Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Ontologie e Linguaggi Ontologici per il Web Semantico

Giovanni Canfora, Daniela Di Fatta,
Giovanni Pilato

RT-ICAR-PA-04-06 **aprile 2004**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Ontologie e Linguaggi Ontologici per il Web Semantico

Giovanni Canfora[2], Daniela Di Fatta[1,]
Giovanni Pilato[1]

Rapporto Tecnico N.6:
RT-ICAR-PA-04-06 **aprile 2004**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo Viale delle Scienze edificio 11 90128 Palermo
[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica Viale delle Scienze 90128 Palermo

Sommario

Attualmente, Internet si presenta come un enorme contenitore di dati di vario genere, in cui le informazioni sono espresse senza fornire indicazioni sui loro significati, il che ne impedisce un' interpretazione automatica e rende imprescindibile l'apporto umano. Il Web Semantico, mediante una serie di linguaggi, si prefigge di attribuire un significato alle informazioni, in modo da permettere la realizzazione di applicazioni automatiche, che possano interpretare i dati. Un ruolo fondamentale, in tale contesto, è svolto dalle ontologie che permettono di rappresentare i significati dei dati di una determinata area di conoscenza, identificando le proprietà e le relazioni esistenti fra loro.

Nel presente rapporto tecnico si analizzano le ontologie, trattandole dal punto di vista formale e si definisce una possibile metodologia utilizzabile per le ontologie.

Particolare attenzione è stata posta all'analisi dei linguaggi che permettono l'implementazione delle ontologie, ed in particolare a OIL, DAML+OIL e OWL; inoltre, sono individuate le caratteristiche da ricercare nei linguaggi ontologici, verificando la loro rispondenza nei linguaggi analizzati.

INDICE

Capitolo 1

Ontologie

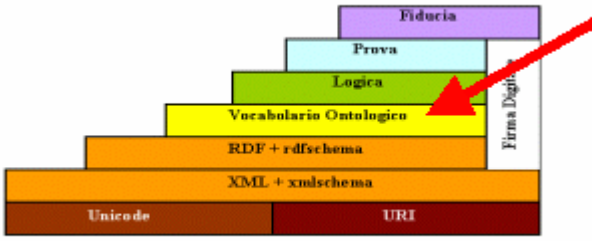

Figura 1.1: Posizione delle Ontologie nell'architettura del Web Semantico

Le ontologie hanno un ruolo fondamentale nello sviluppo del Web Semantico, permettendo, per mezzo di una teoria comune e condivisa sui domini, la realizzazione dell'accesso basato sui contenuti, sull'interoperabilità e della comunicazione tramite il Web.

Per avere applicazioni che possano interagire automaticamente, è necessario avere un modo comune di interpretare le collezioni di informazioni, per ridurre o eliminare la confusione concettuale e terminologica presente in uno specifico settore.

Tale compito è svolto dall'ontologia che costituisce una struttura unificante fra differenti punti di vista e serve come base per abilitare la comunicazione (fra persone, fra persone e sistemi, fra sistemi)[1].

L'ontologia elimina l'ambiguità, fornendo una base ad una base semantica e un vocabolario concettuale condiviso, per permettere la costruzione di descrizioni e la comunicazione.

1.1 Definizione

Nel corso degli anni il termine ontologia è stata usato con significati differenti, generando non poche ambiguità.

La parola ontologia deriva dalle parole greche "*ontos*" e "*logos*", che significano "*la parola dell'essere*".

Il termine più antico per indicare l'odierna ontologia è la "κατηγορια" aristotelica, che classifica in 10 categorie ogni cosa che poteva essere detta o "predicata"[2].

Il concetto di ontologia ha una chiara provenienza filosofica ed è un termine relativamente nuovo; esso indica la disciplina istituzionalizzata nel XIX secolo dai filosofi tedeschi per distinguere lo studio dell'essere in quanto tale dallo studio dei differenti tipi di esseri nelle scienze naturali.[1]

Ma solo recentemente le ontologie sono state prese in considerazione nel campo dell'intelligenza artificiale e, più in generale, dell'informatica.

Applicazioni si sono avute nel campo della "teoria della rappresentazione della conoscenza" con riferimento a settori specifici come la "granularità"[3] e le assunzioni esistenziali [4][5], oppure nel settore della comprensione del linguaggio naturale [6] [7] [8]e dell'acquisizione della conoscenza [9].

Più recentemente, dai primi anni novanta, ci sono state applicazioni nel settore della condivisione della conoscenza [10] [11] [12] [13].

Tuttavia, le ontologie rappresentano un argomento vasto e di non facile definizione, anche alla luce del disaccordo esistente sulle modalità di costruirle ed utilizzarle, nonché sui diversi ruoli loro attribuiti.

Per definire le ontologie è opportuno ripercorrere i lavori più significativi a riguardo, in modo da giungere ad una visione completa dei diversi significati che sono associati.

Nel 1991, Neches [10] individuò l'ontologia come "*l'insieme dei termini basilari e delle relazioni che costituiscono il vocabolario di una specifica area e delle regole per combinare termini e relazioni per estendere il vocabolario*".

In tale definizione, è indicato il modo di procedere per costruire un'ontologia : 1) identificare i termini basilari e le loro relazioni; 2) identificare le regole per combinarle; 3) realizzare definizioni di tali termini e relazioni.

La definizione di Neches chiarisce come l'ontologia non includa solo i termini definiti esplicitamente, ma anche termini che possono essere "desunti" usando delle regole.

[1] Wolffe, C.," Philosophia Prima Sive Ontologia",1729

Nella comunità della rappresentazione della conoscenza, la definizione più rappresentativa di ontologia appartiene a Gruber (1993) [12]: "U*n'ontologia è una specificazione formale ed esplicita di una concettualizzazione condivisa*".

In particolare, nella definizione di Gruber, l'ontologia fornisce una specificazione di una "*concettualizzazione*", tramite "*meta-dati*" che esplicitamente rappresentano la semantica in un modo elaborabile dalle macchine.

La definizione appare maggiormente comprensibile analizzando i termini che la compongono:

a) Il termine "*Concettualizzazione*" si riferisce all' individuazione di un modello astratto del fenomeno del mondo, avendo identificato i concetti rilevanti dei fenomeni . Il corpo di una conoscenza formalmente rappresentata è basata sulla sua "*concettualizzazione*" che indica gli oggetti, i concetti e le altre entità che sono assunte ed esistenti nelle aree di interesse e le relazioni fra loro esistenti [17]. La concettualizzazione, quindi, può essere definita come la struttura formale della realtà come percepita e organizzata da un "*agente automatico*", indipendentemente da quello che è il vocabolario usato e dalla specifica situazione.Volendo fornire un esempio di concettualizzazione, il termine "macchina" e l'equivalente termine inglese "car" devono avere la stessa concettualizzazione, poiché rappresentano due modi diversi di identificare lo stesso concetto.
b) "*Esplicito*" significa che i concetti utilizzati e i vincoli al loro utilizzo sono definiti in modo chiaro e comprensibile.
c) "*Formale*" si riferisce al fatto che l'ontologia deve essere comprensibile dalle macchine e, quindi, deve essere espressa in modo formale.
d) "*Condivisa*" riflette il fatto che l'ontologia deve rappresentare la conoscenza accettata consensualmente dalle diverse comunità.

Altra definizione che chiarisce il significato di ontologia si deve , ancora una volta, a Gruber [13] che, nel 1994, ha definito "*l'ontologia come un comune e condiviso modo di comprendere un campo che può essere comunicato fra persone e sistemi di applicazioni*".

L'ontologia, quindi, può essere vista come un insieme di termini di conoscenza, inclusi il vocabolario, le interconnessioni semantiche e alcune semplici regole di inferenza e regole logiche per particolari scopi [5].

Secondo Gruber, un' ontologia può essere vista come una quintupla composta da classi, istanze, funzioni relazioni, assiomi: le classi corrispondono alle entità del dominio, le istanze sono gli oggetti attuali presenti nel dominio, le funzioni e le relazioni permettono di collegare entità del dominio, gli assiomi restringono i significati e l'uso di classi, di istanze, di funzioni e relazioni.

Risale al del 1998 la definizione di Guarino [14]: "*un ontologia è un insieme di assiomi logici progettati per rendersi conto del significato inteso di un vocabolario*".

Un' analisi delle relazioni fra ontologie e basi di conoscenza può essere realizzata considerando la definizione di Bernars [15]: "*l'ontologia fornisce i significati per descrivere esplicitamente la "concettualizzazione" dietro la conoscenza rappresentata in una base di conoscenza*" .

Altra definizione che chiarisce il rapporto fra ontologia e basi di conoscenza si deve a Swartout [16] "*un'ontologia è un insieme strutturato di termini che descrivono un dominio e che possono essere usati come uno scheletro per la creazione di una base di conoscenza*".

L'ontologia è una particolare base di conoscenza che descrive fatti assunti sempre verida una comunità di utenti, in virtù di un significato comune e condiviso del vocabolario usato (informazioni indipendenti dallo stato); invece, una generica base di conoscenza può anche descrivere fatti e asserzioni relative ad un particolare stato delle cose (informazioni dipendenti dallo stato).

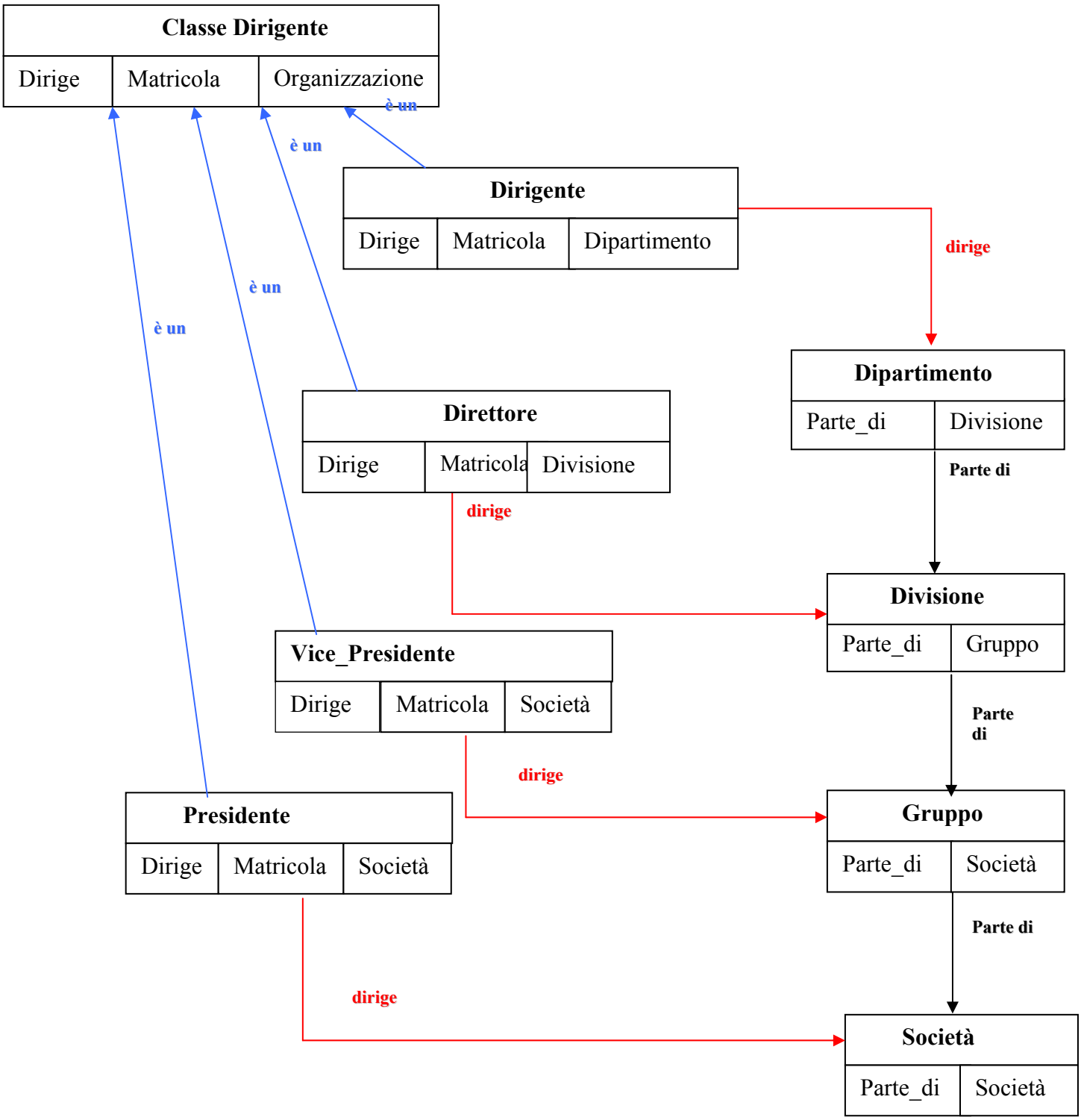

Figura 1.2: Esempio di una ontologia relativa alla struttura di una multinazionale

1.2 Definizione formale dell'ontologia

Una definizione formale del concetto di ontologia può essere ottenuta definendo i concetti visti nella definizione di Gruber [13] del 1993, che introduce il concetto di "*concettualizzazione*" intendendo l' ontologia come come "*la specificazione di una concettualizzazione*".

Questa definizione si riferisce ad una "*relazione estensionale*" che è una relazione matematica ordinaria su D, quindi relativa ad un particolare stato delle cose.
Invece, le "*relazioni intenzionali*", dette anche "*relazioni concettuali*", indipendenti dallo stato delle cose sono come delle funzioni da un possibile mondo verso degli insiemi (mentre le "*relazioni ordinarie*" sono definite su un certo dominio, le "relazioni concettuali" sono definite su uno spazio dei domini).

Definizione 6.2:"*Lo "**spazio dei domini**" è una struttura < D,W >, dove D è un dominio e W è un insieme massimo di stati delle cose."*

Ad esempio, se D è l'insieme dei possibili blocchi sul tavolo, allora W è l'insieme dei possibili posizionamenti dei blocchi.

Definizione 6.3:"*Una "**relazione concettuale**" ρ^n di entità n su uno spazio dominio <D,W> è la funzione $\rho^n: W \rightarrow 2^{D^n}$ da W nell'insieme di tutte le relazioni na-rie su D."*

L'insieme $\mathbf{E}_\rho = \{\rho(w) \mid w \in W\}$ contiene, per una generica relazione concettuale ρ, le estensioni ammissibili di ρ.

Definizione 6.4:"*Una "**concettualizzazione**" per D può essere definita, anche, come una tripla ordinata $\boldsymbol{C} = <D,W,\Re>$, dove $\Re$ è un insieme di relazioni concettuali sullo spazio del dominio <D,W>."*

Una "*concettualizzazione*" può essere vista come un'insieme di relazioni concettuali definite su uno spazio del dominio.

