilizzati all'interno di un documento XML.
- Processor ed Application: sono entrambi costituiti dal software che elabora i documenti XML. Si può supporre che un processore lavori al servizio di una applicazione. Il processore, come viene chiamato dalla specifica, è spesso correntemente riferito all'XML "parser", cioè al programma di analisi che suddivide l'ingresso dei dati di testo in parti più piccole per l'analisi sintattica.
- Markup e Content: i caratteri che costituiscono un documento XML possono essere divisi in markup e in contenuto: possono essere distinti tra di loro applicando alcune semplici regole sintattiche. Tutte le stringhe che costituiscono markup o iniziano con il carattere "<" e terminano con il carattere ">" o iniziano con il carattere "&" e terminano con il carattere ";". Ovviamente, le stringhe che non sono classificate come markup sono content.
- Tag: è un tipo di costrutto di markup che inizia con "<" e termina con ">". I tag si suddividono in tre tipi: "start-tag", ad esempio <section>, "end-tag", ad esempio </section> e l' "empty-element-tag", ad esempio <line-break/>.
- Element: è un elemento logico di un documento che inizia con uno start-tag e termina con un corrispondente end-tag, oppure consiste solamente di un empty-element-tag. I caratteri contenuti tra lo start-tag e l'end-tag sono il contenuto di quell'elemento; essi possono contenere markup, comprendere anche altri elementi che sono chiamati, in questo caso, "child element". Un esempio di elemento è <Nome> Davide </Nome>, dove la stringa "Davide" è il contenuto ed il tag è "Nome".
- Attribute: è un costrutto di markup consistente di una coppia nome/valore che esiste entro uno start-tag od un empty-element-tag. Un esempio di attributo è: , dove l'elemento img ha i due attributi src ed alt.
- XML Declaration: i documenti XML possono dichiarare, nella loro parte iniziale, alcune informazioni su loro stessi. Ad esempio, la riga <?xml version="1.0" encoding="UTF-8" ?> indica che la versione dell'XML utilizzata nel documento è la 1.0 e che la codifica dei caratteri è la UTF-8.

Nelle righe seguenti si riporta un esempio completo di un documento XML, comprendente anche i child element:

```
<?xml version="1.0" encoding="UTF-8"?>
<utenti>
  <utente>
    <nome>Antonio</nome>
    <cognome>Rossi</cognome>
    <indirizzo>Milano</indirizzo>
  </utente>
  <utente>
    <nome>Mario</nome>
    <cognome>Verdi</cognome>
    <indirizzo>Roma</indirizzo>
  </utente>
</utenti>
```

Un documento XML consiste interamente di caratteri appartenenti al repertorio dell'Unicode, ad eccezione di un piccolo numero di caratteri di controllo specificatamente esclusi. L'XML permette l'utilizzo di qualsiasi tipo di codifica dei caratteri che sia basata sull'Unicode: esso fornisce un meccanismo per mezzo del quale un XML processor può affidarsi, senza averne alcuna conoscenza a priori, per determinare quale codifica è stata utilizzata nel documento: questo meccanismo è utile dato che alcune codifiche diverse dalla UTF-8 e dalla UTF-16 non sempre sono riconosciute dal parser XML.

Come già visto prima con il caso dell'Unicode, anche l'XML presenta delle difficoltà o, addirittura, delle impossibilità nel includere alcuni caratteri direttamente in un documento XML. Questo è dovuto al fatto che alcuni caratteri, quali ad esempio "<", "&" sono "key syntax markers", cioè utilizzati per le marcature o anche dal fatto che alcuni di questi caratteri non sono presenti nelle tastiere degli utenti. Per queste ragioni l'XML fornisce delle "escape facilities", cioè delle "scappatoie" per risolvere questi problemi. Ci sono cinque "predefined entities", cioè entità predefinite: `<` rappresenta il carattere <, `>` rappresenta il carattere >, `&` rappresenta &, `'` rappresenta l'apice ' e `"` rappresenta le virgolette " .

All'interno di un documento XML è possibile posizionare dovunque dei commenti, basta che si trovino al di fuori di altri markup. Essi, comunque, non devono apparire nella prima riga o sopra la riga di dichiarazione dedicata al processor per la compatibilità. Un commento ha una sintassi del tipo:

```
<!-- qui posso scrivere il testo del mio commento --> .
```

Le specifiche XML definiscono un "documento XML" come un testo che è "well-formed" (ben definito/ben formattato), cioè esso soddisfa una lista di regole sintattiche che sono fornite nelle specifiche. La lista completa delle specifiche per il documento è discretamente lunga. I punti chiave sono:

- il documento contiene solo caratteri stabiliti in modo appropriato dallo standard Unicode;
- nessuno dei caratteri speciali per la sintattica, quali > e " deve essere utilizzato, tranne che nelle regole del markup;
- i tag di inizio, di fine che delimitano i gli elementi sono correttamente nidificati con nessuna mancanza e nessuna sovrapposizione: devono inoltre corrispondere esattamente (devono cioè essere bilanciati);
- c'è un unico elemento "root" (radice) che contiene tutti gli altri elementi.

La definizione di un documento XML esclude testi che contengano violazioni delle regole di well-formed.

Oltre ad essere well-formed un documento XML deve essere “valid” (valido). Questo significa che esso contiene un riferimento ad un “Document Type Definition” (DTD), che tutti i suoi elementi ed attributi sono dichiarati in quel DTD e che seguono le regole grammaticali che il DTD specifica per loro.

I processori XML si classificano in “validating” (validatori) o “non-validating”, a seconda se essi controllino, o meno, la validità del documento. Un processore che scopre un errore di validità deve essere in grado di segnalarlo senza interrompere il processo che sta eseguendo. Un DTD è un esempio di quello che sarà chiamato “schema” o “grammar”.

Sin dalla pubblicazione della versione 1.0 dell’XML c’è stato un gran lavoro nell’area degli “schema language per l’XML” (linguaggi di descrizione del contenuto di un documento XML, o linguaggi per gli schemi XML). Tali schema language, tipicamente, vincolano l’insieme degli elementi che possono essere usati in un documento, quali attributi possono essere loro attribuiti, l’ordine in cui possono apparire e l’ammissibilità delle relazioni di tipo padre/figlio.

Qui sotto si analizzano, in breve, i più noti schema language:

- **DTD (Document Type Definition)**: è il più vecchio schema language per l’XML. Esso deriva dall’SMGL (Standard Generalized Markup Language), che verrà presentato nelle prossime pagine. Il DTD è un documento attraverso cui si specificano le caratteristiche strutturali di un documento XML attraverso una serie di “regole grammaticali”. Esso, inoltre, definisce l’insieme degli elementi del documento XML, le relazioni gerarchiche tra gli elementi, l’ordine di apparizione e quali elementi ed attributi sono opzionali o meno. Il DTD presenta dei vantaggi: esso è supportato dovunque, grazie alla sua inclusione nello standard XML 1.0; è, rispetto ad altri linguaggi, molto conciso e dà la possibilità di mostrare molte informazioni in una sola schermata. Il DTD definisce quindi un “document type” che raggruppa tutti i vincoli di un documento in una singola raccolta. Tuttavia, sono presenti anche degli svantaggi, ad iniziare dal mancato supporto delle nuove caratteristiche dell’XML, quali i namespace. Inoltre c’è la mancanza di espressività, in quanto certe strutture non possono essere espresse con grammatiche regolari. Un altro aspetto negativo deriva dalla mancanza di leggibilità dei DTD. Due sono le caratteristiche che distinguono i DTD da altri tipi di schema: la prima è il supporto sintattico per includere un DTD all’interno di un documento XML; la seconda è data dalla possibilità di definire delle “entities” (entità) che sono dei pezzi di testo e/o markup che il processore XML inserisce nel DTD e nel documento XML se essi sono referenziati (referenced). La tecnologia DTD è ancora utilizzata in molte applicazioni data la sua vasta diffusione.
- **XML Schema**: è il linguaggio più recente per gli schemi, descritto dalla W3C come il successore ed il sostituto della DTD stessa. La sua sigla è XSD, acronimo di “XML Schema Definition”. Come si vedrà nei prossimi paragrafi, gli XSD sono molto più potenti dei DTD nel descrivere i linguaggi XML. Essi usano un sistema di tipi di dato più ricco e consentono l’uso di vincoli più dettagliati sulla struttura logica di un documento XML. Nei prossimi paragrafi si svilupperà in maniera più dettagliata questo linguaggio.

- RELAX NG: è stato inizialmente specificato dal consorzio OASIS e, attualmente, è uno standard ISO, come una sezione del DSDL (che è presentato più in basso). Gli schemi di tipo RELAX NG possono essere scritti o in una sintassi basata sull'XML o con una sintassi più compatta di tipo non XML; queste due sintassi sono isomorfe. RELAX NG ha una definizione ed una struttura per la validazione più semplici rispetto all'XML Schema; inoltre è in grado di utilizzare strutture di tipi di dato aggiuntive (plug-in).
- Schematron: è un linguaggio che permette di fare asserzioni sulla presenza o la mancanza di modelli all'interno di un documento XML. E' usato nelle espressioni della specifica XPath (vedi sotto).
- ISO DSDL (Document Schema Description Languages): questo standard riunisce un insieme completo di piccoli linguaggi di schema, ciascuno relativo ad un problema specifico. Esso include la sintassi completa e compatta di RELAX NG, il linguaggio per le asserzioni di Schematron e linguaggi per definire i tipi di dato, oltre ad un repertorio di vincoli sui caratteri ed altre caratteristiche importanti. Tuttavia, il DSDL non ha ancora un supporto come l'XML Schema, che resta ancora il più utilizzato nell'ambito lavorativo.

Esistono anche altri linguaggi per gli schemi, che descrivono la struttura di un particolare formato XML e che offrono anche delle piccole facilitazioni per l'elaborazione di singoli documenti XML relativi a questo specifico formato.

Con la pubblicazione dell'XML 1.0, sono state sviluppate un gruppo di specifiche relative all'XML. E' frequente infatti il caso che il termine "XML" sia usato per riferirsi all'XML assieme ad una, o più, di queste tecnologie, che possono essere viste come un parte del nucleo dell'XML. Le principali specifiche sono:

- XML Namespace: permette allo stesso documento di contenere elementi XML presi da diversi vocabolari, senza che accadano conflitti di nome;
- XML Base: definisce l'attributo `xml:base` che può essere usato per impostare la base per risoluzione dei riferimenti relativi degli URI all'interno dell'estensione di un singolo elemento XML;
- XML infoset: chiamato anche XML Information Set descrive un modello di dati astratto per i documenti XML in termini di "information item" (argomento d'informazione). E' utilizzato per descrivere i vincoli sui costrutti XML che sono permessi dai vari linguaggi;
- xml:id: asserisce che un attributo chiamato `xml:id` funziona come un "ID attribute" (attributo identificativo) nel senso utilizzato nei DTD;
- XPath: definisce una sintassi chiamata "XPath expression" la quale identifica uno o più dei componenti interni (quali elementi, attributi, ecc.) inclusi in un documento XML.
- XLS (eXstensible Stylesheet Language): è il linguaggio con cui si descrive il foglio di stile di un documento XML. La sua versione estesa è l'XLST.
- XLST (eXstensible Stylesheet Language Transform): è un linguaggio con una sintassi di base di tipo XML usato per trasformare un documento XML in un altro documento XML, HTML, o in un altro formato non strutturato come un RTF (Rich Text Format). Attualmente è largamente implementato nei software lato server e in molti browser web;

- XLS-FO (eXtensible Stylesheet Language Formatting Objects): è un linguaggio di markup per la formattazione di documenti XML, ad esempio per generare i documenti di tipo PDF;
- XQuery: è un linguaggio di query orientato all'XML, fortemente radicato nell'XPath e nel XML Schema. Fornisce un metodo per accedere e manipolare dati, ottenendo come risultato un documento XML;
- XML Signature: definisce una sintassi e delle regole per creare firme digitali sui contenuti XML;
- XML Encryption: definisce una sintassi e delle regole per criptare il contenuto di un documento XML.

Esistono altre specifiche, nate come parte del nucleo dell'XML, ma che hanno fallito nel trovare una vasta applicazione. Tra queste si segnalano XInclude, XLink e XPointer, che non vengono qui sviluppate.

L'XML è usato principalmente per l'interscambio di dati attraverso Internet. La RFC 3023 ^[3.11] fornisce le regole per la costruzione di "Internet Media Types" da usarsi quando si spedisce documentazione in formato XML. Essa definisce i tipi "application/xml" e "text/xml" che specificano che il dato è in XML, ma non dicono nulla circa la semantica. Il termine "text/xml" sta per essere deprecato.

L'RFC 3023 raccomanda che i linguaggi basati sull'XML siano determinati con il media type, nel formato che inizi con "application/" e che termini con "+xml". Ad esempio, l'SVG viene indicato con "application/svg+xml".

Un'altra specifica, la RFC 3470^[3.12], pubblicata nel 2003, contiene delle linee guida per l'utilizzo dell'XML: in questa sono contemplati molti aspetti per il progetto e lo sviluppo di linguaggi basati sull'XML.

Uno degli obiettivi degli sviluppatori dell'XML è: "...dovrebbe essere facile scrivere programmi che elaborino i documenti scritti in XML". Nonostante questo, la specifica XML non contiene informazioni su come i programmatori dovrebbero fare tale procedimento. L'XML Infoset fornisce un vocabolario per riferirsi ai costrutti interni di un documento XML, ma non fornisce una guida su come accedere direttamente a queste informazioni. A tal fine sono state sviluppate, e sono quindi utilizzate, un certo numero di API per elaborare (leggere e modificare) i documenti scritti in XML. Talune di queste sono state anche standardizzate. Possono essere divise in 4 categorie:

- Stream-oriented APIs: accessibili da un linguaggio di programmazione quale SAX;
- Tree-transversal APIs: accessibili da un linguaggio di programmazione quale DOM;
- XML data binding: che forniscono una "traduzione" automatica tra documento XML e linguaggio di programmazione ad oggetti;
- Declarative transformation languages: quali, ad esempio, XSLT e XQuery.

Le API di tipo Stream-oriented richiedono meno memoria rispetto alle Tree-transversal e al data binding che sono, talvolta, più convenienti per l'uso da parte dei programmatori. L'XQuery ha le stesse funzionalità dell'XSLT, ma è più specifico per la ricerca su grandi database basati sull'XML.

Si presentano, in breve, alcune di queste interfacce di programmazione:

- SAX (Simple API for XML): questa interfaccia permette di leggere in maniera “sequenziale” e di modificare i documenti XML. Attraverso di essa è possibile implementare degli XML parser specifici. E’, inoltre, “event-based” (basata sugli eventi) e reagisce quindi agli eventi di parsing facendo rapporto all’applicazione, ma lasciando al programmatore la gestione dell’evento di parsing.
- DOM (Document Object Model): è più semplice da utilizzare rispetto a SAX. Esso costruisce, partendo dal file XML, un albero dove ogni nodo dell’albero corrisponde ad un elemento del file: per questo motivo fa parte della categoria tree-transversal. Permette la lettura “trasversale” del contenuto del documento.
- Data Binding: qui i dati XML sono resi disponibili come una gerarchia di “custom”; è fortemente tipizzato in classi, al contrario degli oggetti generici creati dal DOM. Questo approccio semplifica lo sviluppo del codice e, talvolta, permette di risolvere problemi durante la compilazione e non durante l’esecuzione del codice.

Storia dell’XML

Il W3C, in seguito alla “guerra dei browser”, ovvero la situazione verificatasi negli anni '90 nella quale la Microsoft e la Netscape Communication introducevano, con ogni nuova versione del proprio browser, un'estensione proprietaria dell'HTML ufficiale, fu costretto a seguire queste estensioni individuali del linguaggio HTML. Il W3C dovette scegliere quali caratteristiche standardizzare e quali lasciare fuori dalle specifiche ufficiali dell'HTML. Fu in questo contesto che iniziò a delinearsi la necessità di un linguaggio di markup che desse maggiore libertà nella definizione dei tag, pur rimanendo in uno standard.

Il "progetto XML", che ebbe inizio alla fine degli anni '80 nell'ambito della “SGML Activity” del W3C, suscitò un così forte interesse a tal punto che la W3C stessa creò un gruppo di lavoro, chiamato “XML Working Group”, composto da esperti mondiali delle tecnologie SGML, ed una commissione, la “XML Editorial Review Board”, deputata alla redazione delle specifiche del progetto.

Nel febbraio del 1998 le specifiche divennero una raccomandazione ufficiale con il nome di “eXtensible Mark-up Language”, versione 1.0. Ben presto ci si accorse che l’XML non era solamente limitato al contesto del Web, ma era qualcosa di più: uno strumento che permetteva di essere utilizzato nei più diversi ambiti, dalla definizione della struttura di documenti, allo scambio delle informazioni tra sistemi diversi, dalla rappresentazione di immagini, alla definizione di formati di dati, ecc.

Versioni dell’XML

L’XML è una sezione dello standard ISO SGML, dal quale è tratto senza cambiamenti. Da esso derivano la separazione tra struttura logica e fisica (elementi ed entità), la disponibilità della validazione basata sulla grammatica (DTD), la separazione tra dati e metadati (elementi ed attributi), i contenuti misti e la separazione del processo dalla rappresentazione. Altre parti derivano dall’HTML e da altre sintassi di altri linguaggi meno noti.

Attualmente, esistono due versioni dell'XML.

La prima, la XML 1.0, fu inizialmente definita nel 1998. Da allora è stata sottoposta a poche revisioni e non è stato dato un nuovo numero ad ogni versione. E', al momento, alla quinta edizione pubblicata il 26 Novembre 2008. Questa versione è largamente implementata e ancora raccomandata per un uso generale.

La seconda versione, XML 1.1, fu inizialmente pubblicata il 4 Febbraio 2004 e, attualmente, è alla sua seconda edizione, pubblicata il 16 Agosto 2006. Essa contiene caratteristiche, alcune controverse, che sono intese a rendere più semplice l'XML per l'uso in certi casi: principalmente permettendo l'uso di caratteri di line-ending usati nell'EBCDIC e l'uso di script e caratteri mancanti dall'Unicode 3.2. L'XML 1.1 non è largamente implementato ed è raccomandato per l'uso solamente da chi necessita delle sue caratteristiche specifiche.

Prima della sua quinta edizione, l'XML 1.0 differiva molto dall'XML 1.1 avendo requisiti rigorosi per i caratteri disponibili per l'uso nei nomi dei elementi e degli attributi e negli identificatori. La quinta edizione dell'XML 1.0 sostituisce alcuni meccanismi dell'XML 1.1, il quale è molto avveniristico ma ha il pregio di ridurre la ridondanza. L'approccio preso nella quinta edizione dell'XML 1.0, ed in tutte le edizioni dell'XML 1.1, è che solo certi caratteri sono proibiti nei nomi e tutte le altre cose sono permesse; questo al fine di consentire l'uso dei caratteri delle future versioni dell'Unicode.

C'è stata una discussione su un "XML 2.0", anche se nessuna organizzazione ha annunciato intenzioni di lavorare su tale progetto. Esiste un "XML Skunk Works" (XML-SW) (lavoro per sconfiggere l'XML), scritto da uno degli sviluppatori originari dell'XML, che contiene alcune proposte alle quali un XML 2.0 potrebbe assomigliare: l'eliminazione del DTD dalla sintassi, l'integrazione dei namespace e l'utilizzo di XML Base e XML infoset come standard di base.

Il Consorzio W3C ha anche un "XML Binary Characterization Working Group" che sta facendo degli studi preliminari sugli "use cases" e sulle proprietà al fine di sviluppare una codifica binaria dell'XML infoset. Costoro non hanno, tuttavia, l'abilitazione a produrre uno standard ufficiale.

La presentazione dell'XML potrebbe dilungarsi ancora per molte pagine, ma ciò esulerebbe dal lavoro di questa tesi. Il lettore che volesse approfondire l'XML può trovare in rete tantissimo materiale, sia tecnico che didattico. Il sito di riferimento resta quello del W3C, nella pagine dedicate all'XML, riportate in bibliografia.^[3.13]

Altro materiale utile si può trovare nel sito XML.gov^[3.14], un sito sponsorizzato da alcune grosse aziende informatiche, quali IBM, Microsoft, Oracle, Sun, ecc. il cui scopo è quello di "portare avanti l'uso degli standard aperti fornendo informazioni educative, aree di discussione e risorse per la collaborazione". Guide, manuali e risposte alle domande più frequenti, FAQ (Frequently asked questions), sull'XML si possono trovare anche nella sezione^[3.15] relativa all'XML del sito html.it, sito che spazia anche su moltissimi altri argomenti informatici relativi all'HTML.

- • -

Dato che l'XML deriva dallo SGML, si accenna anche a questo linguaggio, senza la pretesa di completezza.

Lo "Standard Generalized Markup Language" (SGML), è uno standard per la descrizione logica dei documenti. Discende dal "Generalized Markup Language" della IBM. L'idea centrale dello standard è l'uso di un tipo di marcatura generica chiamata "marcatura descrittiva" che definisce la struttura logica dei documenti. L'organizzazione di un documento non è espressa usando la codifica dei sistemi di scrittura, che è finalizzata alla presentazione grafica, ma sono solo evidenziate le parti in cui è strutturato il documento (ad esempio paragrafi, capitoli) insieme ad altre particolarità del testo come note, tabelle, intestazioni, ecc.

SGML fu inizialmente sviluppato per permettere lo scambio di documenti machine-readable (leggibili da un computer) in progetti governativi, legali ed industriali, che devono rimanere leggibili per diverse decadi. Inizialmente fu usato per pubblicazione di testo e basi di dati, recentemente una delle sue maggiori applicazioni è stata la seconda edizione dell'Oxford English Dictionary (OED), che era ed è interamente formattato usando un linguaggio SGML.

- . - . - . -

Un "XML schema" è una descrizione di un tipo di documento XML, tipicamente espressa in termini di vincoli sulla struttura e sul contenuto dei documenti di quel tipo, che sta sopra e va al di là dei vincoli sintattici di base imposti dall'XML stesso. Questi vincoli sono generalmente espressi usando delle combinazioni di regole grammaticali che governano l'ordine degli elementi.

Esistono dei linguaggi specificatamente sviluppati per esprimere gli schemi XML. Il linguaggio DTD, che è nato come specifica XML, è un linguaggio per gli schemi che ha una capacità relativamente limitata, ma che ha anche altri utilizzi oltre all'XML, per l'espressione degli schemi.

Due linguaggi per gli schemi XML, molto espressivi e largamente utilizzati, sono l'XML Schema (W3C), che verrà presentato nelle prossime righe, ed il già visto RELAX NG.

Il meccanismo per associare un documento XML ad uno schema varia a seconda del linguaggio. L'associazione può essere ottenuta o attraverso il mark-up all'interno del documento stesso o tramite alcuni meccanismi esterni.

Il processo di controllo della conformità di un documento allo schema è detto "validation" (validazione), che, si ricorda, è diverso dal concetto di "well-formed" (ben definito). La validazione di un'istanza di un documento su di uno schema può essere considerata come un'operazione concettuale distinta dal parsing (analisi) XML. Tutti i documenti XML devono essere well-formed, ma non è richiesto che un documento sia valido, a meno che l'analisi del parser XML sia anche "validante", in qual caso il documento è anche conforme al suo schema associato. I documenti sono, quindi, considerati "validi" se soddisfano i requisiti dello schema a cui sono stati associati. Questi requisiti solitamente includono alcuni vincoli: 1) quali elementi ed attributi che devono/possono essere inclusi e quale struttura sia loro permessa; 2) la struttura deve essere specificata da un regolare espressione sintattica; 3) come i caratteri rappresentanti dati devono essere interpretati (ad esempio, booleani, ecc.).

Si fa notare una confusione che capita nell'utilizzo dei termini "Schema" e "schema". Il termine scritto in minuscolo è un termine generico e può riferirsi a

qualsiasi tipo di schema, quale DTD, RELAX NG, ecc. e dovrebbe essere sempre scritto in minuscolo, tranne quando è all’inizio di una frase. Il termine “Schema” è attualmente sempre utilizzato nella comunità dell’XML per riferirsi allo XML Schema del W3C (scritto quindi XML Schema W3C) e presentato qui sotto.

E’ disponibile una lista di schemi XML, divisi per scopo e con un collegamento ad una pagina web relativa ad ognuno di essi, nel sito inglese di Wikipedia alla voce “List_of_XML_schemas”^[3.16].

- • -

L’ “XML Schema (W3C)” è una raccomandazione pubblicata dalla W3C nel Maggio del 2001. E’ pubblicata nella versione 1.0 e divisa in due parti: Structures e Datatype. E’ uno dei diversi linguaggi di schemi XML. E’ stato il primo ed unico schema language ad ottenere lo stato di raccomandazione dal W3C. A causa della confusione tra XML Schema e XML schema, già accennata in precedenza, alcune comunità di utenti si riferiscono al linguaggio della W3C con la sigla WXS che abbrevia “W3C XML Schema”, mentre altri utenti indicano ciò con la sigla XSD che sta per “XML Schema Document”, che è inteso come un documento scritto nel linguaggio XML Schema, solitamente contenente il prefisso namespace XML “xsd” e memorizzato in un file con estensione “.xsd”. Nella bozza della più recente versione, la 1.1, il W3C ha scelto di adottare il termine XSD e, in queste righe, si farà riferimento a questo.

Come tutti gli altri linguaggi per gli schemi XML, l’XSD può essere usato per esprimere un insieme di regole alle quali un documento XML deve essere conforme al fine di essere considerato valido; diversamente dagli altri linguaggi, l’XSD è stato progettato con la particolarità che, se il documento risulta valido, sarà prodotto un insieme di informazioni (infoset) corrispondenti agli specifici tipi di dati. Ciò può essere visto come un “post-validation infoset”, che risulta utile per i software che elaboreranno questi documenti XML.

La caratteristica più evidente, offerta dall’XSD, che non era disponibile all’inizio dell’XML nei DTD, è la presenza dei namespace e dei tipi di dato: questi ultimi definiscono il contenuto di elementi ed attributi, come valori (ad esempio interi, ecc.) e non solo come stringhe di testo. L’XML Schema permette al contenuto di un elemento o attributo di essere validato su di un determinato tipo di dato. Sono disponibili 19 tipi di dato primitivi (interi, booleani, decimali, data, tempo, ecc.) ed è possibile la costruzione di nuovi tipi partendo dai tipi primitivi: 25 tipi derivati sono già definiti per l’utente.

Si è visto che la ragione principale per definire uno schema XML è per descrivere formalmente un documento XML; tuttavia lo schema che ne risulta ha un numero di altri usi che va al di là della semplice validazione.

Lo schema può essere usato per ottenere della documentazione leggibile dall’uomo: questo è usato specialmente dove gli autori hanno fatto uso di commenti o di elementi di annotazione. Al momento non esiste uno standard per la generazione di documentazione.

Lo schema può anche essere utilizzato per generare codice: questo viene riferito come “XML Data Binding”. Questo codice permette al contenuto del documento XML di essere trattato come un oggetto all’interno di un ambiente di programmazione.

Nell'Agosto del 2009 la versione 1.1 dell'XSD è diventata una "raccomandazione candidata", che significa che è nella fase di revisione finale prima di diventare una specifica approvata dal W3C. Le nuove caratteristiche principali sono:

- l'abilità di definire asserzioni sul contenuto del documento tramite le espressioni utilizzate nell'XPath 2.0;
- la possibilità di scegliere il tipo sul quale un elemento sarà validato, basato sui valori degli attributi dell'elemento stesso ("conditional type assignment");
- l'abilità di specificare caratteri speciali, sia per gli elementi che per gli attributi, che possono essere applicati a tutti i tipi presenti nello schema, cosicché essi possano implementare la stessa politica di estensione;
- l'aggiunta di nuovi tipi di dato numerici, quale il precisionDecimal.

Tutte le specifiche tecniche dell'XML Schema nelle versioni 1.0 e 1.1 si possono trovare nel sito del Consorzio W3C nelle pagine relative alle TR ^[3.17]. Nel sito sono presenti anche altre pagine, in lingua inglese, in cui è presentata una definizione dell'XML Schema ^[3.18] oltre ad un tutorial (testo che spiega come lavorare sull'XML tramite esemplificazioni) prodotto dalla W3Schools ^[3.19].

- • -

Poiché sono stati più volte citati nei paragrafi precedenti è utile approfondire un poco i "Namespace" e gli "XML Namespace", visto che sono utilizzati nell'XML.

Un Namespace è un contenitore astratto, o un ambiente, creato per mantenere un raggruppamento logico di singoli identificatori o simboli. Un identificatore, definito in un namespace, è associato con quel namespace. In questo modo, uno stesso identificatore può essere definito in maniera indipendente in vari namespace: il significato associato con un identificatore definito in un namespace può o non può avere lo stesso significato dello stesso identificatore definito in un altro namespace.

I linguaggi che supportano i namespace specificano le regole che determinano a quale namespace appartiene un determinato identificatore.

Nel programmi o nei documenti non è difficile avere migliaia di identificatori. I namespace forniscono un meccanismo per tenere nascosti identificatori locali. I namespace, inoltre, forniscono un meccanismo per raggruppare logicamente identificatori che sono correlati nei corrispondenti namespace, rendendo in tal modo il sistema più modulare.

I dispositivi di memorizzazione (ad esempio, hard disk, USB key, ecc.) e i moderni linguaggi di programmazione forniscono un supporto per utilizzare i namespace. I dispositivi usano come namespace le "directory" (cartelle): questo permette a due file (documenti) con lo stesso nome di essere memorizzati sullo stesso dispositivo a condizione che essi siano memorizzati su differenti directory. In alcuni linguaggi di programmazione (ad esempio, il C++) gli identificatori che nominano i namespace, sono loro stessi associati con un delimitato namespace. In questi linguaggi i namespace possono innestarsi, formando un "namespace tree". Nei linguaggi di programmazione che non forniscono un supporto per i namespace, questi ultimi possono essere emulati tramite alcune convenzioni fatte sugli identificatori.

Nell'XML, la specifica XML Namespace è utilizzata per fornire solamente una denominazione agli elementi e agli attributi in un documento XML. Un istanza (un esempio) di un documento XML può contenere nomi di elementi od attributi

appartenente a uno o più vocabolari XML. Se a ciascun vocabolario è dato un namespace allora una possibile ambiguità, che può nascere tra elementi o attributi con lo stesso nome, è in grado di essere risolta.

Gli XML Namespace sono specificati in maniera precisa nella raccomandazione, elaborata dal W3C. Un XML Namespace è dichiarato usando l'attributo XML riservato "xmlns", il cui valore deve essere un IRI anche se, solitamente, è un riferimento di tipo URI. Ad esempio:

```
xmlns="http://www.w3.org/1999/xhtml"
```

Si fa notare, comunque, che la specifica del namespace non richiede né fa capire che il namespace URI sia usato per recuperare informazioni: il parser XML lo tratta come una stringa. E' anche possibile mappare i namespace in un "prefisso" nella dichiarazione dei namespace stessi. Ad esempio:

```
xmlns:xhtml="http://www.w3.org/1999/xhtml"
```

in questo caso, qualsiasi elemento od attributo il cui nome inizia col prefisso "xhtml" è considerato far parte dello namespace XHTML.

Per un approfondimento tecnico degli XML Namespace si può far riferimento direttamente alla raccomandazione della W3C ^[3.20], relative alle due versioni dell'XML 1.0 e 1.1. I link alle specifiche (Technical Report) sono riportati in bibliografia.

- . - . - . -

Nel prossimo paragrafo si presenta il livello successivo, salendo verso l'alto, della piramide tecnologica del Web Semantico che è relativo alla descrizione della struttura concettuale, o alla modellizzazione, delle informazioni che sono implementate nelle risorse del Web, tutto questo utilizzando un insieme di formati sintattici.

3.3 RDF, RDF/XML, RDF Schema

L'RDF (Resource Description Framework) ^[3.21] è un linguaggio, sviluppato dalla W3C, che è utilizzato per rappresentare le informazioni sulle risorse nel World Wide Web. E' stato progettato, soprattutto, per rappresentare i metadati sulle risorse, ma può anche essere usato per rappresentare informazioni su cose che possono essere identificate sul Web, anche se non è possibile recuperarle direttamente dal Web stesso.

L'RDF è stato anche progettato per tutte quelle situazioni in cui le informazioni debbano essere elaborate dalle applicazioni piuttosto che essere solo rese visibili all'uomo. Per far questo, l'RDF fornisce una struttura comune che permette di esprimere queste informazioni in un modo tale che lo scambio tra applicazioni sia possibile senza perdita di significato. L'RDF consente, quindi, l'interoperabilità tra le applicazioni che si scambiano dati sul Web. Questa abilità di scambio permette alle informazioni di essere utilizzabili anche da altre applicazioni per le quali i dati non erano stati originariamente creati.

Dal punto di vista storico l'RDF, ora mantenuto ed aggiornato dalla W3C, deriva da alcuni lavori precedenti tra i quali si cita il "progetto MCF" (Meta Content Framework) sviluppato da Ramanathan V. Guha inizialmente presso l'Apple Computer tra il 1995 e il 1997 e poi proseguito presso la Netscape Communicator Corporation. Qui, egli ha adattato l'MCF all'utilizzo con l'XML e creato la prima versione del RDF: la MCF era nata come una specifica di un formato per strutturare i metadati sui siti web.

La "W3C RDF Working Group" ha pubblicato la specifica del modello dei dati del RDF come raccomandazione nel 1999. Da allora è iniziato il lavoro di sviluppo di una nuova versione che fu pubblicata nel 2004^[3,22], non come nuova raccomandazione, ma come un insieme di sei relative specifiche, comprendenti: Primer, Concepts, Syntax, Semantics, Vocabulary e Test Cases. Mentre ci sono state poche implementazioni basate sulla raccomandazione del 1999, che deve ancora essere completamente aggiornata, l'adozione delle specifiche migliorative è stata molto rapida, soprattutto da quando sono state rese pubblicamente sviluppabili. Attualmente, all'interno del W3C Semantic Web Interest Group, è in corso una discussione sulla necessità o meno di un aggiornamento dell'RDF e, in caso affermativo, su quale forma dovrebbe prendere.

La specifica principale dell'RDF è costituita da due componenti: l'RDF Model and Syntax che è una raccomandazione pubblicata nel 1999 e l'RDF Schema che è una raccomandazione candidata uscita nel 2000.

L'RDF Model and Syntax espone la struttura del modello RDF e ne descrive una possibile sintassi. Si basa su tre principi chiave:

1. Qualunque cosa può essere identificata da un URI (Universal Resource Identifier);
2. "*The least power*": utilizzare il linguaggio meno espressivo per definire qualunque cosa;
3. Qualunque cosa può dire qualunque cosa su qualunque cosa.

L'RDF Schema espone la sintassi per definire schemi e vocabolari per i metadati. Essendo una parte molto importante dell'RDF, verrà sviluppata in maniera approfondita nel prossimo paragrafo. Nella raccomandazione del 2004 è stato presentato con il nuovo nome di "RDF Vocabulary Description Language 1.0", anche se continua ad essere chiamato RDF Schema.

Qualunque cosa che è descritta dall'RDF è detta "risorsa". Principalmente una risorsa è reperibile sul Web, ma l'RDF può descrivere anche risorse che non si trovano direttamente sul Web. Ogni risorsa è identificata da un URI (Uniform Resource Identifier) che, come si è visto nei paragrafi precedenti, è un identificatore univoco delle risorse che può essere o un URL (Uniform Resource Locator) od un URN (Uniform Resource Name).

