


CORSO I.F.T.S.
"TECNICHE PER LA PROGETTAZIONE E LA GESTIONE DI DATABASE"

Matricola 2014LA0033

DISPENSE DIDATTICHE

MODULO di "DATABASE SEMANTICI"

Ing. Simone Menabeni

Lezione del 22/09/2014

Ontologie e OWL

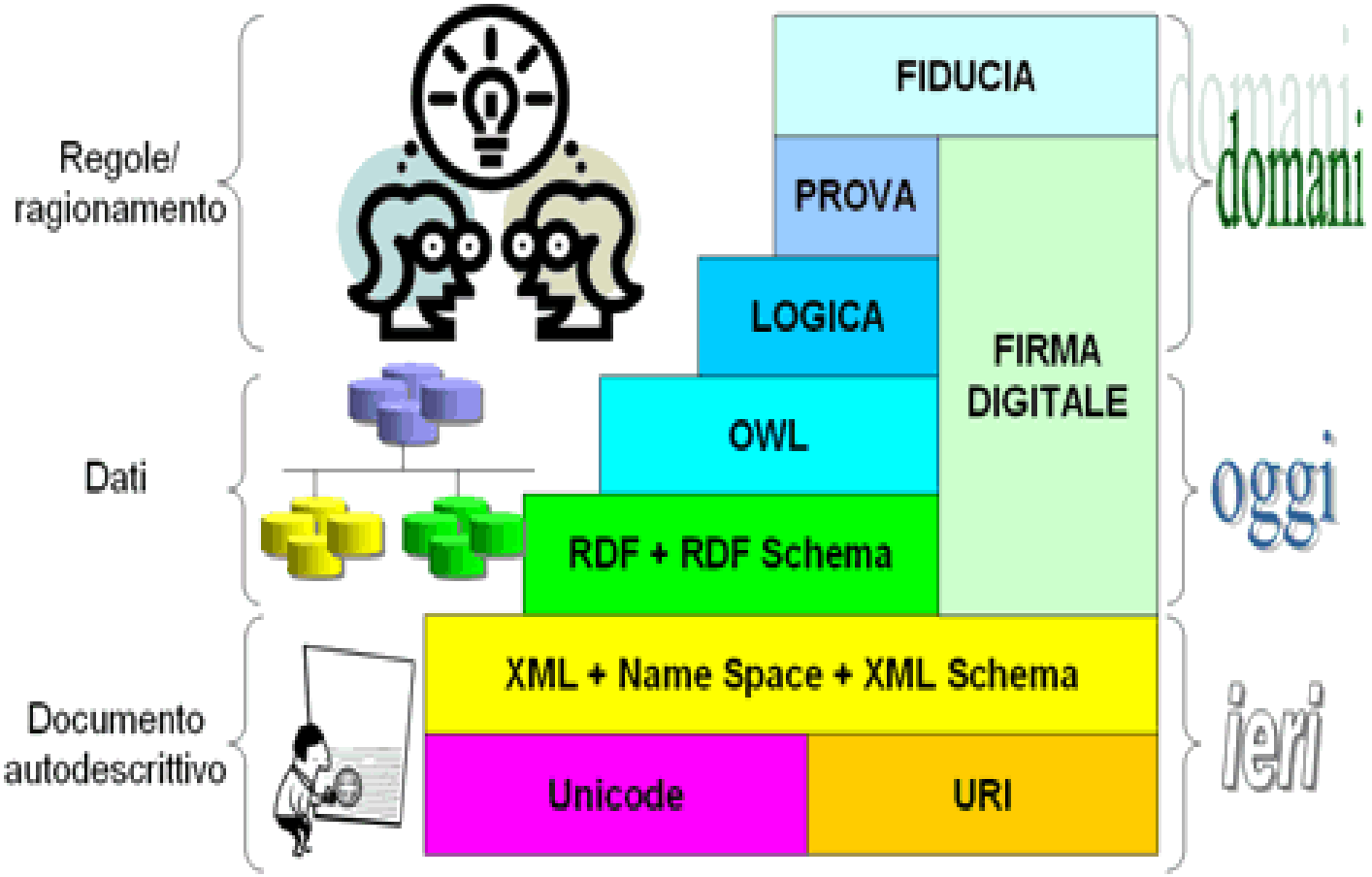
Simone Menabeni
Dipartimento di Ingegneria dell'informazione
Università di Firenze
simone.menabeni@unifi.it

DISIT Lab
http://www.disit.dinfo.unifi.it/

Indice

- Rappresentazione della Conoscenza e Ontologie
  - La conoscenza
  - Linguaggi di Rappresentazione
  - Ragionamento Automatico
  - Sistemi di rappresentazione della conoscenza
- OWL: *Ontology Web Language*
  - Introduzione
  - Descrizione di Classi e Assiomi
  - Proprietà
  - Individui e Fatti
  - Servizi di Ragionamento
  - Interrogare la Conoscenza: SPARQL

Web Semantico: OWL



XML vs RDF

- Si può riassumere brevemente quanto presentato finora.
- L'XML fornisce uno strato "sintattico" per strutturare i documenti, ma non impone nessun vincolo semantico sul significato di questi documenti.
- L'XML Schema è un linguaggio usato per restringere la struttura dei documenti XML e per estenderli grazie ai datatype.
- L'RDF è un modello dei dati per le risorse e per le relazioni tra di esse, che fornisce una semplice semantica per questo modello dei dati e che permette al modello stesso di essere rappresentato nella sintassi dell'XML.
- L'RDF Schema è una vocabolario utilizzato per descrivere le classi e le proprietà delle risorse RDF, con una semantica per la generalizzazione delle gerarchie di tali classi e risorse.

Ontologie

- Il “livello ontologico" del Web Semantico è quello che raccoglie attualmente gli sforzi maggiori, perché centrale nell’opera di sviluppo di un “Knowledgeable Web”, in cui la gestione della conoscenza e il suo trattamento da parte delle macchine siano predominanti.
- Un’ontologia è una descrizione dei concetti e delle relazioni che intercorrono fra di essi.
- Individua gli aspetti che sono ritenuti rilevanti nel contesto di riferimento e quelli che possono essere ignorati: specifica i concetti e come sono legati tra loro, quali sono le proprietà di ciascuno e come queste proprietà sono connesse, mediante regole d’inferenza e logiche.

Ontologie

- Un'ontologia è formata da una raccolta di nomi per i concetti che si vuole definire e dalle relazioni di ordinamento tipo-sottotipo che si vogliono stabilire per un dato dominio o realtà che si intende descrivere e interpretare.
- L'introduzione delle ontologie nel Web consente la strutturazione delle informazioni e permette di superare alcuni aspetti critici del Web tradizionale.

Ontologie - Caratteristiche

1. In primo luogo, le varie sorgenti “producono” informazioni in diversi formati e la creazione di indici per localizzare queste sorgenti è piuttosto complessa, poiché risulta molto difficile ottenere indicazioni da sorgenti audio o video: un’ontologia, viceversa, può facilitare questa operazione, descrivendo in modo formale i contenuti di ogni sorgente e supportando gli utenti nella ricerca di quelle sorgenti che generano un particolare tipo di informazione.
2. L'introduzione del livello ontologico nel Web inteso in senso tradizionale consente di superare anche il problema della mancanza di struttura del Www, dovuta al fatto che HTML è un linguaggio di formattazione e non si occupa di gestire le informazioni per facilitare il loro reperimento: un’ontologia descrive il dominio dal punto di vista strutturale, definendone i componenti e i loro legami.

Ontologie - Caratteristiche

3. Dipendenza dal contesto: definendo mediante un'ontologia il contesto in cui è presentato un certo documento, i termini utilizzati sono propri di quell'ambito e quindi il loro significato non è ambiguo.

Ontologie - Sviluppo

- Il livello dei dati, formato da XML e RDF, e il livello Schema, necessario alla definizione di un vocabolario, non sono però sufficienti per rendere le applicazioni capaci di processare "semanticamente" i documenti.
- Una semantica formale per i primitivi espressi da RDF Schema non è fornita e l'espressività di questi primitivi non è sufficiente per un modellamento ed un ragionamento ontologico, ed è necessario uno strato di linguaggi ulteriore.
- Questo strato aggiuntivo di tecnologie è formato da quei linguaggi che consentono la creazione delle ontologie, e quindi del livello logico del Web Semantico. I linguaggi principali sviluppati a questo scopo, ovvero OIL, DAML+OIL e OWL mirano ad estendere RDF Schema per la definizione dei vocabolari formali necessari a compiere le inferenze.

Ontologie - Sviluppo

- Il contributo di questi linguaggi a RDF Schema sta nell'aggiungere una semantica formale basata su descrizioni logiche e su primitivi più avanzati, come l'uso dei marcatori booleani (AND, OR, NOT, per esempio) e di alcuni assiomi logici.

OWL

- In particolare OWL - Ontology Web Language è un linguaggio di markup semantico derivato da DAML+OIL e strutturato a partire da RDF.
- OWL consente la formalizzazione di uno specifico dominio della conoscenza, permettendo la definizione di classi e delle proprietà di tali classi. Permette la definizione di elementi singoli della classe e la definizione delle proprietà che questi assumono.
- Una ontologia OWL consiste di nessuna o più intestazioni, seguita da zero o più classi, elementi, proprietà degli elementi e istanze.

OWL

- L' "OWL Web Ontology Language", abbreviato in OWL, è un linguaggio progettato per essere utilizzato da parte di quelle applicazioni che necessitano di elaborare il contenuto delle informazioni, invece che presentarlo solamente all'uomo.
- L'OWL è una raccomandazione del W3C, pubblicata nel Febbraio del 2004. E' stata sviluppata dal "Web Ontology Working Group" come parte della "Semantic Web Activity" del W3C ad iniziare dal Novembre del 2001.
- L'acronimo naturale per il "Web Ontology Language" dovrebbe essere "WOL" invece che OWL.
- L'acronimo OWL fu proposto come un acronimo facilmente pronunciabile che vorrebbe augurare una buona fortuna, suggerire buonsenso e ricordare un progetto nell'ambito della rappresentazione della conoscenza degli anni '70 di William A. Martin chiamato "One World Language".

OWL

- L'OWL aggiunge ai linguaggi visti fino ad ora più vocabolario per descrivere classi e proprietà. Ad esempio, relazioni più complesse tra classi quali unione, cardinalità, uguaglianza, ecc.
- L'OWL è diviso in tre sottolinguaggi a potenza espressiva crescente, ognuno dei quali è designato per un uso specifico.