La relazione <D,**R**> , si riferisce ad un particolare stato delle cose e può essere definita come una "**struttura mondo**".
Una "*concettualizzazione*" contiene molte "*strutture mondo*", una per ogni mondo: rappresentano le "*strutture mondo*" intese in accordo con tale "*concettualizzazione*".

Definizione 6.5:"*Considerando una "concettualizzazione" **C** = <D, W, ℜ>, per ogni possibile "mondo" w∈W, "**la struttura intesa**" di w in accordo con C è la struttura* $\boldsymbol{S}_{w\boldsymbol{C}} = <D, \boldsymbol{R}_{wC}>$*, dove* $\boldsymbol{R}_{w\boldsymbol{C}} = \{\rho(w) | \rho \in \Re\}$ *è l'insieme delle estensioni (relative a w) degli elementi di ℜ."*

Possiamo inoltre denotare con $\mathbf{S_C}$ l'insieme $\{\mathbf{S}_{w\mathbf{C}} \mid w \in \mathrm{W}\}$ di tutte le strutture mondo intese di **C**.

Definizione 6.6:"*Un **linguaggio logico L** con un vocabolario V può essere definito come una struttura <**S**,I>, dove **S**=<D,**R**> è una struttura del mondo e I: V→D∪**R** è una funzione interpretazione che assegna elementi di D a simboli costanti di V e elementi di **R** a simboli predicati di V.*"

Un modello fissa una particolare interpretazione estensionale del linguaggio.

Definizione 6.7:*" Una "**interpretazione intenzionale"** tramite una struttura <**C**, ℑ>, dove **C** = <D,W, ℜ> è una "concettualizzazione" e ℑ: V→D∪ℜ è una funzione che assegna elementi di D a simboli costanti di V e elementi di ℜ a simboli predicati di V. Possiamo chiamare" interpretazione intenzionale" un "affidamento ontologico" per L."*

Definizione 6.7: *"Se **K** = <**C**, ℑ> è un "affidamento ontologico" per L , noi sappiamo che L **affida** a **C** tramite **K**, mentre **C** è la **concettualizzazione fondamentale** di **K**."*

Definizione 6.8:*"Dato un linguaggio L con vocabolario V e un suo "affidamento ontologico" **K** = <**C**, ℑ> per L, un modello < **S**, I > sarà **compatibile** con K se:*

1. ***S**∈**SC**;*

2. *per ogni costante c si ha che I(c) =ℑ(c);*
3. *esiste un "mondo" w tale che, per ogni simbolo predicato p, trova la corrispondenza fra tale predicato n in un "estensione" ammissibile di ℑ(p),cioè esiste una "relazione concettuale" r tale che:ℑ(p) = ρ ∧ ρ(w) = I(p)."*

Definizione 6.9: ***"****L'insieme* $\boldsymbol{I_K(L)}$ *di tutti i modelli di L che sono compatibili con K sono chiamati "****modelli intesi"*** *di L in accordo con K."*

In generale, non c'è modo di ricostruire "*l'assegnazione ontologica*" di un linguaggio da un insieme di "*modelli intesi*", poiché un modello non necessariamente riflette un particolare mondo: le relazioni rilevanti considerate possono non essere sufficienti a caratterizzare completamente lo stato delle cose, quindi, un modello può attualmente descrivere una situazione comune a molti stati delle cose.
Allora, è possibile ricostruire la corrispondenza tra parole e "*relazioni estensionali*" stabilite attraverso la "concettualizzazione" fondamentale.
Un insieme di modelli indicati è così, solo, una debole caratterizzazione di una "*concettualizzazione*": esclude delle assurde interpretazioni, senza realmente descrivere il significato del vocabolario.

Definizione 6.10:*" Dato un linguaggio L con "assegnazione ontologica" K, un* ***ontologia*** *per L è un insieme di assiomi progettati in modo tale che l'insieme dei suoi modelli approssimi nel miglior modo l'insieme dei modelli intesi di L in accordo con K."*

La difficoltà di trovare l'insieme degli assiomi, porta alla necessità che un' ontologia introduca altri modelli accanto a quello inteso, per specificare una "concettualizzazione" solo in un modo indiretto, così:

1. può solo approssimare un insieme di modelli intesi;
2. tale insieme di "*modelli intesi*" è solo una debole caratterizzazione di una "concettualizzazione".

Definizione 6.11: *"Una ontologia O per un linguaggio L* ***approssima una concettualizzazione*** *C se esiste una "assegnazione ontologica"* $\boldsymbol{K} = \langle \boldsymbol{C}, \mathfrak{I} \rangle$ *tale che i "modelli intesi" di L in accordo con K siano inclusi nel modello di O."*

Definizione 6.12: "*Un ontologia è* ***assegnata ad una "concettualizzazione"*** *C se:*

1. *è stata progettata per caratterizzare C;*
2. *approssima C."*

La definizione di ontologia di Gruber può essere rivista sulla base delle definizioni precedenti: "*Un' ontologia è una teoria logica per rendere conto del "significato inteso" di un vocabolario formale, cioè la sua "assegnazione ontologica" ad una particolare "concettualizzazione" del mondo. "*
Il "*modello inteso*" di un linguaggio logico usando un vocabolario è forzato dall'"*assegnazione ontologica*".
Un' ontologia, indirettamente, riflette quest'assegnazione e la "*concettualizzazione*" sottostante, approssimando il modello inteso. [14]
E' importate rilevare che un' ontologia dipende dal linguaggio, mentre una "*concettualizzazione*" è indipendente dal linguaggio, quindi, il termine ontologia unisce i due aspetti.

1.3 Ciclo di vita di una ontologia

Per costruire le ontologie non esiste una metodologia standard, ma sono state presentate diverse linee guida [1][18]. In particolare, i passi usati di solito invocati non sono diversi da quelli richiesti dagli approcci dell'ingegneria del software.
La costruzione di una ontologia richiede per potere individuare la conoscenza di uno specifico dominio, un ambiente formato da esperti di settore, utenti finali e specialisti software.
La progettazione di una ontologia è un processo iterativo che evolve attraverso diversi stati prima di raggiungere il suo obiettivo.
Durante il processo di sviluppo di una ontologia, devono essere effettuate le seguenti attività:

1. pianificazione e specificazione;
2. raccolta di dati e acquisizione della conoscenza;
3. concettualizzazione;
4. formalizzazione;
5. integrazione;
6. implementazione;
7. valutazione;
8. documentazione e mantenimento.

1.3.1 Pianificazione e specificazione

Per sviluppare una ontologia è necessario procedere all'individuazione degli obiettivi e dei limiti del processo di creazione, indicandoli in un "**documento di specificazione dei requisiti dell'ontologia**" [19].
Gli obiettivi da seguire nella progettazione sono, quindi, individuati a partire dai motivi per cui si costruisce l'ontologia e dall'utilizzo che si prevede di farne [1].
Per individuare gli obiettivi devono essere applicate opportune tecniche.
Una tecnica di analisi interessante è ottenuta tramite i cosiddetti "*scenari*" [20], rappresentazioni in linguaggio naturale di una determinata realtà, utili per acquisire conoscenza.
In particolare, distinguiamo uno "*scenario di interazione*" che descrive la realtà che s'intende esaminare e le conoscenze che s'intendono trasferire agli elaboratori per la risoluzione dei problemi e gli "*scenari motivatori*" che sono esempi di problemi non risolti utilizzando le ontologie esistenti.
La progettazione delle ontologie avviene basandosi sulla descrizione degli "*scenari*", mentre la realizzazione sarà dettata dalla necessità di risolvere un determinato "*scenario*"[1].

1.3.2 Raccolta dati e acquisizione della conoscenza

La raccolta dei dati è un processo necessario per acquisire la conoscenza e per effettuare la "*classificazione dei concetti*".

La fase di raccolta intende [1]:

1 identificare i concetti chiave e le relazioni nel "*dominio di interesse*";

2 produrre definizioni per tali concetti e relazioni, in formato testo, precise e non ambigue;

3 identificare i termini per riferirsi a concetti e relazioni;

4 accettare i passi precedenti.

I passi da seguire per realizzare la raccolta ed analisi i dati sono i seguenti:

1. preparazione:;
2. raccolta dei dati;
3. analisi preliminare e modellazione informale;
4. controlli;
5. formalizzazione e convalida.

Per la raccolta dei dati e della conoscenza sono utilizzate spesso tecniche di tipo "*bottom-up*".

La "*conoscenza*" può essere estratta dalle sorgenti di conoscenza[2] usando tecniche come interviste, brainstorming, analisi formali e informali su testi e strumenti per l'acquisizione della conoscenza.

Una tecnica possibile può essere la seguente:

- Interviste non strutturate con esperti, per costruire un bozza iniziale del "*documento di specificazione dei requisiti*";
- Analisi informale per studiare i concetti principali;
- Analisi formale per identificare le strutture da individuare (definizioni, affermazioni) e il tipo di conoscenza fornito da ognuno (concetti, attributi, valori e relazioni);
- Interviste strutturate con esperti del settore per raccogliere la conoscenza specifica e dettagliata circa i concetti, loro proprietà e relazioni.

[2] le sorgenti di conoscenza possono essere: esperti umani, libri, manuali, figure, tabelle o altre ontologie

1.3.3 Concettualizzazione

Partendo dalla conoscenza acquisita sul dominio, è possibile "*concettualizzare*" in un modello, cioè individuare le relazioni che riguardano un dato dominio e raccoglierle in un apposito modello, in modo da descrivere i problemi e le relative soluzioni[19].
Lo studio della terminologia permette l'individuazione di relazioni fra termini e definizioni appartenenti alla stessa area di interesse o allo stesso dominio. Tali relazioni sono utili per raggruppare in gruppi di termini relazionati fra di loro.
Tale operazione di raggruppamento permette di suddividere il lavoro di concettualizzazione[1] e prevede le seguenti fasi:

- "*Categorizzazione*" provvisoria per inclusioni ed esclusioni;
- Registrazione delle decisioni mediante note per riferimenti futuri;
- Raggruppamento dei potenziali termini simili e sinonimi insieme;
- Identificazione dei riferimenti semantici trasversali fra aree diverse.

La fase di "*concettualizzazione*" richiede un dominio di conoscenza strutturato con un modello concettuale rappresentato in termini del vocabolario di dominio identificato nell'attività di specificazione dell'ontologia e, per essere realizzata, richiede un certo numero di attività, collegate con uno speciale documento detto "*Rappresentazione Intermedia*" [18]. In tale documento possono essere individuati diversi parti che corrispondono alle diverse fasi da realizzare per ottenere la "*concettualizzazione*". In particolare si ha:

- "***Dizionario dati"o "glossario completo dei termini"***": identifica e raccoglie i concetti utilizzati e potenzialmente utili del dominio, i significati, gli attributi e le istanze, etc.
- "***Alberi di classificazione dei concetti***": organizzano i concetti del dominio in tassonomie usate per riconoscere i concetti correlati e modulare il dominio di conoscenza in ontologie indipendenti.
- "***Tabelle degli attributi di istanze***": forniscono informazioni circa gli attributi e i loro valori nelle istanze.
- "***Tabelle degli attributi di classe***": forniscono informazioni sugli attributi appartenenti alle classi che descrivono i concetti e i loro valori.

- "***Tabelle delle costanti***": forniscono informazioni sulle costanti ovvero elementi caratterizzati dal fatto che assumono sempre lo stesso valore, di solito usati nelle formule per specificare informazioni legate al dominio di conoscenza.
- "***Tabelle delle formule***": descrivono le formule che permettono di derivare valori numerici degli attributi delle istanze dai valori numerici di altri attributi e costanti.
- "***Alberi di classificazione di attributi***": mostrano graficamente attributi e costanti relazionati nella sequenza di inferenza .
- "***Tabelle di istanze***": per ogni istanza presente nel "*dizionario dati*" ed esistenti nel dominio viene creata una relativa tabella.

1.3.4 Tassonomie

Un' ontologia di solito include una **tassonomia di concetti**, utilizzata per conferire all'ontologia una struttura e facilitare la comprensione umana e l'integrazione.[22]
Una tassonomia costituisce una classificazione basata sulle similarità che permette di classificare[3] e categorizzare[4] la conoscenza.
La relazione più importante delle tassonomie è l'inclusione in una categoria più vasta, detta anche "*sussunzione*" che permette di incorporare un concetto in un concetto più generale. Basato su tale relazione è il meccanismo dell'ereditarietà con cui un concetto eredita delle caratteristiche da un altro concetto che lo include.
Per la rappresentazione della struttura delle tassonomie vi sono diversi i tipi di struttura proposti [23] : ad albero, a reticolo e a grafico generale multi-eredità.
Diversi sono anche gli approcci, presentati in letteratura, utilizzabili in fase di costruzione per la tassonomia di concetti. La scelta di un dato approccio e le motivazioni della scelta sono strettamente collegate al tipo di dominio e di dati manipolati [1]:

- "**Approccio Bottom-up**": la tassonomia è costruita determinando prima i concetti di "*livello basso*", cioè quelli relativi a casi specifici, attraverso cui si generalizza per

[3] La classificazione è l'inferenza con cui gli esseri umani determinano se qualcosa appartiene ad una data classe o categoria;
[4] La categorizzazione è l'inferenza che permette di identificare ed organizzare le categorie e le classi di cose relative al mondo

trovare i concetti di "*livello più alto*". Tale tipo di approccio porta ad un numero elevato di dettagli e alla crescita degli sforzi necessari per la creazione delle tassonomie, in quanto prima di potere individuare i concetti di "*livello più alto*" potrebbe essere necessario determinare un numero elevato di concetti di "*livello basso*". I rischi che si corrono in tale approccio sono quelli di possibili inconsistenze. Il campo di applicazione è quello delle ontologie specifiche e "*fatte su misura*" con elevato grado di dettaglio e concetti granulari.

- "**Approccio top-down**": parte dal concetto generico per costruire una struttura attraverso la specializzazione. Tale approccio risulta migliore per controllare il livello di dettagli, in quanto una volta individuati i concetti generali si può procedere alla specializzazione soltanto dei concetti necessari. Tuttavia, la scelta di determinare i concetti partendo dall'alto può portare all'imposizione di categorie arbitrarie di livello alto, con rischi sulla stabilità del modello e conseguente necessità di aggiustamenti. Vantaggi nell'utilizzo di quest'approccio si hanno, ad esempio, nel il riuso delle ontologie e nel mantenimento della coerenza.
- "**Approccio middle-out**": si identificano i concetti fondamentali in ogni dominio, che vengono, successivamente, generalizzati e specializzati per completare l'ontologia. Tale approccio è legato alla possibilità di individuare tematiche rilevanti di gruppo e porta ad un bilanciamento in termini del livello di dettaglio: il livello di dettaglio aumenta solo se necessario, con conseguente diminuzione degli sforzi e aumento della stabilità. Le caratteristiche di tale approccio, quindi, sono l'aumento della modularità e la stabilità dei risultati.