Il modello dei dati dell'RDF è formato da risorse, proprietà e valori. Le proprietà sono delle relazioni che legano tra loro risorse e valori. Un valore è una risorsa od un dato semplice quale, ad esempio, un numero. Questo modello è basato sull'idea di fare "statements" (dichiarazioni) sulle risorse (o sulla rappresentazione delle informazioni) nella forma di "subject - predicate - object" (soggetto - predicato - oggetto). Nel gergo dell'RDF queste espressioni sono conosciute con il nome di "triples" (triple).

Il soggetto una dichiarazione RDF è un URI o un nodo vuoto, che denotano entrambi risorse. Le risorse indicate da un nodo vuoto sono chiamate “anonymous resource” e non sono direttamente identificabili. Il predicato è un URI che denota tratti o aspetti della risorsa ed esprime una relazione tra il soggetto e l’oggetto. L’oggetto è anch’esso un URI o un nodo vuoto oppure una stringa di caratteri Unicode.

Ad esempio, l’affermazione “Alessandro Manzoni ha scritto ‘I promessi sposi’” espressa in RDF è una tripla dove: “Alessandro Manzoni” è il soggetto, “ha scritto” è il predicato e “I promessi sposi” è l’oggetto.

Per essere più precisi, si fa presente l’RDF non utilizza gli URI ma bensì gli URI reference, già introdotti prima, per poter descrivere praticamente tutto. L’RDF definisce quindi una risorsa come qualcosa che è identificabile da un URI reference.

Il modello dei dati RDF permette quindi di definire un modello semplice per descrivere le relazioni tra le risorse, in termini di proprietà identificate da un nome e dai relativi valori. Questo modello ha diversi formati che lo serializzano (cioè diversi formati di file) e quindi il meccanismo con il quale una risorsa o una tripla viene codificata varia da formato a formato. Nelle prossime righe si presenteranno alcuni formati specifici. Questo modello dei dati, tuttavia, non fornisce nessun meccanismo per dichiarare queste proprietà, né per definire le relazioni tra queste proprietà ed altre risorse: questo compito è realizzato dall’RDF Schema.

Gli statement sulle risorse sono rappresentabili da un grafo orientato composto da nodi ed archi: i nodi rappresentano le risorse (o i tipi primitivi), mentre gli archi rappresentano le proprietà (predicati). Una dichiarazione è rappresentata quindi da un grafo composto da due nodi, rappresentanti rispettivamente soggetto ed oggetto, collegati da un arco orientato dal soggetto all’oggetto rappresentante il predicato.

Il grafo è il modello concettuale dell’RDF. Un insieme di statement RDF può essere rappresentato da un multi-grafo (diretto ed etichettato). Tutte le triple possono essere immagazzinate in un contenitore detto “triplestore”, che è un database costruito appositamente per archiviare e recuperare metadati RDF.

Per rappresentare e memorizzare gli statement, l’RDF utilizza delle serializzazioni. Le principali sono:

- XML/RDF: è un formato basato sull’XML, più spesso indicato con il solo termine RDF, visto che è stata introdotto assieme ad altre specifiche che definiscono l’RDF: si distingue, tuttavia, questo modello da quello astratto dell’RDF. Questa serializzazione è specificata nella RFC 3870, ed è definita nel MIME media type con “application/rdf+xml”. Essa segue le specifiche delle raccomandazioni pubblicate nel 2004 ed ha la proprietà fondamentale di essere machine-processable.
- Notation 3: abbreviata in N3, è una serializzazione di modelli RDF, non basata sull’XML. E’ stata introdotta sempre dal W3C e risulta molto semplice da scrivere a mano. Si basa sull’uso di una notazione tabellare. In essa si serializza il grafo descrivendo una risorsa per volta con tutte le sue proprietà. La N3 è strettamente collegata ai formati N-Triples e Turtle.
- Turtle (Terse RDF Triple Language): è anch’essa una serializzazione di grafi RDF. E’ un sottoinsieme di N3, proposta da Davie Beckett. Essa è limitata solo al modello dei grafi dell’RDF e non ha nessuno “stato ufficiale” con nessuna organizzazione di standard. E’ usata spesso al posto di XML/RDF.

- **N-Triples**: è un formato per memorizzare e trasmettere dati. In questa serializzazione il grafo è rappresentato come un insieme di triple nella forma soggetto - predicato - oggetto. Essa è un sottoinsieme di Turtle. E' stata progettata per essere un formato più semplice rispetto all'N3 e al Turtle stesso.

Nelle prossime righe sarà sviluppata la serializzazione RDF/XML e si farà vedere come si creano gli statement in RDF . Per comodità di trattazione si utilizzano alcuni esempi sviluppati dalla W3C in modo tale che il lettore che volesse approfondire il vasto argomento dell'RDF sia facilitato nella lettura dei testi, scritti in lingua inglese, che il consorzio ha pubblicato. Tale documentazione è molto vasta: si pensi che il solo "Primer" (testo di nozioni fondamentali) della raccomandazione del 2004 è composto di oltre cento pagine.

Esempio

Si consideri l'affermazione nella quale si dice che una determinata pagina web, riferita dall'indirizzo web <http://www.example.org/index.html> , è stata creata da una persona di nome John Smith. Scritta per esteso, in lingua inglese, si ha:

<http://www.example.org/index.html> has a creator whose value is John Smith

Questa affermazione può essere rappresentata in uno statement RDF dove:

- il soggetto è <http://www.example.org/index.html>
- il predicato è <http://purl.org/dc/elements/1.1/creator>
- l'oggetto è <http://www.example.org/staffid/85740>

In questo caso il grafo equivalente allo statement è:

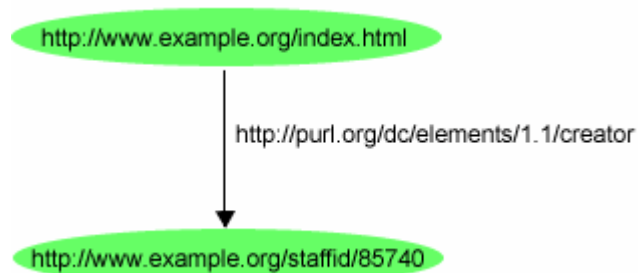


Figura 3.2 – Grafo dell'esempio

Si nota che gli URI reference non sono utilizzati per identificare il solo soggetto della dichiarazione, ma anche per il predicato e per l'oggetto, invece che utilizzare rispettivamente i termini "creator" e "John Smith". Questo ultimo è indicato dal codice 85740, che identifica univocamente John Smith all'interno di uno staff di persone. Questo aspetto sarà approfondito, e quindi diventerà più chiaro, nelle prossime righe.

Se l'affermazione è più complessa e comprende anche la data di creazione del documento (16 Agosto 1999) e la lingua (inglese = en) in cui il documento è stato scritto, gli statement diventano tre. Anche il grafo, ovviamente, cambia.

Gli statement che vanno aggiunti all’esempio sono:

http://www.example.org/index.html has a **creation-date** whose value is **August 16, 1999**
http://www.example.org/index.html has a **language** whose value is **English**

Il grafo diventa ora:

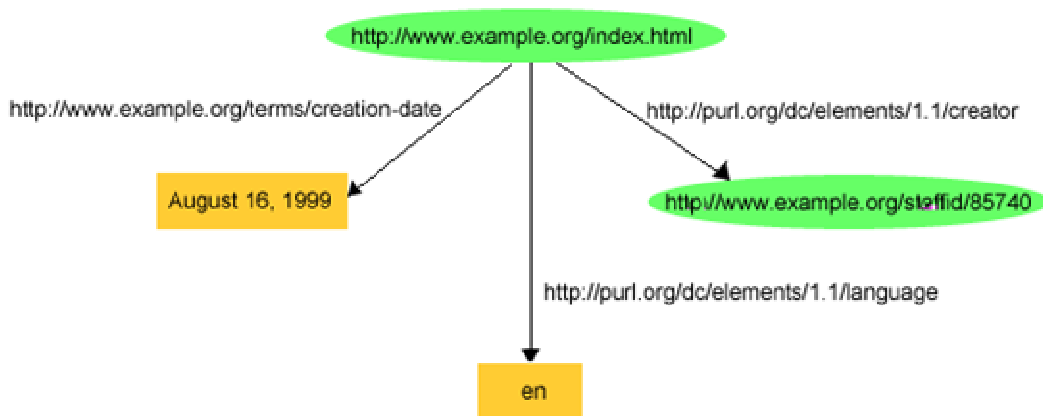


Figura 3.3 – Grafo dell’esempio con gli statement aggiunti

dove gli oggetti delle nuove affermazioni non sono URI, bensì valori costanti (detti “literal”), rappresentati da stringhe di caratteri che rappresentano certi tipi di proprietà dei valori. I literal non possono essere usati come soggetto in un RDF statement.

Dal un punto di vista grafico, i nodi che sono URI sono rappresentati da elissi, mentre i nodi che rappresentano literal sono disegnati con dei rettangoli. I predicati sono indicati con archi e frecce che ne indicano il verso. Per i predicati dell’esempio “creation-date” e “language” valgono le stesse considerazioni fatte prima per il predicato “creator”.

Talvolta, la notazione grafica non è la più felice per rappresentare le dichiarazioni RDF. Si può usare delle notazioni alternative per scrivere le triple. Una di queste è la N-Triple, nella quale ciascuna dichiarazione del grafo è scritta come una semplice tripla di soggetto - predicato - oggetto nell’ordine. Nell’esempio presentato, i tre statement diventano:

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740> .

<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date>
"August 16, 1999" .

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language>
"en" .
```

dove ciascuna tripla corrisponde ad un singolo arco del grafo, compresi i due nodi iniziale e finale.

Si noti che questa notazione richiede che un nodo, quale `<http://www.example.org/index.html>` sia identificato separatamente per ciascuno statement dove esso appare.

La cosa importante da capire per il lettore è che le triple rappresentano esattamente le stesse informazioni rappresentate nel grafo. La cosa fondamentale nell'RDF è il modello del grafo degli statement: la notazione utilizzata per rappresentarlo è secondaria.

La notazione completa delle triple, come si vede qui sopra, richiede che gli URI reference siano scritti per esteso e racchiusi tra parentesi angolari. Questo si trasforma spesso in una riga di caratteri troppo lunga che, talvolta, esce dalla visualizzazione della pagina. Esiste una scrittura abbreviata che è utilizzata in alcune specifiche RDF. Questa forma abbreviata sostituisce un "XML qualified name" (o Qname) scritto senza parentesi angolari, come abbreviazione di un URI reference completo. Un Qname contiene un "prefix", che è stato assegnato ad un namespace URI, seguito dai due punti e poi da un "local name". L'URI reference completo è formato dal Qname attaccando il local name al namespace URI che è stato assegnato al prefisso. Ad esempio, se il prefisso `image` è assegnato al namespace URI `http://example.com/image-stored`, allora il Qname `image:friends` è un'abbreviazione per l'URI reference `http://example.com/image-stored/friends`. Qui il local name è `friends`.

Esistono diversi prefissi Qname che sono molto utilizzati; sono definiti come segue:

```
prefix rdf: , namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#
prefix rdfs: , namespace URI: http://www.w3.org/2000/01/rdf-schema#
prefix dc: , namespace URI: http://purl.org/dc/elements/1.1/
prefix owl: , namespace URI: http://www.w3.org/2002/07/owl#
prefix ex: , namespace URI: http://www.example.org/
                        ( oppure http://www.example.com/ )
prefix xsd: , namespace URI: http://www.w3.org/2001/XMLSchema#
```

Leggendo le prossime righe il significato di essi risulterà più chiaro.

Con queste shorthand (abbreviazioni), l'esempio di prima può essere scritto come:

```
ex:index.html dc:creator          exstaff:85740 .
ex:index.html exterm:creation-date "August 16, 1999" .
ex:index.html dc:language         "en" .
```

dove sono stato utilizzati altri prefissi:

```
prefix exterm: , namespace URI: http://www.example.org/terms/ (per i termini
                        usati da un'organizzazione dell'esempio),
prefix exstaff: , namespace URI: http://www.example.org/staffid/ (per gli
                        identificatori delle persone dello staff dell'organizzazione
                        dell'esempio).
```

Da quando l'RDF usa gli URI reference (abbreviato in URIref) al posto delle parole per nominare le "cose" nelle dichiarazioni, l'RDF si riferisce ad un insieme di URIref (in particolare, ad un insieme inteso per uno specifico uso) come ad un

“vocabulary” (vocabolario). In tali vocabolari, spesso, gli URIref sono organizzati in un modo tale da poter essere rappresentati come un insieme di QName utilizzando un prefisso comune. In altre parole, un namespace di un URIref comune viene scelto da chi ha definito il vocabolario stesso per tutti i termini nel vocabolario. Gli URIref, che sono contenuti nel vocabolario, sono formati appendendo il singolo local name alla fine dell’URIref comune: questo, quindi, forma un insieme di URIref con un prefisso comune.

L’RDF usa questo approccio per definire il proprio vocabolario dei termini che hanno un significato speciale nell’RDF stesso. Gli URIref in questo vocabolario RDF iniziano tutti con <http://www.w3.org/1999/02/22-rdf-syntax-ns#>, convenzionalmente associato con il prefisso QName `rdf:`.

L’ “RFD Vocabulary Description Language” definisce un insieme aggiuntivo di termini che hanno URIref che iniziano con <http://www.w3.org/2000/01/rdf-schema#>, ed è convenzionalmente associato con il prefisso QName `rdfs:`.

L’uso comune dei prefissi URI fornisce un modo conveniente di organizzare gli URIref per un insieme di termini collegati. Questa comunque resta una convenzione, in quanto il modello RDF riconosce solo URIref completi: esso non “guarda all’interno” dell’URIref o non usa alcuna conoscenza sulla loro struttura.

Una particolare organizzazione, un processo, ecc. possono definire un vocabolario che è significativa per loro, usando gli URIref da un qualsiasi numero di altri vocabolari come facenti parte del proprio vocabolario. Un’organizzazione può usare un vocabolario di namespace URIref come l’URL di una risorsa web che fornisce ulteriori informazioni su quel vocabolario. Come visto prima, il prefisso QName `dc:`, associato al namespace URIref <http://purl.org/dc/elements/1.1>, si riferisce al vocabolario della Dublin Core. Accedendo a questo namespace URIref un browser web recupererà ulteriori informazioni sul vocabolario Dublin Core. E’ da notare che l’RDF non assume che un namespace URI identifichi una risorsa web recuperabile.

Gli URIref di diversi vocabolari possono essere liberamente mescolati in un grafo RDF. Inoltre, l’RDF non impone vincoli su quanti statement, che utilizzano un dato URIref come predicato, possano apparire in un grafo per descrivere la stessa risorsa. Nell’esempio di prima, la pagina web creata da John Smith, potrebbe avere altri creatori: in questo caso, nelle ulteriori dichiarazioni, si hanno stesso soggetto, stesso predicato, ma oggetto diverso.

Quanto visto finora sull’RDF illustra alcuni dei vantaggi dati dall’utilizzo degli URI reference come sistema di base per identificare le “cose”. Uno di questi è dato dalla precisione con cui i soggetti degli statement sono identificati; inoltre, come visto, aggiungendo statement si possono aggiungere ulteriori informazioni su di un soggetto. Un discorso simile può essere fatto con i predicati: l’uso di URIref per identificare proprietà permette alle proprietà di essere trattate come risorse loro stesse.

L’uso di URIref per i soggetti, i predicati e gli oggetti nelle dichiarazioni RDF permette lo sviluppo e l’utilizzo di vocabolari condivisi nel Web, dato che le persone possono scoprire ed iniziare ad utilizzare vocabolari già creati da altre persone per descrivere le cose, riflettendo un significato condiviso di questi concetti. Il discorso sull’utilizzo degli URIref nell’RDF potrebbe proseguire per molte altre pagine, ma si rischia di entrare in dettagli troppo approfonditi. Si fa notare che gli URIref non risolvono tutti i problemi di identificazione dato che delle persone potrebbero usare URIref diversi tra loro per identificare una stessa risorsa. Una buona idea è quindi

quella di utilizzare i termini dei vocabolari già esistenti, piuttosto che crearne di nuovi che potrebbero sovrapporsi con quelli di altri vocabolari: esistono vocabolari appropriati, per l'uso in specifiche aree applicative, che sono stati completamente sviluppati. Un esempio viene dal già citato Dublin Core (DC), che è un insieme di attributi (proprietà) per descrivere informazioni di molti tipi.

E' molto importante distinguere tra un qualsiasi significato che l'RDF stesso associa con i termini usati negli RDF statement ed il significato aggiuntivo, definito esternamente, che la gente potrebbe associare con quei termini. Come un linguaggio, l'RDF definisce direttamente solo la sintassi grafica di soggetto, predicato ed oggetto delle triple, oltre che certi significati associati con gli URIref nel vocabolario `rdf: .` L'RDF non definisce il significato dei termini di altri vocabolari, quali il `dc:creator` del DC, che potrebbero essere usati negli statement RDF. I vocabolari specifici quali il DC sono creati, con il significato specifico associato all'URIref definito in essi, esternamente all'RDF. Gli statement RDF che usano URIref da questi vocabolari possono trasportare il significato specifico associato con quei termini alla gente che ha familiarità con questi vocabolari, o alle applicazioni RDF scritte per processare proprio quei vocabolari senza trasportare alcuno di questi significati ad una altra applicazione RDF arbitraria, cioè non specificatamente scritta per processare questi vocabolari.

Si fa qui notare che gli statement RDF sono simili ad un numero di altri formati che sono utilizzati per registrare informazioni, quali le voci in un catalogo che descrive le risorse, le righe in un semplice database relazionale o semplici asserzioni in logica formale. Le informazioni in questi formati possono essere trattate come statement RDF: ciò permette di usare l'RDF per integrare dati da diverse sorgenti.

La parte principale della conoscenza, modellata da un insieme di statement, può essere soggetta ad una operazione chiamata "reification" (reificazione), nella quale ciascun statement (cioè ciascuna tripla soggetto - predicato - oggetto considerata nel suo complesso), è assegnato ad un URI e trattato come una unica risorsa, su cui possono essere fatte dichiarazioni aggiuntive. L'esempio iniziale può essere soggetto a reification diventando: "*Il professore ha detto che Alessandro Manzoni ha scritto I promessi sposi*". L'operazione di reification è talvolta molto importante al fine di dedurre un livello di fiducia/sicurezza od un grado di utilità per ciascun statement. In un insieme di dichiarazioni che sono state reificate ciascuna dichiarazione originale diventa una risorsa essa stessa, ed altre dichiarazioni su di essa possono essere fatte o già esistere, a seconda della necessità dell'applicazione.

Le informazioni contenute nei grafi RDF possono essere interrogate tramite il linguaggio SPARQL, che è un linguaggio che assomiglia all'SQL. Lo SPARQL, che verrà presentato più avanti è dal 2008 un raccomandazione della W3C. Esistono altri linguaggi per interrogare un grafo RDF: senza entrare nello specifico si segnalano RDQL, Versa, RQL e XUL.

Si è visto che l'RDF mette le informazioni in un modo formale tale che una macchina le possa capire. Lo scopo dell'RDF è di fornire un meccanismo di codifica e di interpretazione al fine di descrive le risorse in un maniera tale che un determinato software possa capirle; in altri termini, che un software possa accedere ed utilizzare queste informazioni che altrimenti non potrebbero essere usate.

Esistono molteplici applicazioni che sono basate sull’RDF; tra queste si ricordano:

- Creative Commons: utilizza l’RDF per inserire informazioni sulle licenze (diritti d’autore) nelle pagine web e nei file musicali di tipo mp3.
- FOAF (Friend of a Friend): è progettato per descrivere le persone con i loro interessi e le interconnessioni con le altre persone.
- DOAC (Description of a Career): è un supplemento al FOAF, creato da Ramon A. Parada, che consente di aggiungere informazioni di tipo lavorativo, quali studi fatti, lingue conosciute, esperienze lavorative.
- SKOS (Simple Knowledge Organization System): è una famiglia di linguaggi formali creata per rappresentare glossari, classificazioni, tassonomie e qualsiasi tipo di vocabolario strutturato. Il suo obiettivo principale è di consentire una facile pubblicazione di vocabolari strutturati per il Web Semantico.
- SIOC (Semantically Interlinked On-line Communities): è una tecnologia del Web Semantico progettata per descrivere comunità on-line e per creare connessioni tra discussioni su Internet tramite “message board” (tabellone dei messaggi in Internet su cui gli utenti inviano i loro messaggi o partecipano ai gruppi di discussione) e per le mailing list. Questa tecnologia verrà richiamata nei prossimi paragrafi, assieme ad alcune di queste applicazioni qui introdotte.

Nel Web Semantico ed in alcune sue popolari applicazioni, quali il FOAF, le risorse tendono ad essere rappresentate da URI che intenzionalmente denotano (e che possono essere usati per accedere a) dati effettivi sul Web. L’RDF, in generale, non è limitato alla descrizione di risorse basate su Internet; infatti gli URI che denominano una risorsa non devono essere affatto dereferenziabili. E’ però necessario che i produttori e gli utilizzatori di dichiarazioni RDF siano in accordo sulla semantica degli identificatori di risorse. Tale accordo non deve essere inerente solo all’RDF benché ci siano in uso dei vocabolari controllati, quali ad esempio il Dublin Core Metadata (che è parzialmente mappato in uno spazio URI per l’utilizzo con l’RDF). L’intenzione di pubblicare nel Web ontologie, basate sull’RDF, è spesso necessaria per stabilire, o circoscrivere, il significato voluto dell’identificatore della risorsa che è usato per esprimere il dato in RDF.

Tutte le specifiche tecniche dell’RDF sono riportate nelle pagine del sito W3C relative all’RDF stesso, soprattutto nella specifica “RDF Concepts and Abstract Syntax” che dovrebbe essere consultata dal lettore che vuole approfondire l’aspetto sintattico. L’aspetto semantico di quanto visto nelle righe precedenti è definito nel documento “RDF Semantics”. Questi ed altri documenti sono liberamente scaricabili dal sito della W3C presente nella bibliografia.

In aggiunta alle tecniche di base per descrivere le cose utilizzando gli statement RDF visti finora, dovrebbe essere chiaro al lettore che le persone e/o le organizzazioni hanno bisogno anche di una maniera per descrivere i vocabolari che essi intendono utilizzare in quelle dichiarazioni; nello specifico vocabolari per:

- descrivere tipi di cose (come `exterms:person`);
- descrivere proprietà (come `exterms:age 0 exterm:creation-date`);
- descrivere il tipo di cose che possono servire come soggetto od oggetto nelle dichiarazioni che coinvolgono quelle proprietà (ad esempio, nella specifica

del valore di una proprietà quale `ex:terms:age`, esso dovrebbe essere sempre un `xsd:integer`, cioè di tipo intero).

La base per descrivere tali vocabolari in RDF è il “RDF Vocabulary Description Language 1.0: RDF Schema”, che sarà descritto nel prossimo paragrafo. Si presenta ora, in maniera più approfondita la serializzazione RDF/XML.

- . - . - . -

L’ “RDF/XML” è una sintassi, basata sull’XML, che fornita dall’RDF per scrivere (rappresentare) e scambiare con altre applicazioni i grafi RDF. Diversamente dalle triple, che vengono intese come una notazione abbreviativa, l’RDF/XML è la normativa sintattica per scrivere le dichiarazioni RDF. Si presenta ora un esempio, sempre tratto dalla documentazione presente nel sito W3C, nel quale una dichiarazione viene descritta da un grafo e poi rappresentata in codice RDF/XML. Verranno indicati i passaggi principali, evidenziando gli aspetti degni di nota.

Esempio La dichiarazione, in lingua inglese, è:

`http://www.example.org/index.html` has a `creation-date` whose value is **August 16, 1999**

Il grafo RDF corrispondente a questa singola dichiarazione, con le specifiche viste prima, è:

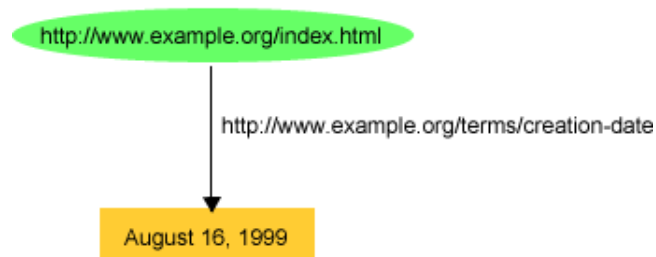


Figura 3.4 – Grafo dell’esempio

dove, alla proprietà `creation-date` è stato assegnato un URIref e la data non è specificata da un literal ma da un formato specifico.

La tripla è rappresentata come:

```
ex:index.html    ex:terms:creation-date    "August 16, 1999" .
```

La sintassi XML/RDF corrispondente al grafo è:

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#"
3.     xmlns:ex="http://www.example.org/terms/">
4.   <rdf:Description
   rdf:about="http://www.example.org/index.html">
5.     <ex:creation-date>August 16,
   1999</ex:creation-date>
6.   </rdf:Description>
7. </rdf:RDF>
```

dove i numeri di linea sono stati aggiunti per chiarire l'esempio.

Si analizzano, brevemente, le varie linee di codice per capirne il significato.

La 1° linea `<?xml version="1.0"?>` è la “XML declaration”, che indica che il contenuto che la segue è codice XML e a quale versione di XML si fa riferimento (in questo caso la versione 1.0).

La 2° linea inizia con un elemento `rdf:RDF`: questo indica che il contenuto XML che segue (che inizia lì e finisce con `</rdf:RDF>` nella 7° linea) è inteso rappresentare codice RDF. Quello che segue l'elemento `rdf:RDF` sulla stessa riga è la dichiarazione di un namespace XML, rappresentato come un attributo `xmlns` dello start-tag `rdf:RDF`. Questa dichiarazione specifica che tutti i tag in questa contenuti con prefisso `rdf:`, sono parte del namespace identificato dall'URIref `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. Gli URIref che iniziano con la stringa `http://www.w3.org/1999/02/22-rdf-syntax-ns#` sono usati per i termini del vocabolario RDF.

La 3° linea specifica un'altra dichiarazione di un namespace XML, questa volta per il prefisso `exterms:`. Questo è espresso come un altro attributo `xmlns` dell'elemento `rdf:RDF` e specifica che il namespace URIref `http://www.example.org/terms/` è associato con il prefisso `exterms:`. Gli URIref che iniziano con la stringa `http://www.example.org/terms/` sono usati per i termini dal vocabolario definito dall'organizzazione dell'esempio, `example.org`. Il `>` alla fine della 3° linea indica la fine dello start-tag `rdf:RDF`. Le linee comprese tra la 1° e la 3° sono generalmente “housekeeping” necessarie cioè ad indicare che questo è contenuto di tipo RDF/XML e per identificare i namespace che si stanno usando all'interno del contenuto dell'RDF/XML.

Le linee dalla 4° alla 6° forniscono il codice RDF/XML per lo specifico statement che è mostrato nel grafo. Un modo semplice di parlare di qualsiasi statement RDF è quello di dire che esso è una “description” (descrizione) e che è fatto sul soggetto dello statement (in questo caso, su `http://www.example.org/index.html`): questo è il modo in cui l'RDF/XML rappresenta gli statement. Lo start-tag `rdf:Description` nella 4° linea indica l'inizio della descrizione di una risorsa e va ad identificare la risorsa (about) su cui lo statement è relativo (il soggetto dello statement) usando l'attributo `rdf:about` per specificare l'URIref della risorsa del soggetto. La 5° linea fornisce un “property element” (elemento proprietà), con il QName `exterms:creation-date` come suo tag, per rappresentare il predicato e l'oggetto dello statement. The QName `exterms:creation-date` è scelto in modo tale che, appendendo il local name `creation-date` all'URIref del prefisso `exterms:` (`http://www.example.org/terms/`) si abbia l'URIref del predicato dello statement `http://www.example.org/terms/creation-date`. Il contenuto di questo property element è l'oggetto dello statement, il literal `August 19, 1999` (il valore della proprietà `creation-date` della risorsa soggetto). Il property element è nidificato all'interno dell'elemento contenente `rdf:Description`, che indica che questa proprietà è applicata alla risorsa specificata nell'attributo `rdf:about` dell'elemento `rdf:Description`. La 6° linea indica la fine dell'elemento `rdf:Description`.

La 7° linea 7 indica la fine dell'elemento `rdf:RDF` che era iniziato nella 2° linea. L'uso di un elemento `rdf:RDF` per racchiudere il contenuto RDF/XML è opzionale in quelle situazioni nelle quali l'XML può essere identificato come RDF/XML dal contesto. Comunque, non fa male imporre l'elemento `rdf:RDF` in ogni caso.

Questo esempio illustra l'idea di base usata dall'RDF/XML per codificare un grafo RDF tramite elementi, attributi, contenuto di elementi e valore degli attributi di tipo XML. Gli URIref dei predicati sono scritti come Qname. Anche gli URIref dei soggetti sono scritti come valori di attributi XML. I nodi di tipo literal diventano elementi contenenti testo o valori di attributo.

La sintassi RDF/XML fornisce anche un numero di abbreviazioni per renderne più facile la scrittura.

La specifica completa dell'RDF/XML è definita nel documento della W3C "RDF/XML Syntax Specification" ^[3.23], riportato in bibliografia.

- -

L' "RDF Schema", talvolta abbreviato in RDFS o RDF-S, è un linguaggio estensibile per la rappresentazione della conoscenza, che fornisce gli elementi di base per la descrizione dei vocabolari RDF, che sono a loro volta progettati per strutturare le risorse RDF.

La prima versione è stata pubblicata dal World Wide Web Consortium nell'Aprile 1998. E' divenuta una raccomandazione nel Febbraio del 2004, con il nome di "RDF Vocabulary Description Language 1.0" ^[3.24], anche se il nome precedente di RDF Schema è ancora molto utilizzato.

Molti dei componenti dell'RDF Schema sono inclusi nel linguaggio OWL Web Ontology Language che permette più espressività e che verrà presentato più dettagliatamente nel prossimo paragrafo.

Nelle pagine precedenti si è visto che l'RDF fornisce un modo per esprimere semplici dichiarazioni sulle risorse, utilizzando proprietà e valori. La comunità degli utenti dell'RDF ha, tuttavia, bisogno anche di definire dei vocabolari, cioè gli insiemi dei termini che essi intendono usare in quelle dichiarazioni. In maniera specifica, i vocabolari sono usati per indicare che, quello che gli utenti stanno descrivendo, specifica i tipi di classi o di risorse; questo inoltre precisa che nella descrizione di quelle risorse saranno usate delle specifiche proprietà. L'RDF da solo non fornisce i mezzi per definire tali classi e tali proprietà: queste saranno descritte come un vocabolario RDF, usando una estensione dell'RDF fornita dall' "RDF Vocabulary Description Language", riferito qui come RDF Schema.

L'RDF Schema non fornisce un vocabolario di classi specifiche per le applicazioni, quali `ex:Person` e di proprietà quali `ex:author`. Esso fornisce, invece, i mezzi necessari per descrivere tali classi e proprietà e per indicare quali classi e proprietà ci si aspetta che possano essere utilizzate assieme.

Dal punto di vista tecnico, si può dire che l'RDF Schema fornisce un "type system" (un sistema di tipi) per l'RDF.

L'RDF Schema è definito nella specifica RDF Semantics come una "estensione semantica" dell'RDF. Essa fornisce meccanismi per descrivere gruppi di risorse collegate e le relazioni tra queste risorse. Le descrizioni del vocabolario RDF Schema sono scritte in codice RDF usando i termini che saranno presentati nelle

prossime pagine. Queste risorse sono usate per determinare le caratteristiche di altre risorse.

L’RDF Schema permette alle risorse di esse definite come istanze di una o più classi. Inoltre, le classi possono essere organizzate in modo gerarchico.

I “mezzi” dell’RDF Schema sono forniti loro stessi nella forma di un vocabolario RDF, cioè un insieme specialistico di risorse RDF predefinite, con il loro specifico significato. Le risorse nel vocabolario RDF Schema hanno un URIref con il prefisso `http://www.w3.org/2000/01/rdf-schema#`, convenzionalmente associato con il prefisso QName `rdfs:`. La specifica utilizza anche il prefisso `rdf:` per riferirsi al namespace RDF `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.

Si presentano ora i costrutti e le proprietà di base dell’RDF Schema, descrivendo le classi e le proprietà.

In un processo di descrizione, un punto importante è quello di identificare i vari tipi di cose che sono descritte. L’RDF Schema si riferisce a queste “tipi di cose” come classi. Una classe corrisponde al concetto più generico di tipo o categoria.

Le classi in RDF possono essere usate per rappresentare qualsiasi tipo di cose: pagine Web, persone, documenti, concetti astratti, ecc. Esse sono descritte usando le risorse dell’RDF Schema `rdfs:Class` e `rdfs:Resource` e le proprietà `rdf:type` e `rdfs:sulClassOf`.

La specifica fornita dalla W3C riporta svariati esempi per chiarire meglio questi concetti. Qui se ne riporta uno, al fine di dare concretezza agli aspetti prettamente teorici. L’esempio ne introduce anche di ulteriori.

Esempio

Si supponga che un’organizzazione, chiamata “example.org” voglia utilizzare l’RDF per fornire informazioni sui diversi tipi di veicoli a motore (motor vehicle).

Nell’RDF Schema, example.org necessiterà di una classe per rappresentare la categoria di quelle cose che sono veicoli a motore. Le risorse che appartengono a una classe sono chiamate istanze. L’organizzazione intende per istanze di questa classe essere quelle risorse che sono dei veicoli a motore.

Nell’RDF Schema una classe è qualsiasi risorsa che ha una proprietà `rdf:type` il cui valore è la risorsa `rdfs:Class`.

Così la classe dei motoveicoli sarà descritta assegnando la classe ad un URIref, chiamato `ex:MotorVehicle` e descrivendo quella risorsa con una proprietà `rdf:type` il cui valore è la risorsa `rdfs:Class`. (qui si è utilizzato `ex:` al posto dell’URIref `http://www.example.org/schemas/vehicles` che è usato come prefisso per gli URIref dal vocabolario example.org).

In questo modo l’organizzazione può scrivere la dichiarazione RDF:

```
ex:MotorVehicle    rdf:type    rdfs:Class .
```

Descrivendo così come una classe, la risorsa `exthings:companyCar` potrebbe essere descritta come un veicolo a motore tramite lo statement RDF:

```
exthings:companyCar    rdf:type    ex:MotorVehicle .
```

La risorsa `rdfs:Class` ha essa stessa un `rdf:type` di `rdfs:Class`. Una risorsa può essere una istanza di più classi.