OWL

- **OWL Lite**: serve come supporto per quegli utenti che hanno bisogno di rappresentare classificazioni gerarchiche e vincoli semplici. Esso fornisce anche una migrazione veloce di thesauri ed di altre tassonomie.
- **OWL DL**: supporta quegli utenti che hanno bisogno della massima espressività, ma che devono conservare la completezza computazionale (cioè tutte le conclusioni sono garantire essere computabili) e la decidibilità (cioè tutte le elaborazioni terminano in un tempo finito).
- **OWL Full**: questo è stato sviluppato per quegli usi che necessitano della massima espressività e della massima libertà sintattica dell'RDF: questo però non dà garanzie computazionali. L'OWL Full permette alle ontologie di aumentare il significato di vocabolari predefiniti, appartenenti sia all'RDF che all'OWL.

OWL

- L'OWL Full può essere visto come un'estensione dell'RDF, mentre l'OWL Lite e l'OWL DL possono essere visti come estensioni di una visione ristretta dell'RDF.
- Ogni documento OWL è un documento RDF ed ogni documento RDF è un documento di OWL Full, ma solo certi documenti RDF saranno un documento valido per l'OWL Lite o per l'OWL DL.

OWL

- Seguendo l'obiettivo del WS, anche l'OWL deve consentire alle informazioni di poter essere raccolte da più sorgenti che sono sparse nel Web. Ciò è in parte fatto permettendo alle ontologie di essere collegate tra loro ed includendo l'esplicita importazione di informazioni da altre ontologie.
- Oltre a questo, si può dire che l'OWL fa una assunzione detta di tipo "Open World", cioè che la descrizione delle risorse non è limitata ad un singolo scopo.
- Facendo una piccola digresssione, l'assunzione Open World dice, tra le altre cose, che: "se una affermazione non può essere dimostrata vera usando la conoscenza attuale, allora non si può ottenere la conclusione che essa sia falsa".
- Le nuove informazioni che si aggiungono non possono ritrattare le informazioni precedenti; esse possono anche essere contraddittorie, ma i fatti e le implicazioni che ne derivano possono aggiungere e mai cancellare informazioni dell'ontologia originaria.

OWL

- Una ontologia OWL è una grafo RDF, che a sua volta è un insieme di triple RDF.
- Come i grafi RDF anche un'ontologia può essere scritta in diverse forme sintattiche.
- Un qualsiasi grafo RDF forma una ontologia di tipo OWL Full. Quindi, il significato dato ad un grafo RDF dall'OWL comprende ed, eventualmente, arricchisce il significato dato al grafo dall'RDF.

OWL - Deduzione

- La semantica di base dell'OWL fornisce il supporto per dedurre da alcuni dati degli altri dati che l'utente non si sarebbe aspettato.
- Un esempio di ciò è data dalla proprietà **owl:sameAs** che è usata per dichiarare che due individui, che apparentemente sembrano diversi, in realtà sono lo stesso. Questo fa si che le informazioni provenienti da diverse sorgenti, ma relative allo stesso individuo possano essere messe assieme.

Struttura Documento OWL

- Un documento OWL consiste di una o più “ontology headers”, seguita da un certo numero di definizioni di classi, di definizioni di proprietà e di fatti sugli individui.
- Nello scrivere un’ontologia la prima cosa da fare è specificare in maniera precisa quali vocabolari saranno utilizzati.
- Un componente iniziale standard di una ontologia comprende un insieme di dichiarazioni di namespace XML (XML namespace declaration) che vengono racchiusi nel tag di apertura **rdf:RDF** : ciò dà un significato preciso a tutti gli identificatori e rende il più leggibile il resto della presentazione dell’ontologia.

Struttura Documento OWL

- Una ontologia OWL inizierà con una namespace declaration del tipo:

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

Struttura Documento OWL

- Una volta che i namespace sono stati stabiliti, normalmente si includono un insieme di asserzioni sull'ontologia, raggruppate sotto il tag **owl:Ontology** , che specificano commenti, controllo delle versioni e l'inclusione di altre ontologie. Questa parte è chiamata "ontology headers".

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

Struttura Documento OWL

- L'attributo **rdf:about** fornisce un nome o un riferimento per l'ontologia. Se il valore è "" allora il nome dell'ontologia è l'URI base dell'elemento owl:Ontology .
- L'attributo **rdfs:comment** fornisce la possibilità di aggiungere commenti.
- **owl:priorVersion** è un tag standard che ha lo scopo di fornire dei "ganci" per il controllo della versione da parte di quei sistemi che utilizzano l'ontologia.
- **owl:import** fornisce un meccanismo di inclusione. Esso prende un singolo argomento che è identificato dall'attributo **rdf:resource** . Dal punto di vista concettuale owl:import serve ad indicare l'intenzione di includere le asserzioni dell'ontologia che verrà importata. L'importare un'altra ontologia porta nell'ontologia corrente l'intero insieme di asserzioni di quell'ontologia. Se quest'ultima importa altre ontologie, tutte sono portate all'interno della prima.
- **rdfs:label** supporta l'utilizzo di etichette per l'ontologia.
- La parte iniziale della definizione dell'ontologia, detta "header ontology definition" si chiude con il tag **</owl:Ontology>** . Seguono poi le definizioni effettive che compongono l'ontologia ed il tutto termina con il tag **</rdf:RDF>** .

Classi

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

Classi

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

Classi - Identificatore

- “Class Identifier” (cioè un URI reference): descrive la classe tramite un class name.
- Una class description di questo tipo è sintatticamente rappresentata come una istanza nominata di owl:Class , una sottoclasse di rdfs:Class .
- Ad esempio: **<owl:Class rdf:ID=”Human”/>** . Questo asserisce la tripla “ex:Human rdf:type owl:Class . “, dove ex: è il namespace dell’ontologia.

Classi - Enumeration

- Una lista precisa di tutti gli individui che insieme formano le istanze della classe.
- È definita tramite la proprietà **owl:oneOf** . Il valore di questa proprietà deve essere una lista di individui che sono istanze della classe. Questa lista è rappresentata tramite il costrutto RDF rdf:parseType=”Collection”.
- Ad esempio, per definire la classe di tutti i continenti il codice RDF/XML sarà:

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

Classi – Property Restriction

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

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

- La classe owl:Restriction è definita come sottoclasse di owl:Class. Fanno parte dei value constraints i costrutti: **owl:allValueFrom** , **owl:someValueFrom** e **owl:hasValue** . Fanno parte dei cardinality constraints i costrutti: **owl:maxCardinality**, **owl:minCardinality** ed **owl:hasValue** .

Restrizioni di proprietà (1)

Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **almeno** un individuo della classe **C** tramite l'operatore **owl:someValuesFrom**

Sintassi ***DL***

$A \equiv \exists R.C$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#R" />
      <owl:someValuesFrom rdf:resource="#C" />
    </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (2)

Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **soli individui** di classe **C** tramite l'operatore **owl:allValuesFrom**

Sintassi ***DL***

$A \equiv \forall R.C$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#R" />
      <owl:allValuesFrom rdf:resource="#C" />
    </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (3)

ESEMPI (da W3C OWL Guide: http://www.w3.org/TR/owl-ref/)

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#hasParent" />
    <owl:someValuesFrom rdf:resource="#Doctor" />
</owl:Restriction>
```

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#hasParent" />
    <owl:allValuesFrom rdf:resource="#Human" />
</owl:Restriction>
```

Restrizioni di proprietà (4)

Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** **sul solo individuo** a tramite l'operatore **owl:hasValue**

Sintassi ***DL***

$A \equiv \forall R.\{a\}$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#R" />
      <owl:hasValue rdf:resource="#a" />
    </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (5)

ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/)

```
<owl:Class rdf:ID="Burgundy">
  ...
  <rdfs:subClassOf rdf:resource="Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTaste" />
      <owl:hasValue rdf:resource="#Dry" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Restrizioni di proprietà (6)

Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **al massimo n individui** tramite l'operatore **owl:maxCardinality**

Sintassi ***DL***

$A \equiv \leq nR$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
   <owl:Restriction>
     <owl:onProperty rdf:resource="#R" />
     <owl:maxCardinality rdf:datatype=
        "&xsd;nonNegativeInteger">n</owl:maxCardinality>
   </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (7)

Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **almeno n individui** tramite l'operatore **owl:minCardinality**

Sintassi ***DL***

$A \equiv \geq nR$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
   <owl:Restriction>
     <owl:onProperty rdf:resource="#R" />
     <owl:minCardinality rdf:datatype=
        "&xsd;nonNegativeInteger">n</owl:minCardinality>
   </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (8)

Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **esattamente n individui** tramite l'operatore **owl:cardinality**

Sintassi ***DL***

A ≡ =nR

Sintassi ***RDF***

```
<owl:Class  rdf:ID="A">
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#R"  />
      <owl:cardinality  rdf:datatype=
          "&xsd;nonNegativeInteger">n</owl:cardinality>
    </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (9)

ESEMPI (da W3C OWL Guide: http://www.w3.org/TR/owl-ref/)

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxCardinality>
</owl:Restriction>

<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent" />
        <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:minCardinality>
</owl:Restriction>

<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:cardinality>
</owl:Restriction>
```

Classi - Intersezione

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

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

Classi - Unione

- L'unione di due o più class description. Vale quanto detto con l'intersezione, con le opportune modifiche. Il costrutto è **owl:unionOf**

Unione

Una classe **A** può essere descritta come *unione* di un numero finito di classi **C_1**, **...**, **C_n** attraverso l'operatore **owl:unionOf**

Sintassi ***DL***

$A \equiv C_1 \sqcup \ldots \sqcup C_n$

Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#C1" />
      ...
      <owl:Class rdf:about="#Cn" />
    </owl:unionOf>
</owl:Class>
```

Classi - Complementazione

- La complementazione di una class description.
- Una dichiarazione **owl:complementOf** descrive una classe per la quale la class extension contiene esattamente quegli individui che non appartengono alla class extension della class description che è oggetto della dichiarazione.

Class Axiom

- La class description forma i blocchi costruttivi per definire le classi attraverso i class axioms.
- La class axiom nella sua forma più semplice è una class description del tipo nr. 1.
- Essa dichiara solo l'esistenza di una classe usando owl:Class con un identificatore di classe.
- Ad esempio, **<owl:Class rdf:ID="Human"/>**. Questo pezzo di codice OWL è corretto, ma non dice molto a riguardo della classe Human.
- La class axiom contiene delle componenti addizionali che dichiarano caratteristiche necessarie e/o sufficienti di una classe.
- L'OWL contiene tre costrutti di linguaggio per combinare le class description in class axiom.

Class Axiom

1. rdfs:subClassOf : permette di dire che la class extension di una class description è un sottoinsieme della class extension di un'altra class description. Questo costrutto è definito come parte dell'RDF Schema. Il suo significato nell'OWL è lo stesso. Ad esempio:

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

   – Questo class axiom dichiara una relazione di sottoclasse tra due classi OWL che sono descritte tramite i loro nomi (Opera e MusicalWork). Qui Opera è sottoclasse di MusicalWork. L'axiom subclass fornisce una definizione parziale: esse rappresentano condizioni necessarie ma non sufficienti per stabilire l'appartenenza di un individuo ad una classe.

