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

Progettazione e realizzazione di un algoritmo per la generazione automatica di un'ontologia basata sullo standard HL7 CDA Rel. 2

Angelo Esposito, Aniello Minutolo, Massimo Esposito, Mario Ciampi

RT-ICAR-NA-2014-3 **Ottobre 2014**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

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

Progettazione e realizzazione di un algoritmo per la generazione automatica di un'ontologia basata sullo standard HL7 CDA Rel. 2

Angelo Esposito, Aniello Minutolo, Massimo Esposito, Mario Ciampi

Rapporto Tecnico N: **RT-ICAR-NA-2014-3** ***Data:*** **Ottobre 2014**

Progettazione e realizzazione di un algoritmo per la generazione automatica di un'ontologia basata sullo standard HL7 CDA Rel. 2

Angelo Esposito, Aniello Minutolo, Massimo Esposito, Mario Ciampi

Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche
Via Pietro Castellino, 111 – 80131 Napoli, Italia
E-mail: {angelo.esposito, aniello.minutolo, massimo.esposito, mario.ciampi}@na.icar.cnr.it

Abstract

Lo sviluppo delle tecnologie informatiche e le nuove normative in materia di digitalizzazione nel settore dell'e-health stanno portando negli ultimi anni ad un aumento rapidissimo delle informazioni sanitarie disponibili in formato elettronico. Una parte di esse sono contenute in forma semi-strutturata o non strutturata all'interno di documenti aventi diversi formati elettronici (XML, HTML, PDF, DOC, testo piano, ecc.), mentre la restante parte è memorizzata, in misura sempre crescente, in sistemi informativi sanitari in forma strutturata adoperando lo standard HL7 CDA 2. Lo standard HL7 CDA 2, come verrà introdotto in questo lavoro, usa il linguaggio XML Schema Language ([XMLSCHEMA 2004]) per definire struttura e vincoli di un documento in formato CDA. Il linguaggio XML ha delle limitazioni legate all'espressività semantica dei concetti definiti nei documenti. Infatti, tale linguaggio permette di rappresentare la struttura dei dati ma non consente la rappresentazione o la deduzione tramite inferenza di complesse relazioni semantiche tra i dati stessi.

In questo ambito, l'utilizzo delle ontologie per la rappresentazione della semantica dei dati sanitari, risulta di particolare interesse nel campo scientifico perché permetterebbe di superare gli attuali limiti del linguaggio XML nella rappresentazione della conoscenza. Con tale obiettivo, questo lavoro presenta una metodologia ed un software sviluppato che consente di generare automaticamente modelli ontologici a partire da documenti in formato XML Schema.

Il software sviluppato è stato usato per la generazione automatica di un modello ontologico a partire dagli XSD Schemas che implementano la versione Clinical Document Architecture Rel. 2 (CDA rel. 2) di Health Level Seven (HL7). La particolarità della metodologia proposta riguarda la definizione di una fase di analisi preliminare degli XSD Schemas per l'ottimizzazione degli stessi eliminando tutte le informazioni non necessarie e semplificando in questo modo sensibilmente l'ontologia di riferimento prodotta.

Il rapporto tecnico si articola in 5 sezioni, nella prima sezione viene descritto brevemente lo standard HL7 CDA 2, nella seconda viene presentata la metodologia proposta, nella terza viene applicata la metodologia prodotta ad un caso di studio, nella quarta viene analizzata l'ontologia prodotta per il caso di studio in esame, il lavoro si conclude con una sezione dedicata alle conclusioni.

Keywords: Automatic ontology generation, standards, XML Schema, OWL, Ontologies, Clinical Document Architecture.

1 Introduzione

Lo standard HL7 nasce nel 1987 per rispondere alla crescente necessità di avere a disposizione uno standard di messaggistica da utilizzare per il trasferimento di informazioni cliniche codificate tra sistemi informativi eterogenei. L'evoluzione più importante, che porta alla versione 3 dello standard HL7, avviene dopo l'introduzione del Reference Information Model (RIM). Il RIM rappresenta il modello informativo globale dal quale tutti gli altri modelli informativi definiti nei diversi domini coperti da HL7v3 sono derivati. Il RIM è specificato tramite un modello UML e definisce l'insieme delle classi che rappresentano entità, ruoli e partecipazioni ammessi.

La modellazione del RIM è basata sulla definizione di sei tipologie di classi principali, dalle quali tutte le altre classi del RIM derivano. Le core Classes del RIM sono:

- **Classe Entity**: che descrive ogni soggetto in grado di interagire con altri soggetti, nell'universo clinico preso in considerazione;

- **Classe Act**: che descrive le azioni che accadono e che identifica il tipo di azione, il soggetto che esegue l'azione e gli altri soggetti (o oggetti) coinvolti.

- **Classe Role**: che descrive il ruolo con il quale un determinato soggetto (entity) partecipa ad una Act.