Nell'esempio, si fa vedere come sia possibile introdurre classi aggiuntive che rappresentano vari tipi speciali di veicoli a motore. Queste classi possono essere descritte come fatto prima per la classe `ex:MotorVehicle` assegnando un URIref per ciascuna nuova classe e scrivendo dichiarazioni RDF che descrivono queste risorse come classi. Si aggiungono le classi `Van` e `Truck` tramite gli statement:

```
ex:Van      rdf:type  rdfs:Class .
ex:Truck    rdf:type  rdfs:Class .
```

E' possibile anche specificare delle relazioni che queste classi hanno con la classe `ex:MotorVehicle`. Una proprietà importante predefinita nell'RDF Schema è quella di sottoclasse. In questo caso si può esplicitare che le due classi sono dei tipi speciali di veicoli a motore. Ciò si specifica con lo statement RDF:

```
ex:Van      rdfs:subClassOf  ex:MotorVehicle .
ex:Truck    rdfs:subClassOf  ex:MotorVehicle .
```

Il significato della relazione `rdfs:subClassOf` è che ogni istanza della classe `ex:Van` è anche una istanza della classe `ex:MotorVehicle`. Un software sofisticato, che "capisce" questo vocabolario, può dedurre l'informazione aggiuntiva che `ex:things:companyCar` è anche una istanza di `ex:MotorVehicle`.

La proprietà `rdfs:subClassOf` è transitiva. Nell'esempio dati gli statement:

```
ex:Van      rdfs:subClassOf  ex:MotorVehicle .
ex:MiniVan  rdfs:subClassOf  ex:Van .
```

l'RDF Schema definisce `ex:MiniVan` essere anche una sottoclasse di `ex:MotorVehicle`.

Una classe può essere una sottoclasse di più classi. L'RDF Schema definisce tutte le classi come sottoclasse della classe `rdfs:Resource`. Questo perché le istanze appartenenti a tutte le classi sono risorse.

L'esempio presentato nelle sue linee fondamentali può essere rappresentato graficamente. Nel grafo sono omesse le proprietà di tipo `rdf:type` che collegano ciascuna delle classi alla classe `rdfs:Class`.

Per comodità di lettura si riporta la figura 3.5 nella pagina seguente.



Figura 3.5 – Grafo dell’esempio Motor Vehicle

Lo schema può essere descritto con le triple:

```

ex:MotorVehicle      rdf:type      rdfs:Class .
ex:PassengerVehicle  rdf:type      rdfs:Class .
ex:Van               rdf:type      rdfs:Class .
ex:Truck             rdf:type      rdfs:Class .
ex:MiniVan           rdf:type      rdfs:Class .

ex:PassengerVehicle  rdfs:subClassOf  ex:MotorVehicle .
ex:Van               rdfs:subClassOf  ex:MotorVehicle .
ex:Truck             rdfs:subClassOf  ex:MotorVehicle .

ex:MiniVan           rdfs:subClassOf  ex:Van .
ex:MiniVan           rdfs:subClassOf  ex:PassengerVehicle .
    
```

L’esempio può essere anche scritto in RDF/XML:

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
    "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://example.org/schemas/vehicles">

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

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

```

<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>

```

L'XML/RDF fornisce anche un modo conciso, chiamato "Typed Node Abbreviation" per descrivere la gerarchia delle classi dei veicoli. Non viene qui riportato, ma la sua sintassi è presentata nella specifica "RDF Primer".

Il secondo punto fondamentale, nella trattazione dell'RDF Schema, riguarda la descrizione delle proprietà. Infatti, gli utenti hanno spesso la necessità di poter descrivere specifiche proprietà che caratterizzano le classi delle cose. Nell'RDF Schema le proprietà sono descritte usando la classe RDF `rdf:Property` e le proprietà dell'RDF Schema `rdfs:domain`, `rdfs:range` e `rdfs:subPropertyOf`.

Tutte le proprietà nell'RDF sono descritte come istanze della classe `rdf:Property`. Così facendo una nuova proprietà, ad esempio `extern:weightInKg` viene descritta assegnando alla proprietà un URIref e descrivendo quella risorsa con una proprietà `rdf:type` il cui valore è la risorsa `rdf:Property`.

La proprietà dell'esempio viene scritta come:

```
extern:weightInKg  rdf:type  rdf:Property .
```

L'RDFS fornisce anche dei vocabolari per descrivere come le proprietà e le classi devono essere intese per essere usate assieme nei dati RDF. L'informazione più importante di questo tipo è fornita attraverso l'uso delle proprietà `rdfs:range` e `rdfs:domain`, usate per descrivere ulteriori specifiche delle proprietà per le applicazioni. Si analizza ognuna di esse nei dettagli.

La proprietà `rdfs:range` è utilizzata per indicare che i valori di una particolare proprietà sono istanze di una classe designata. Ad esempio se `example.org` vuole indicare che la proprietà `ex:author` ha valori che sono istanze della classe `ex:Person`, essa scriverà i tre statement:

```

ex:Person  rdf:type  rdfs:Class .
ex:author  rdf:type  rdf:Property .
ex:author  rdfs:range ex:Person .

```

Queste dichiarazioni indicano che `ex:Person` è una classe, `ex:author` è una proprietà e gli statement che utilizzano la proprietà `ex:author` hanno istanze di `ex:Person` come oggetti.

Una data proprietà può avere nessuna, una o più di una range property. Nell'ultimo caso, il valore delle proprietà sono risorse che sono istanze di tutte le classi che sono state specificate come il range.

Una proprietà di tipo `rdfs:range` può anche essere utilizzata per indicare che il valore di una proprietà è dato da un tipo literal. Per esempio, se `example.org` vuole indicare che la proprietà `ex:age` ha valori dal tipo di dato dell'XML Schema `xsd:integer`, essa scriverà gli statement RDF:

```
ex:age    rdf:type    rdf:Property .
ex:age    rdfs:range  xsd:integer .
```

Il datatype `xsd:integer` è identificato dal suo URIref (l'URIref completo è `http://www.w3.org/2001/XMLSchema#integer`). Questo URIref può essere usato senza dichiarare esplicitamente nello schema che esso identifica un datatype. Comunque, è buona cosa dichiarare esplicitamente che un dato URIref identifica un datatype. Questo può essere fatto usando la classe dell'RDF Schema `rdfs:Datatype`. Per dichiarare che `xsd:integer` è un datatype, `example.org` scriverà lo statement RDF:

```
xsd:integer  rdf:type  rdfs:Datatype .
```

Questo statement dice che `xsd:integer` è l'URIref di un datatype (il quale è assunto essere conforme ai requisiti per i datatype RDF descritti nella specifica RDF-Concepts. E' da notare che una tale dichiarazione non costituisce una definizione di un tipo di dato. Non c'è possibilità di definire datatype nell'RDF Schema. Essi vengono definiti esternamente all'RDF e all'RDFS e sono attribuiti nelle dichiarazioni RDF tramite i loro URIref.

La proprietà `rdfs:domain` è usata per indicare che una particolare proprietà si applica ad una classe designata. Se, nel solito esempio, `example.org` vuole indicare che la proprietà `ex:author` si applica ad istanze della classe `ex:Book`, essa dichiarerà:

```
ex:Book    rdf:type    rdfs:Class .
ex:author  rdf:type    rdf:Property .
ex:author  rdfs:domain ex:Book .
```

Questa dichiarazione indica che `ex:Book` è una classe, `ex:author` è una proprietà che la dichiarazione RDF che usa la proprietà `ex:author` ha istanze di `ex:Book` come soggetti.

Una data proprietà può avere nessuna, una o più di una proprietà di dominio. Nell'ultimo caso si può dire che una risorsa che ha quella proprietà è un'istanza di tutte le classi specificate come dominio.

L'RDF Schema fornisce anche una maniera di specializzare le proprietà come fatto per le classi. Questa relazione di specializzazione tra due proprietà è descritta usando la proprietà, già predefinita, `rdfs:subPropertyOf`.

Ad esempio, se `ex:primaryDriver` e `ex:driver` sono entrambe proprietà, `example.org` potrebbe descrivere queste proprietà, ed il fatto che `ex:primaryDriver` è una specializzazione di `ex:driver`, scrivendo lo statement RDF:

```
ex:driver          rdf:type          rdf:Property .
ex:primaryDriver  rdf:type          rdf:Property .
ex:primaryDriver  rdfs:subPropertyOf ex:driver .
```

Il significato della relazione `rdfs:subPropertyOf` è che se un'istanza `exstaff:fred` è un `ex:primaryDriver` dell'istanza `ex:companyVan`, allora l'RDF Schema definisce `exstaff:fred` come essere anche un `ex:driver` di `ex:companyVan`.

Una proprietà può essere una sottoproprietà di nessuna, una o di più proprietà. Tutte le proprietà dell'RDFS `rdfs:range` e `rdfs:domain` che si applicano ad una proprietà RDF si applicano anche a ciascuna delle sue sottoproprietà.

Qui sotto si presenta l'esempio completo dei Motoveicoli, codificato in RDF/XML, dove sono state applicate tutte le proprietà presentate. Non ci si sofferma su tutti i dettagli minimi per non appesantire la lettura. Qui interessa far vedere al lettore come viene utilizzato l'RDF Schema.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

  <rdfs:Class rdf:ID="MotorVehicle"/>

  <rdfs:Class rdf:ID="PassengerVehicle">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Truck">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Van">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="MiniVan">
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Person"/>

  <rdfs:Datatype rdf:about="&xsd;integer"/>
```

```

<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdf:Property rdf:ID="driver">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>

</rdf:RDF>

```

Finora si è visto come si possono descrivere le classi e le proprietà nell’RDF Schema. Si accenna ora a come possono essere definite le istanze che utilizzano quelle classi e quelle proprietà. Si presenta come esempio un’istanza di `ex:PassengerVehicle` : (per i dettagli si rimanda al già citato RDF Primer)

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
  "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
  ns#"
          xmlns:ex="http://example.org/schemas/vehicles#"
          xml:base="http://example.org/things">

  <ex:PassengerVehicle rdf:ID="johnSmithsCar">
    <ex:registeredTo
  rdf:resource="http://www.example.org/staffid/85740"/>
    <ex:rearSeatLegRoom

  rdf:datatype="&xsd;integer">127</ex:rearSeatLegRoom>
    <ex:primaryDriver
  rdf:resource="http://www.example.org/staffid/85740"/>
  </ex:PassengerVehicle>
</rdf:RDF>

```

Da quanto presentato finora si è visto che l’RDF Vocabulary Description Language descrive le proprietà in termini delle classi di risorse alle quali esse si applicano. Questo è il ruolo di dei meccanismi di “domain” (dominio) e di “range” (portata). Ad esempio, si potrebbe definire la proprietà `ex:author` che ha un domain di `ex:Document` e un range di tipo `ex:Person`. Questo approccio è molto simile al classico sistema dei linguaggi di programmazione orientati agli oggetti, quali Java, che potrebbe tipicamente definire una classe `ex:Book` con un attributo di chiamato `ex:author` di tipo `ex:Person`. In questo caso si è definita la classe in termini delle proprietà che le sue istanze possono avere. In altre parole, lo scopo della descrizione di un attributo in molti linguaggi di programmazione è ristretto alla classe o al tipo in cui esso è definito. Nell’RDF le descrizioni delle proprietà sono indipendenti dalla definizione delle classi ed hanno uno scopo “globale”.

Usando l'approccio dell'RDF Schema è facile definire nuove proprietà aggiuntive che hanno un dominio di `ex:Document` o un range di `ex:Person`. Questo può essere fatto senza ridefinire la descrizione originale di questa classe. Tutto ciò rientra anche nella proprietà principale dell'RDF che è quella di permettere a chiunque di estendere la descrizione delle risorse esistenti.

L'RDF Schema fornisce informazioni sullo schema come “descrizioni aggiuntive” delle risorse, ma non dice come queste descrizioni dovrebbero essere utilizzate dalle applicazioni.

Queste ulteriori informazioni possono essere usate in maniera diversa a seconda di come l'applicazione interpreta la descrizione della proprietà. In altre parole, le dichiarazioni in uno schema RDF sono sempre descrizioni. Esse possono essere “prescriptive” (cioè, normative, che introducono vincoli) ma solo se l'applicazione che interpreta quelle dichiarazioni le considera in quel modo. L'RDF Schema fornisce una maniera di dichiarare queste informazioni aggiuntive. Se queste informazioni sono in conflitto con il dato dell'istanza esplicitamente specificato sarà l'applicazione a determinarlo e ad agirci sopra di conseguenza.

Riassumendo, si è visto che le risorse possono essere divise in gruppi chiamati classi. I membri di una classe sono conosciuti come istanze della classe. Le classi sono loro stesse risorse. Esse sono identificate da un URIref e possono essere descritte usando le proprietà RDF.

L'RDF distingue tra una classe ed un insieme di sue istanze. Associata a ciascuna classe c'è un insieme, detto la “class extension” della classe, che è l'insieme delle istanze della classe. Due classi possono avere lo stesso insieme di istanze, ma essere classi diverse. Una classe può essere un membro della sua class extension e può essere una istanza di se stessa.

Il gruppo di risorse che sono classi dell'RDF Schema è esso stesso una classe che è chiamata `rdfs:Class`. Le proprietà `rdf:type` può essere usata per dichiarare che una risorsa è un'istanza di una classe.

La specifica “RDF Concepts and Abstract Syntax” definisce il concetto RDF di un “RDF datatype”. Tutti i datatype sono classi. Le istanze di una classe che è un datatype sono i membri dello spazio dei valori del datatype.

Di seguito si presentano i principali costrutti dell'RDFS dal punto di vista puramente teorico. Si aggiungono anche dei semplici esempi per una migliore comprensione.

rdfs:Resource

Tutte le cose descritte dall'RDF sono chiamate “resources” e sono istanze della classe `rdfs:Resource`. Questa è la classe di tutto. Tutte le altre classi sono sottoclassi di questa classe. Quindi `rdfs:Resource` è un'istanza di `rdfs:Class`.

rdfs:Class

Questa è la classe delle risorse che sono classi RDF. `rdfs:Class` è un'istanza di `rdfs:Class`. Essa permette di dichiarare una risorsa come una classe per altre risorse.

Un esempio tipico di una `rdfs:Class` è `foaf:Person` nel vocabolario Friend of a Friend (FOAF). Un'istanza di `foaf:Person` è una risorsa collegata alla classe

usando un predicato `rdf:type`. La frase in linguaggio naturale “David is a Person” viene espressa con `ex:David rdf:type foaf:Person`.

rdfs:subClassOf

Se un classe `C` è una sottoclasse di una classe `C'`, allora tutte le istanze di `C` saranno anche istanze di `C'`. La proprietà `rdfs:subClassOf` può essere utilizzata per dichiarare che una classe è una sottoclasse di un'altra. Questo costrutto permette di dichiarare gerarchie di classi, che supportano l'ereditarietà di proprietà di domain e range (definite nelle prossime righe).

La proprietà `rdfs:subClassOf` è una istanza di `rdf:Property` ed è usata per dichiarare che tutte le istanze di una classe sono istanze di un'altra.

Una tripla della forma `C1 rdfs:subClassOf C2` dichiara che `C1` è una istanza di `rdfs:Class`, `C2` è una istanza di `rdfs:Class` e che `C1` è una sottoclasse di `C2`. La proprietà `rdfs:subClassOf` è transitiva.

Ad esempio, la dichiarazione “Every Agent is a Person” diventa `foaf:Agent rdfs:subClassOf foaf:Person`.

rdfs:Literal

La classe `rdfs:Literal` è la classe dei valori literal quali stringhe ed interi. Valori propri quali stringhe di testo sono esempi di RDF literals. I literals possono essere di tipo “plain” o “typed”.

`rdfs:Literal` è un'istanza di `rdfs:Class`. `rdfs:Literal` è una sottoclasse di `rdfs:Resource`.

rdfs:Datatype

`rdfs:Datatype` è la classe dei datatypes. Tutte le istanze di `rdfs:Datatype` corrispondono al modello RDF di un datatype descritto nella specifica RDF Concepts. `rdfs:Datatype` è sia un'istanza che una sottoclasse di `rdfs:Class`. Ciascuna istanza di `rdfs:Datatype` è una sottoclasse di `rdfs:Literal`.

rdf:Property

La specifica “RDF Concepts and Abstract Syntax” descrive il concetto di una “RDF property” (proprietà RDF) come una relazione tra la risorsa soggetto e la risorsa oggetto. Essa descrive la `rdf:Property` come la classe delle proprietà RDF. `rdf:Property` è un'istanza di `rdfs:Class`. Ciascun membro della classe è un predicato RDF.

rdfs:domain

`rdfs:domain` di un predicato RDF dichiara la classe del soggetto in una tripla, il cui secondo componente è il predicato.

`rdfs:domain` è un'istanza di `rdf:Property` che è utilizzata per dichiarare che qualsiasi risorsa che ha una data proprietà è un'istanza di una o più classi.

Una tripla della forma `P rdfs:domain C` dichiara che `P` è un'istanza della classe `rdf:Property`, che `C` è un'istanza della classe `rdf:Class` e che le risorse denotate dal soggetto della tripla, il cui predicato è `P` sono istanze di `C`.

rdfs:range

`rdfs:range` di un predicato RDF dichiara la classe od il tipo di dato dell'oggetto di una tripla, il cui secondo componente è il predicato.

`rdfs:range` è una istanza di `rdf:Property` che è usata per dichiarare che i valori di una proprietà sono istanze di una o più classi.

La tripla `P rdfs:range C` dichiara che `P` è un'istanza della classe `rdf:Property`, che `C` è un'istanza della classe `rdf:Class` e che le risorse denotate dall'oggetto della tripla il cui predicato è `P` sono istanze della classe `C`.

Esempio: se le dichiarazioni seguenti sono usate per esprimere che la proprietà `ex:employer` relativa ad un soggetto, che è del tipo `foaf:Person`, e ad un oggetto, che è del tipo `foaf:Organization`:

```
ex:employer    rdfs:domain    foaf:Person
ex:employer    rdfs:range    foaf:Organization
```

Date le due dichiarazioni precedenti, la tripla seguente richiede che `ex:John` sia necessariamente un `foaf:Person` e che `ex:CompanyX` sia necessariamente una `foaf:Organization`, cioè:

```
ex:John ex:employer ex:CompanyX
```

E' da notare che le strutture di base fornite dal `rdfs:domain` e dal `rdfs:range` non forniscono nessuna maniera diretta di indicare restrizioni alle proprietà che sono locali ad una classe. Sebbene sia possibile combinare l'uso di questi due con gerarchie di `subproperty`, il supporto diretto per tali dichiarazioni è fornito dai linguaggi più ricchi per le web Ontology, quali l'OWL.

rdf:type

`rdf:type` è un'istanza di `rdf:Property` che è usata per dichiarare che una risorsa è una istanza di una classe. Una tripla della forma `R rdf:type C` dichiara che `C` è un'istanza di `rdfs:Class` e `R` è un'istanza di `C`.

rdfs:subPropertyOf

La specifica definisce anche il concetto di "sub-property" (sottoproprietà). La proprietà `rdfs:subPropertyOf` può essere utilizzata per dichiarare che una proprietà è una sottoproprietà di un'altra. Se una proprietà `P` è una sottoproprietà di proprietà `P'`, allora tutte le coppie di risorse che sono collegate tramite `P` sono collegate anche tramite `P'`.

La proprietà `rdfs:subPropertyOf` è un'istanza di `rdf:Property` che è quindi usata per dichiarare che tutte le risorse collegate da una proprietà sono anche collegate da un'altra proprietà.

Una tripla della forma `P1 rdfs:subPropertyOf P2` dichiara che `P1` è un'istanza di `rdf:Property`, `P2` è un'istanza di `rdf:Property` e che `P1` è subproperty di `P2`. Questa proprietà è transitiva.

Esistono anche altri costrutti dell'RDF Schema, che sono definiti nella specifica come "Utility Property", che sono utilizzati per fornire documentazione ed ulteriori informazioni su di uno schema RDF o sulle istanze. I principali sono:

rdfs:comment

`rdfs:comment` è un'istanza di `rdf:Property` che può essere utilizzata per fornire una descrizione di una risorsa in un modo che è leggibile dall'uomo. Questi commenti possono aiutare a capire il significato di classi e proprietà.

rdfs:label

`rdfs:label` è un'istanza di `rdf:Property` che può essere usata per fornire una versione leggibile dall'uomo del nome di una risorsa.

rdfs:seeAlso

`rdfs:seeAlso` è un'istanza di `rdf:Property` che è usata per indicare una risorsa che potrebbe fornire informazioni aggiuntive sulla risorsa del soggetto.

rdfs:isDefinedBy

`rdfs:isDefinedBy` è un'istanza di `rdf:Property` che è usata per indicare una risorsa che definisce la risorsa del soggetto. Può essere utilizzata per indicare un vocabolario RDF nel quale la risorsa è definita. `rdfs:isDefinedBy` è sottoproprietà di `rdfs:seeAlso`.

rdf:value

`rdf:value` è una istanza di `rdf:Property` che può essere usata per descrivere valori strutturati.

La specifica dell'RDF Schema comprende anche vocabolari per la reification degli RDF statement e per la gestione degli RDF Container e delle RDF Collection. In queste righe, non si approfondisce ulteriormente l'argomento per non appesantire la trattazione. Il lettore interessato può scaricare dalla rete nel sito della W3C l'intera specifica: in bibliografia è presente il collegamento.

Si è visto in questo paragrafo che l'RDF Schema fornisce le caratteristiche di base per descrivere i vocabolari RDF: altre capacità più sofisticate sono ipotizzabili e possono diventare utili. Queste proprietà ulteriori sono fornite attraverso gli sviluppi dell'RDF Schema, o in altri linguaggi sempre basati sull'RDF. Altre proprietà che arricchirebbero lo schema, che sono state identificate come utili, ma che non sono fornite dall'RDF Schema, includono:

- “cardinality constraints” (vincoli sulla cardinalità) sulle proprietà, quali ad esempio dire che una persona ha esattamente un padre;
- specificare che una data proprietà, quale `ex:hasAncestor` (ha antenato), è “transitiva”, cioè che se `A ex:hasAncestor B`, e `B ex:hasAncestor C`, allora `A ex:hasAncestor C`;
- specificare che una data proprietà è un identificatore univoco (o una key) per le istanze di una classe particolare;
- specificare che due classi diverse (che hanno `URIref` diverso) in realtà rappresentano la stessa classe;
- specificare che due istanze diverse (che hanno `URIrefs` diverso) in realtà rappresentano lo stesso individuo;
- specificare vincoli sul range o sulla cardinalità di una proprietà che dipende dalla classe delle risorse a cui le proprietà è applicata, ad esempio essere in grado di dire che per una squadra di calcio la proprietà `ex:hasPlayers` ha 11

valori, mentre per una squadra di basket la stessa proprietà dovrebbe avere solo 5 valori;

- l'abilità di descrivere nuove classi in termini delle combinazioni (ad esempio unioni e intersezioni) di altre classi, o dire che due classi sono disgiunte (ad esempio che nessuna risorsa è un'istanza di entrambe le classi).

Le capacità aggiuntive menzionate qui sopra, aggiunte a delle altre, sono l'obiettivo degli "ontology languages" quali DAML+OIL e l'OWL. Entrambi questi linguaggi sono basati sull'RDF e sull'RDF Schema ed entrambi, attualmente, forniscono le proprietà aggiuntive anticipate nelle righe precedenti. Nei prossimi paragrafi verrà approfondito l'OWL. L'obiettivo di tali linguaggi è quello di fornire una semantica aggiuntiva alle risorse che sia elaborabile dalle macchine cioè, di rendere le rappresentazioni delle risorse nelle macchine più simili alle loro controparti nel mondo reale. Mentre tali capacità non sono necessarie per costruire applicazioni usando l'RDF, lo sviluppo di tali linguaggi è una parte molto importante del lavoro degli sviluppatori, come parte dello sviluppo globale del Web Semantico.

3.4 OWL, OWL2, SPARQL, SKOS, RIF, SWRL

L' "OWL Web Ontology Language", abbreviato in OWL, è un linguaggio progettato per essere utilizzato da parte di quelle applicazioni che necessitano di elaborare il contenuto delle informazioni, invece che presentarlo solamente all'uomo. L'OWL facilita una maggior interpretabilità dei contenuti del Web rispetto a quella fornita dall'XML, dall'RDF e dall'RDF Schema, dato che fornisce dei vocabolari aggiuntivi assieme ad una semantica formale. Il OWL Web Ontology Language è un linguaggio di markup per pubblicare e condividere le ontologie nel World Wide Web. E' sviluppato come un'estensione del vocabolario dell'RDF. Esso deriva dal DAM+OIL Web Ontology Language.

L'OWL è una raccomandazione del W3C, pubblicata nel Febbraio del 2004. E' stata sviluppata dal "Web Ontology Working Group" come parte della "Semantic Web Activity" del W3C ad iniziare dal Novembre del 2001. La specifica del linguaggio OWL è attualmente composta da 6 documenti: OWL Overview ^[3.25] (che dà una breve introduzione delle caratteristiche del linguaggio), OWL Guide ^[3.26] (che spiega i rudimenti del linguaggio ed è ricca di esempi), OWL Reference (che dà una descrizione compatta, ma non formale, delle primitive dell'OWL), OWL Semantics and Abstract Syntax (che definisce formalmente il linguaggio), OWL Web Ontology Language Test Cases (contenente un insieme di test cases) ed infine OWL Use Cases and Requirements (contenente un insieme di use cases per un generico web ontology language).

L'acronimo naturale per il "Web Ontology Language" dovrebbe essere "WOL" invece che OWL. Nonostante il personaggio Owl (il Gufo) del cartone animato "Winnie the Pooh" scriva il suo nome WOL, l'acronimo OWL fu proposto senza far riferimento diretto a quel personaggio, ma come un acronimo facilmente pronunciabile che vorrebbe augurare un buona fortuna, suggerire buonsenso e ricordare un progetto nell'ambito della rappresentazione della conoscenza degli anni '70 di William A. Martin chiamato "One World Language".

Il Web Semantico, come già detto più volte, è una visione del Web futuro nella quale alle informazioni viene dato un significato esplicito, in modo da rendere più facile per le macchine elaborare ed integrare in modo automatico le informazioni disponibili nel Web. Il WS si costruirà quindi sull'abilità dell'XML di definire schemi di marcatura “su misura” e sull'approccio flessibile dell'RDF nel rappresentare i dati.

Ricordando la piramide tecnologica, introdotta ad inizio del capitolo, si vede che il livello appena sopra l'RDF, richiesto dal WS, è quello di un linguaggio ontologico che possa formalmente definire il significato della terminologia utilizzata nei documenti Web. Se si prevede che la macchine svolgano dei compiti ragionando su questi documenti, il linguaggio dovrà andare al di là della semantica di base fornita dall'RDF Schema. L'OWL è stato progettato per venire incontro a questa necessità, cioè quella di un Web Ontology Language.

Si può riassumere brevemente quanto presentato finora nei paragrafi precedenti.

L'XML fornisce uno strato “sintattico” per strutturare i documenti, ma non impone nessun vincolo semantico sul significato di questi documenti.

L'XML Schema è un linguaggio usato per restringere la struttura dei documenti XML e per estenderli grazie ai datatype.

L'RDF è un modello dei dati per le risorse e per le relazioni tra di esse, che fornisce una semplice semantica per questo modello dei dati e che permette al modello stesso di essere rappresentato nella sintassi dell'XML.

L'RDF Schema è un vocabolario utilizzato per descrivere le classi e le proprietà delle risorse RDF, con una semantica per la generalizzazione delle gerarchie di tali classi e risorse.

L'OWL, come verrà approfondito nelle prossime righe, aggiunge più vocabolario per descrivere classi e proprietà. Ad esempio, relazioni più complesse tra classi quali unione, cardinalità, uguaglianza, ecc.

L'OWL è diviso in tre sottolinguaggi a potenza espressiva crescente, ognuno dei quali è designato per un uso specifico.

- **OWL Lite:** serve come supporto per quegli utenti che hanno bisogno di rappresentare classificazioni gerarchiche e vincoli semplici. Esso fornisce anche una migrazione veloce di thesauri ed di altre tassonomie.
- **OWL DL:** supporta quegli utenti che hanno bisogno della massima espressività, ma che devono conservare la completezza computazionale (cioè tutte le conclusioni sono garantire essere computabili) e la decidibilità (cioè tutte le elaborazioni terminano in un tempo finito). Esso comprende tutti i costrutti del linguaggio OWL ma, questi ultimi, devono usati sotto certe condizioni, che si vedranno nelle prossime righe. L'aggettivo “DL” è dovuto alla corrispondenza del linguaggio con la “Description Logic” (logica descrittiva) che è un campo della ricerca che ha studiato le logiche che formano la base formale dell'OWL.
- **OWL Full:** questo è stato sviluppato per quegli usi che necessitano della massima espressività e della massima libertà sintattica dell'RDF: questo però non dà garanzie computazionali. L'OWL Full permette alle ontologie di aumentare il significato di vocabolari predefiniti, appartenenti sia all'RDF che all'OWL.

Ciascuno di questi vocabolari, quindi, è un'estensione dei suoi più semplici predecessori, sia per quello che può essere espresso in maniera formale, che in termini della validità delle conclusioni che possono essere dedotte.

Gli sviluppatori di ontologie che adottano l'OWL dovrebbero valutare quale sottolinguaggio meglio si adatta alle loro esigenze. La scelta tra OWL Lite e OWL DL dipende dal livello richiesto dagli utenti di costrutti più espressivi che sono forniti dall'OWL DL. Lo stesso discorso vale per la scelta tra OWL DL e OWL Full: quando si usa l'OWL Full rispetto all'OWL DL, il supporto al ragionamento è meno prevedibile dato che un'implementazione completa dell'OWL Full non esiste ancora. L'OWL Full può essere visto come un'estensione dell'RDF, mentre l'OWL Lite e l'OWL DL possono essere visti come estensioni di una visione ristretta dell'RDF. Ogni documento OWL è un documento RDF ed ogni documento RDF è un documento di OWL Full, ma solo certi documenti RDF saranno un documento valido per l'OWL Lite o per l'OWL DL. A causa di questo è necessaria, quindi, molta attenzione quando si vuole trasferire un documento dall'RDF all'OWL.

Si presentano ora le caratteristiche offerte dall'OWL Lite. Nel seguito sono introdotte anche quelle aggiunte dall'OWL DL e dall'OWL Full. E' da far notare subito che l'OWL DL e l'OWL Full contengono le stesse caratteristiche dell'OWL Lite, ma nell'OWL Full c'è più libertà su come queste ultime possono essere combinate tra loro. L'OWL Lite usa solo alcune delle caratteristiche del linguaggio OWL ed ha più limitazioni nell'uso di queste ultime rispetto agli altri due sottolinguaggi. I prefissi `rdf:` ed `rdfs:` sono utilizzati per gli elementi già presentati nell'RDF o nell'RDF Schema.

Per presentare, come fatto con l'RDF e l'RDF Schema, le caratteristiche dell'OWL, si seguirà la traccia delle specifiche dell'OWL fatta nei documenti "OWL Web Ontology Language - Overview" ed "OWL Web Ontology Language - Guide" ed "OWL Web Ontology Language - Reference". In essi sono presenti vari esempi che spiegano passo passo tutti i costrutti teorici forniti dall'OWL Lite. Nelle prossime righe si farà spesso riferimento ad essi. Il lettore che vuole conoscere tutta la parte più tecnica deve fare riferimento al documento "OWL Web Ontology Language - Semantics and Abstract Syntax".

Alcuni esempi, riportati nelle righe sottostanti, faranno riferimento ad una ontologia chiamata "wine and food", che è una ontologia di tipo OWL DL, creata sulla base di altre due ontologie già esistenti: "wine" e "food". Questa ontologia è stata sviluppata originariamente da McGuinness come un esempio di logica descrittiva. E' stata poi utilizzata all'interno della "DAML ontology library". Le due ontologie complete sono contenute nei due file "wine.rdf" e "food.rdf" scaricabili dal sito della W3C. Il link è riportato in bibliografia.^[3,27]

Tutti gli esempi qui riportati faranno riferimento alla sintassi dell'RDF/XML, presentata nei paragrafi precedenti. Questa infatti è la sintassi normativa dell'OWL, che è stato progettato per avere la massima compatibilità con l'RDF e l'RDF Schema.

L'OWL, come si è visto, è una componente delle tecnologie del Web Semantico. Seguendo l'obiettivo del WS, anche l'OWL deve consentire alle informazioni di poter essere raccolte da più sorgenti che sono sparse nel Web. Ciò è in parte fatto permettendo alle ontologie di essere collegate tra loro ed includendo l'esplicita importazione di informazioni da altre ontologie. Oltre a questo, si può dire che

l'OWL fa una assunzione detta di tipo "Open World", cioè che la descrizione delle risorse non è limitata ad un singolo scopo. Facendo una piccola digressione, l'assunzione Open World dice, tra le altre cose, che: "se una affermazione non può essere dimostrata vera usando la conoscenza attuale, allora non si può ottenere la conclusione che essa sia falsa". Tornando alle ontologie, mentre la classe C1 può essere definita originariamente nell'ontologia O1, essa può essere estesa ad altre ontologie. Le nuove informazioni che si aggiungono non possono ritrattare le informazioni precedenti; esse possono anche essere contraddittorie, ma i fatti e le implicazioni che ne derivano possono aggiungere e mai cancellare informazioni dell'ontologia originaria. Questa possibilità di contraddizione è un fatto che deve essere considerato da chi progetta un'ontologia. Ci si aspetta che in futuro ci siano degli strumenti che aiuteranno a scoprire tali casi.

Per poter scrivere un'ontologia che può essere interpretata in maniera non ambigua e utilizzata dai "futuri" agenti software, sono richieste una sintassi ed una semantica formali per l'OWL. Tutto questo è riportato nella parte della raccomandazione detta "Semantic and Abstract Syntax".

Una ontologia OWL è un grafo RDF, che a sua volta è un insieme di triple RDF. Come i grafi RDF anche un'ontologia può essere scritta in diverse forme sintattiche. Qui si farà riferimento a quella basata sull'RDF/XML.

Come già detto, l'OWL è un'estensione del vocabolario RDF. Un qualsiasi grafo RDF forma una ontologia di tipo OWL Full. Quindi, il significato dato ad un grafo RDF dall'OWL comprende ed, eventualmente, arricchisce il significato dato al grafo dall'RDF.

L'abilità dell'OWL di esprimere informazioni ontologiche sulle istanze che compaiono in molti documenti supporta il collegamento dei dati tra diverse sorgenti in un modo che è basato su dei principi. Questa semantica di base fornisce il supporto per dedurre da alcuni dati degli altri dati che l'utente non si sarebbe aspettato. Un esempio di ciò è data dalla proprietà `owl:sameAs` che è usata per dichiarare che due individui, che apparentemente sembrano diversi, in realtà sono lo stesso. Questo fa sì che le informazioni provenienti da diverse sorgenti, ma relative allo stesso individuo possano essere messe assieme. Questa aggregazione di dati può essere usata per determinare dei fatti che non sono direttamente rappresentati in nessuna delle due sorgenti. Questa abilità, allargabile all'intero Web Semantico, di collegare informazioni da molte sorgenti è una caratteristica molto desiderata e nello stesso tempo molto potente, che può essere utilizzata da molte applicazioni. Un problema che deriva da ciò, è l'abuso che può essere fatto da persone malintenzionate, specie nell'ambito della privacy che facilmente potrebbe essere violata. Sono già allo studio, da parte di organizzazioni, metodi per rendere sicuri questi dati.

Nelle righe che seguono si dà una breve descrizione dei costrutti dell'OWL. In qualche caso sarà presentato qualche esempio. Una trattazione approfondita dei costrutti esula dall'obiettivo di questa tesi. Si presentano questi costrutti per far capire al lettore cosa può essere espresso tramite l'OWL, cioè i vincoli e le specificazioni che aggiungono quel significato aggiuntivo che l'RDF non è in grado di esprimere. L'informazione nell'OWL è racchiusa nelle ontologie, le quali possono essere memorizzate nel World Wide Web.