1.3.5 Formalizzazione

La scelta relativa al grado di formalismo da utilizzare nelle ontologie dipende dalle esigenze: si va da esigenze informali, come in un glossario per condividere la conoscenza fra gli utenti, a necessità più formali come quelle derivanti dall'interoperabilità fra programmi software.

In base alle necessità, quindi, il livello di formalismo varia dalle definizioni informali espresse in linguaggio naturale, alle definizioni espresse in un linguaggio con sintassi e semantica rigorosamente definita (logica del primo ordine).

Ad incidere sulla scelta del livello di formalismo, è per gran parte il livello di automazione richiesto nello svolgimento dei vari compiti che l'ontologia deve supportare: se l'ontologia è solamente utilizzata per comunicare fra persone, la rappresentazione dell'ontologia può essere informale, mentre se l'ontologia deve essere usata da programmi software o "agenti" intelligenti, allora deve essere più precisa.

Uschold e Gruninger[1] hanno proposto di suddividere in quattro livelli il livello possibile di formalismo delle ontologie[5]:

- **"*altamente informale*"**: espresse in linguaggio naturale;
- **"*semi-informale*"**: espresse in una forma ristretta e strutturata di linguaggio naturale, in modo da accrescere la chiarezza e ridurre l'ambiguità;
- **"*semi-formale*"**: espresse in una forma artificiale definita formalmente linguaggio;
- **"*rigorosamente formale*"**: termini meticolosamente definiti con semantica formale, teoremi e prove di tali proprietà per esprimere la correttezza e completezza.

A loro volta le ontologie formali possono essere distinte a seconda del modo in cui i sottotipi differiscono dal supertipo:

- **ontologie assiomatizzate** distinguono i sottotipi mediante assiomi e definizioni in un linguaggio formale;
- **ontologie basate su proto*tipi*** distinguono i sottotipi mediante una comparazione con un membro tipico, detto *prototipo*, per ogni sottotipo.

Il formalizzare un' ontologia non consiste nel sostituire versioni informali con versioni formali, ma nello sviluppare le parti di interesse, aumentandone il formalismo per ottenere un' ontologia operazionale[6] e documentata [7].

La versione informale dell'ontologia, quindi, non costituisce un passo intermedio destinato a venir meno con la formalizzazione, ma ha un ruolo importante nel

[5] Lungo quello che potrebbe essere un continuo
[6] Con descrizioni formali degli attributi semantici rilevanti necessari per concepire il sistema
[7] Con descrizione informale della descrizione formale, possibilmente aumentata da capacità di navigazione,

documentare l'ontologia, sia per facilitare la comprensione sia per permettere il riuso delle ontologie e la reingegnerizzazione[24]: a tale scopo, deve includere definizioni in linguaggio naturale, i commenti e le osservazioni che saranno poi utilizzate dagli utenti umani per appropriarsi dell'ontologia.
I linguaggi formali e la logica sono necessari per l'implementazione automatica.
Il linguaggio naturale e i termini, invece, permettono l'accesso agli umani: per la versione testo delle ontologie si deve raggiungere un appropriato bilanciamento fra precisione tecnica e chiarezza, in modo che sia accessibile a lettori non tecnici e serva come specificazione per produrre codice.

1.3.6 Meta ontologie

La codifica della concettualizzazione catturata nei diversi linguaggi formali implica [1]:

1. "*affidamento*" a termini di base che saranno usati per specificare le ontologie (classi, entità, relazioni) il cui risultato è detto **meta-ontologia,** perché è l'ontologia dei termini di rappresentazione usati per esprimere l'ontologia principale.
2. scelta di un linguaggio di rappresentazione, capace di supportare le meta-ontologie;
3. scrittura del codice.

Per abilitare la condivisione della conoscenza fra "agenti" implementati con differenti sistemi di rappresentazione, la concettualizzazione dovrebbe essere specificata senza essere influenzata da una particolare codifica a livello simbolico.
La scelta di una particolare codifica degli assiomi è dettata solamente dalla convenienza nell'utilizzo di una data notazione o di una data implementazione e dovrebbe essere minimizzata, poiché potrebbero nascere vincoli sul modo di pensare e definizioni inadeguate o incomplete.
Anche se le considerazioni sulla formalizzazione non dovrebbero corrompere la concettualizzazione, può accadere il contrario con conseguente difficoltà nell'eseguire l'astrazione da ogni linguaggio precedentemente usato per individuare nuova concettualizzazione.

1.3.7 Riuso ontologie

Il riuso della conoscenza è un' attività importante nel campo della creazione delle ontologie, perché consente di costruire nuove ontologie utilizzando parti già costruite.
Vi sono vantaggi in termini di risparmio di tempo e risorse. Favorita risulta , inoltre, la realizzazione di ontologie standard. Ma le attività da realizzare per effettuare il riuso risultano impegnative, perché l'affidamento e la concettualizzazione devono essere allineati fra le ontologie riusate e le ontologie desiderate.
Dal punto di vista teorico, il riuso è possibile poiché un concetto può essere rilevante per un particolare compito senza essere necessariamente specifico per quel compito[25].
Elementi chiave per rendere la conoscenza riutilizzabile sono la decontestualizzazione e la generalizzazione, quindi, per riusare un singolo elemento di conoscenza messo a punto per un'altro contesto, è necessario renderlo indipendente da tale contesto e generalizzarlo.
In particolare, la generalizzazione implica una formalizzazione del contesto e lo stabilire una terminologia condivisa [24]. Tuttavia, il riuso di concetti generali per applicazioni specifiche, in certi casi, richiede un notevole sforzo di traduzione dei concetti generali in concetti specifici e questo è un aspetto che va tenuto in considerazione durante la progettazione di un'ontologia quando si deve decidere sul riuso di concetti.
La modularizzazione e l'organizzazione gerarchica sono strategie di progettazione che facilitano la riutilizzabilità [15].
Altra caratteristica importante da considerare, quando si pensa ad ontologie riusabili e condivisibili, è quella di estendibilità di una ontologia [1]: una ontologia dovrebbe essere progetta per anticipare l'uso del vocabolario condiviso, quindi dovrebbe offrire un fondamento concettuale per un intervallo di compiti e non solo per il compito specifico. In tal modo, la rappresentazione dovrebbe essere capace di definire nuovi termini per usi speciali basati sul vocabolario esistente, in modo da non richiedere la revisione di definizioni esistenti.
Emerge dalla pratica che si può sempre adattare una ontologia, ma non è mai possibile riutilizzarla senza adattarla[23].

Un importante ambito di ricerca nell'uso dell'ontologia è quello relativo alla di librerie di ontologie che possono essere adattate a differenti classi di problemi. La sfida è di determinare le ontologie più appropriate per un dato problema.

1.3.8 Integrazione ontologie

Il riuso di differenti ontologie implica che ontologie diverse siano combinate, mediante un processo di integrazione o fusione, al fine di creare una nuova ontologia.
Fondamentale è il processo di allineamento ontologico che permette di accordare due o più ontologie, per renderle consistenti e coerenti.
Il processo di integrazione ontologico comporta i seguenti passi[26]:

- individuare le aree di sovrapposizione nelle ontologie;
- collegare i concetti che sono semanticamente chiusi, attraverso relazioni di equivalenza e l'inclusione in una categoria più vasta (processi di allineamento);
- controllare la consistenza, coerenza e non ridondanza dei risultati.

La fase più critica è quella dell'allineamento dei concetti, perché richiede che siano compresi i significati dei concetti, utilizzando funzioni di similarità semantica.
Nella combinazione di due ontologie possono nascere problemi a causa dello scarso accoppiamento, cioè del fatto che le due ontologie differiscono su uno o più concetti.
In particolare lo scarso accoppiamento può nascere a livello:

- linguaggi (meta-modelli);
- ontologie (modelli).

Al primo livello, troviamo le primitive dei linguaggi usati per specificare una ontologia. L'utilizzo di linguaggi ontologici differenti comporta, infatti, uno scarso accoppiamento fra le ontologie che si intende combinare.
In tale ambito, si distinguono quattro tipi di scarso accoppiamento:

- **a livello sintattico**: fra differenti linguaggi ontologici che usano differenti sintassi;
- **nelle rappresentazioni logiche**: differenza nella rappresentazione di nozioni logiche;
- **nella semantica delle primitive:** costrutti di linguaggi possono avere lo stesso nome in differenti linguaggi, ma differire per la semantica;

- **nell'espressività di linguaggi**: differenza nella espressività fra due linguaggi; infatti, determinati linguaggi possono esprimere concetti non esprimibili in altri linguaggi.

Invece, al livello delle ontologie lo scarso accoppiamento è dovuto alle differenze di modellazione dell'ontologia di dominio, indipendentemente dal linguaggio utilizzato. Si possono riscontrare tre tipologie di scarso accoppiamento:

- **scarso accoppiamento di concettualizzazione**: differenze semantiche fra concettualizzazioni del dominio delle ontologie;
- **scarso accoppiamento di interpretazione**: differenti paradigmi per rappresentare concetti come tempo, causalità, azione e diversi modi di modellare i concetti;
- **scarso accoppiamento terminologico**: lo stesso concetto è rappresentato da differenti nomi o il significato di un termine varia secondo i contesti.

1.3.9 Documentazione

In un'ontologia devono essere comunicate e motivate le scelte effettuate, in modo da minimizzare l'ambiguità, fornendo esempi per facilitare la comprensione.[1]
La documentazione è essenziale sia per i progettisti che per gli utenti.
La documentazione, infatti, permette al progettista la comprensione a posteriori delle scelte fatte in fase di progettazione e facilita la condivisione di quest' ultime con gli altri progettisti con chi poi dovrà mantenere l'ontologia.
Per gli utenti la documentazione abilita e facilita il processo di apprendimento e di appropriazione.
Invece, l'assenza di una solida documentazione costituisce un ostacolo importante all'uso e al riutilizzo delle ontologie esistenti[19].
Importante nel processo di documentazione ontologica è l'annotazione dalle definizioni in linguaggio naturale e dai commenti nella versione formalizzata, svolto il ciclo di vita dell'ontologia.

1.3.10 Valutazione

La valutazione consiste nel verificare e validare le ontologie:

- la "*verifica*" è il processo tecnico che garantisce la correttezza di una ontologia, degli ambienti software associati e della documentazione rispetto ad un protocollo di riferimento, che deve avvenire durante ogni fase e fra le fasi del ciclo di vita;
- la "*validazione*" garantisce che l'ontologia, l'ambiente software e la documentazione corrispondano al sistema che essi suppongono di rappresentare.

In tale ambito, uno dei primi aspetti che devono essere assicurati è la "*coerenza*" dell'ontologia: un'ontologia dovrebbe essere internamente consistente, non ci devono essere parti che vanno in contrasto con altre parti.

"*Coerenza*" dovrebbe esserci negli assiomi e nelle definizioni non assiomatiche (documentazione in linguaggio naturale ed esempi).

Il processo di revisione che consiste nel rivedere le scelte fatte, dovrebbe essere fatto durante tutto il ciclo di vista e ogni modifica e decisione dovrebbe essere opportunamente documentata.

1.3.11 Classificazione

Diverse sono le categorie in cui possono essere classificate le Ontologie secondo differenti parametri, ma la distinzione tra le varie categorie di un'ontologia non è mai netta, poiché una stessa ontologia può appartenere a più categorie.

La classificazione più comune delle ontologie è fatta considerando il **livello di generalità** usato per la descrizione dei domini che permette di individuare le seguenti categorie:

- **"*Ontologie di livello elevato*"**: descrivono concetti molto generali o conoscenza di senso comune come spazio, tempo, oggetti, azioni.
- **"*Ontologie di dominio*"**: descrivono il vocabolario, le teorie e i principi elementari che governano un generico dominio.

- **"*Ontologie di compito*"**: descrivono il vocabolario relativo ad un generico compito o attività, dando una specializzazione dei termini introdotti nell'ontologia di livello elevato.
- **"*Ontologie applicazionali*"**: descrivono concetti relativi ad un dato dominio o compito e, spesso, sono ottenute mediante una specializzazione delle ontologie di dominio e delle ontologie di compito.

Le ontologie possono anche essere classificate in funzione della quantità e del tipo di strutture di concettualizzazione, nonché del soggetto della concettualizzazione[27] .
In particolare, la **quantità ed il tipo delle strutture della concettualizzazione** permettono una suddivisione delle ontologie in:

- **"*Ontologie terminologiche*"**: più che ontologie, sono veri e propri dizionari che specificano la terminologia usata per rappresentare la conoscenza nel dominio di interesse, senza rappresentare la semantica dei termini.
- **"*Ontologie di informazione*"**: specificano la struttura dei dati, fornendo significati per immagazzinare le osservazioni fondamentali concernenti istanze, ma non definiscono i concetti identificati.
- **"*Ontologie di modellazione della conoscenza*"**: specificano la concettualizzazione della conoscenza.

Invece, in relazione al **soggetto della concettualizzazione** distinguiamo:

- **"*Ontologie di applicazione*"**: descrivono i concetti necessari per modellare la conoscenza richiesta per una specifica applicazione, specializzando termini presi da ontologie generiche o estendendo la conoscenza generica e di dominio, in modo da rappresentare metodi e componenti specifici del compito. Le ontologie di applicazione riusano la conoscenza, modificandola per la specifica applicazione di interesse, mentre l'ontologia ottenuta non è riusabile per altri compiti.
- **"*Ontologie di dominio*"**: specificano concetti relativi ad un particolare dominio. Bisogna distinguere fra conoscenza che descrive situazioni in certi domini e le ontologie che specificano vincoli da applicare alla struttura e al contesto della conoscenza di dominio.
- **"Ontologie generiche"**: specificano concetti generici in diversi campi

- **"*Ontologie di rappresentazione*"**: spiegano le concettualizzazioni che portano ai formalismi usati nella rappresentazione della conoscenza, senza nessuna asserzione sul mondo. Le ontologie di rappresentazione descrivono le ontologie di dominio e le ontologie generiche attraverso i significati delle primitive.

Le ontologie possono differire anche per il modo in cui la conoscenza è espressa e, quindi, per il **livello di formalismo** utilizzato per esprimere i termini e i loro significati [1]. Vero è che il livello di formalismo è meglio descritto attraverso un continuo piuttosto che un insieme di classi, tuttavia, possono essere individuati quattro punti in questo continuo:

- **"*Ontologie altamente informali*"**: sono espresse in linguaggio naturale e la definizione dei termini potrebbe essere ambigua a causa dell'intrinseca ambiguità del linguaggio naturale;
- **"*Ontologie semi-informali*"**: sono espresse in una forma rigida e strutturata del linguaggio naturale, migliorando la chiarezza e riducendo le ambiguità.
- **"*Ontologie semi-formali*"**: sono espresse attraverso linguaggi artificiali formalmente definiti, come ad es. Ontolingua;
- **"*Ontologie rigorosamente formali*"**: sono ontologie i cui termini sono precisamente definiti con una semantica formale, teoremi e prove di proprietà.