Class Axiom

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

```
<owl:Class rdf:about="#US_President">
   <equivalentClass rdf:resource="#PrincipalResidentOfWhiteHouse"/>
</owl:Class>
```

   - In questo caso il concetto di "Presidente degli Stati Uniti" è collegato, ma non uguale, al "principale residente della Casa Bianca". L'utilizzo di questo costrutto può essere molto più complesso.

Class Axiom

3. owl:disjointWith: è una proprietà dell'OWL che ha una class description oltre a domain e range. Una dichiarazione di questo tipo asserisce che la class extension delle due descrizioni di classe a riguardo non ha individui in comune.
   – L'esempio più noto di class disjoint è:

```
<owl:Class rdf:about="#Man">
  <owl:disjointWith rdf:resource="#Woman"/>
</owl:Class>
```

OWL - Property

- L'OWL distingue tra due categorie principali di proprietà:
  - "object properties", che collega individui ad individui;
  - "datatype properties", che collega individui a valori dei dati.

OWL – Object Property

- Una object property è definita come un'istanza della classe predefinita in OWL **owl:ObjectProperty**.
- Le object property sono relazioni tra istanze di due classi. Esse consentono di mettere in relazione tra di loro gli oggetti (ad esempio: possiede, insegna, etc.)

OWL – Datatype Property

- Una datatype property è definita come un'istanza della classe predefinita in OWL **owl:DatatypeProperty** .
- Le datatype property sono relazioni tra istanze della classe e literal RDF o datatype di XML Schema. Esse consentono di mettere in relazione gli oggetti con i valori (ad esempio: numeroDiTelefono, nome, dataDiNascita, etc.).

OWL – Property

- Sia **owl:DatatypeProperty** che **owl:ObjectProperty** sono sottoclassi della classe RDF rdf:Property.
- Un property axiom definisce le caratteristiche di una proprietà. Nella forma più semplice, un property axiom definisce l'esistenza di una proprietà.
- Ad esempio:

<owl:ObjectProperty rdf:ID="hasParent"/>

- che definisce una proprietà con la restrizione che i suoi valori dovrebbero essere individui.

OWL – Property

- L'OWL supporta i seguenti costrutti per i property axiom:
  1. RDF Schema constructs: **rdfs:subPropertyOf**, **rdfs:domain** ed **rdfs:range** .
  2. relazioni con altre proprietà. Queste sono **owl:equivalentProperty** e **owl:inverseOf** . Il primo di questi costrutti può essere utilizzato per dichiarare che due proprietà hanno la stessa property extension. Le proprietà hanno una direzione, dal domain al range. Nella pratica, la gente spesso trova utile definire relazioni in entrambe le direzioni. Ad esempio, "persone sono proprietarie di auto" e "auto sono possedute da persone". Il costrutto owl:inverseOf può essere usato per definire tali relazioni inverse tra proprietà. Ad esempio , le relazioni "haFiglio" ed "haGenitore":

```
<owl:ObjectProperty rdf:ID="hasChild">
    <owl:inverseOf rdf:resource="#hasParent"/>
</owl:ObjectProperty>
```

OWL - subPropertyOf

Una proprietà **R** può essere definita come sottoproprietà di un'altra proprietà **S** attraverso l'operatore **rdfs:subPropertyOf**

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
    <rdfs:subPropertyOf rdf:resource="#S" />
</owl:ObjectProperty>
```

OWL - domain

Di una proprietà **R** può essere specificato il dominio attraverso l'operatore **rdfs:domain**

Sintassi ***RDF***

```
<owl:ObjectProperty  rdf:ID="R">
   <rdfs:domain>
     <owl:Class  rdf:about="#C"  />
   </rdfs:domain>
</owl:ObjectProperty>
```

OWL - range

Di una proprietà **R** può essere specificato il codominio attraverso l'operatore **rdfs:range**

Sintassi ***RDF***

```
<owl:ObjectProperty  rdf:ID="R">
    <rdfs:range>
      <owl:Class  rdf:about="#D"  />
    </rdfs:range>
</owl:ObjectProperty>
```

Proprietà equivalente

Una proprietà **R** può essere definita come equivalente a un'altra proprietà **S** attraverso l'operatore **owl:equivalentProperty**

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
   <owl:equivalentProperty rdf:resource="#S" />
</owl:ObjectProperty>
```

Proprietà inversa

Data una proprietà **R** si può definire la proprietà inversa **S** attraverso l'operatore **owl:inverseOf**

Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="S">
    <owl:inverseOf rdf:resource="#R" />
</owl:ObjectProperty>
```

OWL – Property

3. global cardinality constraints (vincoli globali sulla cardinalità): **owl:FunctionalProperty** e **owl:InverseFunctionalProperty** . Una propietà di tipo functional è una proprietà che può avere solo ed esclusivamente un valore y per ciascuna istanza di x, cioè non possono esistere due valori distinti y1 ed y2 tali che le coppie (x,y1) e (x,y2) siano entrambe istanze di questa proprietà. Un esempio per capire questo concetto dichiara che la proprietà marito è un functional, cioè che una donna può avere al massimo un marito; questo è anche un esempio di relazione ontologica che deriva dalla cultura.

```
<owl:ObjectProperty rdf:ID="husband">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:domain rdf:resource="#Woman" />
    <rdfs:range rdf:resource="#Man" />
</owl:ObjectProperty>
```

– Se una proprietà è dichiarata essere “inverse-functional”, allora l’oggetto di una dichiarazione di proprietà determina univocamente il soggetto. Esempi di owl:InverseFunctionalProperty sono: “èCodiceFiscaleDi”, èNumeroDiMatricolaDi”.

OWL – Property

4. logical property characteristics: **owl:TransitiveProperty** e **owl:SymmetricProperty** . Quando una proprietà P è definita essere transitiva, significa che se una coppia (x,y) è un'istanza di P e la coppia (y,z) è anch'essa istanza di P, allora si può dedurre che la coppia (x,z) è anch'essa un'istanza di P. Una proprietà è definita transitiva rendendola un'istanza della classe OWL owl:TransitiveProperty che è una sottoclasse di owl:ObjectProperty .
5. Una proprietà simmetrica è una proprietà dove vale che, se la coppia (x,y) è un'istanza di P, allora la coppia (y,x) è anch'essa un'istanza di P. Una proprietà è definita simmetrica rendendola un'istanza della classe OWL owl:SimmetricProperty che è una sottoclasse di owl:ObjectProperty . Un esempio tipico è la relazione "friendOf" (amico di):

```
<owl:SymmetricProperty rdf:ID="friendOf">
    <rdfs:domain rdf:resource="#Human"/>
    <rdfs:range rdf:resource="#Human"/>
</owl:SymmetricProperty>
```

– Si noti che il domain ed il range di una proprietà simmetrica sono lo stesso.

OWL – Transitività

In OWL è possibile dichiarare che una proprietà è transitiva tramite l'operatore

owl:TransitiveProperty

Sintassi ***RDF***

```
<owl:TransitiveProperty rdf:ID="R">
    <rdfs:domain rdf:resource="#D" />
    <rdfs:range rdf:resource="#D" />
</owl:TransitiveProperty>
```

OWL – Transitività

ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/):

```
<owl:ObjectProperty rdf:ID="subRegionOf">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Region"/>
  <rdfs:range  rdf:resource="#Region"/>
</owl:ObjectProperty>
```

OWL - Simmetria

In OWL è possibile dichiarare che una proprietà simmetrica tramite l'operatore **owl:SymmetricProperty**

Sintassi ***RDF***

```
<owl:SymmetricProperty rdf:ID="R">
    <rdfs:domain rdf:resource="#D" />
    <rdfs:range rdf:resource="#D" />
</owl:SymmetricProperty>
```

OWL - Simmetria

ESEMPIO (da W3C OWL Guide: http://www.w3.org/TR/2004/REC-owl-guide-20040210/):

```
<owl:SymmetricProperty rdf:ID="friendOf">
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range  rdf:resource="#Human"/>
</owl:SymmetricProperty>
```

OWL - Individui

- Gli individui sono definiti con assiomi individuali detti “facts” (fatti). Si considerano due tipi di fatti:
  - fatti riguardanti l’appartenenza ad una classe e valori propri degli individui;
  - fatti che riguardano identità individuale.

- Relativamente ai fatti del primo tipo, si può dire che molti fatti sono degli statement che indicano un’appartenza degli individui ad una classe e dei valori propri degli individui.

OWL - Individui

- Un esempio, preso dalla documentazione sull'OWL, è:

```
<Opera rdf:ID="Tosca">
    <hasComposer rdf:resource="#Giacomo_Puccini"/>
    <hasLibrettist rdf:resource="#Victorien_Sardou"/>
    <hasLibrettist rdf:resource="#Giuseppe_Giacosa"/>
    <hasLibrettist rdf:resource="#Luigi_Illica"/>
    <premiereDate rdf:datatype="&xsd;date">1900-01-14</premiereDate>
    <premierePlace rdf:resource="#Roma"/>
    <numberOfActs
    rdf:datatype="&xsd;positiveInteger">3</numberOfActs>
</Opera>
```

- Qui si vede che "Tosca" è un individuo della classe "Opera" ed i fatti quali la composizione da parte di Puccini e la data in cui è stata presentata per la prima volta, oltre ad altri fatti.

OWL - Individui

- Relativamente ad i fatti del secondo tipo (detti individual identity), l'OWL fornisce tre costrutti per dichiarare fatti sull'identità degli individui.
  1. owl:sameAs è usato per indicare che due URI reference si riferiscono allo stesso individuo, cioè i due individui hanno la stessa "identità". Un esempio classico si fa con riferimento alle persone: in questo caso Bill Clinton e William Jefferson Clinton indicano la stessa persona.

```
<rdf:Description rdf:about="#William_Jefferson_Clinton">
<owl:sameAs rdf:resource="#BillClinton"/>
</rdf:Description>
```

  - Questo costrutto è molto importante quando si definiscono mappe tra varie ontologie, visto che è impossibile che tutti si riferiscano alla stessa cosa (individuo) con lo stesso nome.

OWL - Individui

2. owl:differentFrom collega un individuo ad un altro ed è usato per indicare che due URI reference si riferiscono a individui diversi.
3. owl:AllDifferent fornisce uno strumento per dichiarare che gli individui di una lista sono tutti diversi. Considerando come esempio le opere.