Un documento OWL consiste di una o più “ontology headers”, seguita da un certo numero di definizioni di classi, di definizioni di proprietà e di fatti sugli individui.

Nello scrivere un'ontologia la prima cosa da fare è specificare in maniera precisa quali vocabolari saranno utilizzati. Un componente iniziale standard di una ontologia comprende un insieme di dichiarazioni di namespace XML (XML namespace declaration) che vengono racchiusi nel tag di apertura `rdf:RDF` : ciò dà un significato preciso a tutti gli identificatori e rende il più leggibile il resto della presentazione dell'ontologia.

Una ontologia OWL inizierà con una namespace declaration del tipo:

```
<rdf:RDF
  xmlns      ="http://www.w3.org/TR/2004/REC-owl-guide-
    20040210/wine#"
  xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-
    20040210/wine#"
  xml:base  ="http://www.w3.org/TR/2004/REC-owl-guide-
    20040210/wine#"
  xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-
    20040210/food#"
  xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

dove le prime due dichiarazioni identificano il namespace associato con questa ontologia; la terza identifica l'URI di base del documento dell'esempio; la quarta identifica il namespace della ontologia che supporta il food (cibo) con il prefisso `food:` . La dichiarazione del namespace della quinta riga dice che gli elementi il cui prefisso è `owl:` dovrebbero essere capiti come facenti riferimento a cose estratte dal namespace chiamato `http://www.w3.org/2002/07/owl#` . Questa è la dichiarazione convenzionale dell'OWL, utilizzata per introdurre il vocabolario OWL. Le righe sottostanti fanno riferimento all'RDF, all'RDF Schema ed al datatype dell'XML Schema.

Per questa dichiarazione del namespace `rdf:RDF` esistono altri formati abbreviati, usando la “ENTITY definition”:

```
<!DOCTYPE rdf:RDF [
  <!ENTITY vin  "http://www.w3.org/TR/2004/REC-owl-
    guide-20040210/wine#" >
  <!ENTITY food "http://www.w3.org/TR/2004/REC-owl-
    guide-20040210/food#" > ]>
```

le prime quattro righe delle dichiarazioni precedenti diventano:

```
rdf:RDF
  xmlns      ="&vin;"
  xmlns:vin ="&vin;"
  xml:base  ="&vin;"
  xmlns:food="&food;"
```

Una volta che i namespace sono stati stabiliti, normalmente si includono un insieme di asserzioni sull'ontologia, raggruppate sotto il tag `owl:Ontology`, che specificano commenti, controllo delle versioni e l'inclusione di altre ontologie. Questa parte è chiamata "ontology headers". Si riporta un altro pezzo dell'esempio precedente, spiegando le varie parti:

```
<owl:Ontology rdf:about="">
  <rdfs:comment>An example OWL ontology</rdfs:comment>
  <owl:priorVersion
    rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-
    20031215/wine"/>
  <owl:imports
    rdf:resource="http://www.w3.org/TR/2004/REC-owl-
    guide-20040210/food"/>
  <rdfs:label>Wine Ontology</rdfs:label>
  ...
```

dove è stato usato "" per indicare che manca una parte di testo, inutile per l'esempio.

L'elemento `owl:Ontology` è un luogo dove raccogliere la maggior parte dei metadati OWL per il documento. Esso non garantisce che il documento descriva una ontologia nel senso tradizionale. In alcune comunità, le ontologie non riguardano gli individui, ma solo le classi e le proprietà che definiscono un dominio. Quando si utilizza l'OWL per descrivere un insieme di dati di istanze il tag `owl:Ontology` può essere necessario al fine di registrare le informazioni sulla versione e per importare le definizioni da cui il documento dipende. Pertanto, nell'OWL, il termine "ontologia" è stato ampliato per includere i dati delle istanze.

L'attributo `rdf:about` fornisce un nome o un riferimento per l'ontologia. Se il valore è "" allora il nome dell'ontologia è l'URI base dell'elemento `owl:Ontology`.

L'attributo `rdfs:comment`, come già visto, fornisce la possibilità di aggiungere commenti.

`owl:priorVersion` è un tag standard che ha lo scopo di fornire dei "ganci" per il controllo della versione da parte di quei sistemi che utilizzano l'ontologia.

`owl:import` fornisce un meccanismo di inclusione. Esso prende un singolo argomento che è identificato dall'attributo `rdf:resource`. Dal punto di vista concettuale `owl:import` serve ad indicare l'intenzione di includere le asserzioni della ontologia che verrà importata. L'importare un'altra ontologia porta nell'ontologia corrente l'intero insieme di asserzioni di quell'ontologia. Se quest'ultima importa altre ontologie, anche tutte sono portate all'interno della prima.

`rdfs:label` supporta l'utilizzo di etichette per l'ontologia.

La parte iniziale della definizione dell'ontologia, detta "header ontology definition" si chiude con il tag `</owl:Ontology>`. Seguono poi le definizioni effettive che compongono l'ontologia ed il tutto termina con il tag `</rdf:RDF>`.

Tornando ad un aspetto un po' meno tecnico dell'OWL si presentano ora gli elementi principali di un'ontologia: classi, proprietà, istanze di classi e relazioni tra queste istanze. La presentazione di tutti i costrutti sarebbe molto lunga: come fatto finora, si analizzano solo i principali.

Classi

Le classi forniscono un meccanismo di astrazione per raggruppare le risorse che hanno caratteristiche simili. Come le classi dell'RDF, ogni classe OWL è associata con un insieme di individui, detti "class extension". Gli individui appartenenti alla class extension sono chiamati le istanze della classe.

Le classi OWL possono essere descritte attraverso le "class description", le quali possono essere combinate in "class axiom". Nelle prossime righe questi termini diventeranno più chiari.

Esiste nell'OWL una classe generale chiamata `owl:Thing` che è la classe di tutti gli individui ed la super-classe di tutte le classi dell'OWL. Esiste anche un'altra classe chiamata `owl:Nothing`, che è la classe che non ha nessuna istanza ed è la sottoclasse di tutte le classi dell'OWL.

Una class description è il termine con cui sono indicati i blocchi costruttivi di base di una class axiom, talvolta detta anche "class definition". Una class description descrive una classe OWL.

L'OWL distingue sei tipi di class description:

1. un "class identifier" (cioè un URI reference): descrive la classe tramite un class name. Una class description di questo tipo è sintatticamente rappresentata come una istanza nominata di `owl:Class`, una sottoclasse di `rdfs:Class`. Ad esempio: `<owl:Class rdf:ID="Human"/>`. Questo asserisce la tripla "`ex:Human rdf:type owl:Class`"., dove `ex:` è il namespace dell'ontologia.
2. una lista precisa di tutti gli individui che insieme formano le istanze della classe. Questa è detta anche "enumeration" ed è definita tramite la proprietà `owl:oneOf`. Il valore di questa proprietà deve essere una lista di individui che sono istanze della classe. Questa lista è rappresentata tramite il costrutto RDF `rdf:parseType="Collection"`. Ad esempio, per definire la classe di tutti i continenti il codice RDF/XML sarà:

```
<owl:Class>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Eurasia"/>
    <owl:Thing rdf:about="#Africa"/>
    <owl:Thing rdf:about="#NorthAmerica"/>
    <owl:Thing rdf:about="#SouthAmerica"/>
    <owl:Thing rdf:about="#Australia"/>
    <owl:Thing rdf:about="#Antarctica"/>
  </owl:oneOf>
</owl:Class>
```

3. una "property restriction", cioè una proprietà che restringe la classe. L'OWL distingue due tipi di property restriction: "value constraints" (che mette vincoli sul range della proprietà quando questa è applicata a questa particolare class description) e "cardinality constraints" (che vincola il numero di valori che una proprietà può assumere). Una property restriction ha la forma generale:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="(some property)" />
```

```
...
</owl:Restriction>
```

La classe `owl:Restriction` è definita come sottoclasse di `owl:Class`.

Fanno parte dei value constraints i costrutti: `owl:allValueFrom`, `owl:someValueFrom` e `owl:hasValue`.

Fanno parte dei cardinality constraints i costrutti: `owl:maxCardinality`, `owl:minCardinality` ed `owl:hasValue`.

- l'intersezione di due o più class description. Questo tipo di class constructor, assieme a quello di unione e di complementazione, rappresentano i costruttori di classi più avanzati che sono usati nella Logica Descrittiva. Essi possono essere visti come rappresentanti gli operatori tra classi AND, OR e NOT.

La proprietà `owl:intersectionOf` collega una classe ad una lista di class description. Una dichiarazione di questo tipo descrive una classe per la quale la class extension contiene esattamente quegli individui che sono membri della class extension di tutte le class descriptions nella lista. Un esempio:

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Tosca" />
        <owl:Thing rdf:about="#Salome" />
      </owl:oneOf>
    </owl:Class>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Turandot" />
        <owl:Thing rdf:about="#Tosca" />
      </owl:oneOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

In questo esempio il valore di `owl:intersectionOf` è una lista di due class descriptions, cioè due elenchi, entrambi descriventi una classe con due individui. L'intersezione risultante è una classe con un individuo, cioè Tosca.

- l'unione di due o più class description. Vale quanto detto con l'intersezione, con le opportune modifiche. Il costrutto è `owl:unionOf`.
- la complementazione di una class description. Una dichiarazione `owl:complementOf` descrive una classe per la quale la classe extension contiene esattamente quegli individui che non appartengono alla class extension della class description che è oggetto della dichiarazione.

La class description forma i blocchi costruttivi per definire le classi attraverso i class axioms.

La class axiom nella sua forma più semplice è una class description del tipo nr. 1. Essa dichiara solo l'esistenza di una classe usando `owl:Class` con un identificatore

di classe. Ad esempio, `<owl:Class rdf:ID="Human"/>`. Questo pezzo di codice OWL è corretto, ma non dice molto a riguardo della classe Human.

La class axiom contiene delle componenti aggiuntive che dichiarano caratteristiche necessarie e/o sufficienti di una classe. L'OWL contiene tre costrutti di linguaggio per combinare le class description in class axiom.

1. `rdfs:subClassOf` : permette di dire che la class extension di una class description è un sottoinsieme della class extension di un'altra class description.

Questo costrutto è definito come parte dell'RDF Schema. Il suo significato nell'OWL è lo stesso. Ad esempio:

```
<owl:Class rdf:ID="Opera">
  <rdfs:subClassOf rdf:resource="#MusicalWork" />
</owl:Class>
```

Questo class axiom dichiara una relazione di sottoclasse tra due classi OWL che sono descritte tramite i loro nomi (Opera e MusicalWork). Qui Opera è sottoclasse di MusicalWork.

L'axiom subclass fornisce una definizione parziale: esse rappresentano condizioni necessarie ma non sufficienti per stabilire l'appartenenza di un individuo ad una classe. Entrambe le condizioni sono fornite dal costrutto dell'OWL `owl:equivalentClass` , presentato qui sotto.

2. `owl:equivalentClass` è una proprietà che collega una descrizione di classe ad un'altra descrizione di classe. Il significato attribuito a tale class axiom è che le due class description implicate hanno la stessa class extension, cioè che contengono esattamente lo stesso insieme di individui. Un semplice esempio:

```
owl:Class rdf:about="#US_President">
  <equivalentClass
  rdf:resource="#PrincipalResidentOfWhiteHouse"/>
</owl:Class>
```

In questo caso il concetto di "Presidente degli Stati Uniti" è collegato, ma non uguale, al "principale residente della Casa Bianca". L'utilizzo di questo costrutto può essere molto più complesso.

3. `owl:disjointWith` è una proprietà dell'OWL che ha una class description oltre a domain e range. Una dichiarazione di questo tipo asserisce che la class extension delle due descrizioni di classe a riguardo non ha individui in comune. L'esempio più noto di class disjoint è:

```
<owl:Class rdf:about="#Man">
  <owl:disjointWith rdf:resource="#Woman"/>
</owl:Class>
```

che non ha, sicuramente, bisogno di commento.

Dal punto di vista sintattico, questi tre costrutti visti qui sono delle vere e proprie proprietà, aventi una class description come pure domain e range. Questi costrutti verranno richiamati nelle prossime righe dedicate alle proprietà dell'OWL.

Proprietà

L'OWL, come già anticipato, distingue tra due categorie principali di proprietà:

- “object properties”, che collega individui ad individui;
- “datatype properties”, che collega individui a valori dei dati.

Esistono anche altri tipi di proprietà, dette “annotation properties”, che sono necessarie all'OWL DL per il raginamento semantico.

Una object property è definita come un'istanza della classe predefinita in OWL `owl:ObjectProperty`. Le datatype properties sono relazioni tra istanze della classe e literal RDF o datatype di XML Schema. Esse consentono di mettere in relazione tra di loro gli oggetti (ad esempio: possiede, insegna, etc.)

Una datatype property è definita come un'istanza della classe predefinita in OWL `owl:DatatypeProperty`. Le object properties sono relazioni tra istanze di due classi. Esse consentono di mettere in relazione gli oggetti con i valori (ad esempio: numeroDiTelefono, nome, dataDiNascita, etc.).

Sia `owl:DatatypeProperty` che `owl:ObjectProperty` sono sottoclassi della classe RDF `rdf:Property`.

Un property axiom definisce le caratteristiche di una proprietà. Nella forma più semplice, un property axiom definisce l'esistenza di una proprietà. Ad esempio:

```
<owl:ObjectProperty rdf:ID="hasParent"/>
```

che definisce una proprietà con la restrizione che i suoi valori dovrebbero essere individui.

Molto spesso, i property axioms definiscono caratteristiche aggiuntive delle proprietà. L'OWL supporta i seguenti costrutti per i property axiom:

1. RDF Schema constructs. Questi costrutti sono già stati visti dettagliatamente nell'RDF Schema. Si riporta solo la lista: `rdfs:subPropertyOf`, `rdfs:domain` ed `rdfs:range`.
2. relazioni con altre proprietà. Queste sono `owl:equivalentProperty` e `owl:inverseOf`. Il primo di questi costrutti può essere utilizzato per dichiarare che due proprietà hanno la stessa property extension. Questa proprietà non deve essere confusa con la proprietà di uguaglianza, che viene espressa con il costrutto `owl:sameAs`.

Le proprietà hanno una direzione, dal domain al range. Nella pratica, la gente spesso trova utile definire relazioni in entrambe le direzioni. Ad esempio, “persone sono proprietarie di auto” e “auto sono possedute da persone”. Il costrutto `owl:inverseOf` può essere usato per definire tali relazioni inverse tra proprietà. Ad esempio, le relazioni “haFiglio” ed “haGenitore”:

```
<owl:ObjectProperty rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="#hasParent"/>
</owl:ObjectProperty> .
```

3. global cardinality constraints (vincoli globali sulla cardinalità): `owl:FunctionalProperty` e `owl:InverseFunctionalProperty`. Una

proprietà di tipo functional è una proprietà che può avere solo ed esclusivamente un valore y per ciascuna istanza di x , cioè non possono esistere due valori distinti y_1 ed y_2 tali che le coppie (x,y_1) e (x,y_2) siano entrambe istanze di questa proprietà. Un esempio per capire questo concetto dichiara che la proprietà marito è un functional, cioè che una donna può avere al massimo un marito; questo è anche un esempio di relazione ontologica che deriva dalla cultura.

```
<owl:ObjectProperty rdf:ID="husband">
  <rdf:type rdf:resource="#owl:FunctionalProperty" />
  <rdfs:domain rdf:resource="#Woman" />
  <rdfs:range rdf:resource="#Man" />
</owl:ObjectProperty>
```

Se una proprietà è dichiarata essere “inverse-functional”, allora l’oggetto di una dichiarazione di proprietà determina univocamente il soggetto.

Esempi di `owl:InverseFunctionalProperty` sono: “èCodiceFiscaleDi”, èNumeroDiMatricolaDi”.

4. logical property characteristics: `owl:TransitiveProperty` e `owl:SymmetricProperty` . Quando una proprietà P è definita essere transitiva, significa che se una coppia (x,y) è un’istanza di P e la coppia (y,z) è anch’essa istanza di P , allora si può dedurre che la coppia (x,z) è anch’essa un’istanza di P . Una proprietà è definita transitiva rendendola un’istanza della classe OWL `owl:TransitiveProperty` che è una sottoclasse di `owl:ObjectProperty` .
5. Una proprietà simmetrica è una proprietà dove vale che, se la coppia (x,y) è un’istanza di P , allora la coppia (y,x) è anch’essa un’istanza di P . Una proprietà è definita simmetrica rendendola un’istanza della classe OWL `owl:SymmetricProperty` che è una sottoclasse di `owl:ObjectProperty` . Un esempio tipico è la relazione “friendOf” (amico di):

```
<owl:SymmetricProperty rdf:ID="friendOf">
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range rdf:resource="#Human"/>
</owl:SymmetricProperty>
```

Si noti che il domain ed il range di una proprietà simmetrica sono lo stesso.

Individui

Gli individui sono definiti con assiomi individuali detti “facts” (fatti). Si considerano due tipi di fatti:

- fatti riguardanti l’appartenenza ad una classe e valori propri degli individui;
- fatti che riguardano identità individuale.

Relativamente ai fatti del primo tipo, si può dire che molti fatti sono degli statement che indicano un’appartenza degli individui ad una classe e dei valori propri degli individui. Un esempio, preso dalla documentazione sull’OWL, è:

```
Opera rdf:ID="Tosca">
  <hasComposer rdf:resource="#Giacomo_Puccini"/>
  <hasLibrettist rdf:resource="#Victorien_Sardou"/>
```

```

    <hasLibrettist rdf:resource="#Giuseppe_Giacosa"/>
    <hasLibrettist rdf:resource="#Luigi_Illica"/>
    <premiereDate rdf:datatype="&xsd;date">1900-01-
      14</premiereDate>
    <premierePlace rdf:resource="#Roma"/>
    <numberOfActs
      rdf:datatype="&xsd;positiveInteger">3</numberOfActs>
  </Opera>

```

Qui si vede che “Tosca” è un individuo della classe “Opera” ed i fatti quali la composizione da parte di Puccini e la data in cui è stata presentata per la prima volta, oltre ad altri fatti.

Relativamente ad i fatti del secondo tipo (detti individual identity), l’OWL fornisce tre costrutti per dichiarare fatti sull’identità degli individui.

1. `owl:sameAs` è usato per indicare che due URI reference si riferiscono allo stesso individuo, cioè i due individui hanno la stessa “identità”. Un esempio classico si fa con riferimento alle persone: in questo caso Bill Clinton e William Jefferson Clinton indicano la stessa persona.

```

<rdf:Description rdf:about="#William_Jefferson_Clinton">
  <owl:sameAs rdf:resource="#BillClinton"/>
</rdf:Description>

```

Questo costrutto è molto importante quando si definiscono mappe tra varie ontologie, visto che è impossibile che tutti si riferiscano alla stessa cosa (individuo) con lo stesso nome.

2. `owl:differentFrom` collega un individuo ad un altro ed è usato per indicare che due URI reference si riferiscono a individui diversi.
3. `owl:AllDifferent` fornisce uno strumento per dichiarare che gli individui di una lista sono tutti diversi. Considerando come esempio le opere.

```

<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Opera rdf:about="#Don_Giovanni"/>
    <Opera rdf:about="#Nozze_di_Figaro"/>
    <Opera rdf:about="#Cosi_fan_tutte"/>
    <Opera rdf:about="#Tosca"/>
    <Opera rdf:about="#Turandot"/>
    <Opera rdf:about="#Salome"/>
  </owl:distinctMembers>
</owl:AllDifferent>

```

Qui si dichiara che questi sei URIref puntano tutti ad opere diverse.

La specifica dell’OWL comprende altri argomenti, quali i Datatype e le Annotation, che potrebbero essere approfonditi, ma ciò esula dallo scopo di questo lavoro di tesi. Nella specifica tecnica sono inoltre riportati i costrutti che non sono permessi nelle versioni OWL Lite e OWL DL rispetto alla OWL Full e i vari vincoli per la scrittura del codice imposti dalle versioni minori dell’OWL.

E' interessante, invece, rendere noto che esistono, e sono già utilizzabili, molte ontologie. Esse possono essere consultate da appositi siti web o, in alcuni casi, anche modificate dall'utente. Soprattutto l'ambito biomedico è ricco di ontologie "pronte all'uso". Nel prossimo capitolo si svilupperà la famosa "Gene Ontology" ed alcune applicazioni che la utilizzano. Altre ontologie possono essere trovate ricercando, con i termini appropriati, tra i documenti con estensione ".owl" o ".rdf" oppure utilizzando opportuni motori di ricerca sul Web quale "Swoogle", che è già stato presentato nei capitoli precedenti.

Nel prossimo paragrafo si introduce l'evoluzione dell'OWL, che è rappresentata dall'OWL 2.

- . - . - . -

L' "OWL 2 Web Ontology Language", informalmente detto OWL 2, è un linguaggio ontologico per il Web Semantico con un significato formalmente definito. Le ontologie dell'OWL 2 stabiliscono le classi, le proprietà, gli individui ed i valori dei dati e come questi vengono memorizzati nei documenti del Web Semantico. Le ontologie dell'OWL 2 possono essere utilizzate insieme con le informazioni scritte in RDF; queste ontologie sono loro stesse scambiate principalmente come documenti RDF.

Lo sviluppo dell' OWL 2 è iniziato nell'Ottobre del 2007, da quando il "W3C OWL Working Group" ha iniziato ad estendere l'OWL con nuove caratteristiche. Il 27 Ottobre 2009 l'OWL 2 è stato "promosso" da Candidate Recommendation a Recommendation. Quest'ultima è composta da vari documenti che sono presentati nelle prossime righe. In bibliografia è riportato il link all'Overview che contiene tutti i riferimenti della specifica.^[3.28]

In questo paragrafo verrà introdotto l'OWL 2: si descriverà la sua sintassi, i diversi tipi di semantica, i profiles (sotto-linguaggi disponibili) ed il rapporto tra l'OWL e l'OWL 2.

L'OWL 2 è stato definito in modo da utilizzare tipi di dati definiti nel linguaggio XML Schema Definition (XSD). Attualmente, l'ultima raccomandazione del W3C per XSD è la versione 1.0; la versione 1.1 sta progredendo verso lo stato di raccomandazione. L'OWL 2 è stato progettato per sfruttare i nuovi tipi di dati e le spiegazioni più chiare dell'XSD 1.1 ma, per ora, queste possibilità sono parzialmente inutilizzate. In particolare, fino a che l'XSD 1.1 non diventa una raccomandazione del W3C, gli elementi di OWL 2 che si basano su di esso deve essere considerati facoltativi. Dopo che sarà pubblicato l'XSD 1.1 come recommendation, tali elementi cesseranno di essere facoltativi e dovranno essere considerati come richiesti.

Le ontologie sono dei vocabolari formalizzati di termini, che spesso coprono un dominio specifico e condiviso da una comunità di utenti. Essi precisano le definizioni dei termini per descrivere le loro relazioni con altri termini dell'ontologia. L'OWL 2 è un ampliamento ed una revisione del OWL Web Ontology Language sviluppato dal W3C Web Ontology Working Group (di seguito verrà indicato come "OWL 1"). L'OWL 2 è in fase di sviluppo ed è curato dal gruppo W3C OWL Working Group. Come l'OWL 1, l'OWL 2 è stato progettato per facilitare lo sviluppo delle ontologie e la condivisione tramite il Web, con lo scopo ultimo di rendere i contenuti Web più accessibili alle macchine.

La figura 3.6 fornisce una panoramica del linguaggio OWL 2: essa mostra gli blocchi costruttivi e come si collegano gli uni agli altri. L'ellisse nel centro rappresenta la nozione astratta di una ontologia, che può essere pensata sia come una struttura astratta che come un grafo RDF. Nella parte superiore ci sono le varie sintassi concrete che possono essere utilizzate per serializzare e scambiare le ontologie. Nella parte inferiore ci sono le due caratteristiche semantiche che definiscono il significato di ontologie dell'OWL 2.

La maggior parte degli utenti di OWL 2 utilizzerà una sola sintassi ed una sola semantica: lo schema diventerà molto più semplice, con una sola loro sintassi nella parte superiore ed una sola loro semantica in basso. Gli utenti hanno raramente la necessità di vedere cosa c'è dentro l'ellisse.

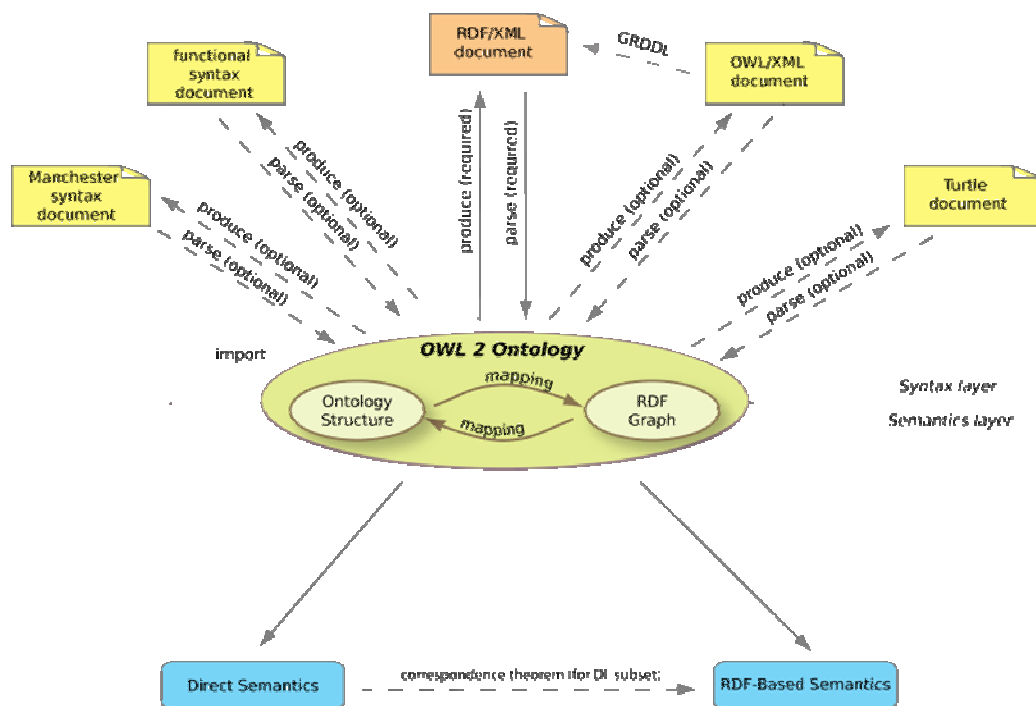


Figura 3.6 – Schematizzazione del linguaggio OWL 2

La struttura concettuale delle ontologie dell'OWL 2 è definita nel documento "OWL 2 Structural Specification". Questo documento utilizza l'UML (Unified Modeling Language) per definire gli elementi strutturali, che sono disponibili nell'OWL 2, spiegando il loro ruolo e la funzionalità in termini astratti e senza fare riferimento ad alcuna sintassi particolare. Esso definisce inoltre la sintassi del functional-style, che segue da vicino le specifiche strutturali e permette alle ontologie dell'OWL 2 di essere scritte in una forma compatta.

Ogni ontologia OWL 2 può anche essere vista come un grafo RDF. Il rapporto fra queste due visioni è specificato dal documento "Mapping to RDF Graph" che definisce una mappatura dalla forma strutturale alla forma grafica RDF, e viceversa. La "OWL 2 Quick Reference Guide" fornisce una semplice panoramica di queste due visioni dell'OWL 2, disposte fianco a fianco.

Dal punto di vista pratico, è necessaria una sintassi concreta per poter memorizzare le ontologie OWL 2 e per poterle scambiare tra le applicazioni. La sintassi principale di scambio dell'OWL 2 è la RDF/XML: questa è davvero l'unica che è supportata da tutto ciò che utilizza l'OWL 2. Una esatta definizione di ciò si trova nel documento "OWL 2 Conformance".

Oltre alla serializzazione RDF / XML per l'interoperabilità tra le applicazioni che usano l'OWL 2, esistono altre sintassi concrete che possono essere utilizzate. Queste includono serializzazioni RDF alternative quali una serializzazione XML come "Turtle" ed una più "leggibile" chiamata "Manchester Syntax", che è utilizzata in vari strumenti di editing ontologici. Infine, anche le sintassi functional-style possono essere utilizzate per la serializzazione, anche se il loro scopo principale è quello di specificare la struttura del linguaggio.

Si è detto prima che il documento OWL 2 Structural Specification definisce la struttura astratta delle ontologie OWL 2, ma non definisce il loro significato. La "Direct Semantics", scritta nel documento "OWL 2 Direct Semantics" e l' "RDF-Based Semantics" presentata nel testo "OWL 2 RDF-Based Semantics" forniscono due modi alternativi di assegnare un senso alle ontologie di tipo OWL 2, con un teorema di corrispondenza che fornisce un collegamento tra i due. Queste due semantiche sono utilizzate dai "ragionatori" e dagli altri strumenti, ad esempio, per rispondere a query per il recupero di istanze.

- La Direct Semantics assegna il significato direttamente alle strutture ontologiche, risultando con una semantica che è compatibile con la semantica del modello teorico della logica descrittiva di tipo SROIQ (che è un pezzo della logica del primo ordine, con utili proprietà computazionali). Il vantaggio di questo stretto legame con l'ampia letteratura sulla descrizione logica e sull'esperienza di implementazione disponibili, possono essere direttamente sfruttati dagli strumenti che usano l'OWL 2. Tuttavia, alcune condizioni devono essere imposte sulle strutture dell'ontologia, al fine di garantire che queste possano essere tradotte in una conoscenza di base di tipo SROIQ. Le ontologie che soddisfano tali condizioni sintattiche sono chiamate "OWL 2 DL ontologies": sono informalmente riferite come "OWL 2 DL" ed interpretate usando la semantica diretta.
- L'RDF-Based Semantics assegna il significato direttamente ai grafici RDF e quindi indirettamente alle strutture ontologiche attraverso la mappatura ai grafici RDF. L'RDF-Based Semantics è pienamente compatibile con la semantica RDF ed estende le condizioni semantiche definite per l'RDF. Inoltre l'RDF-Based Semantics può essere applicata a qualsiasi ontologia di tipo OWL 2 senza restrizioni, come pure ogni OWL 2 Ontology può essere mappata in RDF. Il termine "OWL 2 Full" è usato informalmente per riferirsi ai grafici RDF considerati come ontologie OWL 2 ed interpretati con la RDF-Based Semantics.

Il "correspondence theorem" (teorema di corrispondenza) definisce una precisa e stretta relazione tra la Direct e l'RDF-Based Semantics. Questo teorema, in sostanza, dichiara che, data un'ontologia OWL 2 DL, le deduzioni che si ottengono utilizzando la semantica diretta saranno ancora valide se l'ontologia è mappata in un grafo RDF ed interpretata con la RDF-Based Semantics.

Gli "OWL 2 Profiles" sono sottolinguaggi (nello specifico, sottoinsiemi sintattici) dell'OWL 2, che offrono vantaggi in particolari scenari di applicazione. Tre diversi

profili sono definiti: “OWL 2 EL”, “OWL 2 QL”, e “OWL 2 RL”. Ogni profilo è definito come una “restrizione sintattica” delle specifiche strutturali dell’OWL 2, vale a dire, come un sottoinsieme degli elementi strutturali che possono essere utilizzati in una ontologia conforme. Ciascuno di essi è più restrittivo dell’OWL DL. Ciascuno dei profili fa un compromesso tra i diversi aspetti della potenza espressiva dell’OWL in cambio di diversa capacità computazionale e/o dei benefici dell’implementazione.

- OWL 2 EL: consente algoritmi in tempo polinomiale per tutte le funzioni standard di un ragionamento. E’ particolarmente adatto per quelle applicazioni in cui sono necessaria ontologie molto grandi, ed in cui la potenza espressiva può essere scambiata per la garanzia delle prestazioni.
- OWL 2 QL: consente query congiunte che possono essere risolte in LOGSPACE (più precisamente, AC^0) utilizzando la tecnologia standard dei database relazionali. E’ particolarmente adatto per applicazioni in cui delle ontologie relativamente leggere vengono utilizzate per organizzare un gran numero di individui e dove è utile, o necessario, accedere ai dati direttamente tramite le query relazionali (ad esempio, SQL).
- OWL 2 RL: consente l’implementazione di algoritmi di ragionamento in tempo polinomiale utilizzando le tecnologie dei database rule-extended per operare direttamente sulle triple RDF. E’ particolarmente adatto per quelle applicazioni in cui le ontologie relativamente leggere vengono utilizzate per organizzare un gran numero di individui e dove è utile, o necessario, operare direttamente sui dati in forma di triple RDF.

Qualsiasi ontologia OWL 2 EL, QL o RL è, naturalmente, anche una ontologia OWL 2 e può essere interpretata usando o la Direct o l’RDF-Based Semantics.

L’OWL 2 ha una struttura globale molto simile all’OWL 1. Guardando la figura, si nota che quasi tutte le componenti dell’OWL 2 erano presenti nell’OWL 1, anche se con qualche nome diverso. Il ruolo centrale dell’RDF/XML, il ruolo di altre sintassi e le relazioni semantiche tra la Direct e la RDF-Based non sono cambiati. Ancora più importante è la compatibilità “all’indietro” con l’OWL 1, che è completa a tutti gli effetti: tutte le ontologie di tipo OWL 1 rimangono ontologie valide nell’OWL 2 con le deduzioni che sono identiche in tutti i casi pratici.

L’OWL 2 aggiunge nuove funzionalità rispetto alla OWL 1. Alcune delle nuove caratteristiche sono dette “syntactic sugar” (zucchero sintattico) (ad esempio, l’unione disgiunta di classi), mentre altre offrono nuove espressività, tra cui:

- chiavi;
- vincoli di proprietà;
- tipi di dati più ricchi, intervalli di dati;
- restrizioni di cardinalità più qualificate;
- proprietà asimmetriche, riflessive e disgiuntive;
- funzioni di annotation (annotazione/significato) avanzate.

L’OWL 2 definisce inoltre tre nuovi profili ed una nuova sintassi, la Manchester. Inoltre, alcune delle restrizioni applicate all’OWL DL sono stati attenuate: ne consegue che l’insieme dei grafi RDF che possono essere gestiti dai ragionatori Description Logics è leggermente più grande nell’OWL 2.

Tutto quanto è stato presentato sopra è documentato in dettaglio nel documento chiamato “OWL 2 New Features and Rationale” (OWL 2 nuove funzioni e

motivazione). Il OWL 2 Quick Reference Guide fornisce anche una panoramica delle nuove caratteristiche dell'OWL 2. Qui sotto è presentata una lista strutturata delle categorie dei costrutti, divisa per tipologie e ricavata dalla specifica, che non verrà approfondita:

- 1) Syntactic sugar : DisjointUnion, DisjointClasses, NegativeObjectPropertyAssertion, NegativeDataPropertyAssertion.
- 2) New constructs for properties: Self Restriction, Property Qualified Cardinality Restrictions, Reflexive, Irreflexive, and Asymmetric Object Properties, Disjoint Properties, Property Chain Inclusion, Keys.
- 3) Extended datatype capabilities, Extra Datatypes and Datatype Restrictions, N-ary Datatypes, Datatype Definitions, Data Range Combinations.
- 4) Simple metamodeling capabilities: Punning.
- 5) Extended Annotations: Annotations, Axioms about annotation properties.
- 6) Other Innovations: Declarations, Top and Bottom Properties, IRIs, Imports and Versioning.
- 7) Minor features: Anonymous Individuals, Inverse Properties.