Altra classificazione che può essere effettuata è in **funzione dell'espressività**, cioè delle informazioni che l'ontologia vuole esprimere[28]:

- **"*Vocabolari controllati*"**: sono liste finite di termini che rappresentano la più semplice nozione possibile di ontologia. Un tipico esempio è il catalogo che provvede solo ai termini con un'interpretazione non ambigua.
- **"*Glossari*"**: sono liste di termini e loro significati espressi mediante dichiarazioni in linguaggio naturale. Realizzati principalmente ad uso degli esseri umani, risultano spesso composti da dichiarazioni ambigue, che non possono essere usate da "agenti" automatici.
- **"*Thesauri*"**: aggiungono ai glossari la semantica che emerge dalle definizioni delle relazioni fra i termini (come le relazioni di sinonimia). Tipicamente, non forniscono

una struttura gerarchica esplicita, sebbene questa possa essere dedotta dalla specificazione dei termini.

- **"*Gerarchie informali Is-A*"**: sono ontologie in cui generalizzazione e specializzazione sono ottenuti sebbene non ci sia una gerarchia di tipo "*sotto-classe*" stretta. Includono diverse ontologie presenti sul Web.

  Nelle gerarchie che non sono strettamente di tipo "*is-a*" non sempre l'istanza di una specifica classe appartiene ad una più generica, perciò l'eredità in tal caso non può essere applicata.
- **"*Gerarchie formali Is-a*"**: sono ontologie in cui i concetti sono organizzati secondo una gerarchia stretta di sottoclassi ed è possibile applicare sempre il concetto di eredità.
- **"*Istanze formali*"**: sono ontologie che includono istanze di relazioni formali e rappresentano una naturale estensione di ontologie che obbligano ad una rigida struttura gerarchica.
- **"*Frames*"** (descrizione di proprietà di concetti): sono ontologie i cui concetti sono descritti in termini delle loro proprietà caratteristiche. L'inclusione di proprietà nella descrizione di concetti diviene più interessante quando può essere applicata l'ereditarietà, cosicché proprietà possano essere specificate per un concetto più generale ed essere ereditate.
- **"*Restrizioni di valori*"**: permettono di applicare restrizioni sui valori associati alle proprietà.
- **"*Restrizioni logiche generali*"**: solitamente sono scritte in un linguaggio ontologico molto espressivo che permette la specificazione di restrizioni nella logica del primo ordine su concetti e proprietà.

1.4 Ontologie semplici e complesse

Nel corso degli anni, sono state sviluppate diverse ontologie che possono essere distinte, in base alla complessità, in semplici o complicate.

La costruzione di un'ontologia complicata comporta costi molto elevati,[8]anche proibitivi per determinate applicazioni.

In tal caso la creazione di un'ontologia semplice potrebbe essere la soluzione più adatta.[29]

Un'**ontologia semplice** può essere utilizzata per supportare:

1. **l'interoperabilità**: entità differenti possono usare gli stessi termini appartenenti ad un vocabolario controllato;
2. **la risoluzione delle ambiguità**;
3. **la navigazione e l'organizzazione dei siti**: può essere definita una gerarchia di termini e legami fra vari siti e fra elementi dello stesso sito, in modo che l'utente possa visualizzare la gerarchia e fare le scelte su cosa visualizzare successivamente;
4. **il cosiddetto "scenario d'aspettativa"**: mediante interfacce utenti può essere data una realistica aspettativa del sito, in modo da permettere la valutazione delle caratteristiche del sito;
5. **l'estensione dei contenuti**: mediante argomenti d'alto livello sono definite delle strutture gerarchiche per estendere i contenuti.
6. **l'attività di ricerca**: da una data interrogazione utente si può supportare la ricerca estendendo con termini più specifici appartenenti a categorie della gerarchia.

Esempi di semplici ontologie sono:

- ***DMOZ*** [30] costituisce una grande ontologia semplice, con una struttura di 35000 editori volontari e più di 360000 classi nella tassonomia;
- ***Dublin Core Metadata Initiative (DCMI) Elements and Element Refinements*** [31]uno schema di metadati sviluppati principalmente per facilitare la scoperta di risorse [32]

[8] In termini di tempo, risorse e in senso economico

- ***UMLS*** [33] (Unified Medical Language System), sviluppata dalla libreria nazionale americana di medicina, costituisce una grande ontologia di terminologia medica.

Ontologie complesse possono essere utilizzate per applicazioni più potenti:

1. **controllo della consistenza**: avendo informazioni circa le proprietà e i valori che possono assumere, allora nelle applicazioni può essere fatto il controllo sui tipi;
2. **provvedere al completamento**: un'applicazione può ottenere una piccola quantità d'informazioni da un utente e mediante un'ontologia fornire il resto delle informazioni;
3. **supporto all'inter-operabilità**: l'utilizzo di un vocabolario controllato supporta l'interoperabilità poiché differenti applicazioni o utenti usano lo stesso insieme di termini con conseguente accordo sui significati da dare ai singoli termini;
4. **supporto al processo di validazione**[9]**e controllo per la verifica di testi e dati**.
5. **supportare la configurazione**: possono essere definite classi per descrivere quali componenti sono in un sistema;
6. **supportare la ricerca**.

Esempi di ontologie complesse:

- ***Ariadne Genomics ontology*** [34] è un'ontologia usata per formalizzare dati sulle celle proteine per permettere analisi al elaboratori: l'ontologia definisce le varie proteine, classifica loro in un albero tassonomico e definisce le relazioni semantiche fra loro[32].
- ***CYC*** [35] è una componente fondamentale del progetto CYC. Tale ontologia è stata creata basandosi su microteorie. Ogni microteoria cattura la conoscenza e il ragionamento richiesto per il particolare dominio, come spazio, tempo, causalità o "agenti".
- ***EL Ontology:*** [36] è un esempio di ontologia sviluppata per l'intelligenza artificiale e per il trattamento di linguaggio naturale. E' un'ontologia libera che include tipi, idee, fatti e eventi e anche un motore inferenziale capace di fare complesse inferenze in maniera efficiente.

[9] la validazione garantisce che l'ambiente che si sta esaminando rispettino determinate caratteristiche che si suppone debba avere

- ***Enterprise Ontology*** [37] è una collezione di termini e definizioni rilevanti per imprese d'affari, sviluppata nell'ambito del progetto Enterprise. I maggiori contenuti sono:
  1. Meta-ontologia e tempo: termini usati per definire i termini dell'ontologia (ad es. entità, relazioni, ruoli);
  2. Attività, piani, capacità e risorse: termini relativi ai procedimenti e alla pianificazione;
  3. Organizzazione: termini relativi a come l'organizzazione è strutturata;
  4. Strategia: termini relativi alla pianificazione ad alto livello;
  5. Marketing: termini relativi al marketing, alle vendite e ai servizi.

- ***STEP*** [38] (Standard for the Exchange of Product Model Data) è un tipo di ontologia semi informale usata per l'inter-operabilità e gli scambi di dati di prodotti fra differenti sistemi di elaboratori. L'ontologia STEP descrive i dati relativi ai prodotti attraverso il ciclo di vita del prodotto.
- ***WfMC*** [1] ***(Workflow Management Coalition***) è una terminologia standard contenente definizioni tecniche per termini usati nella specificazione WfMC. Per ogni termine, ci sono una definizione, una discussione dell'uso e un insieme di possibili sinonimi. Questo serve per l'interoperabilità fra differenti sistemi.
- ***WordNet*** [39] è un'ontologia linguistica formata da termini, detti "synset", raggruppati in insiemi semanticamente equivalenti, ognuno assegnato ad una categoria lessicale (nome, verbo, aggettivo, etc.). Ogni synset rappresenta un particolare significato di una parola inglese ed è solitamente espresso come una combinazione unica di sinonimi. Esistono anche supporti multi lingue, come EuroWordNet.

Capitolo 2

Linguaggi Ontologici

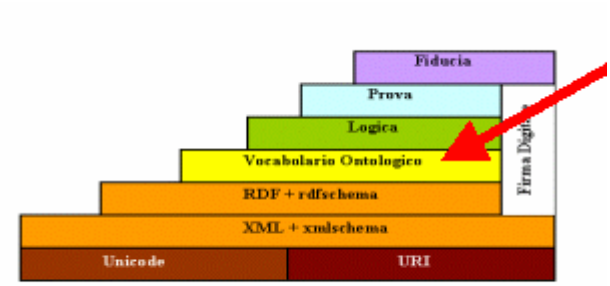

Figura 2.1: Posizione dei Linguaggi Ontologici nell'architettura del Web Semantico

Le ontologie hanno un ruolo fondamentale nella visione del Web Semantico, perché sono una rappresentazione esplicita e condivisa delle conoscenze di un dato dominio, espressa in un modo trattabile dagli elaboratori.

Alla luce di ciò, è importante definire dei linguaggi che permettano di rappresentare le informazioni semantiche contenute nelle ontologie, abilitandone lo scambio in un ambiente distribuito come quello Web.

L'ontologia, quindi, deve essere codificata per mezzo di un opportuno linguaggio capace di esprimere i concetti del dominio d'applicazione.

Nel corso degli ultimi anni sono stati sviluppati diversi linguaggi "ontologici", alcuni poco utilizzati, altri sono diventati importanti per l'utilizzo delle ontologie nelle applicazioni Web.

Le caratteristiche che un linguaggio deve possedere per essere adatto allo sviluppo delle ontologie sono[46]:

- ***Essere sufficientemente intuitivo***: un linguaggio intuitivo è più facile da apprendere e da mantenere per l'autore ontologico; tale caratteristica è collegata alla complessità del linguaggio, poiché risulta difficile realizzare linguaggi complessi ed intuitivi allo stesso tempo.
- ***Sufficiente potere espressivo***: si riferisce alla possibilità di avere un certo numero di primitive di modellazione di base che permettano di esprimere le ontologie in modo

dettagliato; è una delle caratteristiche più importanti, perché ha impatti sul livello di rappresentazione delle ontologie, con conseguenze sugli sforzi richiesti all'autore per creare l'ontologia. Infatti, la mancanza d'espressività non permette la realizzazione di ontologie complesse, che richiedono un certo livello di dettaglio, e complica il lavoro concettuale del progettista che in funzione delle primitive a disposizione, deve rielaborare le parti altrimenti non rappresentabili.

- ***Sintassi ben definita***: è necessaria per avere informazioni trattabili dagli elaboratori e per favorire il processo d'apprendimento e l'uso del linguaggio da parte degli autori ontologici[10]. Una sintassi ben definita permette di realizzare compilatori che possano controllare in modo puntuale il formato, l'ordine delle strutture utilizzate in una data ontologia, per segnalare e permettere la correzione di eventuali errori commessi dall'autore; ciò costituisce un vantaggio per l'autore che in questo modo ha minori possibilità d'errore nell'utilizzo del linguaggio ontologico (avendo a disposizione opportuni strumenti che automaticamente controllano la correttezza sintattica dell'ontologia).
- **Semantica formale ben definita con determinate proprietà di ragionamento**: la semantica permette di definire "*precisamente*" le strutture utilizzate nel linguaggio, dove il termine "*precisamente*" intende che i significati non sono lasciati all'intuizione soggettiva delle singole persone o applicazioni; quindi la semantica permette di interpretare il simbolo sintattico rispettando il loro significato inteso. Un uso della semantica formale permette agli esseri umani di ragionare sulla conoscenza e nel campo ontologico permette di definire relazioni di appartenenza a classi, equivalenza di classi; consistenza, classificazione.

  La semantica è un supporto al ragionamento e la derivazione di conoscenza può essere fatta manualmente o automaticamente. Il supporto al ragionamento è importante perché:

  a) controlla la consistenza delle ontologie e della conoscenza;
  b) controlla le relazioni non individuate fra classi;
  c) classifica automaticamente le istanze in classi;

[10] La necessità di sintassi bene definita per l'apprendimento è sempre minore, perché ormai spesso le ontologie sono realizzate mediante appositi applicativi per la progettazione ontologica invece di utilizzare i linguaggi ontologici.

Normalmente la semantica formale è realizzata facendo corrispondere il linguaggio ontologico ad un formalismo logico conosciuto e usando ragionatori automatici che esistono per questi formalismi; ad esempio OWL usa la logica descrittiva e si appoggia a ragionatori automatici come FaCT e RACER.

- ***Giusto grado di complessità***: un linguaggio troppo semplice, ha un potere espressivo ridotto, quindi, potrebbe non soddisfare le necessità che si possono presentare nel campo ontologico; invece un linguaggio troppo complesso può perdere di vista il potere di rappresentazione e le necessità di ragionamento; inoltre, un'elevata complessità potrebbe limitare il successo e diffusione del linguaggio, poiché potrebbe risultare difficile da usare e apprendere.
- ***Assicurare la completezza, correttezza ed efficienza***: un linguaggio per fra fronte alle esigenze che ci sono nel campo ontologico deve essere completo; deve essere corretto per supportare il progettista nell'utilizzo del linguaggio stesso, permettendo di realizzare strumenti che segnalino eventuali errori sintattici (ad esempio errori di formato) e per evitare che le primitive possano portare ad eventuali errori "concettuali" per mancanza di correttezza e completezza del linguaggio; infine deve essere efficiente per utilizzare minori risorse di elaborazione e facilitare al contempo il lavoro del progettista.
- ***Collegamento con i linguaggi Web (XML, RDF)***: tali linguaggi debbono essere utilizzati per applicazioni Web, quindi necessità di un collegamento con i linguaggi già sviluppati in quest'ambito, per facilitare lo scambio delle ontologie fra applicazioni, assicurando l'interoperabilità nel Web Semantico.

Durante gli ultimi anni sono stati sviluppati diversi linguaggi per l'ontologia: alcuni basati sulla sintassi ***XML***, come ***XOL*** (*Ontology Exchange Language*), ***SHOE*** (*Simple HTML Ontology Extension*) e ***OML*** (*Ontology Markup Language*), oppure ***RDF*** e ***RDF Schema*** che sono linguaggi creati dal ***W3C***. ***OIL*** (*Ontology Inference Layer*), ***DAML+OIL*** e ***OWL*** sono linguaggi costruiti su ***RDF(S)*** per sfruttare le caratteristiche di questi linguaggi.

2.1 Principali linguaggi ontologici.

2.1.1 OIL

OIL (*Ontology Interchange Language*)[40] è un linguaggio sviluppato nel progetto ***On-To-Knowledge IST***, che possiede tre delle caratteristiche fondamentali per un linguaggio ontologico:

- Semantica formale e supporto al ragionamento efficiente, ottenuta dalla Logica Descrittiva (*Description Logic*);[11]
- Primitive di modellazione;
- Proposta standard per notazioni sintattiche della comunità del Web (come ***XML Schema*** e ***RDF Schema***).