```
<owl:AllDifferent>
   <owl:distinctMembers rdf:parseType="Collection">
      <Opera rdf:about="#Don_Giovanni"/>
      <Opera rdf:about="#Nozze_di_Figaro"/>
      <Opera rdf:about="#Cosi_fan_tutte"/>
      <Opera rdf:about="#Tosca"/>
      <Opera rdf:about="#Turandot"/>
      <Opera rdf:about="#Salome"/>
   </owl:distinctMembers>
</owl:AllDifferent>
```

- Qui si dichiara che questi sei URIref puntano tutti ad opere diverse.

OWL - Individui

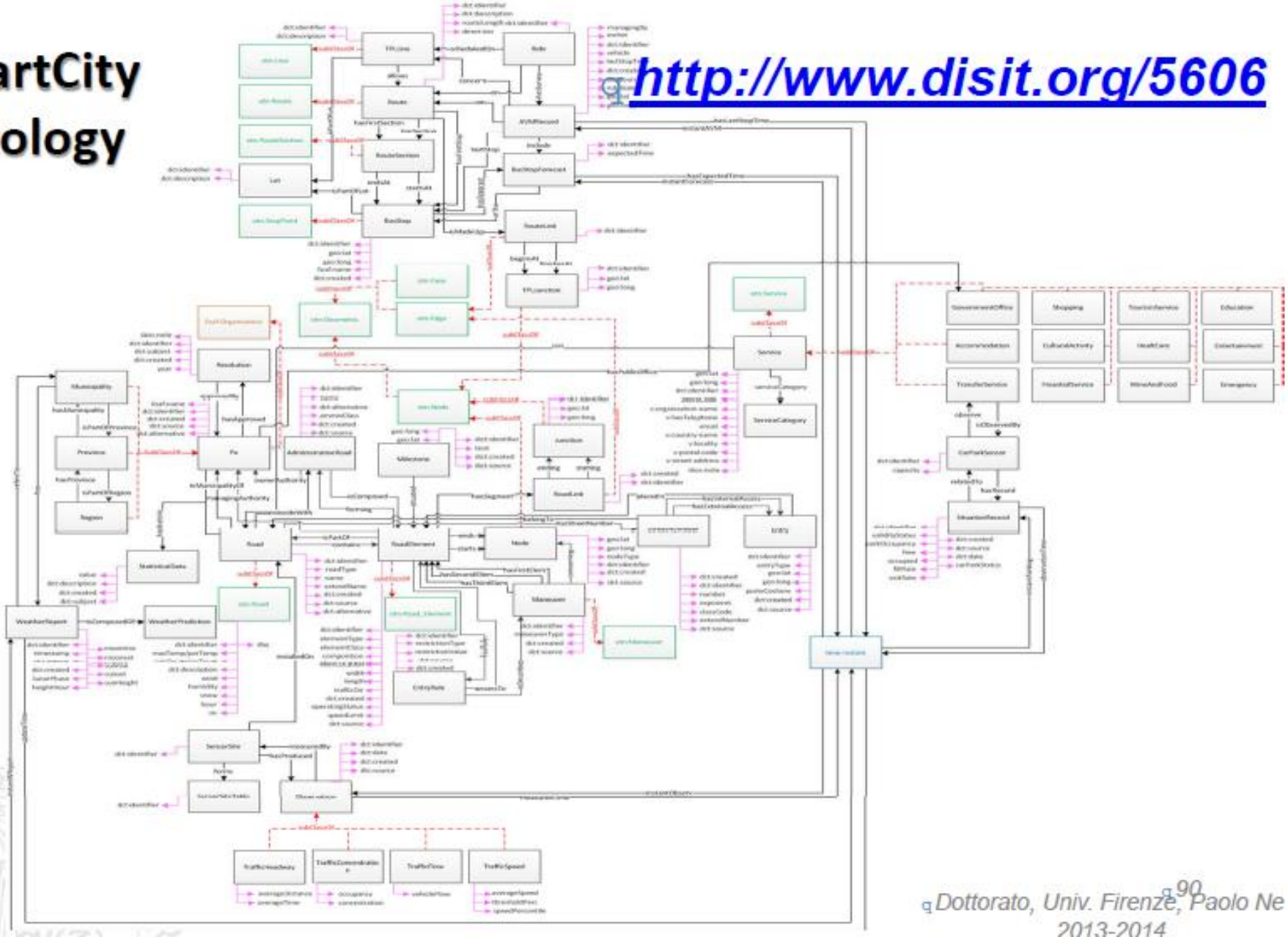
È possibile asserire che due nomi fanno riferimento ad individui distinti tramite l'operatore **owl:differentFrom**

Sintassi ***RDF***

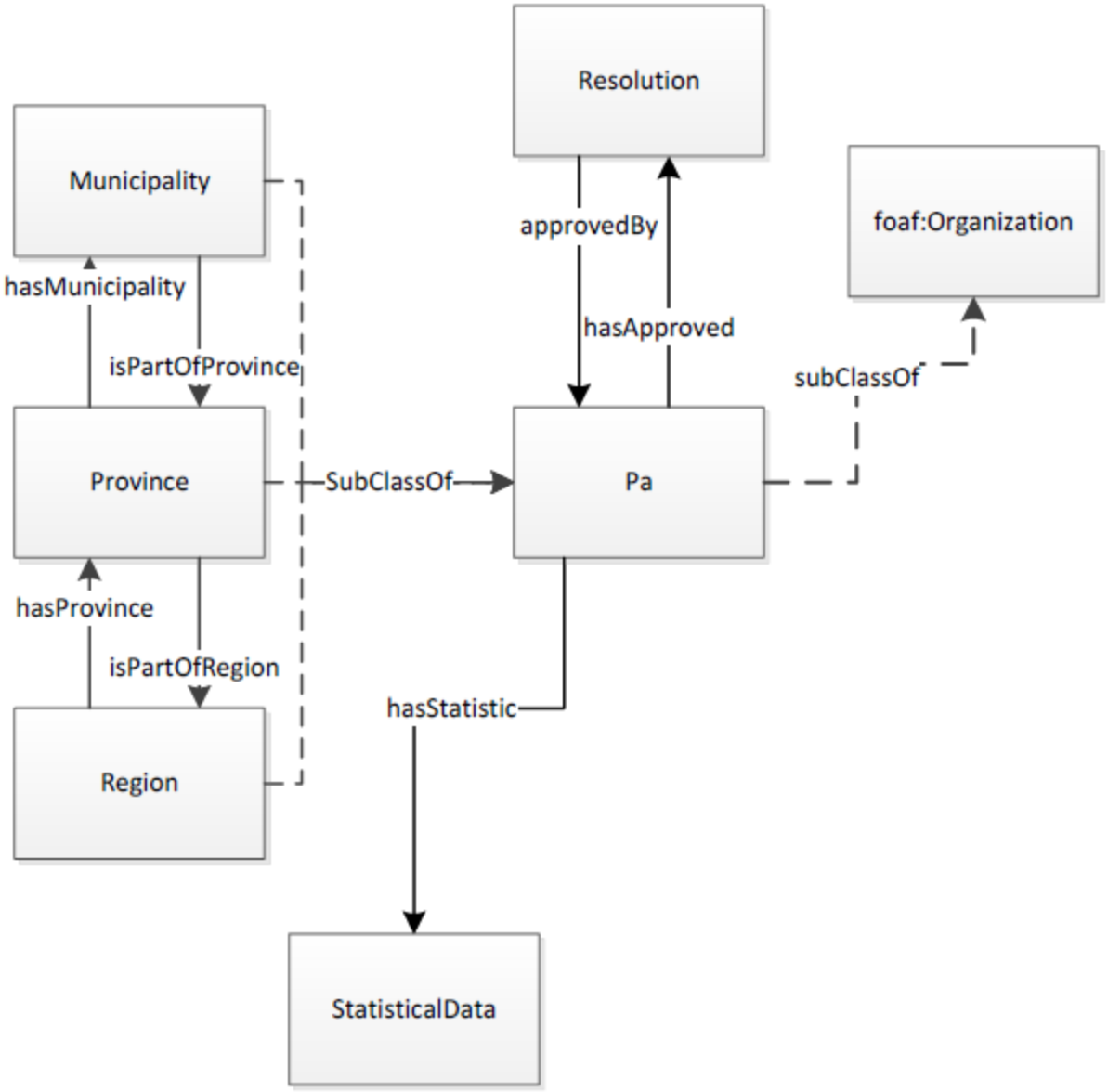
```
<C  rdf:ID="a">
    <owl:differentFrom  rdf:resource="#b"/>
    …
</C>
```

SmartCity Ontology

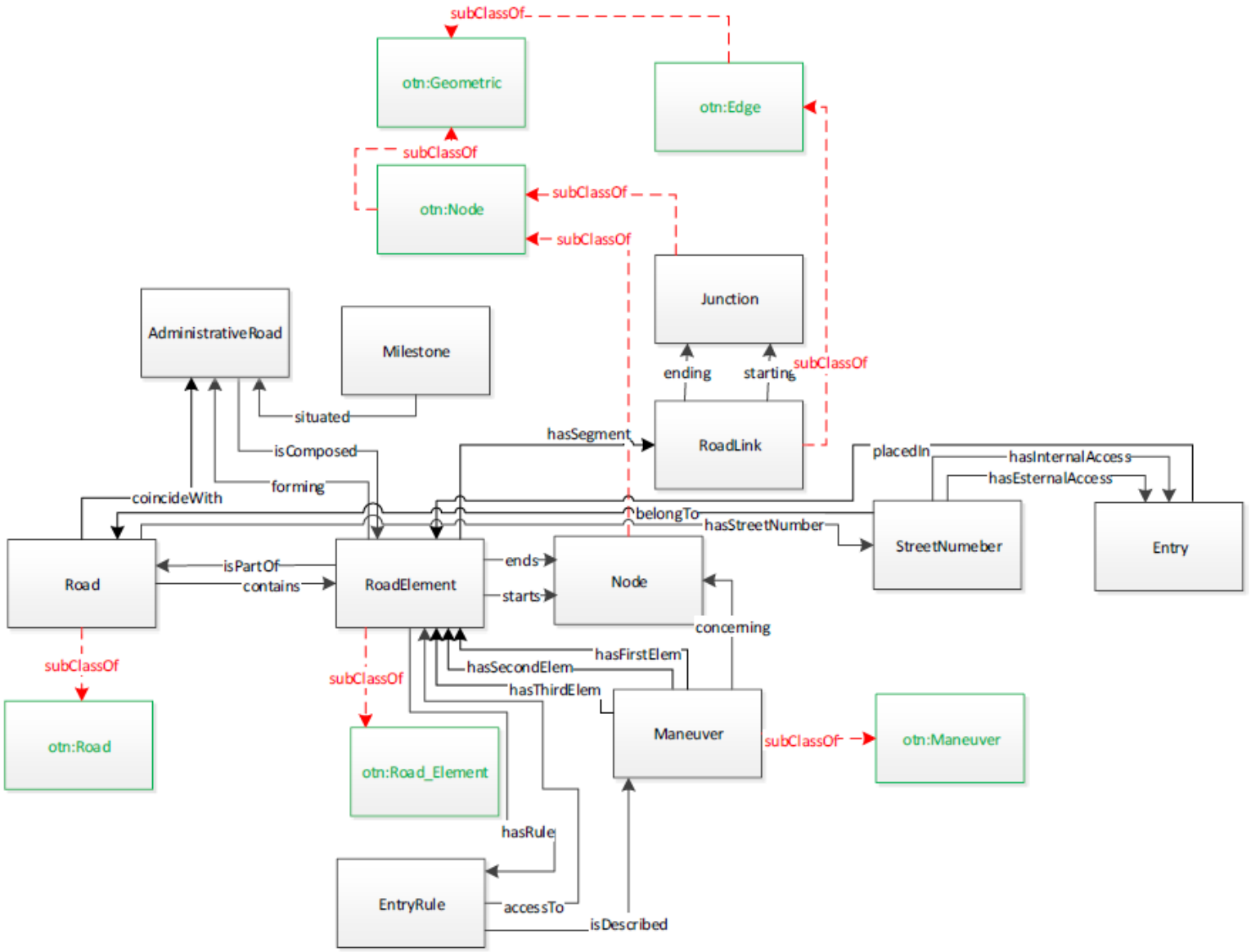
http://www.disit.org/5606



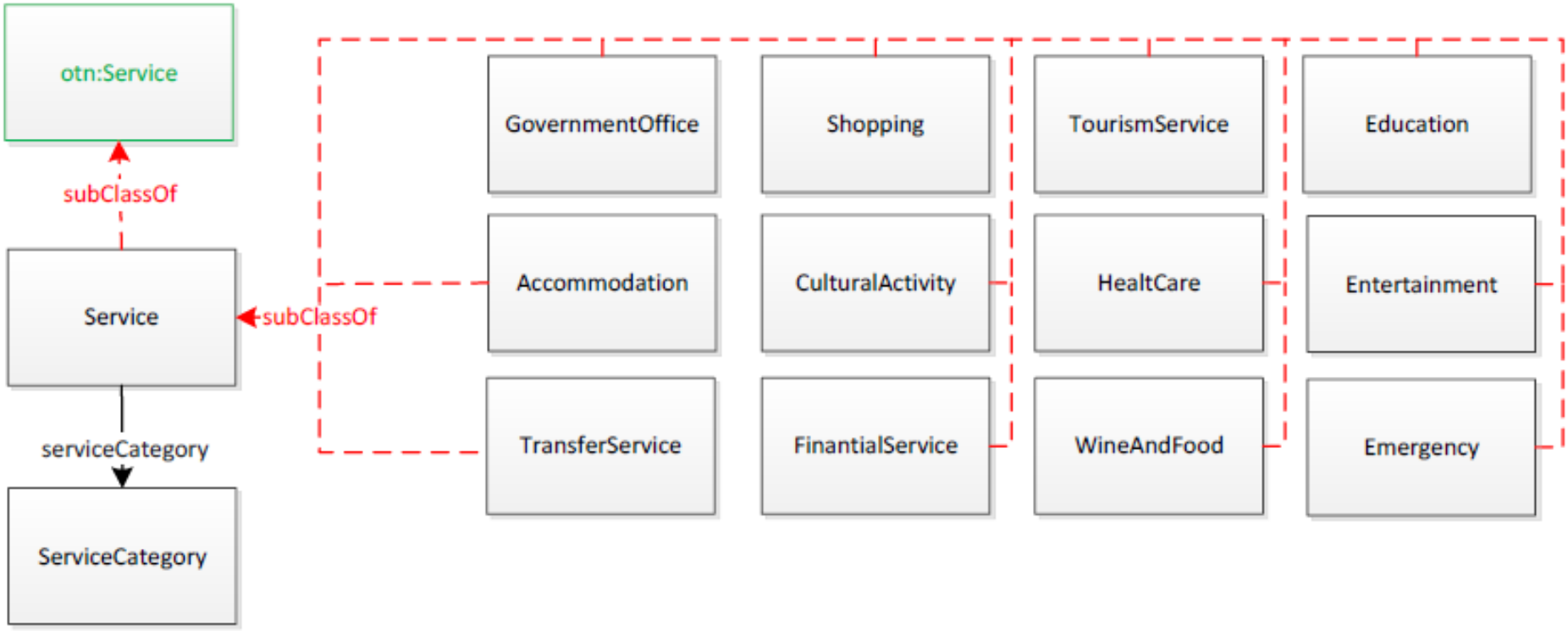
SiiMobility - Administration



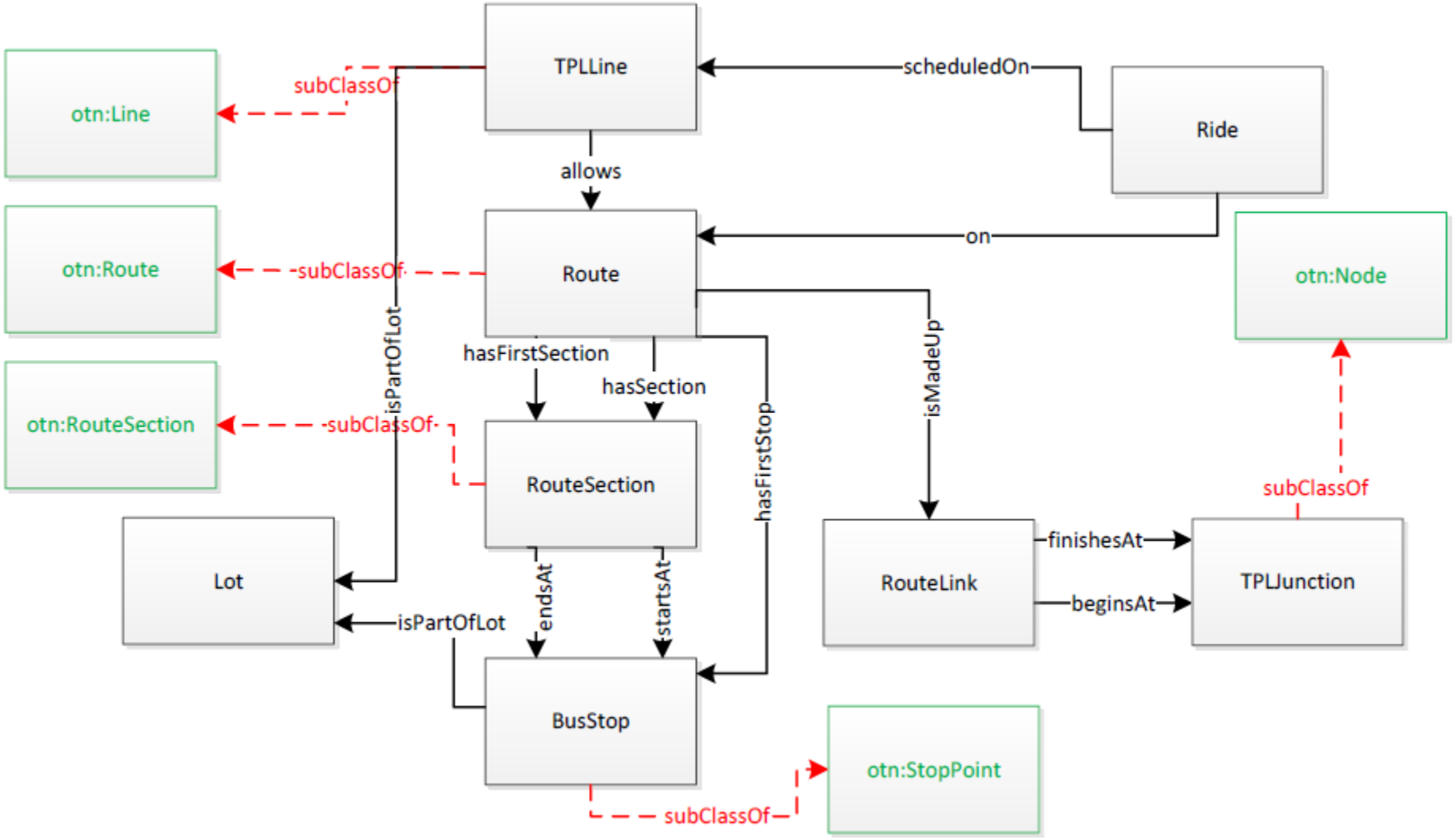
SiiMobility – Street Guide



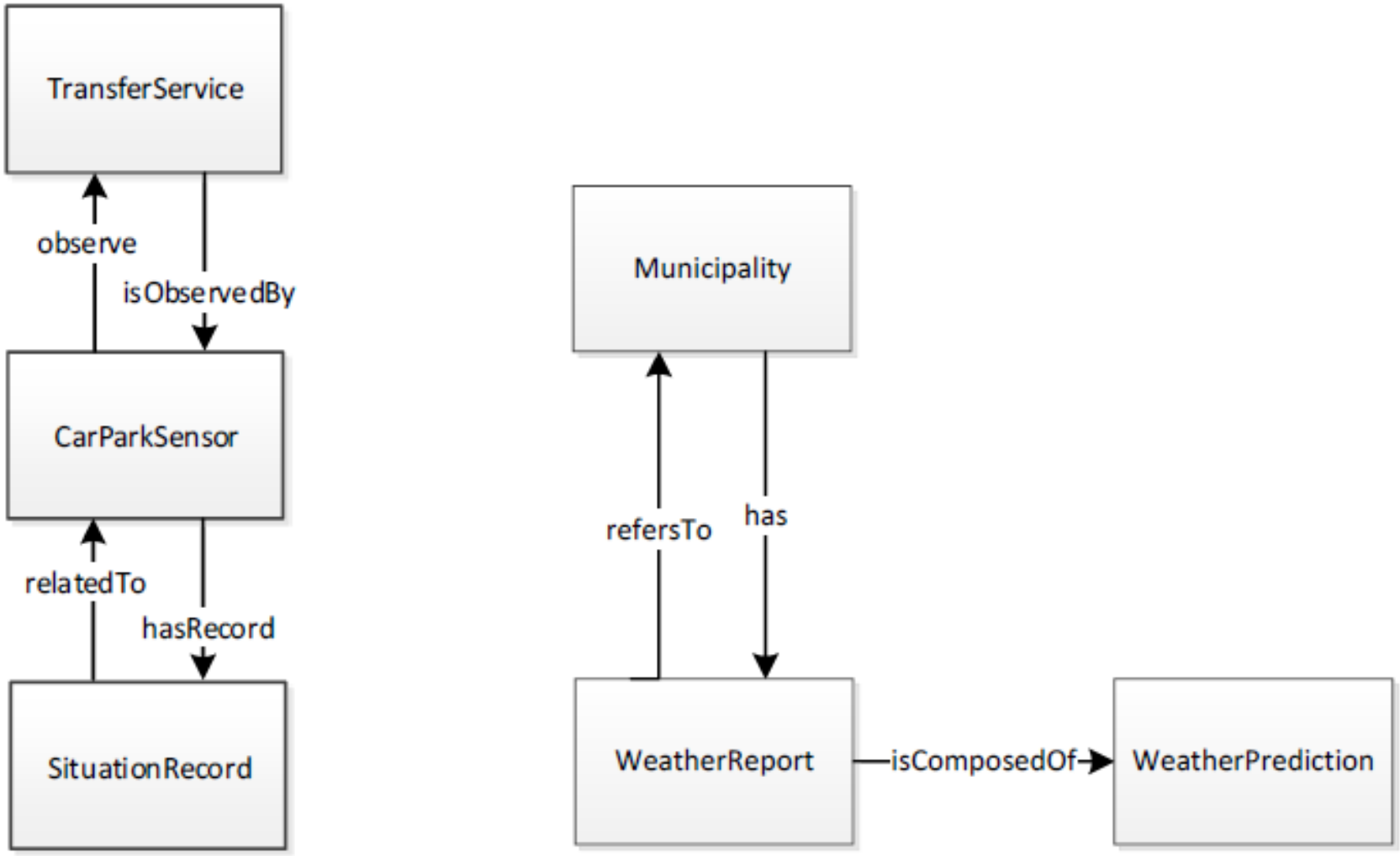
SiiMobility - Service



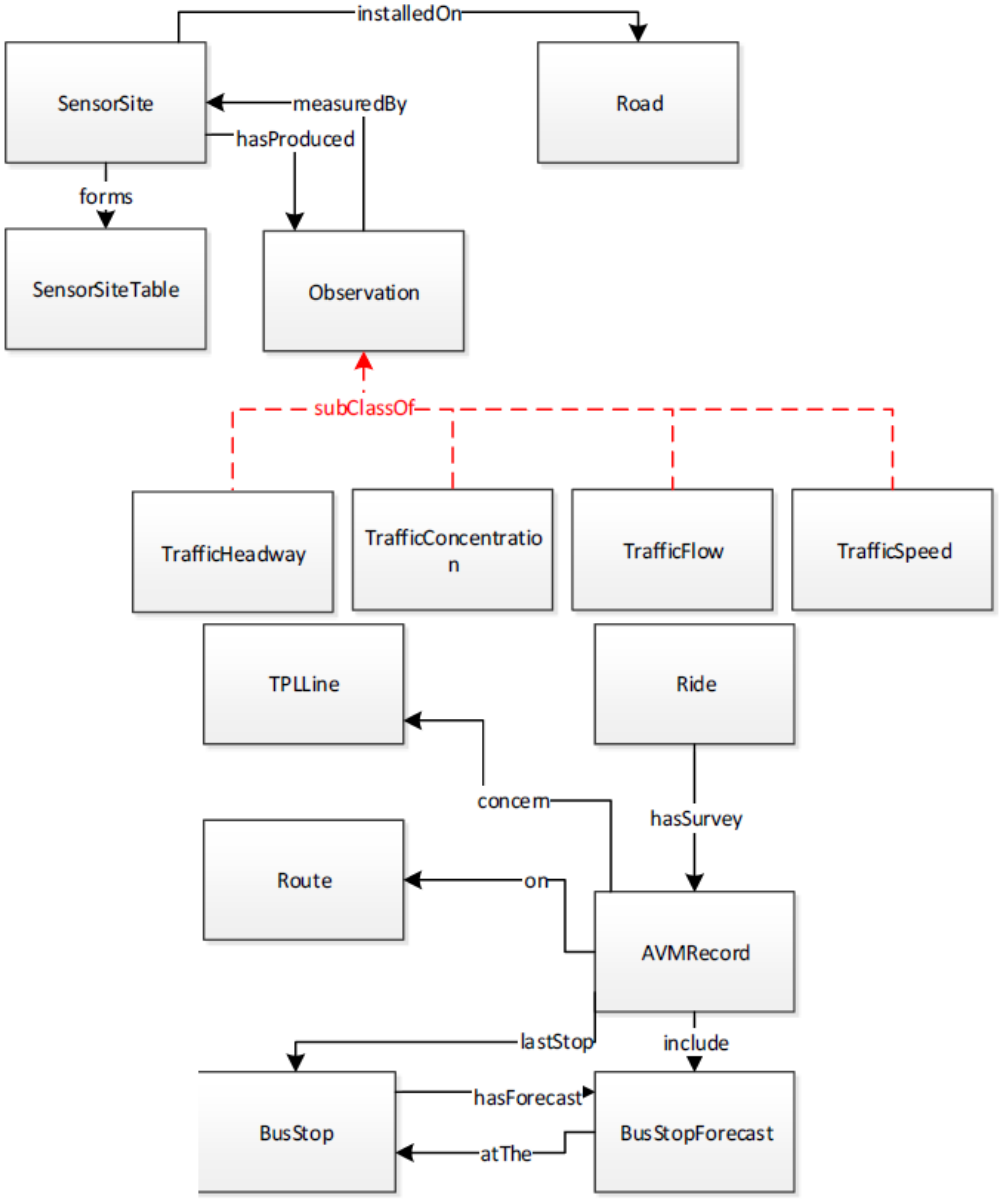
SiiMobility - TPL



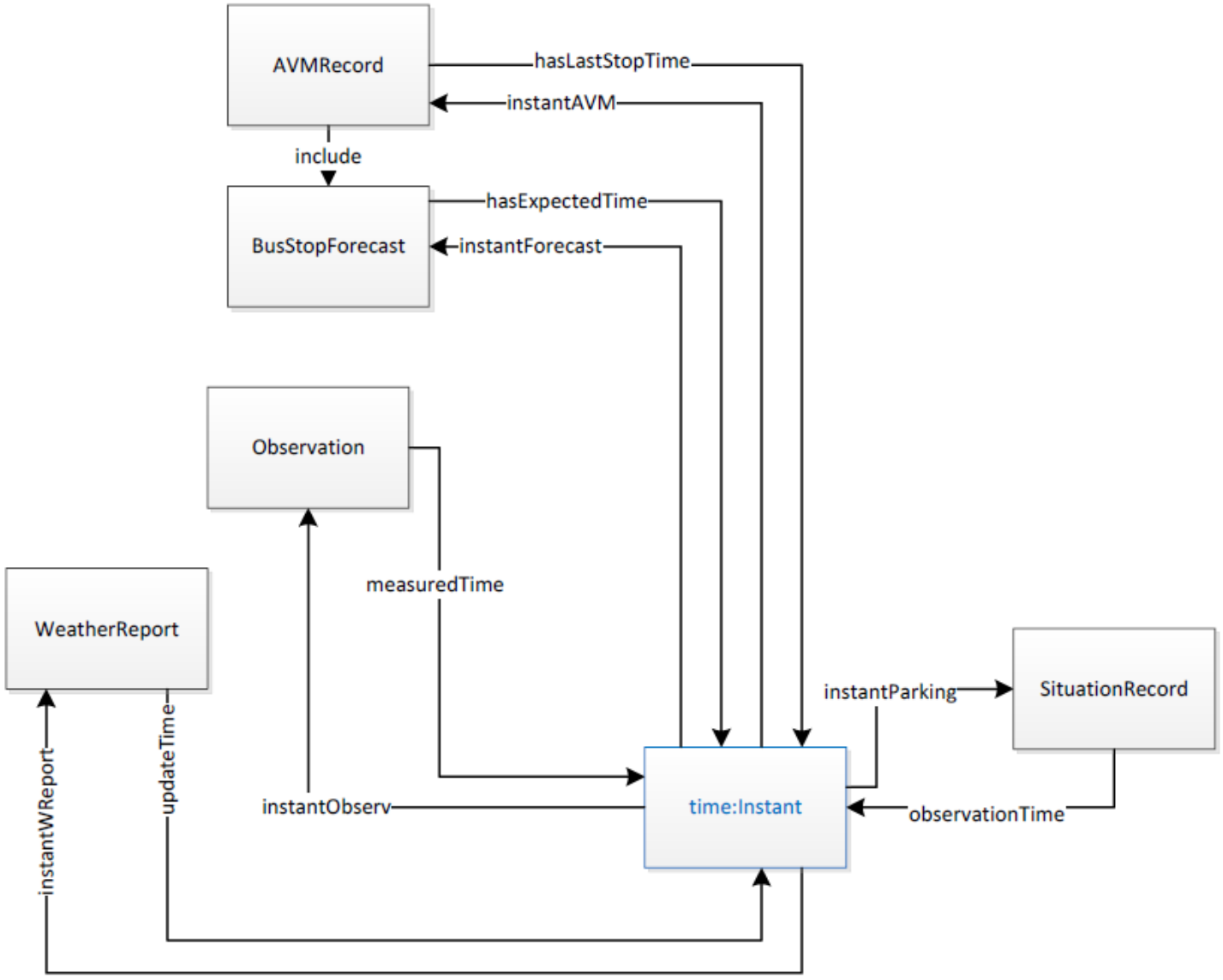