L'OWL 2 è definito dal punto di vista normativo da cinque documenti di specifica di base che ne descrivono la struttura concettuale, una sintassi principale di scambio (RDF/XML), due semantiche alternative (Direct e RDF-Based) e dei requisiti di conformità. Tre documenti aggiuntivi di specificazione descrivono le caratteristiche opzionali che possono essere supportate da alcune implementazioni: i profili di lingua, e due sintassi alternative concrete (OWL/XML e Manchester). Questi documenti sono, tuttavia, tutti piuttosto tecnici e, soprattutto, sono mirati agli implementatori e agli sviluppatori di tool per l'OWL 2. Coloro che cercano una guida più avvicinabile alle caratteristiche e l'utilizzo di OWL 2 possono consultare uno dei documenti per gli utenti (Overview, Primer, New Features and Rationale (motivazione) e Quick Reference Guide). Tutti questi sono scaricabili dal sito della W3C all'indirizzo internet riportato in bibliografia. Ogni documento contiene anche i link agli altri documenti e a quelli che riguardano gli aspetti tecnologici precedenti. Undici sono i documenti che compongono la raccomandazione OWL 2.

Essi sono:

- 1 OWL 2 Web Ontology Language XML Serialization
- 2 OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax
- 3 OWL 2 Web Ontology Language RDF-Based Semantics
- 4 OWL 2 Web Ontology Language Quick Reference Guide
- 5 OWL 2 Web Ontology Language Profiles
- 6 OWL 2 Web Ontology Language Primer
- 7 OWL 2 Web Ontology Language Document Overview
- 8 OWL 2 Web Ontology Language New Features and Rationale
- 9 OWL 2 Web Ontology Language Mapping to RDF Graphs
- 10 OWL 2 Web Ontology Language Direct Semantics
- 11 OWL 2 Web Ontology Language Conformance

- . - . - . - .

Lo “SPARQL” è un linguaggio che permette di fare query (interrogazioni) sulle informazioni contenute nei documenti RDF. Il suo nome è un acronimo ricorsivo che sta per “SPARQL Protocol and RDF Query Language”. E’ stato standardizzato dal “RDF Data Access Working Group (DAWG)” della W3C ed è considerato una tecnologia chiave del Web Semantico. A partire da 15 Gennaio 2008 esso è diventato una raccomandazione ufficiale ^[3.29]. In bibliografia è presente il collegamento al documento ufficiale.

Nel Maggio del 2006 durante un suo intervento, Sir Berners-Lee disse: “SPARQL will make a huge difference”, cioè “lo SPARQL farà una grande differenza”.

Lo SPARQL può essere utilizzato per esprimere query in maniera trasversale tra diverse sorgenti di dati, sia che i dati sia memorizzati nel formato RDF o che essi siano visti come RDF grazie all’aiuto di “middleware”, cioè di applicazioni che si interpongono tra sorgente di dati ed applicazione che li utilizza. Lo SPARQL ha la capacità per fare query su vari modelli di grafo insieme con operazioni su di loro, quali unione, ecc. I risultati delle interrogazioni SPARQL possono risultare essere insiemi di dati o grafi RDF.

La raccomandazione ufficiale, definisce la sintassi e la semantica del linguaggio per interrogare i documenti RDF.

Lo SPARQL è strettamente connesso a due ulteriori specifiche:

- la specifica “SPARQL Protocol for RDF” definisce il protocollo per fare, da remoto, query e su come ottenere i risultati.
- la specifica “SPARQL Query Result XML Format” definisce un formato di documento XML per rappresentare i risultati delle query di tipo SELECT ed ASK.

Nello SPARQL si assumo noti i prefissi `rdf:`, `rdfs:`, `xsd:` ed `fn:`, questo ultimo facente riferimento al <http://www.w3.org/2005/xpath-functions#>. Esso utilizza il formato Turtle per rappresentare i dati, per mostrare in maniera esplicita le triple e per l’utilizzo abbreviato degli IRI con i prefissi.

I risultati delle query vengono presentati in forma tabellare.

Ad esempio:

x	y	z
"Alice"	<http://example/a>	

Un “binding” è una coppia (variabile, termine RDF). In questo insieme di risultati ci sono tre variabili, scritte nell’intestazione della colonna: **x**, **y** e **z**.

Ciascuna soluzione è mostrata come una riga nella tabella. In questo caso la soluzione è unica, nella quale la variabile **x** assume il valore "Alice", la variabile **y** il valore <http://example/a>, mentre la variabile **z** non assume nessun termine RDF.

Uno dei vantaggi dell’utilizzo di SPARQL, deriva dal fatto che esso permette agli utenti di scrivere query che non sono ambigue. E’ da notare che questa non ambiguità globale si radica nel fatto che ogni identificatore nello SPARQL, cioè un URI, è globalmente non ambiguo. In altri linguaggi, quali l’SQL, termini quali "email" o "e-mail" sono invece normalmente usati.

Molte forme di query SPARQL contengono un insieme di tipi di triple chiamate “basic graph pattern”. Questi tipi di triple sono come le triple RDF ma con la differenza che il soggetto, il predicato e l’oggetto possono essere una variabile.

Un basic graph pattern “matches” (si accoppia con) un sottografo di dati RDF quando i termini RDF di quel sottografo possono essere sostituiti per le variabili ed il risultato è un grafo RDF equivalente al sottografo.

Qui sotto riportato un esempio di SPARQL query, tratto dalla documentazione della specifica. E’ presentato per far capire al lettore come “lavora” questo linguaggio di interrogazione. Una query consiste di due parti:

- **SELECT** questa clausola identifica le variabili che devono apparire nel risultato dell’interrogazione.
- **WHERE** questa clausola fornisce il basic graph pattern che deve essere accoppiato con i dati del grafo.

Nell’esempio, il basic graph pattern consiste di un tipo singolo di tripla con una singola variabile (`?title`) posta nella posizione dell’oggetto:

Questa è la tripla RDF che contiene i DATI:

```
<http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

Questa è la QUERY scritta in SPARQL:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1>
  <http://purl.org/dc/elements/1.1/title> ?title .
}
```

E questo è il RISULTATO prodotto dalla query:

title
"SPARQL Tutorial"

Il risultato di una query è una sequenza di soluzioni, corrispondenti al modo con il quale il graph pattern della query si accoppia con i dati. Una query può avere zero, una o più soluzioni. E’ possibile fare il matching con literal RDF, con tipi numerici e con datatype arbitrari. Le query possono contenere anche “blank nodes”: essi sono scritti nella forma “_:” seguiti da un’etichetta di blank node.

Lo SPARQL permette alle query di assumere diverse forme: la query di tipo **SELECT** ritorna un legame con le variabili. La query di tipo **CONSTRUCT** ritorna un grafo RDF: questo grafo è costruito basandosi su di un template (modello) che è usato per generare triple RDF e che è basato sui risultati dell’accoppiamento del graph pattern della query.

Lo SPARQL permette alle query di essere distribuite a più SPARQL endpoints, ognuno dei quali la elabora. Tutti i risultati sono poi raccolti: questa operazione è nota col nome di “federated query” (query associata).

Si è visto che il graph pattern matching produce una sequenza di soluzioni, dove ciascuna soluzione ha un insieme di collegamenti di variabili ai termini RDF. Il costrutto dello SPARQL `FILTER` permette di restringere le soluzioni solo a quelle che soddisfano la condizione del filtro.

Lo SPARQL ha anche altri costrutti che permettono ottenere le sequenze di soluzioni in maniera ordinata (costrutto `ORDER BY`) o presentate senza duplicati (costrutto `DISTINCT`), ecc. E' possibile anche effettuare dei test sui valori ed anche operazioni di conversione di graph pattern. Lo SPARQL supporta anche il test dei vincoli sul grafico RDF sorgente.

Per una panoramica completa delle possibilità offerte dallo SPARQL, ed un ulteriore approfondimento, si può consultare la specifica che è ricca di semplici esempi. Il collegamento è riportato in bibliografia.

- . - . - . - .

Lo SKOS, acronimo di “Simple Knowledge Organization System”, è una famiglia di linguaggi formali che fornisce un modello per esprimere la struttura di base ed il contenuto di schemi concettuali quali glossari, classificazioni, tassonomie e qualsiasi tipo di vocabolario strutturato.

Da un punto di vista più tecnico, lo SKOS è un vocabolario RDF usato per rappresentare KOSs (semi-formal Knowledge Organization Systems) quali sono quelli appena citati qui sopra. Essendo basato sull’RDF, lo SKOS permette a questi sistemi di rappresentazione della conoscenza di essere espressi come dati machine-readable. Questo sistema può essere scambiato tra applicazioni e pubblicato, sempre in un formato machine-readable, sul World Wide Web.

Lo scopo dello SKOS è quello di fornire un “percorso” a basso costo per portare gli “organization system” verso il Web Semantico. Esso fornisce un linguaggio, leggero ed intuitivo, per i modelli concettuali che permette di sviluppare e distribuire in rete nuovi KOS. Esso può essere utilizzato da solo o insieme con altri linguaggi più formali quali l’OWL. Può essere visto come una tecnologia che “fa da ponte”, cioè che fornisce il collegamento mancante tra il formalismo rigoroso dei linguaggi ontologici quali l’OWL ed il caotico, informale e mal strutturato mondo di “collaboration tools” che sono basati sul Web, quali le applicazioni di social tagging. In altre parole, lo scopo dello SKOS non è quello di sostituire gli originari vocabolari concettuali nel loro contesto d’uso iniziale, ma di permettere di portarli in uno spazio distribuito, basato su un modello semplificato e permettendo un loro più ampio riuso ed una migliore interoperabilità.

Dal punto di vista storico, lo SKOS nasce nel 2001 come un risultato del progetto della Comunità Europea “LIMBER” (Language Independent Metadata Browsing of European Resource). Negli anni seguenti fu ulteriormente sviluppato nel “Thesaurus Activity Work Package” nel progetto SWAD-Europe (Semantic Web Advanced Developed) sempre della Comunità Europea. Questo progetto fu designato come supporto dalla W3C Semantic Web Activity. Nel 2003 fu rilasciata la prima versione dello “SKOS Core” e dello “SKOS Mapping”. Al termine del progetto SWAD, lo SKOS, negli anni successivi, è stato supportato e sviluppato dalla W3C che ha pubblicato “SKOS Core Guide” e “SKOS Core Vocabulary Specification”,

documenti ora considerati deprecati. Dal 2006 è iniziato l'iter per far diventare lo SKOS una raccomandazione ufficiale: lo è diventato il 18 Agosto 2009.^[3.30]

Assieme al documento di specifica, è stato pubblicato dal W3C Working Group Note il documento "SKOS Simple Knowledge Organization System Primer" che aiuta l'utente principiante che volesse rappresentare i propri schemi concettuali tramite questo linguaggio.

Il modello dei dati dello SKOS (SKOS data model) è formalmente definito come una ontologia di tipo OWL Full. I dati SKOS (SKOS data) sono espressi come triple RDF e possono essere serializzati tramite qualsiasi sintassi RDF, quale RDF/XML, Turtle, ecc.

Lo SKOS data model considera un "knowledge organization system" (KOS) come uno schema concettuale (concept scheme) comprendente una serie di concetti (concepts). Questi SKOS concept e gli SKOS concept scheme sono identificati da URI che permettono a chiunque di riferirsi a loro in maniera non ambigua ed indipendente dal contesto e rendendo tutto questo parte del Web.

Questo concetti possono essere etichettati con un numero qualsiasi di stringhe lessicali di tipo Unicode ed in un qualsiasi linguaggio naturale. Queste label possono essere anche nascoste. A questi concetti possono poi essere assegnate una o più "notation" (annotazioni) che altro non sono che dei codici lessicali usati per identificare in maniera univoca il concetto all'interno dell'estensione di un dato schema concettuale. Questa notation fornisce un "ponte" verso altri sistemi di identificazione già in uso.

Gli SKOS concept possono poi essere documentati con note di vario tipo. Il modello dei dati dello SKOS fornisce un insieme base di "documentation properties" che supportano le "scope notes" e le note di definizioni ed editoriali. Questo insieme può essere espanso anche da terze persone.

I concetti SKOS possono anche essere collegati ad altri concetti SKOS tramite proprietà di relazioni semantiche. Il modello dei dati dello SKOS fornisce supporto per collegamenti gerarchici ed associativi tra concetti SKOS. Anche qui è possibile una estensione di questo da parte di terze parti.

Oltre a questo, i concetti SKOS possono essere raggruppati in "collection" che possono, a loro volta, essere etichettate e/o ordinate. Questa caratteristica supporta anche l'etichettatura dei nodi all'interno dei thesauri e per quei casi in cui l'ordinamento di un insieme di concetti è significativo oppure fornisce alcune utili informazioni.

I concetti SKOS possono essere mappati verso altri concetti SKOS che sono contenuti in schemi concettuali diversi. Lo SKOS fornisce quattro tipi base di mapping link: hierarchical, associative, close equivalent ed exact equivalent. Questi sono noti come "mapping properties".

Nelle prossime righe si approfondiranno alcuni degli aspetti presentati finora. Per una trattazione completa e dettagliata dello SKOS sarebbero necessarie decine e decine di pagine, ma lo scopo di queste pagine è quello di far capire al lettore le potenzialità di questo nuovo linguaggio.

Prima di fare questo, si apre una piccola parentesi sui rapporti tra lo SKOS, l'OWL e l'RDF.

Gli elementi del modello dei dati dello SKOS sono le classi e le proprietà. La struttura e l'integrità di questo modello è definita dalle caratteristiche logiche di queste classi e proprietà, oltre alle loro interdipendenze. Queste è uno degli aspetti più potenti, ma ancora poco chiari, dello SKOS che in qualche applicazione può lavorare fianco a fianco con l'OWL per esprimere e scambiare conoscenze su di un dominio. Tuttavia, lo SKOS non è un linguaggio formale per la rappresentazione della conoscenza. Si approfondisce questo punto, molto importante per la comprensione teorica dello SKOS. La parte sintattica verrà presentata nelle prossime pagine, con i principali costrutti dello SKOS di base.

La “conoscenza” resa esplicita in una ontologia formale è espressa come un insieme di assiomi e di fatti.

Un thesauro o uno schema di classificazione identifica e descrive, attraverso il linguaggio naturale, un insieme di idee o significati che sono spesso riferiti come “concetti”. Questi ultimi possono essere sistemati ed organizzati in varie strutture che non hanno nessuna semantica formale e che non possono essere interpretati con fiducia come assiomi formali o come fatti sul mondo.

Per includere la “conoscenza” in un thesauro o in uno schema in modo formale è necessario che il thesauro o lo schema vengano “reingegnerizzati” come una ontologia formale. Questo significa che alcune persone devono “trasformare” la struttura ed il contenuto intellettuale del thesauro o dello schema in un insieme di assiomi formali e di fatti: questo comporta una spesa di “intelletto” e di tempo, quindi un costo.

Molto può essere guadagnato dall'utilizzo dei thesauri “così come sono”, cioè come una struttura non formale che consente di muoversi all'interno di un dominio. Utilizzandoli “così come sono”, l'operazione di re-engineering non è necessaria. In aggiunta, alcuni KOS non rappresentano una vista logica del loro dominio. Un'operazione di conversione ad una rappresentazione formale basata sulla logica potrebbe fare dei cambiamenti che risultano in una rappresentazione che è ben diversa dallo scopo originario.

L'OWL, si è visto, fornisce un linguaggio di modellazione dei dati molto potente. Si può perciò utilizzare l'OWL per costruire un modello dei dati che rappresenta i thesauri o gli schemi di classificazione “così come sono”. Questo è quello che fa lo SKOS. Seguendo questo approccio, i “concetti” di un thesauro o di uno schema sono modellati come individui del modello dei dati dello SKOS e le descrizioni informali sui collegamenti tra questi concetti, come espresse dal thesauro o dallo schema, sono modellate come fatti su quegli individui, mai come classi o proprietà. E' da notare che questi sono loro stessi fatti sui thesauri, cioè come “il concetto X che ha un'etichetta preferita Y, è parte del thesauro Z”; questi non sono fatti su come il mondo è organizzato all'interno di un particolare dominio del soggetto, come potrebbe essere espresso in un'ontologia. I dati SKOS sono poi espressi come triple RDF.

Si riporta un esempio, tratto dallo SKOS Reference, che riporta alcuni fatti su di un thesauro:

```

<A> rdf:type skos:Concept ;
      skos:prefLabel "love"@en ;
      skos:altLabel "adoration"@en ;
      skos:broader <B> ;
      skos:inScheme <S> .

<B> rdf:type skos:Concept ;
      skos:prefLabel "emotion"@en ;
      skos:altLabel "feeling"@en ;
      skos:topConceptOf <S> .

<S> rdf:type skos:ConceptScheme ;
      dct:title "My First Thesaurus" ;
      skos:hasTopConcept <B> .

```

Qui il concetto <A> etichettato con “love” o alternativamente “adoration” che appartiene allo schema di concetto <S>, può essere ampliato nel concetto etichettato con “emotion” o “feeling” che è un “top concept” dello schema di concetto <S> intitolato “My first thesaurus”.

Questo è il punto fondamentale per capire la definizione formale di modello dei dati dello SKOS e come ciò può essere implementato in un sistema software.

Definendo uno SKOS data model come una ontologia di tipo OWL Full, il Web Semantico potrà, in seguito, essere utilizzato come un mezzo per pubblicare, scambiare, condividere e collegare dati che coinvolgono questi sistemi di organizzazione della conoscenza. Per questa ragione, per l’espressività dell’OWL Full come linguaggio per la modellazione dei dati e per la possibilità di usare i thesauri e gli classification scheme fianco a fianco con le ontologie formali, l’OWL Full è stato usato per definire il modello dei dati dello SKOS.

Si introduce ora il “core” del modello SKOS, cioè le caratteristiche che sono necessarie per rappresentare i principali KOSs.

Il namespace URI dello SKOS è <http://www.w3.org/2004/02/skos/core#> , indicato con `skos`:

Il vocabolario SKOS è un insieme di URI, che verranno presentati nelle prossime righe, con la spiegazione del loro uso.

L’elemento fondamentale del vocabolario SKOS è, come visto, il “concetto”. I concetti sono l’unità di pensiero e sono il fondamento dei sistemi di organizzazione della conoscenza. Come tali essi esistono nel pensiero come entità astratte che sono indipendenti dai termini usati per etichettarli.

Lo SKOS introduce la classe `skos:Concept` che permette agli implementatori di asserire che una data risorsa è un concetto. Questo è fatto in due passi:

1. creando un URI che identifica univocamente il concetto;
2. facendo una asserzione in RDF, tramite le proprietà `rdf:type` che la risorsa identificata da quell’URI è del tipo `skos:Concept`

Ad esempio:

```
<http://www.example.com/animals> rdf:type skos:Concept.
```

o in Turtle:

```
ex:animals rdf:type skos:Concept.
```

In questo caso la risorsa “animals” è un concetto.

La prima caratterizzazione dei concetti sono le espressioni che sono usate per riferirsi a loro in linguaggio naturale: le “label” (etichette). Lo SKOS fornisce tre proprietà per attaccare etichette alle risorse concettuali, ognuna implicante uno specifico stato di preferita, alternativa e nascosta (non visibile all’utente, ma solo all’applicazione).

Esse sono: `skos:prefLabel` , `skos:altLabel` e `skos:hiddenLabel` .

Ad esempio, con il concetto di animali:

```
ex:animals rdf:type skos:Concept;
skos:prefLabel "animaux"@fr;
skos:altLabel "bêtes"@fr;
skos:hiddenLabel "betes"@fr.
```

In questo esempio l’etichetta preferita, in lingua francese, è “animaux”. Un’alternativa è data dall’etichetta “bêtes”. Un’etichetta nascosta, utile al sistema per correggere errori di battitura, è “betes”, scritta con la “e” normale.

Nei KOSs le “semantic relation” hanno un ruolo cruciale nel definire i concetti. Il significato di un concetto è definito non solo dalle parole del linguaggio naturale delle sue etichette, ma anche dai suoi collegamenti agli altri concetti del linguaggio. Lo SKOS fornisce tre proprietà standard:

- `skos:broader` e `skos:narrower` permettono la rappresentazione di link gerarchici quali sono le relazioni tra un genere e le sue più specifiche specie o, a seconda dell’interpretazione, le relazioni tra un intero e le sue parti. Per asserire che un concetto è un allargamento, una espansione, del significato di un altro si usa la proprietà `skos:broader`. La proprietà `skos:narrower` è utilizzata per dichiarare l’inverso, cioè quando un concetto è una restrizione di un altro. Ad esempio:

```
ex:animals rdf:type skos:Concept;
skos:prefLabel "animals"@en;
skos:narrower ex:mammals.
ex:mammals rdf:type skos:Concept;
skos:prefLabel "mammals"@en;
skos:broader ex:animals.
```

- `skos:related` permette la rappresentazione di link di tipo associativo e non gerarchico, quali sono le relazioni tra un tipo di evento e la categoria di entità alla quale partecipa. Questo costrutto può essere utilizzato per collegare due categorie dove nessuna delle due è più generale o più specifica. Questa proprietà è simmetrica. Ad esempio:

```
ex:birds rdf:type skos:Concept;
skos:prefLabel "birds"@en;
skos:related ex:ornithology.
ex:ornithology rdf:type skos:Concept;
skos:prefLabel "ornithology"@en.
```

Di queste proprietà esistono anche le versioni che soddisfano la proprietà di transitività. Esse sono: `skos:broaderTransitive` e `skos:narrowerTransitive`.

Oltre alla caratterizzazione strutturata dei concetti vista qui sopra, i concetti talvolta necessitano di essere ulteriormente definiti usando una documentazione informale ma leggibile dall'uomo. Lo SKOS fornisce la proprietà `skos:note` per scopi generali di documentazione. Questa è ulteriormente specializzata con altri costrutti che sono:

- `skos:scopeNote` - fornisce alcune possibili parziali informazioni sul significato inteso di un concetto: indica come usare un concetto;
- `skos:definition` fornisce una spiegazione completa del significato inteso di un concetto;
- `skos:example` - fornisce un esempio dell'uso del concetto;
- `skos:historyNote` - descrive i cambiamenti significativi o la forma di un concetto;
- `skos:editorialNote` - fornisce informazioni che sono di aiuto per chi gestirà il concetto dal punto di vista editoriale;
- `skos:changeNote` - documenta i minimi cambiamenti apportati al concetto.

I concetti possono essere creati ed utilizzati come entità a sé stanti. Tuttavia, specialmente nella pratica di indicizzazione, i concetti solitamente vanno dentro a vocabolari, quali thesauri o schemi di classificazione, che sono compilati con cura. Lo SKOS offre la possibilità di rappresentare tali KOS usando la classe `skos:ConceptScheme`. L'esempio qui sotto, preso dalla documentazione ufficiale dello SKOS, mostra come definire una risorsa come schema concettuale (qui si rappresenta un thesauro sugli animali ed il costrutto `dct:` è relativo al vocabolario Dublin Core).

```
ex:animalThesaurus rdf:type skos:ConceptScheme;
dct:title "Simple animal thesaurus";
dct:creator ex:AntoineIsaac.
```

Una volta che la risorsa schema concettuale è stata creata, essa può essere collegata con il concetto che essa contiene usando la proprietà `skos:inScheme`. Nell'esempio:

```
ex:mammals rdf:type skos:Concept;
skos:inScheme ex:animalThesaurus.
ex:cows rdf:type skos:Concept;
skos:broader ex:mammals;
skos:inScheme ex:animalThesaurus.
ex:fish rdf:type skos:Concept;
skos:inScheme ex:animalThesaurus.
```

Al fine di fornire un accesso efficiente al punto d'entrata di una gerarchia concettuale di tipo *broader/narrower*, lo SKOS definisce la proprietà `skos:hasTopConcept`. Questa permette ad uno di collegare un concept scheme ai concetti più generali che esso contiene. Nell'esempio di prima:

```
ex:animalThesaurus rdf:type skos:ConceptScheme;
skos:hasTopConcept ex:mammals;
skos:hasTopConcept ex:fish.
```

Si fa notare che il lettore dovrebbe essere consapevole che ci sono delle sottili differenze tra gli schemi concettuali SKOS e i tradizionali KOS, principalmente dovuti al contesto del Web Semantico degli SKOS. Una importante caratteristica degli SKOS è che è possibile per lo stesso concetto essere collegato a diversi schemi concettuali, tramite la proprietà `skos:inScheme`. Inoltre è da notare che il vocabolario SKOS offre un supporto che è limitato per il contenimento di informazioni KOS in uno schema concettuale.

Il rappresentare un KOS tramite lo SKOS non serve solo come un meccanismo di pubblicazione, ma permette ad esso anche a partecipare ad una rete di schemi concettuali. Nel Web Semantico, la vera potenzialità dei dati avrà libero sfogo quando esso sarà collegato. Man mano che i concetti di diversi schemi concettuali saranno collegati assieme, essi inizieranno a formare uno schema concettuale globale che sarà distribuito ed eterogeneo. Un Web di schemi concettuali può servire come fondamenta per quelle nuove applicazioni che permettono la navigazione significativa tra i KOS. Nelle prossime righe si presentano quelle caratteristiche dello SKOS che permettono l'interconnessione degli schemi concettuali. Questo spiega come mettere in relazione risorse concettuali ad altre risorse nel Web Semantico.

Ad ogni concetto SKOS è assegnato un URI: questo rende possibile riferirsi senza ambiguità ad un concetto in qualsiasi applicazione che utilizzi lo SKOS. Questo è utile per stabile relazioni semantiche tra concetti pre-esistenti. Questo “mapping” è cruciale per quelle applicazioni, quali ad esempio quelle di information retrieval, che usano diversi KOS contemporaneamente, dove questi KOS hanno scopi sovrapposti e necessitano di essere “accordati” da un punto di vista semantico.

Una caratteristica importante del mapping è il poter dichiarare che due concetti da diversi schemi hanno un significato confrontabile e specificare come questi significati possono essere confrontati, anche se essi vengono da diversi contesti e possono seguire diversi principi di modellazione. I “mapping concettuali” sono attesi come essere un vantaggio chiave del rendere disponibili i KOS nel Web Semantico utilizzando lo SKOS.

Lo SKOS fornisce diverse proprietà che mappano i concetti tra diversi schemi concettuali. Questo può essere fatto asserendo che due concetti hanno un significato simile usando le proprietà `skos:exactMatch` e `skos:closeMatch`. Due concetti da schemi concettuali diversi possono anche essere mappati usando proprietà che si dispongono parallelamente alle relazioni semantiche di `broader`, `narrower` e `related`. Esse sono: `skos:broadMatch`, `skos:narrowMatch` e `skos:relatedMatch`.

Un esempio ben fatto è quello presente nel Primer, dove due schemi concettuali rappresentano diversi punti di vista sugli animali:

```
ex1:referenceAnimalScheme rdf:type skos:ConceptScheme;
    dct:title "Extensive list of animals"@en.
ex1:animal rdf:type skos:Concept;
    skos:prefLabel "animal"@en;
    skos:inScheme ex1:referenceAnimalScheme.
ex1:platypus rdf:type skos:Concept;
    skos:prefLabel "platypus"@en;
    skos:inScheme ex1:referenceAnimalScheme.
```



```

ex2:eggSellerScheme rdf:type skos:ConceptScheme;
  dct:title "Obsessed egg-seller's vocabulary"@en.
ex2:eggLayingAnimals rdf:type skos:Concept;
  skos:prefLabel "animals that lay eggs"@en;
  skos:inScheme ex2:eggSellerScheme.
ex2:animals rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:inScheme ex2:eggSellerScheme.
ex2:eggs rdf:type skos:Concept;
  skos:prefLabel "eggs"@en;
  skos:inScheme ex2:eggSellerScheme.

```

E' possibile mappare i concetti contenuti in `ex1:referenceAnimalScheme` nei concetti contenuti in `ex2:eggSellerScheme` usando la seguente asserzione di mapping:

```

ex1:platypus skos:broadMatch ex2:eggLayingAnimals.
ex1:platypus skos:relatedMatch ex2:eggs.
ex1:animal skos:exactMatch ex2:animals.

```

Una asserzione `skos:closeMatch` indica che due concetti sono sufficientemente simili che possono essere usati in maniera intercambiabile in quelle applicazioni che considerano i due schemi concettuali a cui appartengono. Questa proprietà non è definita come essere transitiva.

Anche un'asserzione `skos:exactMatch` indica similarità semantica. Essa è una sottoproprietà di `skos:closeMatch`. Essa denota un ancor più alto grado di affinità: il collegamento può essere sfruttato attraverso un'ampia gamma di applicazioni e schemi. Questa proprietà è definita come essere transitiva.

Le proprietà `skos:broadMatch` e `skos:narrowMatch` sono usate per dichiarare un mapping di tipo gerarchico tra due concetti.

La proprietà `skos:relatedMatch` è usata per dichiarare un mapping di tipo associativo link tra due concetti.

Oltre alle caratteristiche sopra presentate, lo SKOS propone un numero di elementi del vocabolario e/o delle linee guida che si occupano delle rappresentazioni più sofisticate di cui un utente potrebbe aver bisogno; queste funzioni rendono lo SKOS compatibile con un grande estensione di approcci che modellano il KOS. Questi elementi sono progettati specialmente per venire incontro alle necessità quali sono evidenziate nel documento "SKOS Use Cases and Requirements", dove sono raccolte in un piccolo numero di use cases. Le principali sono:

- raggruppamento di concetti basati su specifici criteri;
- documentazione avanzata per mezzo di risorse complesse;
- stabilire relazioni tra le etichette dei concetti;
- creazione di concetti complessi a partire da quelli più semplici;
- accesso a relazioni di tipo transitivo e gerarchico;
- rappresentazioni di annotazioni per i concetti.

Il documento "Primer" si conclude con una nota generale sull'estendibilità del modello SKOS, tracciando la strada per un preciso e più specializzato raffinamento dei costrutti presentati nel vocabolario.

Nelle prossime righe si presenteranno, in forma descrittiva, altre tecnologie che sviluppano il Web Semantico. Qui inizia la linea di confine tra ciò che la ricerca ha prodotto come standard, e che è stato visto fin qui, e le nuove tecnologie (formati standard, applicazioni, ecc.) che sono allo studio per il raggiungimento degli obiettivi del WS. Si presentano ora altri due “mattoncini” della piramide: si inizia con il RIF, passando poi all’SWRL.

- • - • - • -

Il “RIF”, acronimo di “Rule Interchange Format“ è uno standard in fase di sviluppo all’interno della “W3C Semantic Web Activity”. Quando sarà completato esso diventerà una delle componenti del Web Semantico, principalmente assieme con l’RDF e l’OWL. Sebbene fosse stato immaginato da molti come un “rules layer”, cioè “uno strato di regole” per il WS, in realtà il progetto del RIF si basa sull’osservazione che esistono già molti "rules languages" e quello di cui si ha effettivamente bisogno è lo scambio di queste regole tra i linguaggi stessi.

Il RIF comprende tre dialects, un “Core Dialect” che viene esteso nel “Basic Logic Dialect”(BLD) e nel “Production Rule Dialect” (PRD).

The “RIF Working Group” fu istituito nel 2005. Tra i suoi obiettivi c’era quello di attrarre coloro che creano ed impongono regole nei mercati commerciali. Il working group iniziò con più di 50 membri e due responsabili dell’industria Christian de Sainte Marie della ILOG e Chris Welty dell’IBM.

L’atto di sviluppare un formato di scambio tra sistemi di regole già esistenti fu influenzato da una conferenza nella primavera del 2005 nella quale fu chiaro che un linguaggio di regole non dovrebbe rispondere alle esigenze di tutte le parti interessate. Il Dott. Welty descrisse questi risultati provenienti dalla conferenza come “Nash Equilibrium”.

Regole e sistemi di regole

Una regola è probabilmente una delle più semplici nozioni dell’informatica: essa è un costrutto del tipo IF-THEN. Se qualche condizione (la parte IF) che è controllabile in qualche insieme che la sostiene, allora la conclusione (la parte THEN) è processata. Derivando un po’ dalle loro radici nella logica, i sistemi di regole usano una nozione di predicato che mantiene o no alcuni dati oggetto.

Ad esempio, il fatto che due persone sono sposate potrebbe essere rappresentato con predicati quali MARRIED(LISA,JOHN). MARRIED è un predicato che può essere predetto per mantenere LISA e JOHN. Aggiungendo la nozione di variabili, una regola potrebbe essere qualcosa come:

I
F MARRIED(?x, ?y) THEN LOVES(?x, ?y)

Ci si potrebbe aspettare che per ogni coppia di ?x e ?y (ad esempio LISA e JOHN) per i quali il predicato MARRIED vale (holds), alcuni sistemi che potrebbero capire questa regola vorrebbero concludere che il predicato LOVES vale anche per quella coppia.

Le regole sono un semplice modo di codificare la conoscenza e sono una drastica semplificazione della Logica del Primo Ordine, per la quale è relativamente facile

implementare motori che facciano inferenza e che possono processare le condizioni ed ottenere le giuste conclusioni.

Un sistema di regole è un'implementazione di una particolare sintassi e semantica di regole, la quale può estendere la semplice nozione descritta sopra per includere quantificatori esistenziali, funzioni logiche di disgiunzione, unione, negazione e molte altre caratteristiche. I sistemi di regole sono stati studiati ed implementati sin dalla metà degli anni '70 ed hanno visto una significativa ascesa durante l'apice dei così chiamati "sistemi esperti".

I dialects standard del RIF sono Core, BLD and PRD, che sono legati ad una vasta lista di datatypes con funzioni built-in e predicati su quei datatypes.

- Core: comprende in sottoinsieme comune di molte rule engines;
- BLD: aggiunge caratteristiche al Core dialect che non sono direttamente disponibili, quali funzioni logiche, uguaglianza nella parte THEN, ecc. ;
- PRD: aggiunge la nozione di regola "forward-chaining" (essa inizia con i dati disponibili ed utilizza le regole dell'inferenza per ottenere più dati fino a che lo scopo non sia stato raggiunto.

Lo stato attuale del lavoro di sviluppo del RIF è presentato nella pagina web "Rule Interchange Format Working Group", il cui indirizzo internet è riportato in bibliografia.^[3.31]

In data 1 Ottobre 2009 il RIF è arrivato allo stato di raccomandazione candidata, quindi ad un passo dalla raccomandazione definitiva. Tutte queste sono raccolte in cinque documenti: "RIF Basic Logic Dialect", "RIF Datatypes and Built-Ins 1.0", "RIF Framework for Logic Dialects", "RIF RDF and OWL Compatibility", "RIF Core Dialect", raccolti assieme ad altri documenti di bozza, nella pagina intitolata "RIF Rule Integration Format current status"^[3.32] nel sito della W3C.

- -

Il Semantic Web Rule Language (SWRL) è una proposta (proposal) di un linguaggio basato su regole (rules) per il Web Semantico. Esso è una combinazione dei sottolinguaggi dell'OWL Web Ontology Language (OWL Lite e OWL DL) con quelli del Rule Markup Language (Unary/Binary Datalog Rule ML).