OIL eredita la semantica formale ed il supporto al ragionamento dalla Logica Descrittiva, al punto che si può pensare di inquadrare ***OIL*** all'interno del livello logico, oltre che ontologico [41].

La sintassi è ben definita in ***XML***, basata sulla definizione del tipo documento e sulla definizione ***XML Schema***, mentre ***RDF*** e ***RDF(S)*** forniscono una base per la definizione delle istanze e ontologie.

In particolare, ***RDF(S)*** introduce una sintassi standardizzata e un insieme di primitive di modellazione standard (relazioni "instance-of" e "subclass-of").

L'utilizzo dei ***namespace*** permette di utilizzare in un documento ***RDF*** elementi provenienti da diverse ontologie, in modo da fare affermazioni con un linguaggio a proposito di un oggetto definito in termini di un altro linguaggio.

Sebbene la stessa tecnica possa essere applicata a qualsiasi linguaggio per la rappresentazione della conoscenza, il fatto che ***OIL*** supporti entrambe le primitive di modellazione di base di ***RDF(S)*** ("***rdfs:subClassOf***" e "***rdfs:subPropertyOf***"), lo rende facilmente integrabile all'interno del lavoro svolto dal ***W3C***.

Ogni ontologia scritta con ***RDF(S)***è utilizzabile da una qualsiasi ontologia scritta con una sua estensione, come ***OIL***, mentre il processo inverso non è ottenibile facilmente.[12]

[11] La *Logica Descrittiva* descrive la conoscenza in termini di concetti e vincoli, in modo da utilizzarla per derivare automaticamente classificazioni tassonomiche.

L'uso del vocabolario solo di ***RDF(S)***, consentirebbe di avere la massima comapitibilità possibile con applicazioni esistenti, però applicazioni capaci di riconoscere il potere espressivo di ***OIL*** (e non solo di ***RDF(S)***) sarebbero in grado di migliorare i servizi che forniscono.

La proposta di ***OIL*** prevede di utilizzare:

1. ***RDF(S)*** per descrivere le primitive di modellazione;
2. lo schema risultante, contenente la meta-ontologia di ***OIL***, per descrivere una specifica ontologia in ***OIL***;
3. gli schemi dei due punti precedenti per descrivere istanze dell'ontologia ***OIL*** .

OIL è composto da quattro sottolinguaggi organizzati in strati, in modo tale che il sottolinguaggio dello strato successivo ingloba il sottolinguaggio dello strato precedente: ogni strato addizionale aggiunge funzionalità e complessità al precedente.

Gli strati creati in ***OIL*** sono:

- **Core OIL** raggruppa le primitive ***OIL*** che hanno corrispondenza diretta nelle primitive ***RDF(S)***; tale linguaggio coincide con ***RDF(S)***, con l'eccezione della caratteristica di reificazione, sicché ogni agente per ***RDF(S)*** può elaborare una ontologia scritta in ***Core OIL***.
- **Standard OIL** è il modello completo, che usa più primitive di quelle definite in ***RDF(S)***: in particolare permette di determinare le primitive principali di modellizzazione che provvedono ad un adeguato potere espressivo, così da specificare in maniera puntuale la semantica, rendendo di fatto attuabile l'inferenza;
- **Instance OIL** aggiunge esempi relativi a concetti e regole del modello precedente;
- **Heavy OIL** è lo strato per le future estensioni di ***OIL***, che includerà le capacità addizionali di rappresentazione e ragionamento.

Tale architettura a strati ha tre principali vantaggi per le applicazioni che lo utilizzano:

- è utilizzato il linguaggio più adatto, senza la necessità di lavorare con un linguaggio che offre una maggiore espressività e complessità del necessario;
- se non necessario si può lavorare ad un basso livello di complessità;

---

[12] Un linguaggio basato su *RDF(S)*, come *OIL*, di solito contiene nuovi aspetti e quindi un nuovo vocabolario che un'applicazione basata su *RDF(S)* non riconosce, quindi la compatibilità totale non è possibile.

- nel caso in cui è necessario lavorare ad un alto livello di complessità, sono comprensibili le ontologie espresse in un linguaggio ontologico più semplice.

OIL lavora principalmente su due meta-livelli:

- il contenitore di ontologia: che è l'insieme di informazioni a proposito dell'ontologia, come il titolo, l'autore, la descrizione, relazioni verso altre ontologie e così via;
- la definizione dell'ontologia stessa

I costrutti principali per la definizione dell'ontologia prevedono:

- definizione di classi;
- definizione di slot, detti anche ruoli o attributi;
- espressioni di classe e vincoli sugli slot.

Le principali primitive di modellazione sono le classi (o frame) e le proprietà associate alle classi chiamate attributi.

Le relazioni possono essere definite come attributo di una classe, ma anche come entità indipendenti di un certo dominio.

La definizione delle gerarchie di classi è l'elemento principale di ogni linguaggio ontologico.

Invece di usare tipi singoli nelle espressioni, possono essere combinate classi in espressioni logiche indicanti intersezioni, unione e complementi.

Gli Slot costituiscono relazioni fra classi e possono essere dichiarate, insieme con assiomi logici che dichiarano sia che sono funzionali (ad es.hanno al più un valore), transitivi, simmetrici.

Vincoli sugli intervalli possono essere definiti nella dichiarazioni degli Slot: ad esempio si può definire il numero di valori distinti che uno slot può avere, vincoli sui valori o sul tipo di valori.

I vincoli sul tipo di valori, comporta che i valori della proprietà sono di un dato tipo, mentre quelli sui valori implicano che lo slot ha al più quei valori specificati.

Una ontologia ***OIL*** è una semantica formale ottenuta trovando la corrispondenza di ogni classe in un insieme di oggetti e ogni slot in un insieme di coppie di oggetti, in base ai vincoli specificati dalla definizione delle classi e dei slot.[42]

OIL ha alcune limitazioni:

- Presenta una espressività limitata del second'ordine: molti linguaggi esistenti per la definizione di ontologie (come ***KIF***) includono qualche meccanismo di reificazione, che permette il trattamento di proposizioni del linguaggio come oggetti, rendendo possibile esprimere proposizioni di proposizioni. Non è richiesta l'intera espressività della logica del second'ordine, poiché renderebbe intrattabile qualsiasi problema di calcolo.
- Impossibilità di usare istanze per definire le classi: ***OIL*** non permette di definire le classi enumerando le istanze.
- Mancanza del "*ragionamento di riferimento*": sebbene fornisca un meccanismo per ereditare valori dalle superclassi, tali valori non possono essere sovrascritti senza che ne nasca una inconsistenza nella definizioni delle classi.
- Mancanza di proprietà algebriche quali riflessività, antisimmetria, asimmetria, ordine parziale e totale.
- *Modularizzazione*: come ***XML***, ***OIL*** utilizza i namespace, ma non fornisce meccanismi più sofisticati come la possibilità di rinominare, ristrutturare e ridefinire ontologie importate.
- Mancanza di domini concreti: ***OIL*** attualmente non supporta domini come interi, stringhe ecc.
- *Regole e assiomi*: possono essere espresse solo un numero fisso di proprietà algebriche (transitività, simmetria, inversione) e non c'è nessun meccanismo per la descrizione di assiomi arbitrari che devono valere per gli elementi dell'ontologia

2.1.2 DAML+OIL

DAML+OIL[43] (*DARPA Agent Markup Language plus Ontology Inference Layer*) è un linguaggio standard per la rappresentazione ontologica che nasce dalla fusione di ***DAML*** e ***OIL***.

L'obiettivo principale di questo linguaggio, è di integrare la logica descrittiva di ***OIL*** in ***DAML*** e di modificare ***DAML*** per rispondere alle richieste della comunità scientifica.

In particolare, sono stati aggiunti operatori per definire classi, relazioni e tipi, è gestita la cardinalità e i vincoli nei domini ed intervalli; è possibile applicare l'unione, disgiunzione, inversione e la regola transitiva[44].

La semantica usata risulta chiara grazie all'uso di una semantica aggiornata della logica del primo-ordine.

2.1.2.1 Tipi di dati

DAML+OIL supporta l'intervallo dei tipi di dati definiti in ***XML Schema***: sia le primitive di tipi di dati (come stringhe, decimali), sia tipi derivati complessi.
Tipi sono definiti, quindi, mediante il costrutto "***xsd:simpleType***".
Ad esempio volendo definire un tipo numerico per i numeri della lotteria:

```
<xsd:simpleType name="NumLotteria">
        <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxinclusive value="90"/>
        </xsd:restriction>
</xsd:simpleType>
```

2.1.2.2 Elemento classe

Una ontologia è costituta da descrizioni di oggetti, per cui risulta utile definire dei tipi base di oggetti, mediante l'elemento "*Classe*".
Una "*Classe*" rappresenta il sottoinsieme, del dominio di interesse, costituito da tutti gli oggetti che sono di quel tipo.
La definizione dei tipi di oggetti, viene realizzata in **DAML+OIL** mediante la relazione "***daml:Class"***.
Ad esempio, consideriamo le tre classi "*Balena*", "*Delfino*", "*Mammifero*" e "*Animale-Acquatico*". Risulta evidente che le classi "*Balena*" e "*Delfino*" sono dei sotto-insiemi della classe "*Mammifero*", in quanto entrambi sono dei tipi di mammiferi; inoltre alla classe "*Animale-Acquatico*" appartiene (è una sottoclasse) la classe "*Mammifero*".
Quindi in **DAML+OIL** possiamo dire definire le seguenti classi:

```
<daml:Class rdf:ID="Balena">
        <rdfs:subClassOf rdf:resourse="#Mammifero"/>
</daml:Class>
```

```
<daml:Class rdf:ID=”Delfino”>
        <rdfs:subClassOf rdf:resourse=”# Mammifero”/>
</daml:Class>

<daml:Class rdf:ID=”mammifero”>
        <rdfs:subClassOf rdf:resourse=”# Animale-Acquatico”/>
</daml:Class>

<daml:Class rdf:ID=”Balena”>
        <rdfs:subClassOf rdf:resourse=”#Mammifero”/>
        <rdfs:subClassOf rdf:resourse=”# Animale-Acquatico”/>
</daml:Class>
```

Negli esempi precedenti si nota l’utilizzo del costrutto “***rdfs:subClassOf”***, che permette di definire che la classe soggetto è una sottoclasse della classe oggetto: ad esempio, la classe “*Delfino*” è una “*sotto-classe*” della classe “*Mammifero*”.
L’ultima definizione dell’esempio precedente evidenzia la possibilità che una stessa classe possa essere “*sotto-classe*” di più classi.
Altra possibilità del linguaggio ***DAML+OIL*** è che si può definire che una classe è equivalente ad una espressione logica sopra il nome della classe.
Ad esempio, possiamo definire che la classe “Animale Acquatico” è data dalla disgiunzione delle classi “*Mammiferi*” e “*Pesci*”:

```
<daml:Class rdf:about =”#Animale-Acquatico”>
        <daml:disjointUnionOf rdf:parseType=”daml:collection”>
                <daml:Class rdf:about=”#Mammiferi”/>
                <daml:Class rdf:about=”#Non-mammiferi”/>
        </daml:disjointUnionOf >
</daml:Class>
```

L'elemento "***daml:disjointUnionOf***" permette di asserire che due classi risultano disgiunte; nell'esempio precedente la classe "*Animale-Acquatico* " non può contemporaneamente appartenere alla classe "*Mammifero*" e alla classe "*Non-mammifero*", quindi, le due classi risultano disgiunte.
Possono essere definite altre relazioni fra classi come equivalente ad una espressione booleana su altre classi tramite :

- "***daml:intersectionOf***": intersezione;
- "***daml:unionOf***" :unione;
- "***daml:complementOf***" :complemento di;
- "***daml:oneOf***": permette di definire classi enumerando i suoi elementi.

2.1.2.3 Proprietà

Le proprietà relazionano due elementi e ***DAML+OIL*** definisce due tipi di proprietà:

- "***daml:ObjectProperty***": relaziona membri di differenti classi;
- "***daml:DatatypeProperty***": relaziona l'elemento di una classe ai valori consentiti di un tipo di dati.

In particolare ha un unico nome è può avere un intervallo ***RDF(S)*** e restrizioni al dominio:

```
<daml:ObjectProperty rdf:ID="Parenti">
        <rdfs:domain rdf:resource="#Animale"/>
        <rdfs:range rdf:resource="#Animale"/>
</daml:ObjectProperty>
```

La prima evoluzione di ***DAML+OIL*** rispetto a ***RDF(S)*** è la possibilità di definire una relazione di equivalenza o di inverso rispetto ad un'altra relazione.
Come in ***RDF(S)*** è possibile definire gerarchie di relazioni usando l'operatore "***rdfs:subpropertyOf***".
Altre relazioni:

- "***daml:UniqueProperty***": indica che esiste una sola risorsa che soddisfa la proprietà;
- "***daml:transitiveProperty***": definisce la transitività di una relazione.

2.1.2.4 Individui

DAML+OIL definisce singoli individui utilizzando la notazione ***RDF*** "***rdf:ID***", che permette di specificare valori e proprietà relative ai tipi di dati.
Inoltre, estende il vocabolario base ***RDF*** per descrivere risorse attraverso le proprietà:

- "***daml:differentIndividualFrom***": specifica che due individui sono differenti;
- "***daml:sameIndividual***": specifica che due individui sono gli stessi malgrado non hanno lo stesso identificativo.

2.1.2.5 Vincoli sulle proprietà

Le proprietà sono utilizzate per mettere in relazione fra loro individui e permettono di introdurre le proprietà associate.
Per descrivere una proprietà viene utilizzato il costrutto "***daml:ObjectProperty***. Ad esempio volendo descrivere la proprietà che un uomo è "*Europeo*" si ha:

<**daml:ObjectProperty** rdf:ID="èEuropeo">

I tipi di individui che possono soddisfare una data proprietà, possono essere limitati mediante la definizione di un intervallo di valori possibili per la proprietà considerata.
Per fare ciò si può definire una classe di tutte le possibili origini valide e utilizzare il costrutto "***rdfs:range***":

<**rdfs:range** rdf:resourse="#Origini"/>

mentre il costrutto "***rdfs:domain***" permette di definire il dominio della proprietà:

<**rdfs:domain** rdf:resourse="#UOMO"/>

Le classi, come detto, definiscono proprietà comuni dei suoi membri e ***DAML+OIL*** permette di definire due tipi di vincoli relativi alle proprietà di "*oggetti*" appartenenti a delle classi:

- di tipo;
- numeriche.