SiiMobility – Sensori Parcheggio e Previsioni Meteo



SiiMobility – Sensori Traffico



SiiMobility - AVM



OSIM Ontology

- **L'ontologia di dominio di OSIM è composta da 4 ontologie diverse**
  - *Academy Life Ontology (Unifi) modella l'ateneo fiorentino in termini di docenti, corsi, strutture di affiliazione, facoltà,* gruppi di ricerca, laboratori, ecc…
  - *Friend of a Friend* (FOAF) modella le persone in termini di professori, ricercatori, phd e relazioni tipo nome, indirizzo, e-mail, settore scientifico, relazioni di conoscenza, di co-autore di pubblicazioni, ecc …..
  - Simple Knowledge Organization System (SKOS) che modella ed organizza semanticamente le competenze delle persone e dei corsi.
  - Time Ontology (TIME) che modella I concetti di intervalli ed istanti temporali per quantificare temporalmente i fatti asseriti nell'ontologia.

Friend Of A Friend Ontology (FOAF)

- **Descrizione dello schema**

  – *Friend of a Friend* (FOAF) descrive le persone in termini di links tra di esse. Tali collegamenti riguardano:

    - Relazioni di conoscenza
    - Appartenenza ad Organizzazioni
    - Membro di …

  – *Descrive anche le attività delle persone*:

    - Documentazione prodotta
    - Topic di interesse
    - etc…

FOAF Core

- Agent
- Person
- name
- title
- img
- depiction (depicts)
- familyName
- givenName
- knows
- based_near
- age
- made (maker)
- primaryTopic (primaryTopicOf)
- Project
- Organization
- Group
- member
- Document
- Image

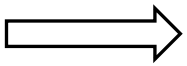
Social Web

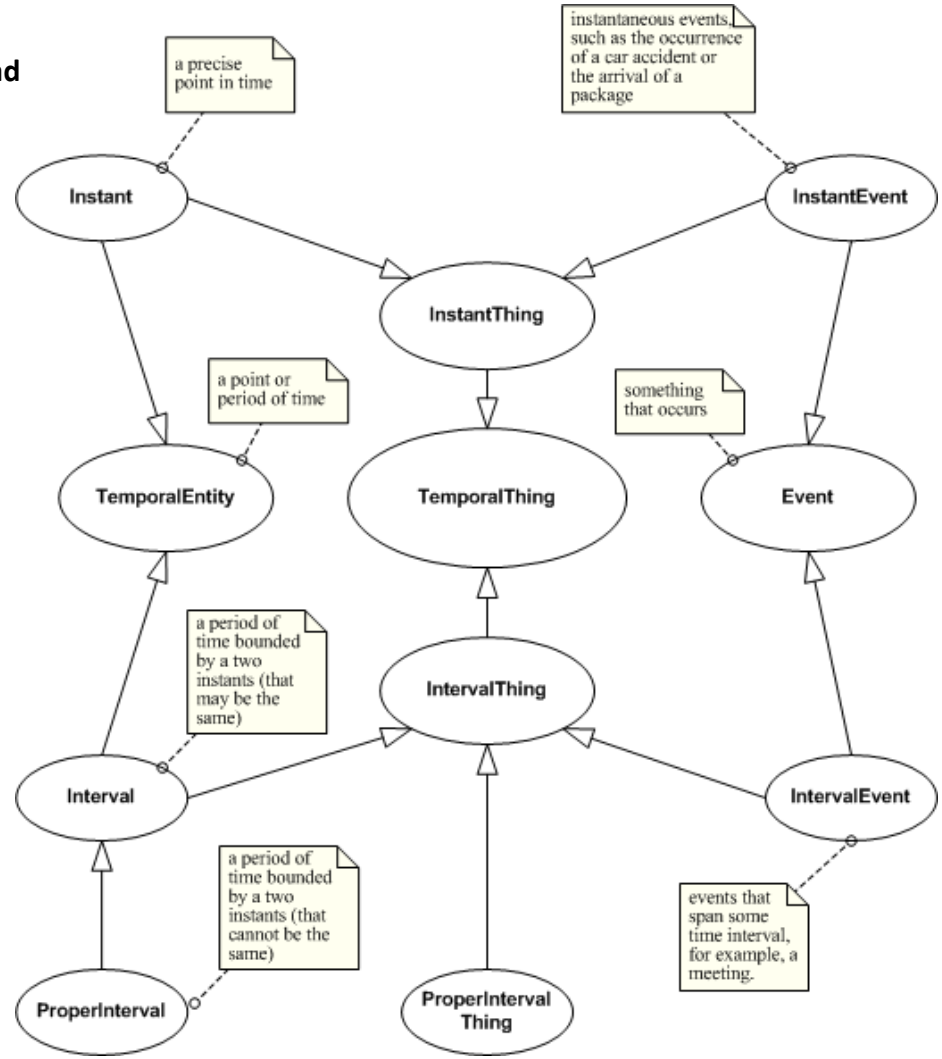
- nick
- mbox
- homepage
- weblog
- openid
- jabberID
- mbox_sha1sum
- interest
- topic_interest
- topic (page)
- workplaceHomepage
- workInfoHomepage
- schoolHomepage
- publications
- currentProject
- pastProject
- account
- OnlineAccount
- accountName
- accountServiceHomepage
- PersonalProfileDocument
- tipjar
- sha1
- thumbnail
- logo

Time Ontology

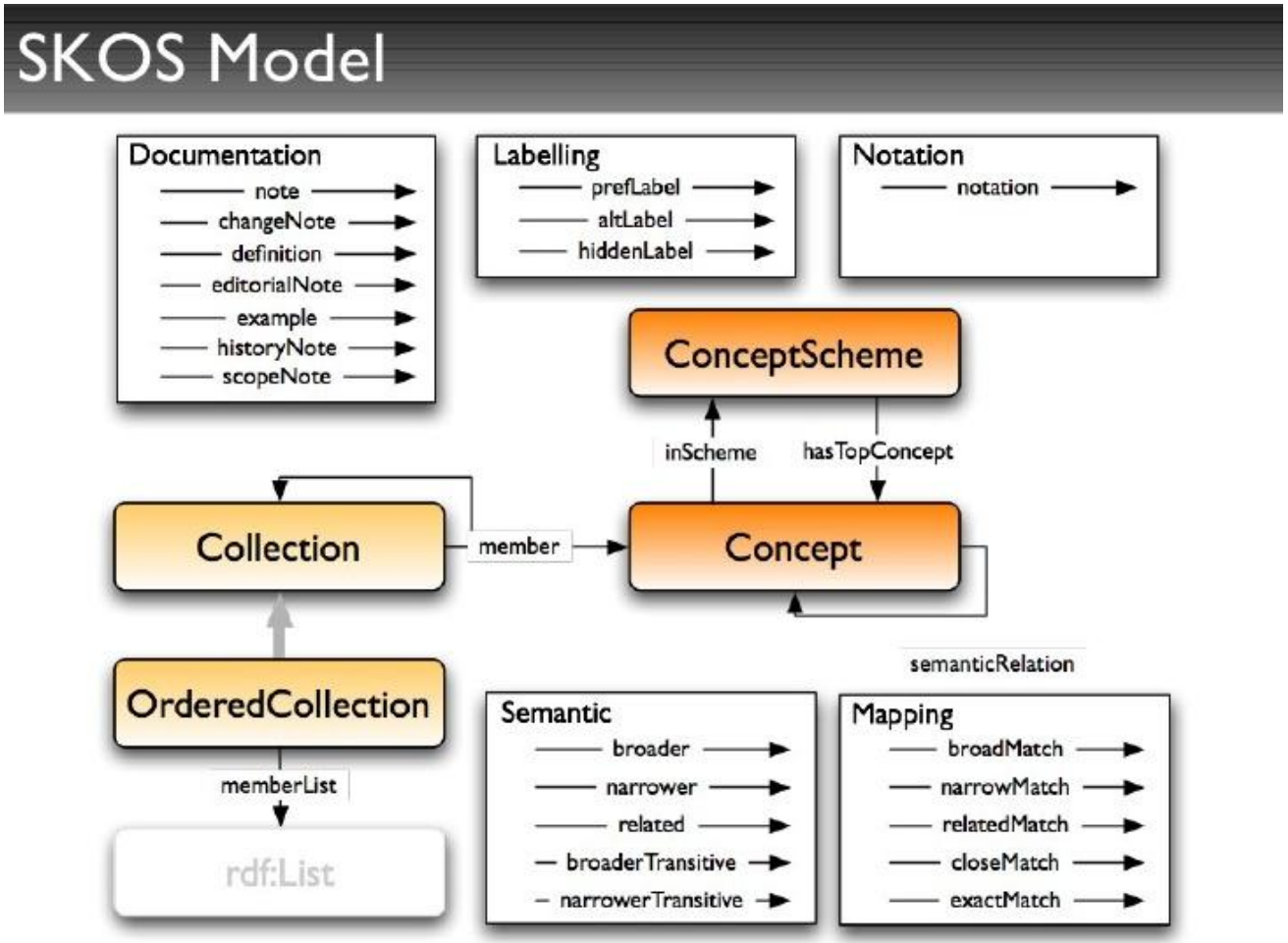
Si specifica il valore di un istante ad una certa granularità:

- **Ora**
- **Giorno**
- **Mese**
- **Anno**
- **…**





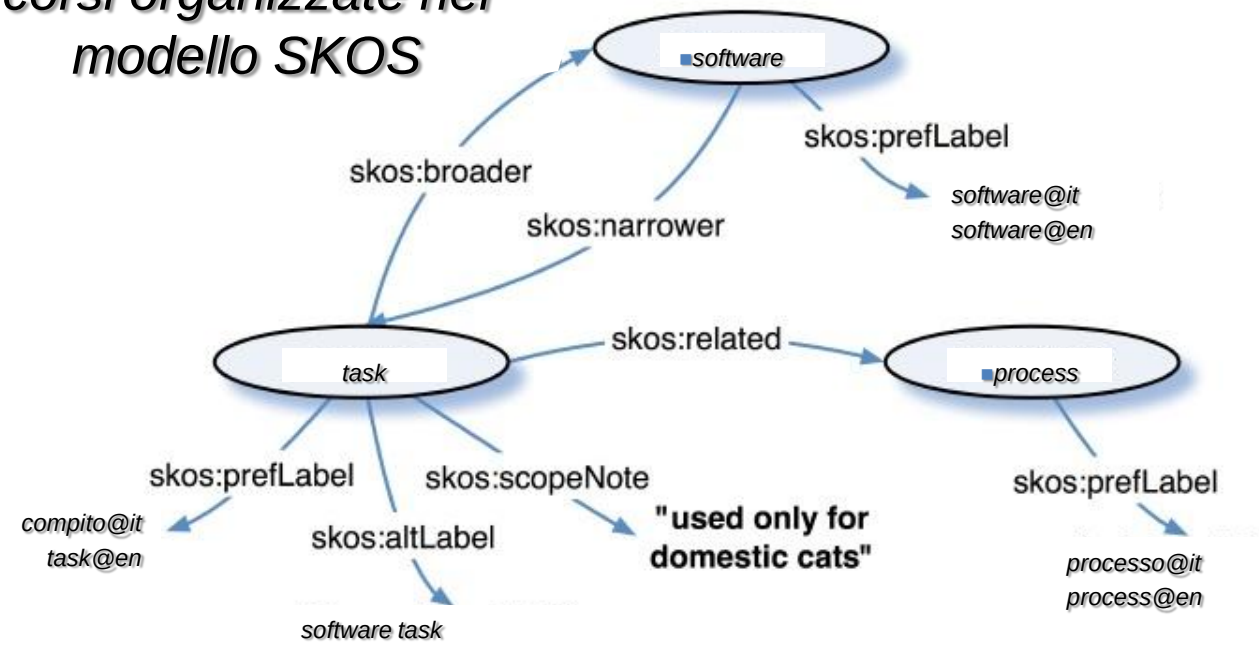
Simple Knowledge Organization System (SKOS)

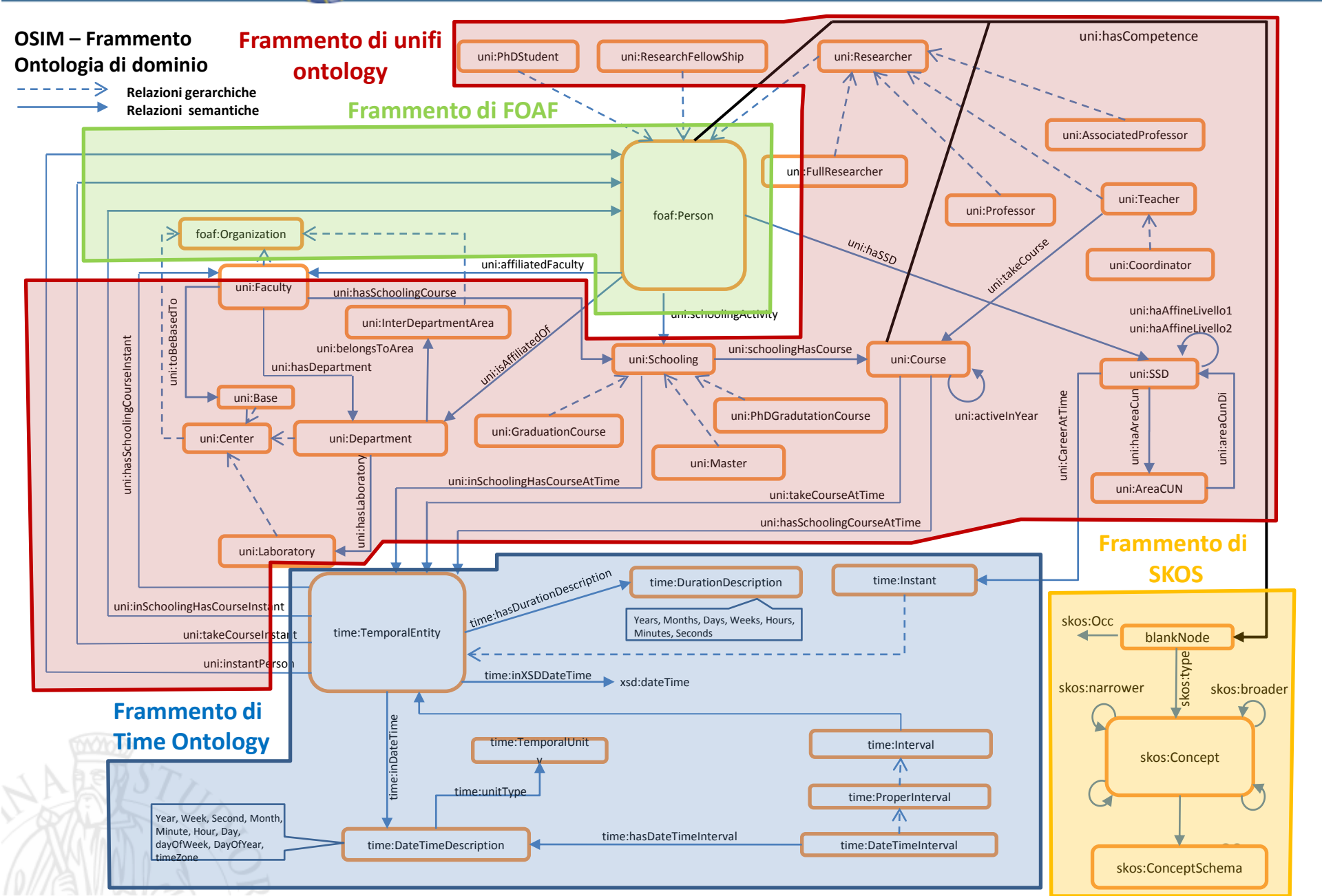


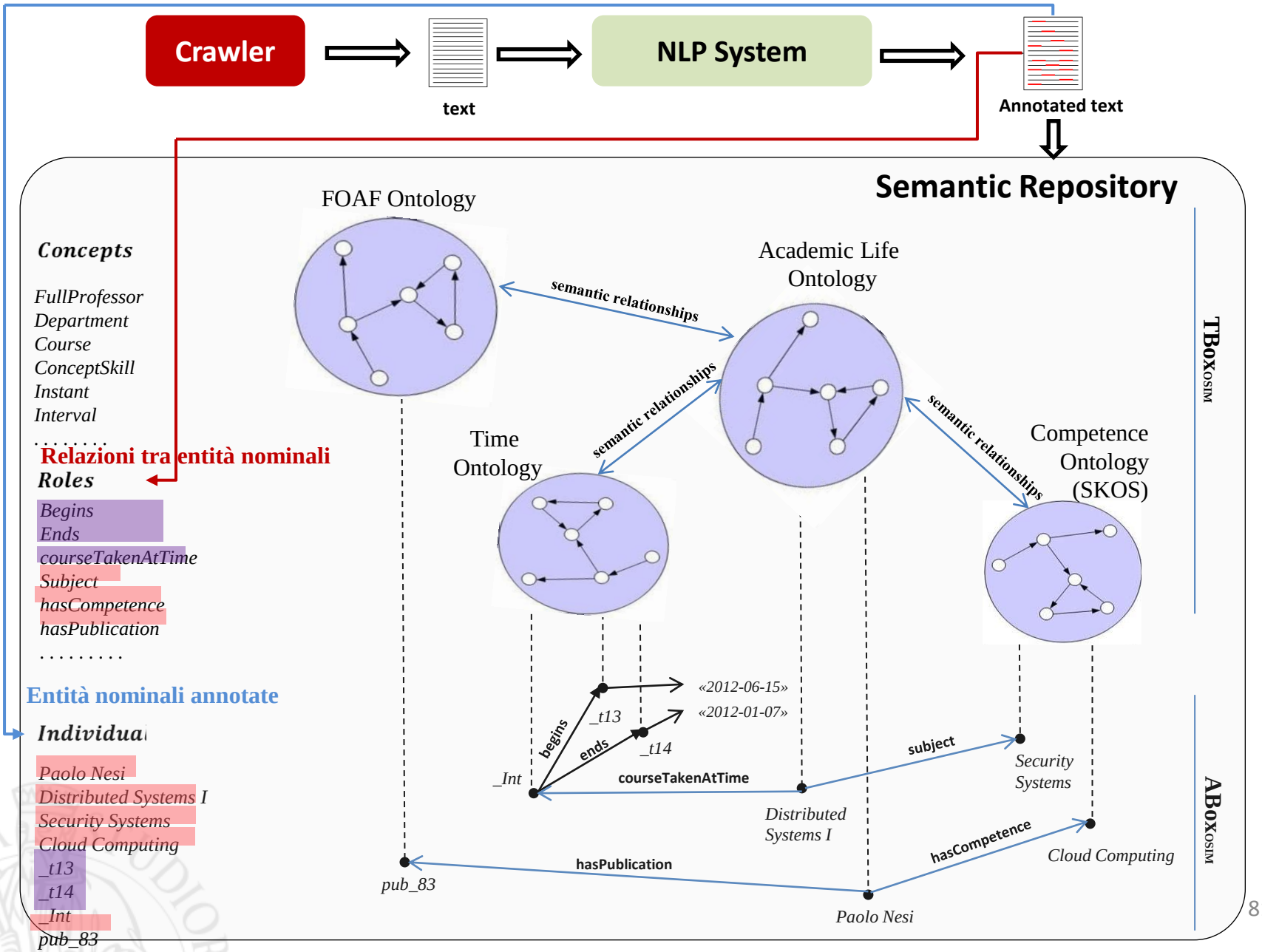
SKOS di OSIM

SKOS Example