Questa specifica^[3.33], che non è una raccomandazione W3C, è stata presentata nel Maggio del 2004 alla W3C dal Consiglio Nazionale delle Ricerche del Canada, dalla Network Inference e dall'Università di Stanford in associazione con Joint US/EU, uno specifico Comitato per gli Agent Markup Language.

La pubblicazione di questa "submission" da parte della W3C indica, tuttavia, che i suoi contenuti non sono approvati dal Consorzio e che il W3C non ha e non fornirà nessuna risorsa alle questioni in essa contenute. Questo documento non è il risultato di un gruppo della W3C, ma è pubblicato come un input potenziale per gli altri processi della W3C. Nelle prossime righe si presenta l'SWRL nelle sue linee principali.

Le regole proposte nell'SWRL hanno la forma di un'implicazione tra un "antecedent" (body) e un "consequent" (head). Il significato qui inteso può essere letto come: "ogni volta che le condizioni specificate nell'antecedent "hold" (cioè

valgono, sono vere), allora (anche) le condizioni nel consequent “must also hold” (devono valere)”.

Sia l’antecedent che il consequent consistono di zero o più atoms (atomi, cioè dei più piccoli componenti dell’elemento). Un antecedent vuoto è trattato banalmente come vero, cioè è soddisfatto da ogni interpretazione, così anche il consequent deve essere soddisfatto da ogni interpretazione. Un consequent vuoto è considerato banalmente come falso, cioè non è soddisfatto da nessuna interpretazione, così anche l’antecedent non deve essere soddisfatto da nessuna interpretazione.

La sintassi utilizzata nell’SWRL si astrae da qualsiasi sintassi di scambio dell’OWL per consentire un facile uso ed una valutazione di questo linguaggio da parte dell’utente. Questa sintassi estende la sintassi teorica dell’OWL. Tuttavia, questa non è una sintassi formale. Essa viene qui specificata per mezzo di una versione dell’Extended Backus-Naur form (EBNF). [Sintassi minima. I valori terminali sono racchiusi tra virgolette; quelli non terminali sono in grassetto e non virgolettati. Le alternative o sono separate da barre verticali o sono date con diverse rappresentazioni. I componenti che occorrono almeno una volta sono racchiusi da parentesi angolari; quelli che possono apparire zero o più volte sono racchiusi tra parentesi graffe.]

Si fa riferimento ai soliti namespace visti finora. I nomi di questa sintassi sono degli URI reference, come già definiti.

Una ontologia OWL, nella sintassi astratta, contiene una sequenza di assiomi e di fatti. Qui si propone di estenderla con dei “rule axioms” indicati con:

axiom ::= rule

Una rule axiom consiste di un antecedent e di un consequent, ciascuno dei quali consiste di un insieme, anche vuoto, di atoms. A questi rule axioms può essere assegnato un URI reference che serve ad identificare la regola.

Un axiom rule ha una forma del tipo:

```
rule ::= 'Implies(' [ URIreference ] { annotation }
antecedent consequent ') '
antecedent ::= 'Antecedent(' { atom } ') '
consequent ::= 'Consequent(' { atom } ') '
```

Il significato della regola è quello anticipato nelle prime righe. Regole con un antecedent vuoto possono essere usate per fornire fatti senza condizioni.

Gli atom in queste regole possono essere della forma C(x), P(x,y), sameAs(x,y) o differentForm(x,y), dove C è una descrizione OWL, P è una proprietà OWL e x,y sono o variabili, o individui OWL o valori di dati OWL.

Essi sono così composti:

```
atom ::= description '(' i-object ') '
| dataRange '(' d-object ') '
| individualvaluedPropertyID '(' i-object i-object ') '
| datavaluedPropertyID '(' i-object d-object ') '
| sameAs '(' i-object i-object ') '
```

```

    | differentFrom '(' i-object i-object ')'
    | builtIn '(' builtinID { d-object } ')'
builtinID ::= URIreference

```

In maniera informale si può dire:

- un atom $C(x)$ vale se x è un'istanza della class description o un data range C ;
- un atom $P(x,y)$ vale se x è collegato a y tramite la proprietà P ;
- un atom $sameAs(x,y)$ vale se x è interpretato come lo stesso oggetto y ;
- un atom $differentFrom(x,y)$ vale se x e y sono interpretati come oggetti diversi;
- $builtin(r,x,...)$ vale se la relazione built-in r vale sulle interpretazioni degli argomenti.

Gli atomi possono riferirsi ad individui, dati letterali, variabili di individui o variabili di dati. Quindi:

```

i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral

```

Esiste un modo informale di scrivere queste regole, che ha il vantaggio di essere leggibile dall'uomo. Una regola ha la forma:

antecedent \Rightarrow consequent

dove antecedent e consequent sono congiunzioni di atomi scritti $a_1 \wedge \dots \wedge a_n$. Le variabili sono indicate con la convenzione di prefissarle con un punto interrogativo.

Il “model-theoretic semantics” per lo SWRL è una semplice estensione della semantica dell'OWL. L'idea di base è di definire dei “bindings”, cioè delle estensioni delle interpretazioni permesse dall'OWL, che mappano le variabili agli elementi del dominio. Una regola è soddisfatta da una interpretazione se e solo se ogni binding che soddisfa l'antecedent soddisfa anche il consequent. Le condizioni semantiche, che sono legate agli assiomi e alle ontologie, restano immutate. Ad esempio, una interpretazione che soddisfa una ontologia se e solo se essa soddisfa ogni assioma, regole comprese, ed i fatti nell'ontologia.

Non si approfondisce ulteriormente l'argomento che entrerebbe in ambiti troppo teorici. Si presenta un semplice esempio che mostra l'utilizzo di queste regole.

Questo esempio potrebbe asserire che la combinazione delle proprietà `hasParent` e `hasBrother` implica la proprietà `hasUncle` (haGenitore e haFratello implica haZio). La regola può essere scritta in maniera informale, leggibile dall'uomo, come:

`hasParent(?x1,?x2) ^ hasBrother(?x2,?x3) \Rightarrow hasUncle(?x1,?x3)`

Nella sintassi astratta dell'SWRL si scrive:

```

Implies(Antecedent(hasParent(I-variable(x1) I-variable(x2))
                   hasBrother(I-variable(x2) I-variable(x3)))
        Consequent(hasUncle(I-variable(x1) I-variable(x3))))

```

Esiste una sintassi, detta “XML Concrete Syntax”, formata dalla combinazione del OWL Web Ontology Language XML Presentation Syntax con il RuleML XML Syntax, nella quale l’esempio sopra verrebbe codificato con il codice seguente:

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom
      swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom
      swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom
      swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Da qui sarebbe poi semplice passare ad una “RDF Concrete Syntax”, ma la presenza delle variabili nelle regole va al di là della semantica dell’RDF. Una traslazione dall’XML Concrete Syntax all’RDF/XML potrebbe essere portata a termine estendendo le trasformazioni XSLT per l’OWL XML Presentation Syntax.

Non si approfondisce ulteriormente questo linguaggio, visto che esso non è attualmente uno standard: è infatti ancora in fase di studio e di sviluppo. E’ stato qui riportato per far capire al lettore a che livello è arrivata la ricerca in questo ambito. Per un approfondimento, si rimanda il lettore al documento intero il cui collegamento è riportato, come al solito, in bibliografia.

3.5 POWDER

In questo paragrafo si presenta un’altra tecnologia che, pur non essendo inclusa nella piramide tecnologica, raffigurata ad inizio capitolo, è oramai parte integrante del Web Semantico: il POWDER.

Il “Protocol for Web Description Resources”, in sigla POWDER, è un meccanismo, per descrivere e trovare risorse Web: è inoltre di aiuto agli utenti nel decidere se una data risorsa è di interesse oppure no. Il POWDER ha molte possibilità d’uso: dal fornire un metodo migliore per descrivere le risorse Web, alla creazione di marchi di fiducia per aiutare la scoperta di contenuti, dalla protezione dei bambini da contenuti Web inadatti, fino alle ricerche nell’ambito del Web Semantico.

La parte iniziale del lavoro è stata impostata nel Febbraio del 2007. Dal 1 Settembre del 2009 il POWDER è divenuto una W3C recommendation ^[3,34], sviluppata dal

“POWDER Working Group”. La “POWDER Suite” è formata da vari documenti, ognuno con uno scopo diverso:

1. POWDER: Primer - è il testo con le nozioni fondamentali.
2. POWDER: Use Cases and Requirements - contiene le premesse sulle quali è stato creato il POWDER, oltre a vari casi d'uso.
3. POWDER: Description Resources - è la raccomandazione che contiene la definizione e la struttura del Description Resources (DR).
4. POWDER: Grouping of Resources - è un'altra raccomandazione che definisce il metodo con il quale possono essere definiti gli insiemi di risorse alle quali si applicano i DR.
5. POWDER: Formal Semantics - è la parte di raccomandazione che descrive in maniera dettagliata la semantica formale del POWDER e del POWDER-S e del GRDDL transformation process.
6. POWDER: Test Suite - fornisce dati sui quali possono essere provate e/o testate applicazioni POWDER.
7. POWDER: Web Description Resources (WDRS) POWDER-S Vocabulary - contiene la definizione del vocabolario RDF che è usato per esprimere il POWDER-S.
8. XML Schema for POWDER and POWDER-BASE - è il namespace document/schema per il POWDER.

Sono disponibili anche una varietà di strumenti che sono utili all'implementazione del POWDER. I dettagli di questi e le nuove versioni possono essere trovate nella POWDER home page ^[3.35].

Il POWDER specifica un protocollo per pubblicare metadati sulle risorse nel Web, utilizzando l'RDF, l'OWL e l'HTTP. Esso sostituisce la precedente specifica chiamata PICS. La PICS (Platform for Internet Content Selection) è una specifica, sempre creata dalla W3C, che usa i metadati per etichettare le pagine web. Tutto questo al fine di aiutare i genitori e gli insegnanti nel controllo di quei contenuti che i ragazzi possono accedere tramite Internet.

Ci sono due tipi di POWDER: uno complesso, semanticamente ricco chiamato POWDER-S (Semantic POWDER) ed una versione più semplice, chiamata solo POWDER, che è da intendersi come meccanismo primario di trasporto per le Description Resources (descrizione delle risorse). Il POWDER-S può essere generato in maniera automatica a partire dal POWDER.

La versione più semplice ha una “semantica operativa” relativamente semplice che è leggibile dall'uomo ed è scritta in XML.

La versione semanticamente più ricca, conosciuta come POWDER-S, permette al POWDER di sfruttare il Web Semantico ampiamente e di codificare le semantiche formali che consolidano le semantiche operative.

Esiste anche una terza versione provvisoria del POWDER, chiamata POWDER-BASE che è riservata ai processori e che qui non verrà trattata.

Una trasformazione (GRDDL) “Gleaning Resource Descriptions from Dialects of Language” può, in maniera automatica, generare un documento di tipo POWDER-S come codice RDF/OWL da un documento di tipo POWDER. I dettagli di questa trasformazione sono definiti nel documento Formal Semantics.

Dal punto di vista operativo, non ci sono restrizioni su quale forma utilizzare, tuttavia si fa notare che la versione più semplice è da utilizzarsi come meccanismo primario di scambio tra sistemi.

Il POWDER-S è progettato per facilitare l'incorporazione di informazioni di tipo POWDER in sistemi che sono largamente basati nell'RDF; è da notare che tali sistemi necessitano dell'implementazione di una estensione semantica che fa questo.

Una "Description Resource" (DR) è semplicemente un'affermazione: qualcuno ha fatto qualche dichiarazione su una data risorsa o su di un gruppo di risorse. Tuttavia, molti utenti dovrebbero aver fiducia della persona che ha fatto l'affermazione prima di decidere se, o no, fidarsi del contenuto della risorsa.

Se una RD è fatta direttamente da un fornitore di contenuti, che è garantito come avere un certo livello di qualità, allora a quel contenuto può essere data fiducia con più facilità. Comunque, tutto questo non sarà sempre sufficiente. Questo perché una descrizione della risorsa può essere pubblicata da chiunque, dovunque e per descrivere qualsiasi cosa; un utente finale potrebbe, ragionevolmente, voler interrogare l'autore citato per sapere se effettivamente egli ha fatto la dichiarazione e, in caso positivo, quando l'ha fatta e se ora la rifarebbe.

Per alcune situazioni questo non potrebbe ancora essere sufficiente per l'utente finale. Per facilitare l'ulteriore estensione della fiducia, è stato fornito un mezzo per permettere la certificazione delle DR. Una DR che è stata "certificata" guadagna immediatamente in fiducia attraverso la verifica dell'affermazione originale che è fatta da una terza parte ritenuta fidata.

Attraverso la combinazione di questi strumenti, una varietà di questioni possono essere risolte su di un data risorsa web, o un gruppo di risorse, senza dover effettivamente recuperare ed esaminare la risorsa stessa.

Esempi di richieste che potrebbero essere fatte usando il POWDER sono: quali risorse sono descritte dalla DR, quale è la loro descrizione, chi l'ha creata, quando, da quando e fino a quando questa è considerata valida, tutti sono d'accordo con essa ed inoltre, esistono altre descrizioni su quel gruppo di risorse.

Il lettore attento si dovrebbe chiedere perché una persona dovrebbe voler usare il POWDER. Ogni giorno la quantità di dati sul Web aumenta sempre; al contrario il numero di persone che si dedicano alla gestione di queste informazioni sembra diminuire. Per far fronte a queste limitazioni, gli utenti necessitano di un sistema dove un contenuto più "su misura" viene dato loro. L'utente medio "vuole ottenere il dato on-line, trovare esattamente quello che vuole e poter proseguire".

Il POWDER offre dei mezzi affidabili attraverso i quali si possono distribuire le informazioni senza dover caricare l'utente dell'onere di verificare e controllare ogni aspetto. Sia i fornitori di informazioni che i fruitori sono interessati ad ottenere il più alto ritorno per i loro investimenti individuali di tempo e sforzi. Si approfondiscono ora le principali caratteristiche offerte dal POWDER.

Una delle caratteristiche generali più potenti del POWDER, e che è stata una sfida non risolta finora, è il "grouping" (raggruppamento). Esso permette di offrire informazioni semanticamente più ricche su interi gruppi di risorse che sono on-line, precisando il modo in cui il gruppo è definito. Un processore può usare un singolo

documento POWDER per estrarre informazioni su anche tantissime altre risorse. Inoltre, il mantenimento di una tale DR è semplificata, dato che si possono definire molti descrittori in un singolo documento.

I metodi per raggruppare risorse spaziano da una lista di singoli IRI (il POWDER utilizza questi al posto degli URI), attraverso la specifica di cose quali nomi di domini e path, fino all'accoppiamento di IRI su espressioni regolari. Le richieste fatte ad un processore di documenti POWDER sono, tuttavia, sempre manipolate a livello di singola risorsa. Ciò che sarà ottenuto come risultato sono delle triple RDF sulla risorsa, che permettono ad una applicazione di analizzare il dato e decidere come agire caso per caso.

Tradizionalmente, i metadati sono sempre collegati ad una singola risorsa e sono incastrati al suo interno. In questo caso tutta la risorsa deve essere completamente recuperata per determinare se è di interesse oppure no. Il POWDER descrive on-line le risorse che possono essere d'interesse. Si scrive il termine "possono" in quanto sarà il richiedente a decidere a priori se recuperare o meno tutta la risorsa, in base alle informazioni contenute nella descrizione. Questo incrementa altre caratteristiche quali l'efficienza e la precisione nell'information retrieval: in questo modo, inoltre, si riduce il traffico di rete e il carico del server.

Un'altra caratteristica interessante del POWDER è il "profile matching", cioè il recupero di risorse a seconda delle preferenze dell'utente, delle possibilità dei dispositivi e dello stato attuale della risorsa al momento della distribuzione del suo contenuto.

Una caratteristica ulteriore è la "trustmark", che può essere vista come un "marchio di fiducia". Le risorse e le DR che le descrivono possono essere connesse una a ciascun'altra in un web of trust. Partner di questo tipo di web possono essere autorità di certificazione che verificano la veridicità delle affermazioni fatte nelle DR. Il POWDER permette molti modelli possibili con i quali le asserzioni e le affermazioni possono essere fatte, autenticate e riportate all'utente finale.

Il POWDER rende facile creare e pubblicare le "semantic annotation" in maniera indipendente dal loro relativo contenuto. Queste annotation possono coprire grandi quantità di contenuto, così com'è creato, con soli pochi ed occasionali cambiamenti necessari alle annotation.

I motori di ricerca semantici possono ritornare dei risultati che sono filtrati tramite dei parametri che sono basati sulla cultura locale. Le informazioni fornite sul Web possono incidere sull'utente in modi diversi che dipendono dal contesto dal quale sono recuperati. Le semantic annotation permettono anche la disambiguazione dei termini.

Infine, un documento POWDER può essere usato come un set di istruzioni che permettono ad un crawler di recuperare ulteriori informazioni su tutti i collegamenti, puntando alla DR nei vari documenti POWDER, al fine di creare una raccolta di triple circa quelle risorse.

Fin qui sono stati presentati i vantaggi del POWDER e le ragioni perché esso sia un metodo convincente per definire piccole quantità di dati che possono essere applicate a un grande quantità di contenuti.

Si presentano ora degli esempi, relativi agli ambiti fin qui visti, di come “lavora” il POWDER nel mondo reale.

Relativamente ai trustmarks, il POWDER offre vari metodi che permettono di fare ed utilizzare dichiarazioni di fiducia. Ad esempio, un browser web potrebbe fornire un’indicazione se un sito è fidato oppure no, basandosi su altre risorse (siti) che danno queste informazioni sulla fiducia.

Per quanto riguarda l’accessibilità, il POWDER permette ai motori di ricerca e ai portali di fornire dei link personalizzati per quegli utenti che sono conformi ad un determinato tipo di accesso. Inoltre, questi motori e/o portali possono presentare od evidenziare i siti migliori che si basano sulle preferenze dell’utente.

In maniera simile, il POWDER può essere utilizzato per identificare quelle risorse che sono conformi allo standard “mobileOK” della W3C. Un utente che vuole navigare il Web usando il suo telefonino può richiedere al provider di collegarlo solamente a quei siti che sono appropriati per essere visualizzati nel suo display.

Un altro fatto importante riguarda la tutela dei minori che si collegano alla rete e che scaricano materiale. Questa è una priorità per qualsiasi responsabile di siti o per i fornitori di servizi sul web. Questi ultimi dovrebbero includere delle caratteristiche che offrano ai genitori l’abilità di definire il tipo od il livello dei contenuti che vorrebbero fossero permessi vedere ai loro figli. Al momento, solo pochi service provider offrono una tale possibilità.

Una ulteriore applicazione del POWDER riguarda la possibilità di adattare il contenuto, che può essere ottenuto dalla rete quale audio, video, ecc, in base alla banda del collegamento di cui l’utente dispone. Inoltre, se l’utente ha una connessione “privilegiata” in quanto a pagamento, il server deve riconoscere questo e far accedere l’utente a quel contenuto che è stato descritto come “premium”.

Dal punto di vista della semantica, i siti che usano la DR comunicano l’importanza del soggetto del loro contenuto in una maniera che è più accurata e con una più grande rilevanza all’utente che ne fa richiesta. Inoltre, nell’ambito della ricerca, un uso “giudizioso” della DR può portare, a lungo andare, un sito ad essere più apprezzato da molta gente.

Sviluppando un adeguato e dettagliato vocabolario, la DR può essere usata per identificare il valore di un sito come un sito che è “giusto” o come un sito che è “intrinsecamente sbagliato”.

L’utilizzo del POWDER dipende dai risultati che si intendono ottenere:

- se l’obiettivo è quello di fornire dei metadati in una maniera nuova e più efficiente, allora basta creare la DR in XML, cosicchè essa possa essere letta ed utilizzata dagli strumenti più appropriati;
- se l’obiettivo è quello di usare tali metadati semanticamente, allora è necessario convertire il documento XML in dati di tipo RDF.

Per quanto riguarda l’utilizzo vero e proprio del POWDER, il processo è una sequenza e può essere intervallato in vari modi o terminato nel punto appropriato per ottenere il risultato desiderato.

L'utente che vuole descrivere contenuti deve:

1. creare una o più Description Resource all'interno di un documento POWDER e renderlo disponibile nel Web. Deve cioè creare un documento XML che contiene le necessarie componenti di una DR.
2. rendere il documento POWDER scopribile. Questo non è altro che essere sicuri che la DR possa essere acceduta dal pubblico generalmente dal Web.
3. eventualmente aggiungere fiducia alla descrizione. La fiducia è una parte integrante dei documenti POWDER: ci sono vari metodi di certificare i documenti.

L'utente che vuole accedere alle triple RDF, ottenute dai documenti POWDER, deve:

1. utilizzare un POWDER processor per leggere ed elaborare i documenti POWDER.
2. utilizzare uno strumento per fare query sui dati RDF per ottenere la descrizione della risorsa e l'insieme delle risorse che hanno una data proprietà.

Se invece qualcuno volesse fare dell'inferenza basata sui dati contenuti nei documenti POWDER, dovrebbe:

1. convertire il documento POWDER in un documento di tipo POWDER-S, tramite una trasformazione GRDDL.
2. utilizzare un motore che fa inferenza per recuperare le proprietà di una data risorsa. Per far questo si fa notare che un'estensione all'RDF/OWL deve essere stata implementata nel motore al fine di poter supportare ed utilizzare il POWDER-S.

Nelle prossime righe si fa vedere come si può creare una Description Resource. Il primo passo da fare è quello di creare una struttura di documento POWDER in XML e dichiarare il namespace:

```
<?xml version="1.0"?>
<powder
  xmlns="http://www.w3.org/2007/05/powder#"
  xmlns:ex="http://example.org/vocab#">
</powder>
```

Il prossimo passo è quello di dire chi ha creato il documento. Tutti i documenti POWDER hanno esattamente un elemento `attribution` e al suo interno un elemento `issuedby` che punta ai dettagli della persona od organizzazione che ha pubblicato il documento POWDER. Gli autori, per far questo, possono utilizzare o il vocabolario FOAF o il Dublin Core. Supponendo, ad esempio, che il profilo del publisher sia:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <foaf:Organization rdf:ID="me">
    <foaf:name>The POWDER Company</foaf:name>
    <foaf:homepage
  rdf:resource="http://authority.example.org" />
```

```

    </foaf:Organization>
</rdf:RDF>

```

si ottiene il documento POWDER, in cui si può indicare anche la data di creazione:

```

<?xml version="1.0"?>
<powder
  xmlns="http://www.w3.org/2007/05/powder#">
  <attribution>
    <issuedby
      src="http://authority.example.org/company.rdf#me" />
    <issued>2007-12-14T00:00:00</issued>
  </attribution>
</powder>

```

Il passo successivo è quello di aggiungere una Description Resource effettiva, tramite l'elemento `dr`. La DR deve contenere essa stessa almeno un insieme di IRI.

Questo è l'obiettivo (scope) della DR, cioè quello che essa descrive. Nell'esempio lo scope è "everything on example.com". Tutte le DR devono contenere almeno un elemento dall'insieme degli IRI e non possono essere vuote o contenere elementi da altri namespace. L'esempio diventa quindi:

```

<?xml version="1.0"?>
<powder
  xmlns="http://www.w3.org/2007/05/powder#">
  <attribution>
    <issuedby
      src="http://authority.example.org/company.rdf#me" />
    <issued>2007-12-14T00:00:00</issued>
  </attribution>
  <dr>
    <iriset>
      <includehosts>example.com</includehosts>
    </iriset>
  </dr>
</powder>

```

L'elemento chiave finale di una Description Resource è la vera descrizione. Ci sono due modi di fornirla: come RDF (in un "descriptor set") o come uno o più tag (in un "tag set"). Una DR deve contenere almeno uno di loro, ovviamente non vuoto.

L'esempio del DR dichiara che nel considerare l'entità riferita dall'elemento `issuedby`, tutte le risorse all'interno del suo scope sono descritte da tutti elementi descrittivi. Si riporta l'esempio di prima, fornito dalla W3C, dove queste descrizioni sono fornite con un insieme dei descrittori. Questo documento POWDER contiene una singola DR.

```

<?xml version="1.0"?>
<powder xmlns="http://www.w3.org/2007/05/powder#"
  xmlns:ex="http://example.org/vocab#">
  <attribution>
    <issuedby
      src="http://authority.example.org/company.rdf#me" />
    <issued>2007-12-14T00:00:00</issued>

```

```

</attribution>

<dr>

  <iriset>
    <includehosts>example.com</includehosts>
  </iriset>

  <descriptorset>
    <ex:color>red</ex:color>
    <ex:shape>square</ex:shape>
    <displaytext>Everything on example.com is red and
square</displaytext>
    <displayicon
  src="http://authority.example.org/icon.png" />
  </displayicon>
  </descriptorset>

</dr>

</powder>

```

La parte restante della specifica POWDER fornisce i costrutti per pubblicare una DR ed i metodi per collegare una risorsa ad un documento POWDER che la descrive. La specifica illustra poi i metodi per dare fiducia al documento, tramite la proprietà `authenticate`, `certifiedby` e `supportedby`. Il lettore interessato può trovare tutte queste informazioni, corredate da molteplici esempi, negli altri documenti facenti parte della specifica POWDER, compresa la parte relativa alla POWDER-S.

- . - . - . -

Si ritorna ora all'aspetto del Web Semantico relativo a quei futuri obiettivi che esso si pone. Nei prossimi paragrafi si presentano i componenti della piramide tecnologica posti più in alto che rappresentano ciò che gli sviluppatori del WS prospettano per il futuro. Nelle prossime righe non si parlerà più di tecnologie specifiche, ma si parlerà di concetti più generali che potrebbero diventare parte dei nuovi sistemi.

3.6 Logica, dimostrazioni, firme digitali e fiducia

Relativamente a questi ambiti, la documentazione che si trova a riguardo è molto scarsa, dato che questi ambiti nascono dalla visione del Web Semantico di Tim Berners-Lee.

Una panoramica di tutto questo è stata fatta qualche anno fa, presso l'Università de L'Aquila, dalla dott.ssa Antonella Dorati assieme alla prof. Stefania Costantini.^[3.36] Nelle righe sottostanti si riporta la parte più significativa del documento, fondendola con gli esempi tratti da un articolo scritto da Aaron Swartz^[3.37], i cui riferimenti precisi sono riportati in bibliografia.

L'obiettivo futuro che si pongono gli sviluppatori del WS è quello di realizzare dei sistemi in grado di formulare ogni principio logico e permettere alle macchine di ragionare usando questi principi.

Nell'ambito della logica si vuole costruire un linguaggio che utilizza questa logica per riuscire a realizzare le inferenze (conclusioni ottenute tramite procedimenti

deduttivi partendo da alcune premesse) tra i molteplici dati, che possono anche essere estratti in maniera automatica, al fine di dare all'utente le effettive informazioni ricercate;

Swartz riporta nel suo articolo un semplice esempio per far capire come intendere queste cose. Si immagini che un'azienda decide che se qualcuno dei suoi agenti riesce a vendere più di 1000 prodotti otterrà un avanzamento di carriera. Un programma "intelligente" che raccoglie tutti gli ordinativi per ogni agente può seguire questa regola: "Se l'agente X ha superato le 1000 vendite, promuovilo al grado di agente di secondo livello.". Il programma che vede che l'agente Brown ha venduto 1005 prodotti, deduce che Brown è agente di secondo livello.

Una volta che si saranno costruiti questi sistemi "logici", questi potranno essere utilizzati per provare qualcosa. Si vuole cioè che queste informazioni, ottenute come conclusioni della logica, siano valide;

Nell'esempio di Swartz si supponga che dai record di vendita risulti che Brown ha venduto 180 prodotti di tipo A, 390 prodotti di tipo B e 650 prodotti di tipo C. Dalle regole incorporate nel programma si ha che $180+390+650=1220$ ed inoltre i prodotti di tipo A, B e C sono tutti prodotti venduti. Inoltre 1220 è maggiore di 1000. Mettendo insieme tutte queste regole logiche in una prova, il programma dimostra che l'agente Brown è un agente di secondo livello.

Utenti, anche sparsi nel mondo, possono scrivere istruzioni logiche. Le macchine possono seguire questi link "semantici" per costruire dimostrazioni. Mentre da un punto di vista operativo è molto difficile costruire queste dimostrazioni; più facile invece è controllarle. Così facendo, si dovrebbe iniziare a costruire un Web di "processori di informazioni". Nella visione futura, alcuni di questi processori forniranno solamente dati che altri processori potranno utilizzare per costruire regole. I processori "più intelligenti" saranno delle macchine euristiche che seguendo queste regole, ed eventuali istruzioni fornite dall'utente, riusciranno a trarre in maniera autonoma delle conclusioni. Queste saranno pubblicate nel Web come prove o dimostrazioni.

Si definisce, infatti, "procedimento euristico", un metodo di approccio alla soluzione dei problemi che non segue un chiaro percorso, ma che si affida all'intuito e allo stato temporaneo delle circostanze, al fine di generare nuova conoscenza.

Tutti questi risultati avranno bisogno anche di un controllo sull'autenticità di quanto viene affermato. Questo ambito viene garantito tramite la firma digitale. Questa, costruita tramite un sistema di crittografia, attesta l'autenticità delle varie asserzioni presenti sul Web e permette di scoprirne la loro provenienza. In altre parole, l'utente o l'ente che pubblica materiale (documenti, codice, ecc.) nel Web se ne assume la responsabilità. Questa firma può essere allegata direttamente ai documenti web stessi. Chi la incontrerà potrà essere sicuro dell'autenticità del documento. Ogni utente potrà dire al proprio programma a quali firme credere e a quali no. Ogni utente firmerà il suo personale livello di fiducia con il quale poi la macchina a deciderà a cosa e quanto credere.

E' però difficile aver fiducia in un gran numero di persone e questo potrebbe limitare l'utilizzo del Web. E qui fa la sua comparsa quello che viene chiamato "Web of Trust" (Web di Fiducia). Lo scenario che si vuole realizzare è che un utente comunica che ha fiducia in una persona. A sua volta questa persona ha fiducia in altre persone e queste ultime in altre. Tutte queste relazioni di fiducia si aprono come

un ventaglio e formano il Web of Trust. Ognuna di queste relazioni avrà un grado di fiducia, associato con essa. E' da notare che anche la sfiducia è utile come la fiducia.

Si supponga che un programma, durante una ricerca, trovi un documento al quale nessuno ha dato esplicita fiducia, ma che nemmeno ha dato esplicita sfiducia. Il programma darà più fiducia, o credibilità, a questo documento piuttosto che a uno che per qualche ragione è stato completamente sfiduciato.

Nel 1997 Tim Berners-Lee propose un bottone marcato "Oh, yeah?"^[3.38] (Ah, sì?) che quando cliccato porterebbe la macchina a provare a fornire ragioni per dar fiducia a quei dati che sono presenti in quel momento nel browser dell'utente.

L'obiettivo finale di questa visione è quello di un Web che offra riservatezza e che ispiri sempre più fiducia da parte di chi ne usufruisce.

Capitolo 4

Un'applicazione già operativa: GoPubMed

In questo capitolo si presenta una applicazione, già operativa, che sfrutta alcune delle tecnologie del Web Semantico che sono state finora sviluppate.

La ricerca di questo tipo di applicazioni è stata un po' laboriosa in quanto non sempre si riesce a sapere quali tecnologie semantiche sono utilizzate nelle applicazioni che il Web offre, dal momento che queste tecnologie "lavorano dietro le quinte".

Al fine di ottenere delle informazioni aggiornate, nei mesi scorsi, ho provato a contattare vari ricercatori e/o persone che si occupano di Web Semantico. Ho iniziato dal prof. Massimo Marchiori, del Dipartimento di Matematica dell'Università di Padova, dal quale non ho avuto nessuna risposta.

Consigliato dal mio relatore, ho partecipato poi a due conferenze, tenutesi a Padova nei mesi scorsi: la IIR 2010 e la CSB 2010. Lo scopo della mia partecipazione era quello di riuscire a contattare persone e/o enti che potessero darmi delle "dritte" su applicazioni interessanti dove la semanticità venisse effettivamente applicata. Va detto subito che le conferenze non erano specifiche sul Web Semantico, ma quest'ambito veniva considerato in alcuni interventi che ho seguito con interesse. In bibliografia sono riportati i link alle pagine web relative alle conferenze, nelle quali si possono trovare ulteriori link ad ogni specifico intervento dei relatori.

LA IIR 2010 ^[4.1], abbreviativo di "First Italian Information Retrieval", svoltasi a fine Gennaio 2010, aveva lo scopo primario di far incontrare ricercatori e specialisti nel campo dell'Information Retrieval e sulle discipline relative allo scambio di informazioni. Una relazione interessante è stata quella della dott.sa Silvia Calegari del Dipartimento di Informatica, Sistemi e Comunicazioni (DISCo) dell'Università di Milano-Bicocca che ha presentato "GranOnto: un'ontologia granulare per la diversificazione dei risultati delle ricerche". L'ambito si è rilevato, purtroppo, una ricerca molto teorica e distante da quello che interessava per il lavoro di questa tesi.

La CSB 2010 ^[4.2], abbreviativo di "Cultura Senza Barriere", svoltasi a metà Febbraio 2010, presso il Dipartimento di Psicologia dell'Università di Padova, si proponeva di far incontrare persone ed enti che a vario titolo si occupano, come si deduce dal titolo, di espandere la cultura soprattutto nell'ambito della disabilità. Gli argomenti spaziavano dalla pubblicazione in rete di contenuti in maniera efficace, al progetto di una biblioteca virtuale di testi in formato alternativo, all'accessibilità alla rete per le persone con disabilità, agli audiolibri, alla situazione dei siti web dei musei italiani e tantissimi altri, i cui abstract sono riportati nelle pagine del sito web riportato in bibliografia. Due sono stati i seminari che ho ritenuto fossero attinenti allo scopo di questa tesi. Il primo è stato quello di Giampiero Lotito; il secondo quello di Oreste Signore.

Giampiero Lotito, co-fondatore del “Italian Company Facility” ha presentato una descrizione, purtroppo solo verbale, di “Facility”, un motore di ricerca semantico attualmente in realizzazione, che dovrebbe uscire ed essere operativo nei prossimi mesi. L’intervento è stato molto teorico e si è concentrato sull’evidenziazione dei problemi dei motori di ricerca attuali, quali Google. Il Facility dovrebbe, a parer suo, risolvere questi problemi. Alla conferenza non è stata fatta nessuna dimostrazione pratica di come questo applicativo dovrebbe funzionare. L’argomento sarebbe stato interessante da sviluppare in questa tesi. Al termine della conferenza ho parlato con Lotito che non si è dimostrato disponibile a fornirmi ulteriori informazioni. Facendo una ricerca in rete, l’unico riferimento utile a tal scopo è un’articolo ^[4.3] di Edward Henning, della rivista on-line “thinkdigit.com” che ha intervistato Lotito nel Luglio del 2009. Il link all’articolo completo è riportato in bibliografia.

Un incontro molto interessante e proficuo, ma purtroppo troppo breve, è stato quello fatto con il dott. Oreste Signori del CNR di Pisa, attualmente responsabile della W3C-IT. Egli, alla conferenza ha presentato una relazione sulle “Tecnologie del Web Semantico per le digital libraries” che, vista la brevità del tempo si è trasformata in una piccola illustrazione di base di che cos’è il Web Semantico. Più utile è stata la nostra chiaccherata durante la pausa, nella quale egli mi ha indirizzato verso l’applicazione GoPubMed e l’attività di ricerca e sviluppo della W3C “Semantic Web Health Care and Life Sciences”. Nei prossimi paragrafi si presenta questa prima applicazione. Molto materiale relativo all’attività della W3C si trova nelle pagine del sito ufficiale W3C, seguendo i link riportati nella pagina iniziale.

Il “GoPubMed” è un motore di ricerca, basato sulla conoscenza, per i testi biomedici. Utilizza, come tavola dei contenuti, al fine di strutturare i milioni di articoli contenuti nel database MEDLINE, la “Gene Ontology” (GO) e le “Medical Subject Headings” (MeSH). Questo motore di ricerca permette ai biologi e/o ai medici di trovare dei risultati più significativi alle loro ricerche in ambito biomedico, il tutto in maniera più veloce. Nelle prossime pagine si riprenderà la presentazione del GoPubMed. Si presentano ora il MEDLINE, la Gene Ontology e le Medical Subject Headings.

4.1 MEDLINE (PubMed), Gene Ontology, MeSH

La “MEDLINE” ^[4.4] (Medical Literature Analysis and Retrieval System Online) è una base dati bibliografica di scienze della vita e di informazione biomedica. Comprende le informazioni bibliografiche di articoli da riviste accademiche che si estendono nelle branche della medicina, infermieristica, farmacia, odontoiatria, medicina veterinaria e assistenza sanitaria in generale.

MEDLINE comprende anche gran parte della letteratura scientifica prodotta nell’ambito della biologia e della biochimica, nonché da altre discipline non direttamente collegate alla medicina, quali l’evoluzione molecolare.

MEDLINE è redatto dalla National Library of Medicine (NLM) degli Stati Uniti, ed è liberamente disponibile su Internet e ricercabile attraverso l’interfaccia “PubMed”, il sistema della National Center for Biotechnology Information, della NLM, “Entrez” ed altri motori di ricerca.

Questo database contiene oltre 18 milioni di record da circa 5200 pubblicazioni selezionate in 37 lingue, che si estendono dalla biomedicina e alla sanità, dal 1950 ad oggi. Originariamente gli articoli nel database partivano dal 1965, ma questo è stato migliorato, ed ora tutto quello che è stato prodotto dal lontano 1950 è disponibile all'interno dell'indice principale.

MEDLINE utilizza il “Medical Subject Headings” (MeSH) per il recupero delle informazioni. I motori progettati per la ricerca su MEDLINE (come Entrez e PubMed) generalmente utilizzano una espressione booleana combinazione di termini MeSH, di parole astratte, del titolo dell'articolo, del nome dell'autore, della data di pubblicazione, ecc.

Entrez e PubMed possono anche rintracciare articoli simili ad uno dato, basandosi su di un sistema di valutazione matematico che tiene conto della somiglianza del contenuto dell'abstract e dei titoli di due articoli.

MEDLINE funziona come una risorsa importante per i ricercatori biomedici ed i professionisti che scrivono nelle riviste di tutto il mondo. Insieme con la Cochrane Library ed una serie di altre banche dati, MEDLINE facilita la medicina “evidence-based” (che si basa sull'evidenza, sulle dimostrazioni). MEDLINE influenza inoltre i ricercatori nella scelta delle riviste in cui pubblicare: pochi ricercatori biomedici oggi considerano la pubblicazione in un giornale che non sia indicizzato da MEDLINE, questo perchè gli altri ricercatori non potrebbero trovare o citare il loro lavoro.

Circa 5.000 riviste biomediche sono indicizzate in MEDLINE. I nuovi giornali non sono inclusi automaticamente o immediatamente. La selezione si basa sulle raccomandazioni di un comitato, il “Literature Selection Technical Review Comiteee”, sulla base dell'ambito scientifico e della qualità della rivista. Il “Journals Database” (uno dei database Entrez) contiene informazioni, come ad esempio l'abbreviazione del nome l'editore, su tutte le riviste comprese in Entrez, compreso PubMed.

La ricerca sul MEDLINE è effettivamente una competenza acquisita; gli utenti inesperti a volte sono frustrati con il gran numero di oggetti restituiti da semplici ricerche. Al contrario di quello che si può credere, una ricerca che restituisce migliaia di articoli non è garantita essere esaustiva.

A differenza di un tipico motore di ricerca internet usando, la ricerca su MEDLINE tramite il PubMed richiede l'investimento di poco tempo. L'utilizzo del database MeSH per definire l'oggetto di interesse è uno delle modi più utili per migliorare la qualità di una ricerca. L'uso di termini MeSH uniti con ambiti (come la data di pubblicazione o il tipo di pubblicazione) o qualificatori (quali gli effetti negativi o di prevenzione e controllo), fa sì che la ricerca del testo e/o di una parola sia un'altra. Trovare un articolo su un argomento e cliccando poi sul link “Related Articles” (Articoli correlati) si ottiene una collezione di articoli classificati in maniera simile che rendono possibile espandere una ricerca che aveva fornito pochi risultati.

L'accesso on line può essere fatto tramite i seguenti motori di ricerca:

- PubMed
- MEDSUM- Un'alternativa a PubMed che restituisce grafici / tabelle di dati di riepilogo.

- Authoratory- Esplora informazioni di contatto, interessi professionali, connessioni sociali.
- GoPubMed - Esplora PubMed / MEDLINE con la Gene Ontology
- HubMed - Una interfaccia alternativa al database PubMed.

Una caratteristica importante di MEDLINE è che i record sono indicizzati tramite il vocabolario Medical Subject Headings (MeSH).

Una panoramica completa del database MEDLINE è fatta nella pagina dell'Istituto Nazionale della Salute degli Stati Uniti (National Institutes of Health, NIH)-Biblioteca Nazionale di Medicina (National Library of Medicine, NLM). Qui sono presenti spiegazioni e i link a tutti gli altri componenti, quali motori di ricerca, manuali, ecc.

- • -

“Entrez PubMed”o, più semplicemente “PubMed” ^[4.5] è un database bibliografico on-line, accessibile liberamente, di citazioni ed abstracts di giornali e pubblicazioni biomediche. E' prodotto dal National Center for Biotechnology Information (NCBI) presso la National Library of Medicine (NLM) del National Institutes of Health (NIH) degli Stati Uniti. Questa banca dati è integrata in Entrez, il più ampio sistema di informazioni biologiche, chimiche e mediche dell'NCBI.

PubMed, con oltre 18 milioni di riferimenti bibliografici, derivati da circa 5.400 riviste biomediche consente l'accesso al MEDLINE.

In aggiunta alle citazioni di MEDLINE, il PubMed contiene anche:

- Citazioni “in-process” che forniscono un record per un articolo prima che esso sia indicizzato con il MeSH de aggiunto al MEDLINE o convertito in uno stato di “fuori dal campo di applicazione”.
- che precedono la data in cui un giornale che è stato selezionato per l'indicizzazione in MEDLINE.
- Alcune citazione dell'OLDMEDLINE che non sono ancora state aggiornate con il vocabolario attuale e convertite allo stato di MEDLINE.
- Citazioni ad articoli che sono out-of-scope (fuori dal campo di applicazione) da certe riviste MEDLINE, principalmente general scienze generali e riviste di chimica generale, per le quali gli articoli di scienze naturali sono indicizzati con il MeSH per il MEDLINE.

- • - • - • -

La “Gene Ontology”(GO) ^[4.6] è una grande iniziativa bioinformatica che ha lo scopo di unificare la rappresentazione dei geni e degli attributi dei prodotti genici di tutte le specie e tra tutti i database esistenti. Tre sono gli obiettivi del “Gene Ontology project”: il primo è quello di conservare e sviluppare ulteriormente il proprio vocabolario controllato di geni ed attributi dei prodotti genici; il secondo è quello di aggiungere note esplicative sui geni ed i prodotti dei geni, rendendoli simili e completando questi dati con delle annotazioni (annotazioni, commenti, significati); il terzo è quello di fornire degli strumenti che facilitino l'accesso a tutti gli aspetti dei dati forniti dal Gene Ontology project.

Il Gene Ontology project è supportato dal “GO Consortium” che è formato da un gruppo di ricercatori attivamente impegnati nella gestione di un gran numero di database contenenti modelli di organismi e proteine di varie specie. Queste persone si occupano, inoltre, dello sviluppo dei software e di tutti quegli strumenti utili alla Gene Ontology. Il link alla pagina web del consorzio è riportato in bibliografia.

La GO è una parte di un tentativo di classificazione più vasto, l’ “Open Biomedical Ontologies” (OBO). La GO fu costruita originariamente costruito nel 1998 da un gruppo di ricercatori che studiavano il genoma dei tre modelli di organismo: *Drosophila melanogaster*, *Mus musculus* e *Saccharomyces cerevisiae*. Molti altri database di modelli di organismo sono stati messi insieme dal consorzio, contribuendo non solo all’annotazione dei dati, ma anche allo sviluppo di ontologie ed altri strumenti per la visualizzazione e/o l’applicazione di tali dati.

Il Gene Ontology project fornisce un’ontologia di termini definiti che rappresentano le proprietà dei prodotti genici. L’ontologia comprende tre domini:

- cellular component: la parte di una cellula o il suo ambiente extracellulare;
- molecular function: l’attività elementare di un prodotto genico a livello molecolare;
- biological process: le operazioni o gli insiemi di eventi a livello molecolare, con un ben preciso inizio e una fine, che sono attinenti al funzionamento di intere unità viventi quali tessuti, organi ed organismi.

Ciascun termine all’interno della Gene Ontology, detto “GO term”, ha un nome che può essere una parola, una stringa di parole, un identificatore alfanumerico univoco, una definizione con la fonte citata ed anche un namespace che indica il dominio al quale il termine appartiene. I termini possono avere anche sinonimi, che sono classificati come essere esattamente equivalenti al nome del termine o che siano una versione allargata o rimpicciolita o che abbiano una qualche attinenza.

Qui sotto si presenta un esempio di un termine della Gene Ontology (GO Term).