Usando i vincoli, si può definire che il padre di ogni persona è un uomo e che esiste un solo padre per ogni persona:

```
<daml:Class rdf:ID="Persona">
        <rdfs:subClassOf rdf:resource="#Animale"/>
        <rdfs:subClassOf>
                <daml:Restriction>
                        <daml:onProperty rdf:resource="#Padre"/>
                        <daml:toClass rdf:resource=#Uomo/>
                </daml:Restriction>
                <daml:Restriction daml:cardinality="1">
                        <daml:onProperty rdf:resource="#Padre"/>
                </daml:Restriction>
        </rdfs:subClassOf>
</daml:Class>
```

In particolare per i "*vincoli di tipo*" ci sono:

- "***daml:toClass***": indica di che tipo deve essere il membro di una classe;
- "***daml:hasClass***": indica che ogni elemento di una classe è relaziona con al più un oggetto di un certo tipo;
- "***daml:hasValue***": indica che ogni oggetto della classe è relazionato con uno specifico oggetto.

Per i "*vincoli numerici*", ci sono le seguenti relazioni:

- "***daml:minCardinality***": definisce il confine inferiore;
- "***daml:maxCardinality***": definisce il confine superiore
- "***daml:cardinality***": definisce i valori esatti

Più vincoli possono essere applicati alla stessa relazione per ridurre la dimensione delle specificazioni, in particolare possono essere definite le seguenti restrizioni che permettono di combinare restrizioni di tipo e numeriche:

- ” ***daml:minCardinalityQ***”: definisce il confine superiore;
- ”***daml:maxCardinalityQ***”: definisce il confine superiore;
- ”***daml:cardinalityQ***”: definisce i valori esatti.

Utilizzando le restrizioni, la classe definita nel precedente esempio può essere riscritta come:

```
<daml:Class rdf:ID="Persona">
        <rdfs:subClassOf rdf:resource="#Animale"/>
        <rdfs:subClassOf>
                <daml:Restriction daml:cardinalityQ="1">
                        <daml:onProperty rdf:resource="#Padre"/>
                        <daml:hasClassQ rdf:resource=#Uomo/>
                </daml:Restriction>
        </rdfs:subClassOf>
</daml:Class>
```

2.1.3 OWL

OWL (*Web Ontology Language*)è l’ultimo dei linguaggi ontologici in ordine temporale, realizzato dal “**W*3C Web Ontology Working Group***” (*WebOnt*)[13],per descrivere le ontologie Web e le basi di conoscenza associate.
In fase di progettazione si è deciso di basarsi su di un altro linguaggio ontologico Web, ***DAML+OIL***, cercando di superare i ben conosciuti problemi di tale linguaggio come l’esiguità dell’espressività semantica.
Per definire il linguaggio ***OWL*** sono stati utilizzati ***RDF*** e ***RDF(S)*** e, per la definizione della sintassi ***RDF*** e ***XML***.

[13] http://www.w3.org/2001/sw/WebOnt/

Dato che la sintassi di questi due linguaggi non è molto leggibile, allora si è fatto ricorso ad altre forme sintattiche:

- una sintassi basata su ***XML*** che non segue le convenzioni di ***RDF*** , per cui è facilmente leggibile dall'uomo;
- una sintassi astratta più compatta e leggibile rispetto alla sintassi di ***XML*** e di ***RDF/XML***, che è usata nei documenti di specificazione del linguaggio.
- una sintassi grafica basata sulla convenzione del linguaggio ***UML***, che è facilmente comprensibile vista l'enorme diffusione di ***UML***.

OWL, come ***DAML+OIL***, ha classi (e sottoclassi), proprietà (e sottoproprietà), vincoli sulle proprietà e classi individuali.

Inoltre, permette di trattare le informazioni sulle classi e sul tipo di dati, definisce costrutti per le classi come "***owl:subClassOf***" e "***owl:disjointWith***", permette combinazioni booleane di classi con costrutti come: "***owl:intersectionOf***", "***owl:unionOf***","***owl:complementOf*** "e tratta le classi enumerative (liste).

In ***OWL*** ha inoltre forme quantificate come il quantificatore universale "***owl:allValuesFrom***".[45]

I sostanziali cambiamenti rispetto a ***DAML+OIL***, sono lo spostamento delle restrizioni numeriche qualificate, l'abilità di vedere la simmetria delle proprietà, la rimozione di alcuni costrutti poco diffusi.

Ci sono anche altri cambiamenti minori, come quelli relativi ai nomi di vari costrutti, che sono stati cambiati per indicare potenziali cambiamenti alla sintassi per ***OWL***, ma in generale sono stati mantenuti i nomi previsti in ***DAML+OIL***.

2.1.3.1 Intestazione

I documenti ***OWL*** sono chiamati "*ontologie* ***OWL***" e sono di tipo ***RDF***, in cui l'elemento radice è un elemento "***rdf:RDF***" che specifica un certo numero di ***namespace***. Ad esempio:

```
<rdf:RDF
    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/XLMSchema#">

Il documento ***OWL*** può iniziare con una collezione di asserzioni, raggruppate sotto elementi "***owl:Ontology***", contenenti commenti, la versione di controllo e le inclusioni di altre ontologie tramite l'elemento "***owl:imports***".
Per "***owl:imports***" vale la proprietà transitiva, quindi, se l'ontologia importa l'ontologia B e questa a sua volta importa l'ontologia C, allora A importa l'ontologia C.
In ***OWL***, sono stati aggiunti degli elementi che permettono di effettuare una gestione delle diverse versioni delle ontologie e la definizione di specificatori di compatibilità e incompatibilità fra le versioni.
Fra gli elementi ***OWL*** per la definizione delle versioni:

- "***owl:priorVersion***": indica la versione corrente dell'ontologia(fa parte delle informazioni di testata);
- "***owl:versionInfo***": contiene informazioni circa l'attuale versione in formato stringa;
- "***owl:backwardCompatibleWith***":l'ontologia indicata nell'elemento è una versione precedente e compatibile rispetto a quella che contiene l'elemento; quindi, gli identificatori della versione precedente hanno la stessa interpretazione nella nuova versione.
- "***owl:incompatibleWith***": l'ontologia che contiene l'elemento è l'ultima versione dell'ontologia indicata, ma non è compatibile all'indietro con questa ontologia.

2.1.3.2 Tipi di dati

Sebbene ***XML Schema*** provvede ad un meccanismo per costruire tipi di dato definiti dall'utente, tali tipi di dati derivati non possono essere tutti usati su ***OWL***.
Il documento di riferimento elenca tutti i tipi di dati che possono essere usati con ***XML*** Schema e include i più frequenti tipi usati come interi, booleani, tempo e data.

2.1.3.3 Elemento classe

La definizione delle classi avviene mediante l'elemento "***owl:Class***".

Ad esempio la definizione di una classe '*macchina*' è ottenuta nel seguente modo:

```
<owl:Class rdf:ID="macchina">
        <rdfs:subClassOf rdf:resource="#parco_macchine"/>
</owl:Class>
```

In ***OWL*** sono state predefinite due classi basi:

- "***owl:Thing***" è la classe più generale che contiene tutte le altre;
- "***owl:Nothing***" è la classe vuota.

Inoltre, possono essere definite relazioni fra le classi: l'equivalenza è definita attraverso "***owl:equivalentClass***", mentre la disgiunzione di elementi di una classe da quelli di altre classi attraverso "***owl:disjointWith***" .

La definizione delle proprietà di equivalenza può essere fatto nella definizione della classe, oppure è possibile riferirsi alla classe attraverso "***rdf:about***".

Ad esempio possiamo definire che la classe '*macchina*' è disgiunta dalla classe '*nave*' ed '*aereo*':

```
<owl:Class rdf:about="macchina">
        <owl:disjointWith rdf:resource="#nave"/>
        <owl:disjointWith rdf:resource="#aereo"/>
</owl:Class>
```

e che la classe '*macchina'* è equivalete alla classe "veicolo a motore con quattro ruote"

```
<owl:Class rdf:ID="macchina">
        <owl:equivalentClass rdf:resource="#VeicoloMotoreQuattroRuote"/>
</owl:Class>
```

La relazione "***rdfs:subClassOf***" permette di definire delle classi come sottoclassi di altre, in modo da realizzare il concetto di eredità di proprietà.

In tal modo si realizza la sussunzione di una gerarchia con parenti multipli per ogni classe figlio.

Possono essere definite combinazioni booleane di classi (unione, intersezione, complemento) mediante:

- "***owl:unionOf***"
- "***owl:complementOf***"
- "***owl:disjointWith***"
- "***owl:intersectionOf***"

2.1.3.4 Relazioni

In ***OWL*** ci sono due tipi di relazioni:

- "**Relazioni oggetto**" che collegano oggetti ad altri oggetti;
- "**Relazioni dei tipi di dati**" che relazionano oggetti a valori di tipi di dati.

OWL non ha nessun tipo di dato predefinito, ma permette di utilizzare i tipi di dati di ***XML Schema***.

```
<owl:ObjectProperty rdf:ID="éguidata">
        <owl:domain rdf:resource="#macchina"/>
        <owl:range rdf:resource="#autista"/>
</owl:ObjectProperty>
```

Normalmente gli utenti in ***XML Schema*** definiscono dei tipi di dati che sono usati (importati) nell'ontologia ***OWL***.

OWL permette di considerare le "proprietà inverse" attraverso "***owl:inverseOf***".

L'equivalenza di proprietà possono essere definite attraverso "***owl:equivalentProperty***".

```
<owl:ObjectProperty rdf:ID="guidare">
        <rdfs:range rdf:resource="#macchina"/>
        <rdfs:domain rdf:resource="#autista"/>
        <owl:inverseOf rdf:resource="#èGuidataDa"/>
</owl:ObjectProperty>
```

2.1.3.5 Vincoli sulle proprietà

La relazione "***rdfs:subClassOf***"definisce una relazione di tipo sottoclasse, in modo che ogni istanza della sottoclasse sia un'istanza della classe.
Inoltre, è possibile definire che una data classe soddisfa determinate condizioni e, tutte le istanze di quella classe soddisferanno quelle condizioni: la relazione "***owl:allValuesFrom***"specifica la classe di valori che la proprietà attraverso "***owl:onProperty***" può avere.
Nell'esempio seguente si ottiene che solo chi abbia la '*patente*' possa essere '*autista di automobili*':

```
<owl:Class rdf:about="#AutistaDiAutomobili">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#DeveAvere"/>
                        <owl:allValuesFrom rdf:resource="#Patente"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

La relazione "***owl:hasValue***" specifica il valore che deve avere la proprietà specificata attraverso "***owl:onProperty***".
L'elemento "***owl:Restriction***" permette di definire dei vincoli e in generale contiene un elemento "***owl:onProperty***" e uno o più dichiarazioni di vincolo.
Un tipo di dichiarazione di vincolo è quella che definisce il tipo di "*valori che la proprietà può avere*" e ciò si può ottenere mediante:

- "***owl:allValuesFrom*** "
- "***owl:hasValue***"
- "***owl:someValuesFrom***"

Altri tipi di vincoli sono quelli "*di cardinalità*", che si ottengono mediante i costrutti: "***owl:minCardinality***", "***owl:maxCardinality***" e "***owl:Cardinality***".

Nell'esempio, ogni '*macchina*' deve avere almeno una '*persona che ha la patente*', inoltre è stato specificato che '1' deve essere interpretato come un intero non negativo e che si utilizza la dichiarazione ***namespace*** xsd definita nell'elemento testata del documento ***XML Schema***:

```
<owl:Class rdf:about="#macchina">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#DeveAvere"/>
                        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
                                1
                        </owl:minCardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Altre proprietà che possono essere definite direttamente :

- "***owl:TransitiveProperty***" proprietà transitiva;
- "***owl:SymmetricProperty***" proprietà simmetrica;
- "***owl:FunctionalProperty***" proprietà che ha al più un unico valore per ogni oggetto;
- "***owl:InverseFunctionalProperty***" proprietà per cui due differenti oggetti non possono avere lo stesso valore.

2.1.3.6 Enumerazione

La definizione di elementi enumerativi è possibile mediante l'elemento "***owl:oneOf***", usato per definire una classe ottenuta elencando tutti i suoi elementi.
Nell'esempio seguente si ha l'enumerazione di tutti i giorni della settimana:

```
<owl:oneOf rdf:parseType="Giorni">
        <owl:Thing rdf:about="#Lunedì"/>
        <owl:Thing rdf:about="#Martedi"/>
        <owl:Thing rdf:about="#Mercoledi"/>
```