Competenze di docenti e corsi organizzate nel modello SKOS




OSIM – Frammento Ontologia di dominio
Relazioni gerarchiche
Relazioni semantiche
Frammento di unifi ontology
Frammento di FOAF
uni:PhDStudent
uni:ResearchFellowShip
uni:Researcher
uni:hasCompetence
uni:AssociatedProfessor
uni:FullResearcher
uni:Professor
uni:Teacher
uni:Coordinator
foaf:Person
foaf:Organization
uni:haSSD
uni:takeCourse
uni:affiliatedFaculty
uni:Faculty
uni:hasSchoolingCourse
uni:InterDepartmentArea
uni:haAffineLivello1
uni:haAffineLivello2
uni:toBeBasedTo
uni:belongsToArea
uni:hasDepartment
uni:isAffiliatedOf
uni:Schooling
uni:schoolingHasCourse
uni:Course
uni:SSD
uni:hasSchoolingCourseInstant
uni:Base
uni:Center
uni:Department
uni:GraduationCourse
uni:PhDGradutationCourse
uni:activeInYear
uni:CareerAtTime
uni:haAreaCun
uni:areaCunDi
uni:Master
uni:AreaCUN
uni:hasLaboratory
uni:inSchoolingHasCourseAtTime
uni:takeCourseAtTime
uni:hasSchoolingCourseAtTime
uni:Laboratory
Frammento di SKOS
uni:inSchoolingHasCourseInstant
uni:takeCourseInstant
uni:instantPerson
time:TemporalEntity
time:hasDurationDescription
time:DurationDescription