```

id:          GO:0000016
name:        lactase activity
namespace:   molecular_function
def:         "Catalysis of the reaction: lactose + H2O = D-
             glucose + D-galactose." [EC:3.2.1.108]
synonym:     "lactase-phlorizin hydrolase activity" BROAD
             [EC:3.2.1.108]
synonym:     "lactose galactohydrolase activity" EXACT
             [EC:3.2.1.108]
xref:        EC:3.2.1.108
xref:        MetaCyc:LACTASE-RXN
xref:        Reactome:20536
is_a:        GO:0004553 ! hydrolase activity,
             hydrolyzing O-glycosyl compounds

```

Il vocabolario GO è stato progettato per essere “neutro” rispetto alle varie specie ed include termini applicabili ai prokaryotes (procariote=microrganismo unicellulare sprovvisto di membrana nucleare o organelli cellulari) e agli eukaryotes (eucariote=organismo composto di cellule con membrana e contenenti peduncoli ciliati e nuclei con cromosomi) e agli organismi sia singoli che multicellulari.

L'ontologia GO è strutturata come un grafo che è diretto ed aciclico. Ciascun termine ha relazioni definite con uno o più altri termini nello stesso dominio o, talvolta, in domini diversi.

Questa ontologia non è statica. Aggiunte, correzioni e modifiche ad essa sono suggerite e raccomandate sia dai membri delle comunità di ricerca, sia da coloro che fanno le annotation, come pure da chi è direttamente coinvolto nel GO project. Le modifiche suggerite sono poi riviste dagli editori dell'ontologia e, se queste modifiche si rivelano appropriate, vengono implementate.

La “genome annotation” è la pratica di catturare dati su un prodotto genico: per far questo la GO annotation usa i termini dell'ontologia GO. I membri del Consorzio presentano i loro commenti, per l'integrazione e la diffusione, nel sito della GO, dove questi possono essere visualizzati o scaricati direttamente. In aggiunta agli identificatori dei prodotti genici e ai termini rilevanti, le GO annotation contengono i seguenti dati:

- il “reference” (riferimento) usato per fare l'annotazione (ad esempio, articoli di giornale, ecc.);
- un “evidence code” che indica il tipo di prova/dimostrazione su cui si basa l'annotation;
- la data e l'autore dell'annotation.

Si riporta qui sotto un esempio di GO annotation.