```
        <owl:Thing rdf:about="#Giovedi"/>
        <owl:Thing rdf:about="#Venerdi"/>
        <owl:Thing rdf:about="#Sabato"/>
        <owl:Thing rdf:about="#Domenica"/>
</owl:oneOf>
```

2.1.3.7 Strati del linguaggio

Anche ***OWL*** si compone di tre linguaggi a complessità crescente, in modo che ogni utilizzatore possa utilizzare il linguaggio di cui necessità, senza la necessità di sfruttare tutte le potenzialità del linguaggio ***OWL***.
I linguaggi di livello superiore contengono quelli di più basso livello, così si può affermare che estendono i linguaggi di più basso livello.

OWL Lite è il linguaggio più semplice, fra quelli proposti per ***OWL*** e, soddisfa le necessità elementari degli utenti di una classificazione gerarchica, con semplici caratteristiche di vincolo.
In tale linguaggio non è prevista la definizione degli operatori unione e negazione, inoltre la cardinalità risulta ristretta (0/1).
Inoltre, è possibile definire un'ontologia di classi, proprietà e istanze (individui) di queste classi e proprietà.
OWL utilizza la relazione "***rdfs:subClassOf*** "per definire delle classi come sottoclassi di altre,che così ereditano proprietà da classi parenti.
Le proprietà possono essere definite usando la proprietà:

- "***owl:objectProperty***" assegna relazioni fra elementi di classi distinte;
- "***owl:datatypeProperty***" assegna relazioni fra elementi classe e tipi di dati ***XML***
- "***owl:subproperty***","***owl:domain***" e, "***owl:range***" un dominio di una data proprietà è la classe per cui il primo argomento della proprietà è specificato; un range di una data proprietà è la classe per cui il secondo argomento è specificato.

Tale versione semplifica di ***OWL***, risulta facile da comprendere e da implementare, di contro lo svantaggio è che presenta una espressività limitata.

OWL DL (Description Logic), nasce dall'estenzione di ***OWL Lite*** e, porta al massimo l'espressività senza perdere la completezza computazionale.
Include il vocabolario completo ***OWL***, interpretato sotto un numero di semplici vincoli.
In particolare:

- permette vincoli sulla cardinalità, non più limitata a 0 o 1.
- c'è la separazione di tipo: gli identificatori di classe non possono simultaneamente essere proprietà o individui ed in maniera analoga, proprietà non possono essere individui.

Possono essere definite classi basate su specifici valori di proprietà usando il costrutto "***owl:hasValue***" e create espressioni di tipo classe usando combinatori booleani (insieme di operatori) come "***owl:unionOf***","***owl:intersectionOf***" e "***owl:complementOf***".
Inoltre, è possibile enumerare classi mediante il costrutto "***owl:oneOf***" o specificare che sono disgiunte attraverso "***owl:disjointWith***".
In ***OWL DL*** si ha una limitazione nel modo in cui i costrutti di ***OWL*** e ***RDF*** possono essere usati.
Il vantaggio di tale linguaggio è che permette un supporto efficiente al ragionamento, mentre lo svantaggio è che si perde la piena compatibilità con ***RDF***: un documento ***RDF*** deve essere limitato o esteso per diventare un documento valido ***OWL DL***, di contro ogni documento valido ***OWL DL*** è ancora un documento valido ***RDF***.

OWL Full consente la massima espressività e libertà sintattica di ***RDF*** senza garanzie computazionali.
Costruito estendendo ***OWL DL***, permette il trattamento delle classi simultaneamente come collezioni (la classe *extension*) e come individui o istanze(la classe intension).
In tale linguaggio è incluso il vocabolario ***OWL*** completo, interpretato più ampiamente che in ***OWL DL***, con la libertà data da ***RDF***.
L'intero linguaggio ***OWL*** è chiamato ***OWL Full***, e usa tutte le primitive di tale linguaggio, che possono essere combinate in modo arbitrario con ***RDF*** e ***RDF Schema***.
Inoltre, include la possibilità di cambiare il significato di una primitiva pre-definita (***RDF*** o ***OWL***), applicando le primitive del linguaggio.

Altra significativa differenza da ***OWL DL*** è che una ***owl:DatatypeProperty*** può essere marcata come una ***owl: InverseFunctionalProperty***.
Il vantaggio di ***OWL Full*** è la totale compatibilità con ***RDF***, sintattica e semantica: ogni documento valido ***RDF*** è anche un documento valido ***OWL Full***, e ogni conclusione ***RDF***/***RDF Schema*** è anche una conclusione valida ***OWL Full***.
Lo svantaggio è che il linguaggio è così potente da essere ingestibile, rendendo difficile il supporto al ragionamento.[46][47][48]
Le caratteristiche che favoriscono ***OWL*** sono:

- Maggiore espressività rispetto a ***RDF(S)***;
- Potere espressivo sufficiente ad effettuare la modellazione di ontologie;
- Supporta la logica descrittiva.

2.2 Altri linguaggi ontologici

CycL[35]è un linguaggio formale nel quale la sintassi deriva da predicati del primo ordine, nato per descrivere l'intelligenza artificiale di sistemi di elaboratori, ma non avendo raggiunto il suo scopo costituisce una grande ontologia formalizzata.
La sintassi consiste di termini appartenenti alla logica dei predicati del primo ordine, estesa con concetti del secondo ordine come meccanismi per le conclusioni o tecniche di strutturazione.
I concetti di ***CycL*** sono:

- Costanti: rappresentano il vocabolario(le parole) per esprimere gli assiomi;
- Predicati: descrivono relazioni fra termini;
- Variabili: rappresentano termini di una formula;
- Formule: combinano i termini in frasi significative, che formano delle proposizioni chiuse ***CycL*** (senza variabili libere); un insieme di espressioni ***CycL*** formano una base di conoscenza.
- Microteorie: è un insieme di formule nella base di conoscenza;ogni formula appare al più in una microteoria e le microteorie stesse possono apparire in una formula

KIF[49](Knowledge Interchange Format) è un linguaggio basato sul predicato logico del primo ordine, progettato per lo scambio di conoscenza fra sistemi di elaboratori ed in particolare per esprimere e scambiare ontologie.
KIF è un linguaggio funzionale per la rappresentazione della conoscenza di tipo dichiarativo, che concettualizza il mondo in termini di oggetti e relazioni fra tali oggetti e non è pensato per l'interazione fra gli uomini, anche se può essere utilizzato per tale scopo.
Le caratteristiche di ***KIF*** sono:

- semantica dichiarativa;
- logicamente comprensivo;
- provvede a significati per la rappresentazione di conoscenza circa la conoscenza (meta-informazioni).

XOL[50]è stato progettato dalla comunità di bioinformatica statunitense per lo scambio di definizioni ontologiche tra insiemi eterogenei di sistemi software nel loro dominio.
Tale linguaggio è stato costruito su ***Ontololingua*** e ***OML***.
Attualmente non ci sono applicativi che permettono lo sviluppo di ontologie usando ***XOL***, ma poiché usa la sintassi ***XML***, può essere utilizzato un qualunque editor ***XML*** per realizzare file ***XOL***.

SHOE[51]è stato sviluppato all'Università del Maryland come estensione di ***HTML***.
Tale linguaggio introduce primitive per definire ontologie ed esempi di dati sulle pagine Web, incorporando conoscenza semantica comprensibile ai elaboratori nei documenti ***HTML*** o in altri documenti Web.
Recentemente è stata adattata la sintassi ***SHOE*** a ***XML*** e ciò rende possibile agli "agenti" di raccogliere significative informazioni circa pagine Web e documenti, migliorando i meccanismi di ricerca e raccolta della conoscenza.

OML[52]è parzialmente basato su SHOE, tanto da essere considerato una serializzazione ***XML*** di ***SHOE***. ***OML*** e ***SHOE*** condividono diverse caratteristiche.
In particolare esistono quattro diversi livelli di ***OML***:

- **OML Core** è relativo agli aspetti logici del linguaggio ed è incluso dal resto degli strati;
- **Simple OML** mappa direttamente a ***RDF(S)***;
- **Abbreviated OML**, include caratteristiche grafiche concettuali;
- **Standard OML** è la più espressiva versione di ***OML***.

Anche per ***OML*** non esistono attualmente applicativi speciali per definire ontologie, quindi, si devono utilizzare applicativi editor per ***XML***.

2.3 Confronto fra i diversi linguaggi ontologici

2.3.1 Componenti principali delle ontologie

Le ontologie, come visto precedentemente, realizzano un vocabolario comune relativo ad una data area di conoscenza e, definiscono con differenti livelli di formalità il significato di termini e le relazioni fra loro esistenti.
Per rappresentare le ontologie, un linguaggio ontologico necessita di un certo numero di elementi fondamentali per specificare la conoscenza. Tali componenti sono[12]: "*concetti*", "*relazioni*", "*funzioni*", "*assiomi*" e "*istanze*".

Concetti

I "*concetti*" rappresentano qualcosa che è "*detto*" in un dato dominio di conoscenza, per rappresentare un pensiero definito e idealmente configurato, in modo esauriente ed insostituibile, formulabile in modo logico e realizzabile praticamente.
Le ontologie permettono di raggruppare gli oggetti di un dato dominio mediante gruppi che tengono in considerazione le proprietà comuni. Quindi nelle ontologie, i "*concetti*" possono essere definiti come le descrizioni di queste proprietà comuni.
Nel campo dei linguaggi ontologici sono stati usati diversi nomi per definire i "*concetti*" (classe, oggetto e categoria) e diversi sono pure i tipi possibili (ad esempio: reali o fittizi, elementari o composti).
Importante per rappresentare i "*concetti*", è l'individuazione e rappresentazione delle "*caratteristiche*" o "*attributi*", che permettono di rappresentare le qualità peculiari, per effettuare una distinzione che è utile anche per la fase di classificazione.

Gli attributi relativi ai "*concetti*" (indicati nei linguaggi ontologici anche come slot, funzioni o proprietà), possono essere di diversi tipi: locali o globali, di classe o relativi alla singola istanza della classe.

Una volta individuati, i "*concetti*"possono essere ordinati mediante relazioni di tipo "*sottoclasse*" e "*superclasse*", spesso organizzate in tassonomie.

Per elaborare i "*concetti*" è spesso necessario definire degli insiemi disgiunti di concetti, le cosiddette "*partizioni*", che permettono di rappresentare concetti appartenenti alla stessa ontologia, ma disgiunti fra di loro. Altre caratteristiche utili nella definizione di attributi e classi sono:

- valori di default assegnate agli attributi se non sono stati definiti valori espliciti;
- definizione di tipi in modo da vincolare le corrispondenti istanze ad avere determinate caratteristiche;
- vincoli di cardinalità usati per vincolare il numero di minimo e massimo valori;
- possibilità di definire della documentazione anche in linguaggio naturale per gli attributi, fondamentale per la manutenibilità e la leggibilità.

Tassonomie

I concetti spesso sono organizzate in tassonomie, per organizzare la conoscenza ontologica tramite relazioni di generalizzazione e specializzazione attraverso cui si può applicare l'eredità semplice o multipla.

Fra le primitive che si possono trovare nei linguaggi ci sono: "*sotto-classe di*" che permette di specializzare concetti generali in concetti più specifici; le "*disgiunzioni*" che sono partizioni in cui tutti i concetti e quindi sono sottoclassi di un concetto comune; "decomposizione esaustiva in sottoclassi" rappresenta una decomposizione disgiunta completa; "*non sottoclasse di*" usato per definire che un concetto non è una specializzazione di altri concetti [14].

[14] Questo tipo di conoscenza è di solito rappresentato usando la negazione della subclass primitiva.

Relazioni e funzioni

Le ***relazioni*** rappresentano interazioni fra concetti del dominio e attributi, formalmente definite come un sottoinsieme del prodotto di n insiemi:

$$R: C1 \times C2 \times ... \times Cn$$

Le ***funzioni*** sono un tipo speciale di relazioni, caratterizzate dal fatto che il valore dell'n-esimo argomento è unico in corrispondenza ad n-1 argomenti assegnati. Formalmente le funzioni possono essere definite come:

$$F: C1 \times C2 \times ... \times Cn\text{-}1 \rightarrow Cn$$

Le caratteristiche richieste per relazioni e funzioni nelle ontologie sono: possibilità di definire relazioni e funzioni con n argomenti arbitrari; possibilità di vincolare il tipo degli argomenti e di definire il tipo di vincoli di integrità per controllare la correttezza degli argomenti; definire la loro semantica usando assiomi o regole; definizioni operazionali per inferire valori di argomenti con procedure, formule o regole.

Assiomi

Gli assiomi rappresentano affermazioni che sono sempre vere e possono essere usate per diversi scopi, ad esempio per verificare la correttezza o dedurre nuove informazioni. Nel Web Semantico attualmente gli assiomi non sono molto usati, ma la loro importanza è destinata a crescere, perché nuova conoscenza può essere dedotta automaticamente dalle informazioni possedute e, possono essere dedotte eventuali inconsistenze.

Istanze, fatti e affermazioni

Le "*istanze*" sono usate per rappresentare elementi del dominio d'interesse collegati ad uno specifico "*concetto*".
I "*Fatti*" sono relazioni che mantengono insieme elementi, mentre le "*affermazioni*" sono asserzioni di un "*fatto*" realizzato da una "*istanza*".

Quindi ad esempio se consideriamo la classe "*Mammifero*", possiamo dire che l'elemento "*Balena*" è una sua "*istanza*" , mentre la frase "*La Balena vive nel mare*" è un "*Fatto*" e "*Il Biologo dice che la Balena vive nel mare*" è una "*affermazione*".
Le informazioni nel Web vengono definite usando "*istanze*" di "*concetti*" e "*relazioni*", e le "*affermazioni*" sono importanti, perché nell'ambiente distribuito del Web Semantico, le risorse saranno capaci di fare "*affermazioni*".

2.3.2 Risultati confronto fra i principali linguaggi per le ontologie

In tale paragrafo è effettuato un confronto fra i principali linguaggi per le ontologie, analizzati nell'ambito di questa tesi.
In particolare, il raffronto avviene a partire dai componenti fondamentali per le ontologie e di caratteristiche aggiuntive[95][96]:

Principali Elementi	
Concetti	Rappresentano qualcosa che è "detto" in un dato dominio
Relazioni	Sono interazioni fra concetti del dominio e attributi
Funzioni	Sono un tipo speciale di relazione dove il valore dell'n-esimo argomento è assegnato in modo unico in corrispondenza ad n-1 argomenti assegnati.
Istanze	Sono usate per rappresentare elementi del dominio collegati ad uno specifico "*concetto*"
Assiomi	modellano affermazioni che sono sempre vere
Concetti	
Negazione	Negazioni di concetti
Congiunzione	Insiemi di concetti congiunti
Partizione	Insiemi di concetti disgiunti
Documentazione	Definizione di documentazione dei concetti
Attributi	
Attributi di istanza	Attributi i cui valori possono essere diversi per ogni istanza dello stesso concetto
Attributi di classe	Attributi i cui valori sono uguali per tutte le istanze dello stesso concetto
Attributi locali	Attributi che possono avere lo stesso nome associato a differenti concetti
Attributi globale	Attributi in cui non è specificato il dominio, quindi sono applicati a tutti i concetti dell'ontologia
Aspetti di attributi	