Years, Months, Days, Weeks, Hours, Minutes, Seconds
time:Instant
time:inXSDDateTime
xsd:dateTime
Frammento di Time Ontology
time:inDateTime
time:TemporalUnit
time:unitType
time:Interval
time:ProperInterval
Year, Week, Second, Month, Minute, Hour, Day, dayOfWeek, DayOfYear, timeZone
time:DateTimeDescription
time:hasDateTimeInterval
time:DateTimeInterval
skos:Occ
blankNode
skos:type
skos:narrower
skos:broader
skos:Concept
skos:ConceptSchema


Crawler
text
NLP System
Annotated text
Semantic Repository
FOAF Ontology
Academic Life Ontology
Time Ontology
Competence Ontology (SKOS)
semantic relationships
semantic relationships
semantic relationships
TBoxOSIM
ABoxOSIM
Concepts
FullProfessor
Department
Course
ConceptSkill
Instant
Interval
. . . . . .
Relazioni tra entità nominali
Roles
Begins
Ends
courseTakenAtTime
Subject
hasCompetence
hasPublication
. . . . . .
Entità nominali annotate
Individual
Paolo Nesi
Distributed Systems I
Security Systems
Cloud Computing
_t13
_t14
_Int
pub_83
«2012-06-15»
«2012-01-07»
_t13
_t14
begins
ends
_Int
courseTakenAtTime
subject
Security Systems
Distributed Systems I
hasPublication
hasCompetence
Cloud Computing
pub_83
Paolo Nesi