```
Gene product:  Actin, alpha cardiac muscle 1,
                UniProtKB:P68032
GO term:       heart contraction ; GO:0060047 (biological
                process)
Evidence code: Inferred from Mutant Phenotype (IMP)
Reference:    PMID:17611253
Assigned by:  UniProtKB, June 06, 2008
```

I file che contengono l'ontologia GO sono liberamente scaricabili dal sito internet della GO in vari formati, tra cui un RDF/XML, oppure possono essere acceduti direttamente on-line usando un browser specifico per la GO chiamato AmiGO. Si può accedere direttamente a questo browser tramite il link nella home page della GO. Si segnala l'esistenza di “OBO-Edit”, un software open source ed indipendente dalla piattaforma, che permette di modificare le ontologie sviluppate. E' implementato in Java ed utilizza un approccio graph-oriented per visualizzare e modificare le ontologie. Anch'esso può essere liberamente scaricato seguendo i link nella pagina internet della Gene Ontology.

Il GO project fornisce anche mappature dei suoi termini ad altri sistemi di classificazione.

Nel Gennaio del 2008 la Gene Ontology conteneva oltre 24.500 termini applicabili ad una grande varietà di organismi biologici.

Per un approfondimento di questo argomento si consiglia di visitare la pagina web riportata in bibliografia, ricca di moltissime informazioni preziose.

- -

Il “Medical Subject Headings” (MeSH) ^[4.7] è un enorme vocabolario controllato (o sistema di metadati) ideato con l'obiettivo di indicizzare la letteratura scientifica in ambito biomedico. Il thesaurus è stato creato dalla “National Library of Medicine” (NLM) degli Stati Uniti, che continua a gestirlo. Il MeSH viene adoperato per l'indicizzazione degli articoli delle oltre 5000 riviste mediche presenti nel database bibliografico MEDLINE /PubMed e nel catalogo dei libri della NLM.

Il vocabolario può essere consultato e scaricato gratuitamente da tutti gli utenti di internet attraverso il PubMed. L'edizione annuale stampata su carta è stata interrotta nel 2007: ora il MeSH è disponibile solo on-line.

Nei database MEDLINE o PubMed il contenuto di ciascun articolo è indicizzato con 10-15 descrittori, di cui solo uno o due termini indicano l'argomento principale: questi sono detti “major” (principali) e identificati con l'apposizione di un asterisco. L'utilità di queste operazioni può essere verificata nelle operazioni di ricerca in quanto il ricorso al vocabolario controllato MeSH e ai descrittori permette di essere molto selettivi e di ridurre enormemente il “rumore” che si otterrebbe se si utilizzassero solo le parole libere del linguaggio comune

La versione del 2009 del MeSH contiene 25.186 “subject headings”, conosciuti anche come “descriptors” (descrittori). Molti di questi sono accompagnati da una breve descrizione o definizione, collegata ai relativi descrittori ed a una lista di sinonimi o termini molto simili, più noti come “entry term”. A causa di queste liste di sinonimi, il MeSH può essere visto come un thesauro.

I descrittori o i subject headings sono organizzati gerarchicamente. Un dato descrittore può apparire in diversi posti dell'albero gerarchico. Una locazione nell'albero ha con sé una etichetta sistematica conosciuta come “tree number” e, di conseguenza, un descrittore può avere vari tree number.

I termini del vocabolario sono organizzati, dal più generale al più specifico, in sedici categorie identificate da lettere dell'alfabeto (“A” per l'anatomia, “B” per gli organismi, “C” per le malattie, ecc.), ciascuna delle quali a sua volta è suddivisa in sottocategorie, identificate da numeri, sempre più specifiche a mano a mano che si procede in basso: la specificità del termine è proporzionale alla lunghezza del numero. Grazie alla struttura ramificata che scaturisce da questa classificazione, si ottengono degli elenchi di termini definiti “alberi”.

Ad esempio, nel caso della categoria “C” (Malattie), avremo ventitré sottocategorie (“C01”: Infezioni batteriche e micotiche, “C02”: Malattie virali, “C03”: Malattie parassitarie, “C04” Neoplasie, “C05” Malattie del sistema muscolo scheletrico, “C06” malattie dell'apparato digerente, ecc.) ciascuna delle quali ne contiene numerose altre.

Come detto prima, un dato descrittore può apparire in più punti dell'albero gerarchico, e quindi può essere descritto da più codici alfanumerici.

Nel caso, per esempio, delle “**Neoplasie dello stomaco**”, patologia neoplasica (“**C04**”) e dell'apparato digerente (“**C06**”) potremo ottenere i seguenti alberi:

- Neoplasie [C04]
 - Neoplasie per sede [C04.588]
 - Neoplasie dell'apparato digerente [C04.588.274]
 - Neoplasie Gastrointestinali [C04.588.274.476]
 - **Neoplasie dello stomaco**
[**C04.588.274.476.767**]

oppure

- Malattie dell'apparato digerente [C06]
 - Neoplasie dell'apparato digerente [C06.301]
 - Neoplasie Gastrointestinali [C06.301.371]
 - **Neoplasie dello stomaco** [C06.301.371.767]

I tree numbers di un dato descrittore sono soggetti a cambiamenti man mano che il MeSH è aggiornato. Ogni descrittore ha comunque un identificatore alfanumerico univoco che non viene mai cambiato.

Le catorie del livello principale nella gerarchia dei descrittori MeSh, nella versione inglese, sono:

- Anatomy [A]
- Organisms [B]
- Diseases [C]
- Chemicals and Drugs [D]
- Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
- Psychiatry and Psychology [F]
- Biological Sciences [G]
- Physical Sciences [H]
- Anthropology, Education, Sociology and Social Phenomena [I]
- Technology and Food and Beverages [J]
- Humanities [K]
- Information Science [L]
- Persons [M]
- Health Care [N]
- Publication Characteristics [V]
- Geographic Locations [Z]

La gerarchia completa può essere trovata nelle pagine internet relative al MeSH.

Originariamente il MeSH era solo in inglese: ora è disponibile in molti altri lingue. La traduzione dei MeSH dalla lingua inglese a quella italiana è curata dall'Istituto Superiore di Sanità. La traduzione è stata avviata nel 1986 nell'ambito dell'Unified Medical Language System, un progetto della NLM che aveva lo scopo di creare un linguaggio medico utilizzabile dai sistemi informatici. Il MeSH permette anche il recupero di documenti da lingue diverse.

Ulteriori informazioni sui MeSH possono essere trovate in lingua inglese nel sito della National Library of Medicine, il cui link è riportato in bibliografia, o in lingua italiana nel sito dell'Istituto Superiore di Sanità.^[4.8]

4.2 GoPubMed



Figura 4.1 – Logo di GoPubMed

Il “GoPubMed” ^[4.9] è un motore di ricerca, basato sulla conoscenza, per i testi biomedici. Utilizza, come tavola dei contenuti, al fine di strutturare i milioni di articoli contenuti nel database MEDLINE, la “Gene Ontology” (GO) e le “Medical Subject Headings” (MeSH).

Il GoPubMed è stato riconosciuto con il “2009 red dot”: riconoscimento del migliore tra i migliori nella categoria delle interfacce grafiche utente per la comunicazione e gli strumenti interattivi. E’ uno dei primi motori di ricerca Web 2.0.

Il sistema è stato sviluppato alla “Technical University” di Dresden (Dresda-Germania) da Michael Schroeder ed il suo team e alla Transinsight ^[4.10] (Transinsight GmbH, Tatzberg 47-51, D-01307 Dresden, Germany).

La Transinsight sviluppa due prodotti:

- il GoPubMed PRO: un’aggiunta di potenza semantica al PubMed ed una sua versione più limitata che si chiama “GoPubMed”.
- GoGene: una nuova piattaforma sviluppata per la spiegazione dei geni, delle proteine e delle interazioni con le malattie.

Alcuni studi recenti, hanno valutato che un professionista della “life sciences” impiega una media di 12.4 ore in una settimana per ricercare informazioni , non solo su Internet, ma anche negli archivi di altre aziende/società, nei loro computer e tramite le email. Le tecnologie di ricerca semantica del GoPubMed PRO, affermano i loro realizzatori, possono significativamente ridurre il numero di ore spese da 12.4 a solo 1. Tutto questo è possibile tramite una classificazione basata sulla conoscenza (knowledge-based classification), che è la piattaforma del successo di GoPubMed. In aggiunta, il GoPubMed PRO include una “Ontology Generation” ed una piattaforma per l’editing che accelera la modellazione di nuovi domini.

Si fa notare che il motto del GoPubMed è: “Searching is now sorted - Save up to 90% of your time” (La ricerca è ordinata - Risparmia fino al 90% del tuo tempo).

Essendo il GoPubMed un motore di ricerca implementato anche per essere venduto, nei siti internet che ne parlano, a cominciare dal quello della Transinsight, non sono spiegate quali tecnologie specifiche, viste nei capitoli precedenti, siano state utilizzare. Alla base di tutto ci sono la “Gene Ontology” (GO) e le “Medical Subject Headings” (MeSH), già presentate nelle righe precedenti.

In una tabella si possono presentare le caratteristiche principali che contraddistinguono le due versioni. Il “si” indica che la caratteristica fa parte del pacchetto, mentre il “no” indica che essa non è presente.

Caratteristiche	GoPubMed	GoPubMed PRO
	gratuito	a pagamento
Intelligent semantic search	si	si
Semantic navigation	si	si
Advanced semantic statistics	si	si
Advanced search features	si	si
Integrated semantic community platform	si	si
Search your own sources	no	si
Use and edit your own ontologies	no	si
Semi automated ontology generation support	no	si
Multilingual semantic support	no	si
Integrated folksonomy	no	si
Improved text mining	no	si
Discover Information	no	si

Si descrivono, in breve, le particolarità di ognuna di esse:

- **Intelligent semantic search:** questa è una nuova caratteristica delle ricerche, che possono essere fatte anche senza l'utilizzo delle parole chiave. Utilizzando la conoscenza nella forma di semantic networks (delle ontologie), il GoPubMed dà origine a due vantaggi. Il primo è che la ricerca viene sicuramente portata a termine (completeness); il secondo vantaggio è dato dalla possibilità di un filtraggio molto veloce dei risultati.
- **Semantic navigation:** è possibile interrogare il sistema anche senza digitare nessun tasto. Solo cliccando su uno o più termini dell'ontologia, la piattaforma espanderà la query usando dei sinonimi e dei "children" (discendenti) del concetto. Questa è la vera ricerca semantica che non si ancora vista in altri sistemi che sono attualmente disponibili sul mercato.
- **Advanced semantic statistics:** questa caratteristica offre un modo alternativo di avere una panoramica della letteratura biomedica. Una "bibliometric analysis platform" fornisce vari dati statistici tenendo conto di tutti i dati quali autore, rivista/giornale, collocazione, ente/istituzione, lista della rete di autori, informazioni geografiche, così come lo sviluppo di un argomento nel corso del tempo. Gli utenti ottengono così informazioni complete che permettono loro di acquisire rapidamente nuove conoscenze.
- **Advanced search features:** il sistema permette, inoltre, la progettazione di query avanzate. La chiave è che tutte le categorie possibili, come gli autori, affiliazioni, ecc. si basano su conoscenze di base. Una finestra di semantica, a completamento automatico, mostra i concetti più probabili per un certo contesto. Questa caratteristica, semplice ma potente, accelera la struttura della query, oltre a fornire nuovi spunti per l'utente,
- **Integrated semantic community platform:** questa piattaforma collega tra loro professionisti che condividono interessi od attività di ricerca simili. Il sistema integrato di mash-up semantico identifica la collaborazione delle persone e dei gruppi di persone su Internet o all'interno delle organizzazioni/aziende.

La versione PRO, aggiunge delle caratteristiche più sofisticate al semplice GoPubMed:

- **Search your own sources:** permette la ricerca su siti internet selezionati, sul proprio desktop, nella rete intranet aziendale, su altre basi di dati e su Microsoft SharePoint. GoPubMed PRO funziona con la maggior parte formati di file e delle API per database. L'evidenziazione interna è disponibile per molti formati PDF in modo che i significati possono essere identificati a colpo d'occhio. Ricerche semantiche possono essere condotte tramite "concetto + sinonimi + discendenti". GoPubMed PRO garantisce pienamente la completezza e la trasparenza dei risultati della ricerca attraverso il collegamento dei meta dati con le ontologie ed i contenuti.
- **Use and edit your own ontologies:** è possibile creare e personalizzare le proprie ontologie, cioè il proprio "tavolo di contenuti". Il GoPubMed PRO facilita l'inclusione di altre ontologie con informazioni più dettagliate, per esempio, sulle malattie. Il sistema è stato sviluppato in stretta collaborazione con i clienti che già utilizzavano versioni precedenti ed è stato progettato con l'obiettivo di mantenerlo il più semplice possibile, ma non semplicistico. Il software è progettato per gli esperti del dominio biomedico con poca o nessuna conoscenza del progetto di ontologie e/o di conoscenze d'ingegneria. Il sistema supporta i formati OWL e OBO.
- **Semi automated ontology generation support:** la piattaforma "Ontology Generation" offre un approccio rapido per la creazione delle proprie ontologie. Questo strumento automatizzato di generazione dei termini permette l'interrogazione di GoPubMed e l'estrazione e la messa in graduatoria dei termini, così come le loro abbreviazioni, il salvataggio e caricamento dei termini dagli appunti, l'utilizzo di termini di filtraggio con le espressioni regolari e l'aggiunta di termini per l'ontologia.
- **Multilingual semantic support:** il software può gestire ontologie rappresentate in più lingue. Il "text mining" (estrazione dei testi) può trovare testi in altre lingue con le qualità semantiche. Ciò significa che la navigazione e la ricerca possono essere effettuate nella lingua madre degli utenti, ma anche i testi in lingua non nativa possono essere recuperati. Il vantaggio di questo è la piccola quantità di testi di ritorno.
- **Integrated folksonomy:** tramite il GoPubMed PRO gli utenti possono categorizzare facilmente dati estratti dai testi. Uno avanzato strumento "redattore semantico" è integrato nella piattaforma per migliorare rapidamente i dati estratti dai testi. Questa caratteristica consente anche al sistema di apprendere nel corso del tempo e porta la qualità del text mining a raggiungere spesso il 95-100% di accuratezza.
- **Improved text mining:** GoPubMed PRO integra i più recenti algoritmi che sono stati applicati in partnership con BioCreative. Inoltre, la nuova qualità può essere valutata in GoGene, la nuova piattaforma sviluppata sempre dalla Transinsight per la chiarificazione dei geni, delle proteine e le interazioni con le malattie.
- **Discover Information:** la funzione di Discovery permette agli utenti di trovare concetti collegando due diversi concetti che non sono connessi. Questa nuova funzionalità, che è resa possibile attraverso la semantica, permette di scoprire la vera conoscenza.

L'utilizzo del GoPubMed necessita di un collegamento alla rete Internet di media velocità e di uno qualsiasi dei browser disponibili: Internet Explorer, Google Chrome, ecc. anche in versioni non recentissime.

L'utilizzo del GoPubMed PRO necessita dei seguenti requisiti di sistema:

- Database Server:
 - 2 CPU cores at 2 GHz
 - 16 GB RAM
 - l'accesso ad almeno 300 GB di HDD veloce per il database
 - MySql 5.x database
- Application Server:
 - 4 CPU cores at 2,5 GHz
 - 16 GB RAM
 - 50 GB HDD per l'applicazione
 - Tomcat 5.5.x Webserver, Java Runtime 1.5.

- • -

In queste righe si presenta l'interfaccia grafica che è utilizzata dall'utente per inserire le proprie query ed ottenere i risultati, assieme ad un esempio che è stato estratto dalla documentazione a corredo di GoPubMed.

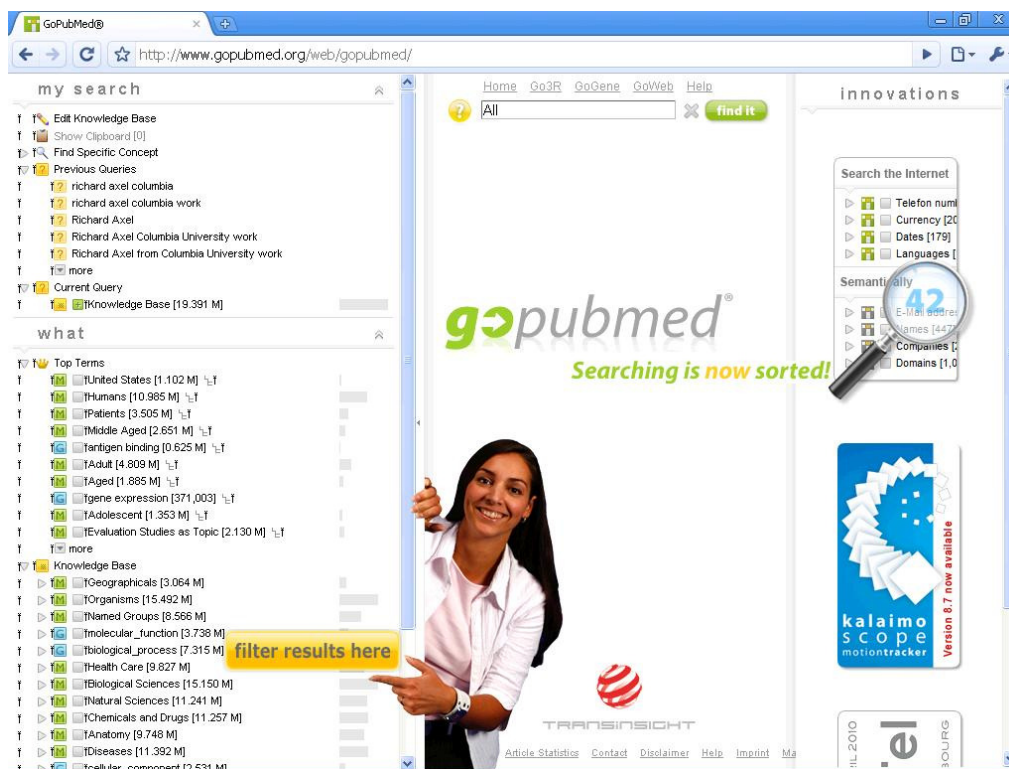


Figura 4.2 – Home page di GoPubMed

Si è detto che il GoPubMed permette il ritrovamento più veloce e significativo delle informazioni necessarie attraverso l'uso di conoscenze di base in campo biomedico. Il GoPubMed non assegna un grado di importanza a queste informazioni, ma lascia il compito di ciò all'utente nella maniera che sarà presentata nelle prossime righe.

Il GoPubMed recupera gli abstract di PubMed per la query di ricerca e ordina le informazioni rilevanti per le 4 categorie di livello superiore: "What", "Who", "Where" e "When" mostrando i concetti, le persone/istituzioni, i luoghi/giornali e le date che hanno una rilevanza per la ricerca dell'utente.

Il numero tra parentesi quadra indica il numero di citazioni che fanno parte di quella categoria. La striscia grigia dà una percentuale "visiva" del numero di citazioni di ogni sottocategoria rispetto alle altre. La figura 4.3 illustra tutto quanto è stato detto finora.

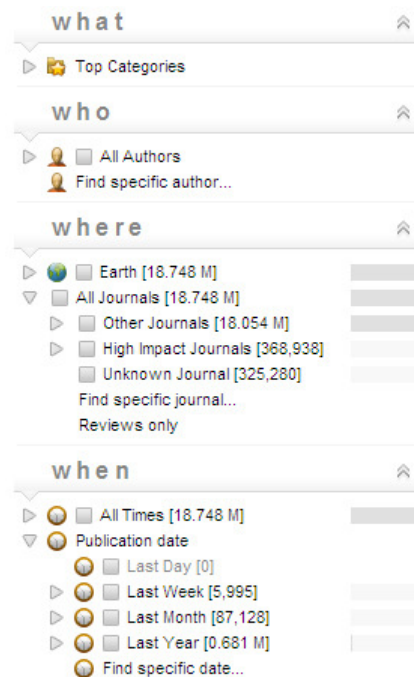





Figura 4.3 – Categorie What, Who, Where e When

Si presentano le varie **categorie principali**.

La categoria "What"

Il GoPubMed recupera le citazioni in accordo con la query dell'utente. Le citazioni sono linkate (presenti) nella categoria "What". Durante il recupero GoPubMed recupera gli abstract rilevanti in PubMed che contengono i concetti biomedici che sono relativi alla ricerca. Questi concetti biomedici sono raggruppati in "Top categories" (categorie principali). I documenti sono annotati con le "Top Categories".

I concetti che appartengono alla Gene Ontology sono indicati con il simbolo , quelli appartenenti alla MeSh sono indicati con il simbolo , mentre quelli che appartengono all' Universal Protein Resource sono indicati con il simbolo .

Il GO descrive i geni e i prodotti genici nei diversi organismi. Ci sono tre aree di base: la funzione molecolare dei prodotti genici, il loro ruolo nei processi biologici multi-step e la loro localizzazione nei componenti cellulari.

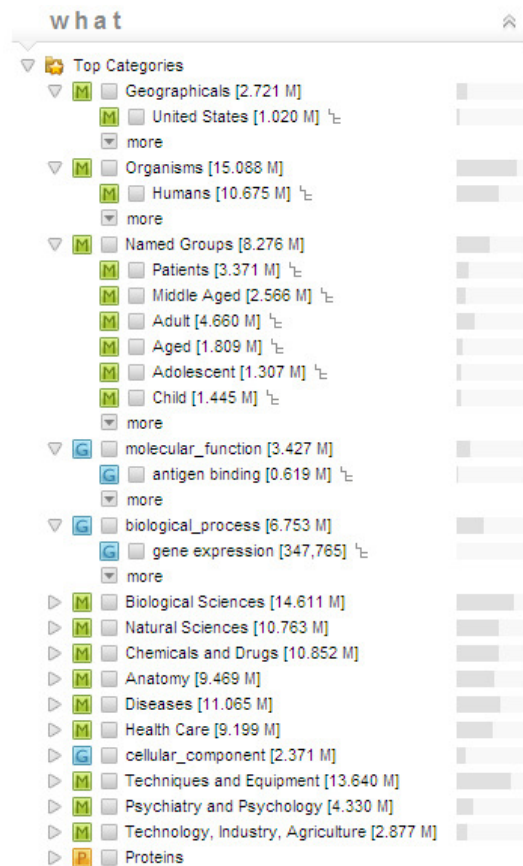


Figura 4.4 – Categoria What

La gerarchia MeSH descrive: Anatomy, Biological Sciences, Chemical and Drugs, Diseases, Health Care, Natural Sciences, Organisms, Psychiatry and Psychology, Techniques and Equipment, Named Groups, Technology, Industry e Agricult.

Altri headings (intestazioni) più specifici si trovano ad un livello più stretto.

Spostando il cursore sopra un concetto, si può vederne una breve descrizione, i suoi sinonimi e la sua posizione nella struttura ad albero.

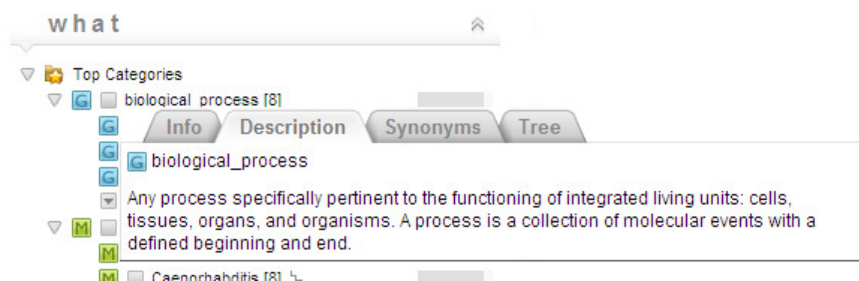


Figura 4.5 – Description di un concept

Ordinando gli abstract secondo le gerarchie di concetto di GO e MeSH si permette la ricerca combinata in biologia molecolare e in medicina. La categoria “What” aiuta ad esplorare sistematicamente i risultati. L’albero sulla sinistra serve come indice al fine di strutturare i milioni di articoli della base di dati PubMed/MEDLINE. Navigando

l'albero si possono ridurre i risultati della ricerca da migliaia a pochi elementi, il tutto in pochi secondi.

La categoria “Who”

Questa categoria aiuta l'utente a trovare autorevoli studiosi/specialisti ed i centri nell'area biomedica. Dopo aver effettuato una ricerca di tutti gli autori degli abstract rilevanti sono elencati sotto la categoria “Who”.



Figura 4.6 – Categoria Who

Spostando il cursore sul nome degli autori e cliccando su “Profile”, il suo profilo sarà mostrato. Nuove tecniche sofisticate per la disambiguazione della persona sono state utilizzate per classificare esattamente i nomi più di 3 milioni di autori che sono presenti nella banca dati PubMed. Ad esempio, il prof. dott. Marino Zerial.

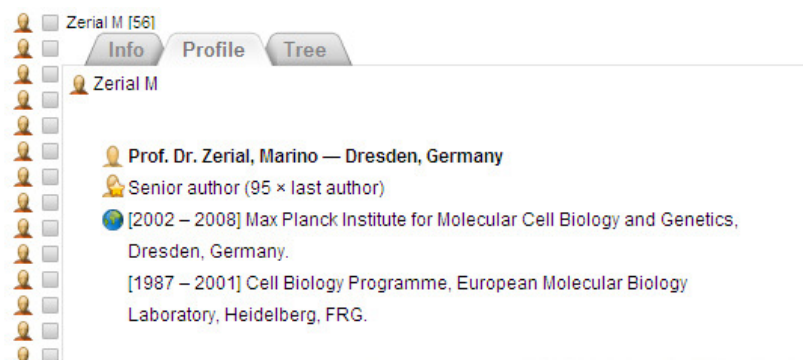


Figura 4.7 – Profile di un author

Se due articoli condividono lo stesso autore, il GoPubMed valuta le loro proprietà simili. Il sistema, quindi, tiene conto del fatto che l'autore di ogni documento pubblica spesso su temi di ricerca simili, con gli stessi co-autori e nelle stesse riviste. I temi di ricerca sono quindi collegati ai concetti della rete semantica in background. Più concetti due articoli hanno in comune e più breve è la distanza semantica della rete ed è più probabile che gli articoli sono stati scritti dalla stessa persona. Tale approccio porta ad una altissima precisione. Se in qualsiasi momento il sistema non è esatto, può essere corretto dagli utenti stessi.

La categoria “Where”

La localizzazione geografica delle persone, dei centri, delle università, ecc., si trovano sotto la categoria “Where”. Inoltre, sono elencate in questa categoria anche le riviste che risultano di top level per la query.

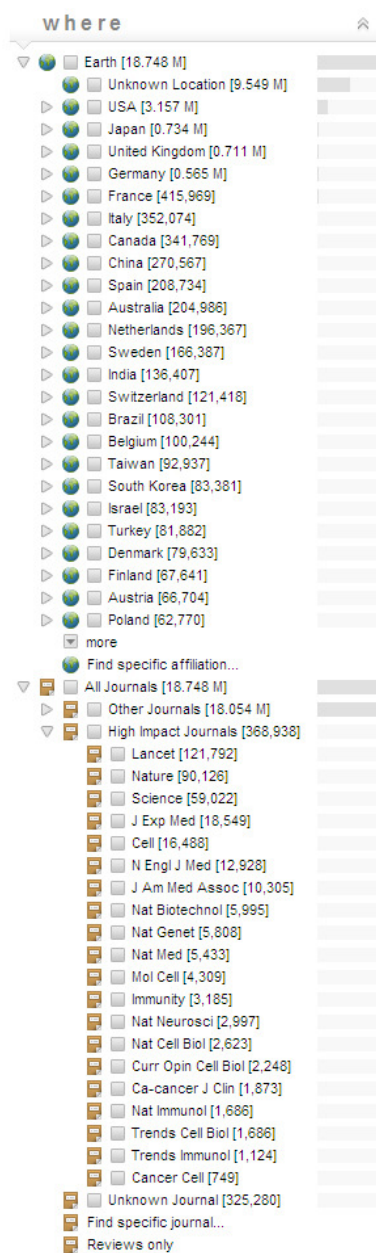


Figura 4.8 – Categoria Where

Le riviste “High Impact” (ad alto fattore d’impatto) e le “only Reviews” (Recensioni) possono essere selezionate qui. Cliccando su queste opzioni, le query troveranno riviste ad alto fattore d’impatto ed, in questo caso, le recensioni verranno generate in maniera automatica.

La categoria “When”

Gli anni delle pubblicazioni, che possono essere utili per la ricerca, sono riportati nella categoria “When”. Utilizzando la sezione “Publication date” (Data di pubblicazione) e cliccando su “today” (oggi), “last week” (la scorsa settimana), “month” (mese), “year” (anno), ecc., la finestra temporale per la ricerca può essere cambiata.

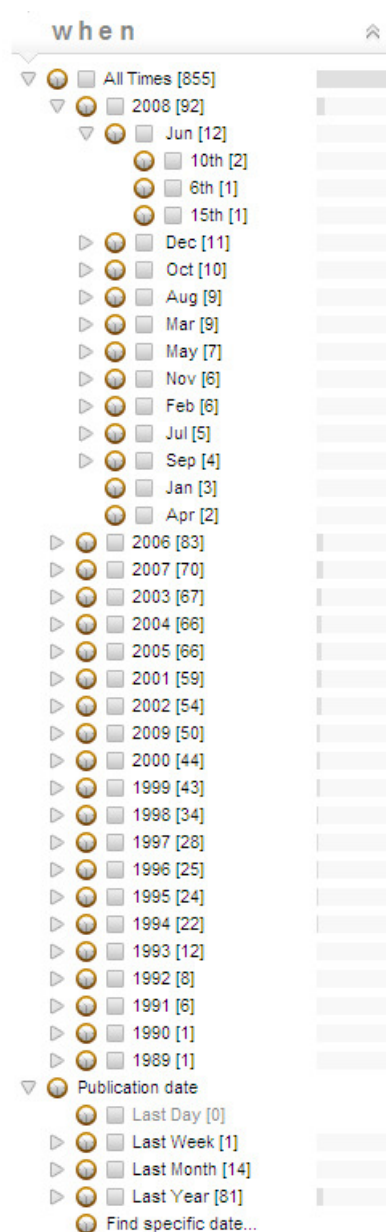


Figura 4.9 – Categoria When

Nelle prossime pagine si mostra come si effettua una semplice ricerca. La ricerca di concetti, autori, riviste, articoli, ecc. è fatta usando le stesse strategie in uso nel PubMed.

All'apertura della home page, nella finestra in alto a destra, dove è presente il punto di domanda giallo, l'utente può scrivere le parole chiave della sua ricerca e premere INVIO o cliccare su "find it". Durante la digitazione il sistema propone già dei suggerimenti (completamento automatico).

E' possibile fare una ricerca più strategica utilizzando dei termini chiave da aggiungere dopo il termine ricercato.

- per trovare i concetti nei titoli e/o negli abstract si aggiunge [TIAB];
- per trovare il nome di un autore si aggiunge [AU];
- per trovare un ente, università, istituto, ecc. si aggiunge [AD].





Ad esempio, scrivendo "Celiac disease" (celiachia) si ottiene il risultato, che viene visualizzato nella parte sinistra dello schermo.



Figura 4.10 – Frame di ricerca e di risposta

che indica che sono stati trovati 13.360 documenti semanticamente relativi a quell'argomento.

Nella parte in basso a destra sono elencati i titoli dei vari documenti che sono stati recuperati. Accanto al titolo di ogni documento ci sono 4 icone che permettono di operare sul documento:


-  mostra l'abstract completo del documento;
-  aggiunge l'articolo agli appunti;
-  alterna tra abstract o snippets mode per tutti i documenti;
-  apre il menù che consente di esportare il documento (sono possibili vari formati, quali Endnote, BibTeX, text plain, XML, RDF o FullText.).

Sotto ogni titolo di ogni documento sono riportati, e visualizzati grazie ad un'icona, l'autore, il giornale/rivista dove è stato pubblicato l'articolo, la riga dell'abstract dove il termine è contenuto e l'ente di affiliazione dove è stato elaborato.

I termini sottolineati sono i link che permettono l'accesso ad altri contenuti o dello stesso autore o della rivista o dell'ente.

Cliccando su “Related articles” si dà il via ad una nuova ricerca su argomenti che sono correlati con quel documento.

Il “PMID:numero” (PubMedID:numero) dà l’accesso diretto al documento (identificato dal numero dell’ID), memorizzato nel MEDLINE, tramite l’interfaccia PubMed.

Tramite l’icona  si può collegarsi ai concetti relativi che sono memorizzati sulla enciclopedia on-line Wikipedia.

Merita attenzione anche la parte in alto a destra della pagina GoPubMed relativa alla ricerca fatta dall’utente e detta “my search”.

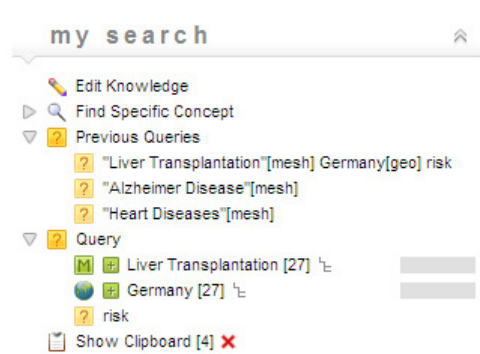







Figura 4.11 – Frame my search

Qui è possibile personalizzare ulteriormente la ricerca:

- Previous Queries: dà accesso alle ricerche fatte precedentemente;
- Query: qui si vedono le differenti parti della query fatta dall’utente. Cliccando sul disegnetto  si può aprire il cammino completo nella categoria What.
- le caratteristiche di raffinamento della ricerca sono mostrate nell’albero alla sinistra del termine . Cliccando semplicemente su esso si attivano i filtri  “Require” o  “Exclude”. Essi selezionano tutti i documenti che citano od escludono un concetto di una categoria ed i suoi discendenti.
- Clipboard: sono gli appunti che raccolgono tutti i documenti che l’utente ha selezionato tramite il tasto apposito. Cliccando sulla  il contenuto è cancellato.

Le citazioni risultanti di una ricerca sono mostrate in una pagina dieci alla volta. Per visualizzare le altre si può cliccare sopra:

Page **1** 2 3 4 5 6 7 8 9 10 11 ... 1878 1879

Fin qui è stato presentato come fare una semplice ricerca con il GoPubMed.

Esso permette di fare ricerche anche più complesse ed articolate del tipo:

1. Which molecular pathways are inhibited by aspirin? (Quali processi molecolari sono inibiti da una aspirina?)

2. What is Richard Axel from Columbia University working on? (Su cosa sta lavorando Richard Axel della Columbia University?)
3. Which cellular component is the protein Rab5 associated with? (A quale componente cellulare è associata la proteina Rab5?)

La ricerca di “aspirin inhibited“, “Richard Axel Columbia” o “Rab5” su un motore di ricerca classico restituirà un gran numero di risultati, non strutturati, che non rispondono alla domanda originale dell'utente. Un motore di ricerca classica non ha alcuna conoscenza di base sui processi molecolari, le funzioni e le componenti cellulari. GoPubMed utilizza un tale background di conoscenze sui percorsi biologici, le funzioni molecolari, cellulari e componenti, per citarne alcuni. I risultati della ricerca, indicizzati dalla macchina utilizzando le conoscenze di base, permettono così agli utenti di esplorare un ampio campo di risultati in modo strutturato. Con tale conoscenza di base, le macchine possono effettivamente rispondere alle domande.

1. GoPubMed scopre che il processo citato più frequentemente di “aspirin inhibits” è il processo di ciclo-ossigenasi (cyclooxygenase pathway). La risposta si vede dal maggior numero di elementi (5) presenti nella sottocategoria Top term della categoria What relativi al termine cyclooxygenase pathway.
2. GoPubMed scopre che uno degli argomenti più frequentemente citati per "Richard Axel Columbia" è l'attività del recettore olfattivo (olfactory receptor activity) (8 occorrenze trovate). Per questo lavoro egli ha ottenuto un Premio Nobel nel 2004.
3. GoPubMed scopre che la componente cellulare più frequentemente citata per "Rab5" è il endosome. La risposta si vede dal maggior numero di elementi (8) presenti nella sottocategoria Top term della categoria What relativi al termine endosome.

Una esemplificazione dettagliata di come si può arrivare a queste risposte per altre vie necessiterebbe di molte altre pagine e, per questo approfondimento, si rimanda il lettore agli esempi riportati nella documentazione ufficiale, scaricabile dal sito del GoPubMed.

Quello che interessava far notare al lettore è la possibilità, tramite il mouse, di spostarsi all'interno delle informazioni che la pagina di GoPubMed illustra. Si può passare da una categoria ad un'altra con un semplice click o addentrarsi nelle ramificazioni di un albero di conoscenza attraverso uno o più filtri. Si è visto che la parte che interessa di una ricerca può essere riportata negli appunti per una successiva esportazione o rielaborazione. Molte altre sono le possibilità offerte dal GoPubMed, soprattutto nella versione PRO che non ho potuto provare, ma che le righe precedenti hanno ben descritto.

Si è visto che Il GoPubMed propone un modo di ricerca che va al di là della semplice lista di articoli che un motore di ricerca quale PubMed od uno di tradizionale può offrire. E' questa una strada molto interessante, in evoluzione dalla quale ci si possono aspettare grandi novità per il futuro.

Conclusioni

L'obiettivo principale del Web Semantico, si è visto, è quello di consentire alle macchine di estrarre la conoscenza disponibile sul Web, che è spesso disponibile in formati eterogenei, e di combinarla per poter estrarre nuova conoscenza.

Questo processo è possibile solo se si riesce a rappresentare, esportare e condividere la conoscenza mediante una rappresentazione che sia indipendente dallo specifico ambiente operativo.

Le tecnologie del W3C qui viste (XML, RDF, RDFS, OWL, SPARQL, ecc.) consentono di raggiungere parzialmente questo obiettivo.

La figura qui sotto riassume il ruolo di queste tecnologie per combinare le informazioni provenienti da varie fonti che, per loro natura, sono eterogenee.

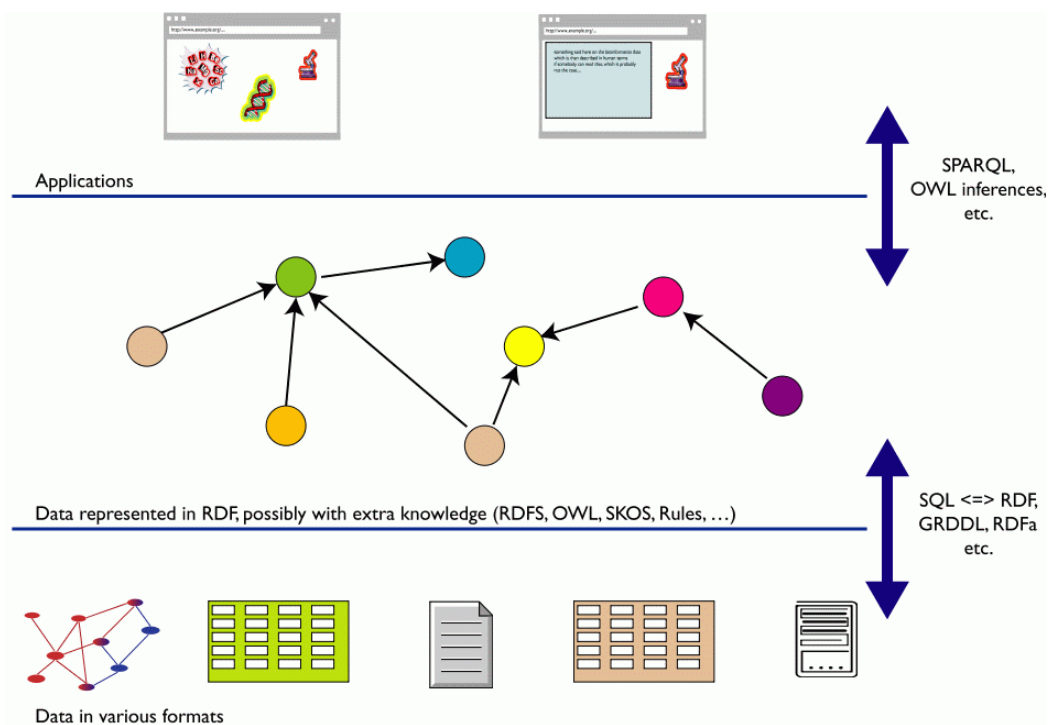


Figura C1- Il ruolo delle tecnologie per combinare informazioni provenienti da fonti eterogenee

In basso ci sono i dati, o più in generale le risorse. Gli URI riescono ad identificarle univocamente.

L'RDF riesce a rappresentare questi dati con una struttura. Con l'RDF Schema e l'OWL si riesce ad aggiungere più vocabolario all'RDF che permette di esprimere relazioni più complesse tra le classi.

Lo SPARQL permette di fare semplici interrogazioni sui dati espressi in RDF.

Lo SKOS riesce a rappresentare glossari, classificazioni, tassonomie e qualsiasi tipo di vocabolario strutturato, per una loro più facile pubblicazione.

Le regole codificate permettono di fare delle piccole inferenze in maniera automatica. Ad esempio, specificando che Y è figlio di X1 e che X2 è fratello di X1, si può far concludere alla macchina che X2 è zio di Y.

Questo dà un'idea del tipo di inferenza che le macchine riescono a fare in maniera automatica. Essa è molto limitata. Da quanto scritto e visto finora, il lettore dovrebbe aver capito che la visione iniziale di Tim-Berners Lee di Web Semantico, con la presenza degli agenti semantici che elaborano la conoscenza è ben lungi dall'essere realizzata.

Attualmente, non tutte le risorse, che appartengano o meno al Web, sono in esso rappresentabili e/o rappresentate. Le pagine Web, scritte da qualche anno, non hanno nessuna marcatura che ne possa illustrare il contenuto semantico. Si ha quindi il problema di come marcarle e di chi deve fare questo lavoro. Inoltre, bisogna vedere se, chi crea nuovi documenti da pubblicare nel Web, è disposto a farlo seguendo le raccomandazioni del W3C.

Si è visto, anche grazie all'esempio di GoPubMed, che finora si riescono a rappresentare piccoli domini di conoscenza, in questo caso l'ambito delle pubblicazioni biomediche che sono contenute nel database MEDLINE. Questo è fatto grazie ad una ontologia, la Gene Ontology, che rappresenta il dominio di conoscenza con tutte le sue relazioni ed il MeSH che definisce l'oggetto di interesse.

Nel corso degli ultimi anni l'attenzione degli sviluppatori si è rivolta nello sviluppo di quello che viene chiamato il "Web of Data", dove si è concentrati nella rappresentazione in classi dei domini di conoscenza con tutte le proprietà che legano tra loro tutti gli individui che appartengono ad una determinata classe. Tutte queste proprietà vengono espresse come affermazioni.

In definitiva si può dire che, finora, si è riusciti a rappresentare ambiti ben definiti di conoscenza, ma non ad elaborarli in maniera automatica.

Per i prossimi anni, è da attendersi un ulteriore raffinamento ed incremento dei metadati utilizzati per esprimere quantità di dati sempre più vaste, delle ontologie che illustrano le relazioni tra questi ed altri dati e delle regole che permettono di aumentare il grado di inferenza possibile.

Appendice

Indice delle abbreviazioni

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
AJAX	Asynchronous JavaScript and XML
BLD	Basic Logic Dialect
BMP	Basic Multilingual Plane
CERN	European Organization for Nuclear Research – Centro Europeo per la Ricerca Nucleare
CGI	Common Gateway Interface
CNR	Consiglio Nazionale delle Ricerche
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DAM	DARPA Agent Markup Language
DBMS	DataBase Management System
DCOM	Distributed Component Object Model
DHTML	Dynamic HTML - Dynamic Hyper Text Markup Language
DOAC	Description of a Career
DOM	Document Object Model
DSDL	Document Schema Description Languages
DTD	Document Type Definition
EBCDIC	Extended Binary Coded Decimal Interchange Code
EBNF	Extended Backus-Naur form
eRDF	Embedded RDF - Embedded Resource Description Framework
FAQ	Frequently asked questions
FOAF	Friend of a Friend
FTP	File Transfer Protocol
GO	Gene Ontology
hex	esadecimale
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP Secure - Hyper Text Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
ID	identification - identity

IMAP	Internet Message Access Protocol, a volte anche chiamato Interactive Mail Access Protocol
IRI	Internationalized Resource Identifier
ISBN	International Standard Book Number
ISO	International Organization for Standardization
ISTI	Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - Pisa
JSP	JavaServer Pages
JSON	JavaScript Object Notation
KOS	Knowledge Organization System
LCS	Laboratory for Computer Science (presso CERN)
LIMBER	Language Independent Metadata Browsing of European Resource
MEDLINE	Medical Literature Analysis and Retrieval System Online
MeSH	Medical Subject Headings
MFC	Meta Content Framework
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusetts Institute of Technology
MTON	Message Transmission Optimization Mechanism
NCBI	National Center for Biotechnology Information
NIH	National Institutes of Health
NITS	National Institute for Standards and Technology
NLM	National Library of Medicine
OASIS	Organization for the Advancement of Structured Information Standards
OBO	Open Biomedical Ontologies
OED	Oxford English Dictionary
OIL	Ontology Inference Layer
OMG	Object Management Group's
OWL	Ontology Web Language - Web Ontology Language (in alcuni testi)
PDF	Portable Document Format
PHP	Personal Home Page (prima definizione) - acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti
PICS	Platform for Internet Content Selection
POWDER	Protocol for Web Description Resource
PRD	Production Rule Dialect
PSL	Process Specification Language
R2RML	Relational Data Base to RDF Mapping Language
RDB2RDF	Relational Data Base to RDF
RDF	Resource Description Framework
RDFa	Resource Description Framework - in - attributes
RDFS	RDF Schema - Resource Description Framework Schema
REST	REpresentational State Transfer
RFC	Request for Comments

RIF	Rule Interchange Format
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SGML	Standard Generalized Markup Language
SIOC	Semantically Interlinked On-line Communities
SKOS	Simple Knowledge Organization System
SSL	Secure Sockets Layer
SMIL	Synchronized Multimedia Integration Language
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SWAD	Semantic Web Advanced Developed
SWRL	Semantic Web Rule Language
TC	Technical Committee
TCP	Transmission Control Protocol
TIOBE	The Importance Of Being Earnest (di Oscar Wilde)
TLS	Transport Layer Security
TR	Technical Report
UCS	Universal Character Set
UFT	Unicode Transformation Format
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
WAI	Web Accessibility Initiative
W3C	World Wide Web Consortium
W3C-IT	World Wide Web Consortium - Italia
WSDL	Web Services Description Language
WSDM	Web Services Distributed Management
WWW	World Wide Web
WXS	W3C XML Schema
XHTML	eXtensible Hypertext Markup Language
XLS	eXtensible Stylesheet Language
XLS-FO	eXtensible Stylesheet Language Formatting Objects)
XLST	eXtensible Stylesheet Language Transform
XML	eXtensible Markup Language
XMLS	XML Schema - eXtensible Markup Language Schema
XML-SW	XML Skunk Works
XRDS	eXtensible Resource Descriptor Sequence
XRI	eXtensible Resource Identifier
XSD	XML Schema Definition

Bibliografia

- [1.1] Tim Berners-Lee, *The Semantic Web*, in “Scientific American”, Maggio 2001
- [1.2] Tim Berners-Lee, *L'architettura del nuovo Web*, Feltrinelli, Milano, 2002
- [1.3] sito web ufficiale del W3C:
<http://www.w3.org>
- [1.4] sito web W3C con gli standard sul web semantico
<http://www.w3.org/standards/semanticweb/>
- [1.5] sito web ufficiale del W3C Italia:
<http://www.w3c.it>
- -
- [2.1] sito web di TIOBE Programming Community Index
<http://www.tiobe.com/content/paperinfo/tpci/index.html>
- [2.2] sito web del Consorzio OASIS (Organization for the Advancement of Structured Information Standard):
<http://www.oasis-open.org/>
- [2.3] sito web della W3C Web Services Activity:
<http://www.w3.org/2002/ws/>
- [2.4] sito web dell'enciclopedia on-line Wikipedia in lingua inglese:
<http://www.wikipedia.org/>
- sito web dell'enciclopedia on-line Wikipedia in lingua italiana:
<http://it.wikipedia.org/wiki/>
- [2.5] Tim Berners-Lee, Mark Fischetti, *Weaving the Web*, Harper, San Francisco, 1999
- [2.6] Paolo Bouquet e Roberta Ferrario, *Introduzione al numero monografico Semantic Web*, 2003
<http://www.swif.uniba.it/lei/ai/networks/03-2/introduzione.pdf>
- [2.7] sito web di Dublin Core
<http://dublincore.org/>
- [2.8] sito web di Swoogle
<http://swoogle.umbc.edu/>
- [2.9] sito web di Ontaria:
<http://www.w3.org/2004/ontaria>
- [2.10] sito web del National Institute for Standards and Technology, NITS:
<http://www.nist.gov/>

[2.11] Vittoria Shannon, Tim Berners-Lee, *A more revolutionary Web*, in International Herald Tribute, 23 Maggio 2006
<http://www.iht.com/articles/2006/05/23/business/web.php>

[2.12] sito web della W3C Semantic Web Activity:
<http://www.w3.org/2001/sw/Activity.html>

- -

[3.1] sito web ufficiale dell'International Organization for Standardization (ISO):
www.iso.org/

[3.2] sito web ufficiale di Unicode Consortium:
<http://www.unicode.org/consortium/consort.html>

[3.3] sito web ufficiale di Internet Assigned Numbers Authority:
<http://www.iana.org/>

[3.4] Uniform Resource Identifier (URI): Generic Syntax specifiche tratte dal sito ufficiale della The Internet Engineering Task Force (IETF) – RFC 3986
<http://www.apps.ietf.org/rfc/rfc3986.html>

[3.5] sito web ufficiale della The Internet Engineering Task Force (IETF)
<http://www.ietf.org/>

[3.6] W3C Interest Group Note “Cool URI’s for the Semantic Web”
<http://www.w3.org/TR/cooluris>

[3.7] L’attuale specifica degli URI: RFC 3987
<http://tools.ietf.org/html/rfc3987>

[3.8] sito web della W3C Internationalization (I18n) Activity
<http://www.w3.org/International/>

[3.9] sito web dell'OASIS Extensible Resource Identifier (XRI) TC
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xri

[3.10] Extensible Markup Language (XML) 1.0 (Fifth Edition) - W3C Recommendation
<http://www.w3.org/TR/REC-xml>

[3.11] L’attuale specifica degli Internet Media Type: RFC 3023 del 2001
<http://tools.ietf.org/html/rfc3023>

[3.12] Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols: RFC 3470 del 2003
<http://tools.ietf.org/html/rfc3470>

[3.13] sito web della W3C - XML home page:
<http://www.w3.org/XML>

[3.14] sito web XML.gov home page:
<http://xml.gov>

-
- [3.15] sito web [html.it](http://xml.html.it/), contenente pagine relative all'XML:
<http://xml.html.it/>
- [3.16] lista di schemi XML ordinata per scopo, sul sito Wikipedia.org
http://en.wikipedia.org/wiki/List_of_XML_schemas
- [3.17] W3C XML Schema 1.0 Specification
- XSD 1.0 Primer <http://www.w3.org/TR/xmlschema-0/>
 - XSD 1.0 Structures <http://www.w3.org/TR/xmlschema-1/>
 - XSD 1.0 Datatypes <http://www.w3.org/TR/xmlschema-2/>
 - Tools <http://www.w3.org/XML/Schema#Tools>
- W3C XML Schema 1.1 Specification
- XSD 1.1 Structures <http://www.w3.org/TR/xmlschema11-1/>
 - XSD 1.1 Datatypes <http://www.w3.org/TR/xmlschema11-2/>
- [3.18] Definizione dell'XML Schema da parte del W3C
<http://www.w3.org/XML/Schema#dev>
- [3.19] Il tutorial, in inglese, della W3Schools sull'XML Schema
<http://www.w3schools.com/schema/default.asp>
- [3.20] XML Namespace - W3C Recommendation
- XML 1.0 (third edition) 2009
<http://www.w3.org/TR/REC-xml-names/>
 - XML 1.1 (second edition) 2006
<http://www.w3.org/TR/2006/REC-xml-names11-20060816/>
- [3.21] sito web della W3C - Resource Description Framework home page:
<http://www.w3.org/RDF/>
- [3.22] W3C RDF W3C Recommendation - Concepts and Abstract syntax
<http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- [3.23] W3C RDF/XML Syntax Specification
<http://www.w3.org/TR/rdf-syntax-grammar/>
- [3.24] W3C RDF Schema Specification
<http://www.w3.org/TR/rdf-schema/>
- [3.25] W3C OWL Features
<http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- [3.26] W3C OWL Guide
<http://www.w3.org/TR/2004/REC-owl-guide-20040210/>
- [3.27] Wine and Food Ontology
<http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf>
<http://www.w3.org/TR/2004/REC-owl-guide-20040210/food.rdf>
- [3.28] W3C OWL 2 Overview
<http://www.w3.org/TR/owl2-overview/>
- [3.29] W3C SPARQL Query Language for RDF - W3C Recommendation
<http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>
-

- [3.30] SKOS Simple Knowledge Organization System - W3C Recommendation
<http://www.w3.org/TR/2009/REC-skos-reference-20090818/>
- [3.31] RIF W3C Rule Interchange Format Working Group Charter
<http://www.w3.org/2005/rules/wg/charter.html>
- [3.32] RIF Rule Integration Format - current status
http://www.w3.org/standards/techs/rif#w3c_all
- [3.33] SWRL Semantic Web Rule Language- W3C Member Submission
<http://www.w3.org/Submission/SWRL/>
- [3.34] POWDER Protocol for Web Description Resource - W3C Recommendation
<http://www.w3.org/TR/2009/REC-powder-formal-20090901/>
- [3.35] W3C POWDER Working Group home page
<http://www.w3.org/2007/powder/>
- [3.36] articolo “Approcci al Web Semantico” di Antonella Dorati e Stefania Costantini
<http://www.websemantico.org/articoli/approcciwebsemantico.php>
<http://costantini.di.univaq.it/>
- [3.37] articolo “The Semantic Web in breadth” di Aaron Swartz
<http://logicerror.com/semanticWeb-long>
http://en.wikipedia.org/wiki/Aaron_Swartz
- [3.38] Tim Berners-Lee: personal view sull’ User Interface nel Design Issue, pubblicato nel settembre 1997
<http://www.w3.org/DesignIssues/UI.html#OhYeah>

- . - . - . -

- [4.1] IIR 2010 - First Italian Information Retrieval - convegno
<http://ims.dei.unipd.it/websites/iir10/index.html>
- [4.2] CSB 2010 - Cultura Senza Barriere - convegno
<http://www.culturasenzabarriere.org/>
- [4.3] “Facility: the next evolutionary step in search?” di Edward Henning
http://www.culturasenzabarriere.org/wp-content/uploads/2009/11/Art_Henning.pdf
- [4.4] MEDLINE -U.S. National Library of Medicine- MEDLINE facts sheet
<http://www.nlm.nih.gov/pubs/factsheets/medline.html>
- [4.5] PubMed - home page
<http://pubmed.gov/>
<http://www.ncbi.nlm.nih.gov/pubmed/>
- [4.6] Gene Ontology website and Gene Ontology Consortium
<http://www.geneontology.org/>
<http://www.geneontology.org/GO.consortiumlist.shtml>

[4.7] MeSH - home pages

<http://www.ncbi.nlm.nih.gov/mesh>

<http://www.nlm.nih.gov/mesh/>

[4.8] MeSH in lingua italiana - Istituto Superiore di Sanità

<http://www.iss.it/site/mesh/Index.aspx>

[4.9] GoPubMed - home page

<http://www.gpubmed.org>

<http://www.gpubmed.org/web/gpubmed/>

<http://www.gpubmed.com>

<http://www.gpubmed.com/web/gpubmed/>

[4.10] sito web della Transinsight

<http://www.transinsight.com>

- . - . - . - . -

Altri link utili:

[a] W3C Markup Validation Service

<http://validator.w3.org>

[b] W3C Semantic Web Activity News

<http://www.w3.org/blog/SW>

[c] sito web aggiornato sullo standard RDF

<http://planetrdf.org>

[d] sito web in italiano sul Web Semantico, gestito da Pasquale Popolizio, non più aggiornato, contenente le traduzioni di alcuni vecchi articoli:

www.websemantico.org

Ringraziamenti

Un primo, doveroso, ringraziamento va alla mia Famiglia che, in tutti questi anni, mi ha sostenuto, permettendomi di raggiungere questo importante traguardo della mia vita. Un pensiero particolare va all'ultima nonna rimasta in vita: a 99 anni compiuti sempre mi ricorda nelle sue preghiere e mi aspetta per festeggiare assieme. Da non mettere in secondo piano zii e cugini tutti. Un ringraziamento particolare va anche alla famiglia Rizzo, con la relativa compagnia di amici.

Un ringraziamento particolare va anche al prof. Michele Moro, per la Sua disponibilità ed i preziosi consigli, senza i quali non avrei potuto concludere oggi il mio percorso di studi.

Un altro grande grazie va fatto agli amici di oggi e di ieri, con quali ho condiviso tanti momenti piacevoli. Volendo nominarli tutti rischierei di dimenticare qualcuno. Cominciamo dalle mitiche compagnie del "Parkcaravan" e dello "Staffblackout", passando dai colleghi nuotatori delle varie piscine del circondario, dalle aula studio di Padova, a quella della Biblioteca comunale di Este e di Monselice, senza dimenticare quella mitica del Patronato Redentore di Este. Un ringraziamento a tutto il personale per la cordialità e la disponibilità di spazi ed attrezzature.

Un saluto ed un ricordo va fatto a tutto il personale e agli allievi della scuola Morini Pedrina di Este dove ho lavorato, piacevolmente, per oltre 10 anni ed alle altre aziende con cui ho collaborato.

Un ringraziamento lo faccio anche a tutti quelli che ho conosciuto e che mi hanno incoraggiato a portare a termine quest'impresa. Ringrazio anche quelli che non avrebbero scommesso su di me: le loro "frecciatine" mi hanno dato ulteriore carica.