Valori di Default	Assegnazione di un valore agli attributi nel caso di una mancata assegnazione esplicita
Restrizioni di Tipo	Restrizione del tipo possibile per un attributo

Restrizioni di Cardinalità	Definizioni del minimo e massimo numero di valori
Altre restrizioni	
Tassonomie di Concetti	
Sottoclassi-di	Specializza concetti generali in concetti più specifici
Decomposizione disgiuntiva	Una partizione dove tutti i concetti sono sottoclassi di un concetto comune, ma che non è necessariamente completa (ci potrebbero essere istanze del concetto non incluse in nessuna delle sottoclassi)
Decomposizione esaustiva in sottoclassi	Rappresenta la completa decomposizione disgiuntiva, in quanto ogni istanza del concetto della superclasse deve essere una istanza di ogni concetto della partizione
Non sottoclasse di	Indica che un concetto non è la specializzazione di un altro concetto
Relazioni e funzioni	
n-arità	È possibile definire relazioni e funzioni di arità arbitraria
Restrizioni di tipo	È possibile definire dei vincoli sui tipi degli argomenti
Restrizioni di integrità	Sono dei vincoli che possono essere inseriti per controllare la correttezza dei valori degli argomenti
Definizioni operazionali	
Assiomi	
Logica del primo ordine	Possibilità di avere assiomi che rispettano la logica del primo ordine
Logica del secondo ordine	Possibilità di avere assiomi che rispettano la logica del secondo ordine
Istanze	
Istanze di concetti	Sono usate per rappresentare elementi del dominio collegati ad uno specifico "*concetto*"
Fatti	Sono relazioni che mantengono insieme elementi
Affermazioni	Sono asserzioni di un "*fatto*" realizzato da una "*istanza*"

Tabella 2.1: Descrizione delle caratteristiche rispetto a cui viene effettuato il raffronto fra i linguaggi ontologici

	RDF(S)	OIL	DAML+OIL	OWL
Principali Elementi				
Concetti	+	+	+	+
Relazioni	+	+	+	+
Funzioni	-	+	+	+
Istanze	+	+	+	+
Assiomi	-	+	+	+
Concetti				
Negazione	-	+	+	+
Congiunzione	-	+	+	+
Partizione	-	+	+	+
Documentazione	+	+	+	+
Attributi di Concetti				
Attributi di istanza	+	+	+	+
Attributi di classe	-	+	+	+
Attributi locali	+	+	+	+
Attributi globali	+	+	+	+
Aspetti di attributi				
Valori di Default	-	-	-	-
Restrizioni di Tipo	+	+	+	+
Restrizioni di Cardinalità	-	+	+	+
Altre restrizioni	+	+	+	+
Tassonomie di Concetti				
Sottoclassi-di	+	+	+	+
Decomposizione disgiuntiva	-	+	+	+
Decomposizione esaustiva	-	+	+	+
Non sottoclasse di	-	+	+	+
Relazioni e funzioni				
n-arità	+/-	+/-	+/-	+/-
Restrizioni di tipo	+	+	+	+
Restrizioni di integrità	-	-	-	-
Definizioni operazionali	-	-	-	-
Assiomi				
Logica del primo ordine	-	+/-	+/-	+/-
Logica del secondo ordine	-	-	-	-
Istanze				
Istanze di concetti	+	+	+	+
Fatti	+	+	+	+
Affermazioni	+/-	+/-	+/-	+/-

Tabella 2.2: Raffronto fra i principali linguaggi ontologici sulla base dei principali caratteristiche delle ontologie. La convenzione usata è di mettere "+" se la caratteristica è supportata, "-" se non è supportata, "+/-" se non è supportata direttamente nel linguaggio, ma può essere rappresentata sfruttando altre caratteristiche del linguaggio.

Dall'analisi si evince che il linguaggio meno espressivo e meno adatto a rappresentare ontologie complesse è ***RDF(S)***, perché non permette di ottenere diverse caratteristiche (per esempio non rappresenta assiomi e istanze). Invece, i due linguaggi più completi sono ***DAML+OIL*** e ***OWL*** che, rispetto alle caratteristiche analizzate, si comportano allo stesso modo. Ciò è da ascrivere al fatto che per la progettazione di ***OWL*** si è deciso di basarsi su ***DAML+OIL***, cercando di superare i ben conosciuti limiti di tale linguaggio, tra cui l'esiguità dell'espressività semantica.

I cambiamenti sostanziali di ***OWL*** rispetto a ***DAML+OIL*** sono la non implementazione delle restrizioni numeriche qualificate[15], l'abilità di vedere la simmetria delle proprietà e la rimozione di alcuni costrutti poco diffusi. [46]

Alcune caratteristiche non sono state implementate in nessuno dei linguaggi analizzati, ad esempio le caratteristiche di default (per gli attributi), i vincoli di integrità e le definizioni operazionali (per relazioni e funzioni).

Riguardo la logica, nessuno dei linguaggi implementa quella del secondo ordine, mentre tutti ad esclusione di RDF(S) permettono di realizzare la logica del primo ordine, utilizzando altre caratteristiche dei linguaggi.

[15] Rappresentate in ***DAML+OIL*** mediante i costrutti "*daml:hasClassQ*", "*daml:cardinalityQ*", "*daml:minCardinalityQ*", e "*daml:maxCardinalityQ*".

Conclusioni

Nella realizzazione delle applicazioni per il Web Semantico, come visto, l'ontologia assume un ruolo fondamentale per ottenere la comprensione automatica dei significati relativi ai dati presenti sul Web.

L'ontologia, dovendo rappresentare la "*conoscenza*" in una data area, risulta spesso di elevata complessità, quindi, è importante realizzare una opportuna fase progettuale e fare una corretta scelta del linguaggio ontologico.

La scelta di un'adatta metodologia di progettazione, permette di "*guidare*" il progettista nella realizzazione dell'ontologia e nella individuazione della conoscenza da rappresentare.

Risulta, quindi, importante l'individuazione delle diverse fasi da realizzare e i relativi risultati che devono essere ottenuti per eseguire il progetto ontologico.

La scelta del linguaggio ontologico, risulta anch'essa importante, in quanto una scelta non corretta potrebbe portare all'utilizzo di un linguaggio con limitato potere espressivo, che non permette di rappresentare l'ontologia o ad un linguaggio troppo complesso che accresce gli sforzi del progettista. Quindi bisogna effettuare la scelta partendo dalla conoscenza dei diversi linguaggi presenti, in modo che possa essere individuato quello che più si confà alle necessità d implementazione.

Dal confronto fra i linguaggi si è visto che i linguaggi analizzati ad esclusione di ***RDF(S)***, permettono di implementare la maggior parte delle caratteristiche ontologiche analizzate.

Con *OIL* e poi in modo più efficace con *DAML+OIL* e *OWL*, i linguaggi ontologici hanno raggiunto caratteristiche tali da permettere di rappresentare ontologie anche complesse, permettendo di rappresentare tutte le caratteristiche delle ontologie:

	RDF(S)	OIL	DAML+ OIL	OWL
Principali Elementi				
Concetti	+	+	+	+
Relazioni	+	+	+	+
Funzioni	-	+	+	+
Istanze	+	+	+	+
Assiomi	-	+	+	+

Tabella 1: Raffronto fra i principali linguaggi ontologici sulla base dei principali caratteristiche delle ontologie. La convenzione usata è di mettere "+" se la caratteristica è supportata, "-" se non è supportata, "+/-" se non è supportata direttamente nel linguaggio, ma può essere rappresentata sfruttando altre caratteristiche del linguaggio.

Mentre, per quanto riguarda la capacità di operare nel campo della logica, i linguaggi hanno una ridotta operatività, quindi allo stato attuale volendo implementare il livello logico si deve pensare di ricorrere ad opportuni linguaggi logici che sono in fase di progettazione.

	RDF(S)	OIL	DAML+ OIL	OWL
Assiomi				
Logica del primo ordine	-	+/-	+/-	+/-
Logica del secondo ordine	-	-	-	-

Tabella 2: Raffronto fra i principali linguaggi ontologici sulla capacità di implementare logica del primo o del secondo ordine. La convenzione usata è di mettere "+" se la caratteristica è supportata, "-" se non è supportata, "+/-" se non è supportata direttamente nel linguaggio, ma può essere rappresentata sfruttando altre caratteristiche del linguaggio.

Bibliografia

[1] Uschold M. and Gruninger M., *" Ontologies: Principles, methods and applications."* Knowledge Engineering Review, Vol. 11, pp 93-136, (1996)

[2] Sowa J. F., "*Guided Tour of Ontology"*; http://www.jfsowa.com/ontology/guided.html (2000)

[3] Hobbs J. R., "*Granularity*", in Proceedings of IJCAI '85, pp 432-435 (1985)

[4] Hobbs J. R., "*Ontological Promiscuity*", in Proceedings of ACL '85, pp.61-69 (1985)

[5] Hirst G., "*Existence Assumptions in Knowledge Representation"*, Artificial Intelligence n. 49, pp199-242 (1991)

[6] Hobbs J. R., Croft W., Davies T., Douglas E. and Laws K., "*Commonsense Metaphysic and Lexical Semantics*", Computational Linguistics, n. 13, pp 241-250 (1987)

[7] Bateman J.A., Kasper R.T., Moore J.D. and Whitney R.A. , "*A General Organization of Knowledge for Natural Language Processing*", Mouton de Gryter, pp 393-439 (Berlin 1990)

[8] Klose G., Lang E. and Pirlen T. "*Ontologie und Axiomatik derWissensbasis von LILOG"* , Springer-Verlag (Berlin 1992)

[9] Alexander J.H., Freiling M.J., Shulman S.J., Staley J.L., Rehfus S. and Messick S.L., "*Knowledge Level Engineering: Ontological Analysis", in* Proceedings of AAAI 86 pp 963-968 (Philadelphia 1986)

[10] Neches R., Fikes R., Finin T., Gruber T., Patil R., Senator T. and Swartout W.R., "*Enabling Tecnology for Knowledge Sharing and Reuse",* AI Magazine 12(3), pp.36-56 (1991)

[11] Musen M.A., "*Dimensions of Knowledge Sharing and Reuse",* Computers and Biomedical Research, n.25, pp 435-467 (1992)

[12] Gruber T.R., "*A Translation Approach to Portable Ontology Specifications*," Knowledge Acquisition, vol. 5, pp199–220 (1993)

[13] Gruber T. R., "*Toward principles for the design of ontologies used for knowledgesharing", IJHC*S, n. 43(5/6): pp 907-928 (1994)

[14] Guarino N.; **"** *Formal ontologies and information systems***"** in Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'98), (Trento, Italy, June 1998).

[15] Bernaras A., Laresgoiti I. and Corera, J.; *"Building and reusing ontologies for electrical network applications*"; in Proceedings of the 12th European Conference on Artificial Intelligence (ECAI), pp 298-302,. John Wiley Sons, Ltd. (Chichester, England, 1996)

[16] Swartout B., Patil R., Knight K. and Russ., T., *"Towards distributed use of largescale ontologies."* in Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW), (Alberta, Canada, 1996).

[17] Genesereth M. R. and Nilsson N. J. , "*Logical Foundation of Artificial Intelligence".* Morgan Kaufmann, (Los Altos, California 1987).

[18] Gomez-Perez, A.. *"A framework to verify knowledge sharing technology*. Expert Systems with Application, n. 11(4), pp 519-529. (1996)

[19] Fernandez M., Gomez-Perez A. and Juristo N., *"METHONTOLOGY: From Ontological Arts Towards Ontological Engineering".* in Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, pp 33-40, (Stanford, USA March 1997) .

[20] Caroll J.M., *"Scenario-Based Design",* cap. 17 , John Wiley & Sons (1997)

[21] Cocho O., Gòmez-Perez A.," *Ontology Languages for the Semantic Web*"; IEEE INTELLIGENT SYSTEMS, pp 54-60 (January -February 2002)

[22] Guarino, N. and Welty, C.. *"Towards a methodology for ontology-based model engineering"* in Proceedings of ECOOP-2000 Workshop on Model Engineering. (Cannes, France 2000)

[23] Mizoguchi R., Ikeda M., ***"****Towards Ontology Engineering****",*** in Proceedings of The Joint 1997 Pacific Asian Conference on Expert systems /Singapore International Conference on Intelligent Systems, pp. 259-266 (1997)

[24] Fensel,"*Lessons Learned from Applying AI to the Web*", *J. Cooperative Information Systems*, vol. 9, no. 4, pp. 361–382 (December 2000)

[25] Guarino N., *"Understanding, Building and Using Ontologies. A Commentary to "Using Explicit Ontologies in KBS Development"*, by van Heijst, Schreiber, and Wielinga. International Journal of Human and Computer Studies vol. 46 n. 2/3, pp. 293-310 (1997)

[26] McGuiness, D.L., Fikes R.E. , Rice J. e Wilder S.; *"An enviroment form merging and testing large ontologies"* in Proceedings of the seventh international conference (KR2000), pp 483-493 (San Francisco 2000)

[27] Van Heijst G., Schreiber A. and Wielinga B.J.; **"** *Using explicit ontologies in kbs development. International Journal of Human-Computer Studies*", pp 184-292 (1997)

[28] Lassila O. and McGuinness D.; *"The role of frame-based representation on the semantic Web"*; Electronic Transactions on Artificial Intelligence (ETAI) Journal: area The Semantic Web (2001)

[29] McGuiness D. L., *"Ontologies Come of Age"* in Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press (2002)

[30] http://www.dmoz.com/

[31] www.dublincore.org/usage/terms/dc/current-elements/ (Marzo 2003)

[32] Greenberg J., Sutton S., Grant Cambell D., "*Metadata: A Fundemental Component of the Semantic Web",* in Bulletin of the American Society for Information Science and Technology, pp16-18 (April/May 2003)

[33] http://www.nlm.nih.gov/research/umls/

[34] www.ariadnegenomics.com/technology/ontology.html

[35] Lenat D.and Guha R.V., "*Building large knowledge-based systems: Representation and inference in the CYC project*", Addison Wesley (1990)

[36] Episodic Logic (http://www.cs.rochester.edu/research/epilog/) (November 2000)

[37] Uschold M., King M., Moralee S. & Zorgios Y., "*The Enterprise Ontology*", *Knowledge Engineering Review* , Vol. 13 (1998)

[38] NIST STEP References : (http://ats.nist.gov/stepmod/related.html#step)

[39] Miller G.A., "*WORDNET: A lexical database for English*", *Communications of ACM* (11), pp 39-41 (1995)

[40] Horrocks I. et al., "OIL in a Nutshell," *Proc. ECAI '00 Workshop on Application of Ontologies and PSM*s (Germany, Berlin 2000)

[41] Fensel D. et al., *"OIL : An Ontology Infrastructure for the Web",* IEEE Intelligent Systems pp38-45 (March/April 2001)

[42] Horrocks I., Fensel D., Broekstra J., Decker S., Erdmann M. , Goble C., van Harmelen F., Klein M., Staab S., Studer R. and Motta E., *"The Ontology Inference Layer OIL, technical report",* http://www.ontoknowledge.org/oil (Vrije Universiteit Amsterdam, NL. 2001)

[43] Horrocks I. and Van Harmelen F., "*ReferenceDescription of the DAML+OIL Ontology Markup Languag*e", www.daml.org/2000/12/reference.html (draft report 2001)

[44] Scott Cost R. et al., "*IT talks: A Case Study in the Semantic Web and DAML+OIL*", IEEE INTELLIGENT SYSTEMS pp.40-47 (March/April 2002)

[45] Daconta M.C., Obrst L.J., Smith K.T.; *"The Semantic Web:A Guide to the Future of XML, Web Services, and the Knowledge Management"*, pp 230-237; Wiley Publishing; pp.1-26(2003)

[46] Antoniou G. and Van Harmelen F., **"***Web Ontology Language: OWL***"** (April 2003)

[47] McGuinness D. and Van Harmelen F. *" OWL Web Ontology Language Overview"* http://www.w3.org/TR/2003/WD-owl-features-20030331/ (W3C Working Draft 31 March 2003)

[48] Smith M., Welty C. and McGuinness D.; *"OWL Web Ontology Language Guide"*, http://www.w3.org/TR/owl-guide/ (W3C Recommendation 10 February 2004)

[49] Genesereth M.R., *"Knowledge Interchange Forma,"* in Proceeding Second Intenational Conference of Principles of Knowledge Representation and Reasoning (KR 91), J. Allenet et al., eds., Morgan Kaufmann, pp 238–249 http://logic.stanford.edu/kif/kif.html (San Francisco, 1999)

[50] Karp R., Chaudhri V. and Thomere J., *"XOL: An XML-Based Ontology Exchange Language(version 0.4)"* www.ai.sri.com/~pkarp/xol (August 1999)

[51] Luke S. and Heflin J., "*SHOE 1.01 ProposedSpecification*", SHOE Project www.cs.umd.edu/projects/plus/SHOE/spec1. 01.htm (February 2000)

[52] Kent R., "*Conceptual Knowledge Markup Language (version 0.2). 1998*", www.ontologos.org/CKML/CKML%200.2.html (1998)

[53] http://www.w3.org