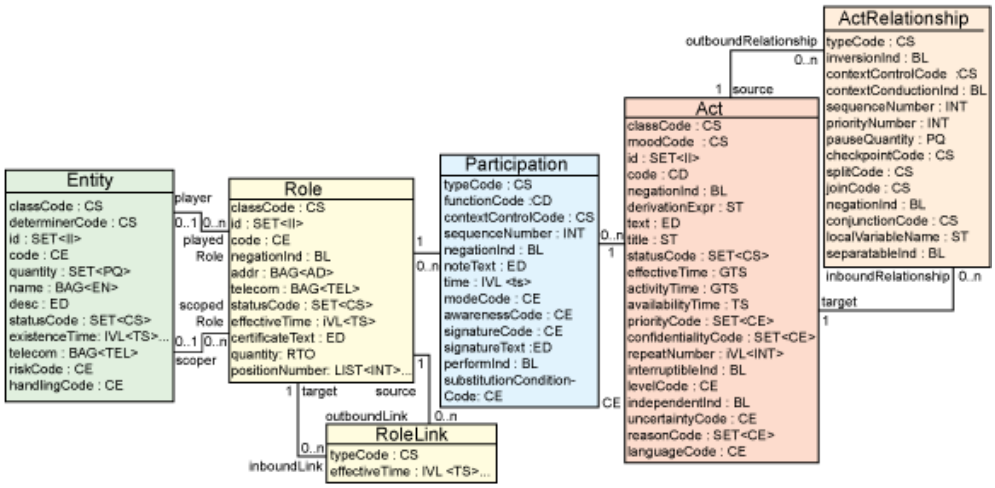


Figura 1 - RIM core classes

- **Classe Participation**: che completa la descrizione del processo sanitario, fornendo informazioni su come una Entity, che svolge uno specifico Role in una determinato Act, o che partecipa all'esecuzione di una Act stesso;
- **Classe ActRelationship**: che permette di scomporre una Act complessa in una serie di Act più semplici;
- **Classe RoleLink**: che consente di associare fra loro due classi Role;

Tutte le classi del RIM hanno una serie di attributi che consentono di descrivere le caratteristiche degli oggetti che appartengono a ciascuna classe.

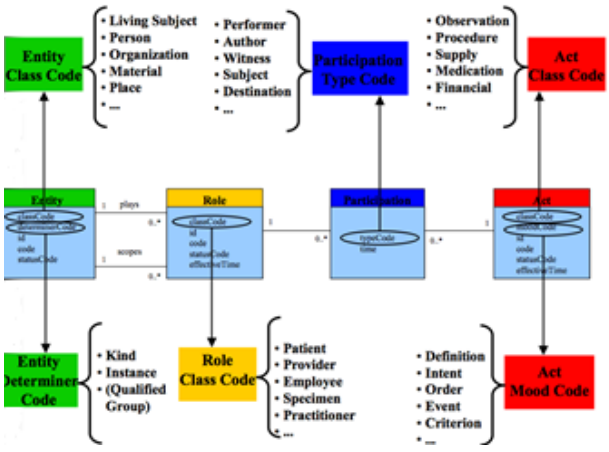


Figura 2 - RIM CORE STRUCTURAL ATTRIBUTES

Tutti gli standard basati su HL7 v.3, sono definiti a partire dal RIM mediante un processo formale di definizione del modelli informativo di contesto. Tale processo prevede i seguenti passi (**Errore. L'origine riferimento non è stata trovata.**):

- **Creazione del D-MIM (Domain Message Information Model)**: è un sottogruppo del RIM che include una serie di classi, attributi e relazioni che possono essere utilizzati per creare messaggi in un particolare dominio.
- **La definizione dell'R-MIM (Refined Message Information Model):** è un sottogruppo del D-MIM che viene utilizzato per esprimere il contenuto informativo di un messaggio o di un insieme di messaggi.
- **La specifica dell'HDM (Hierarchical Message Description):** permette di specificare l'ordine e vincoli di un particolare insieme di attributi e relazioni delle classi di interesse del RIM, da cui, attraverso l'Implementation Technology Specification (ITS), vengono generati gli schemi dei messaggi da scambiare sulla base di una specifica tecnologia (XML, pipe & bar, UML etc).

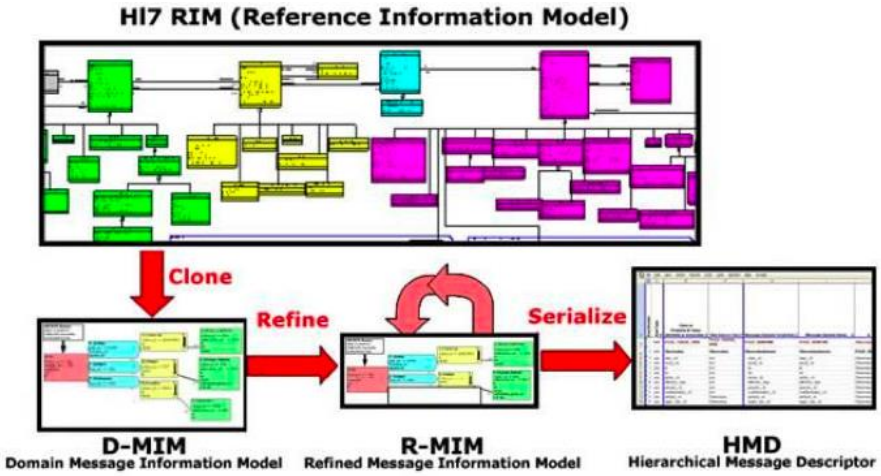
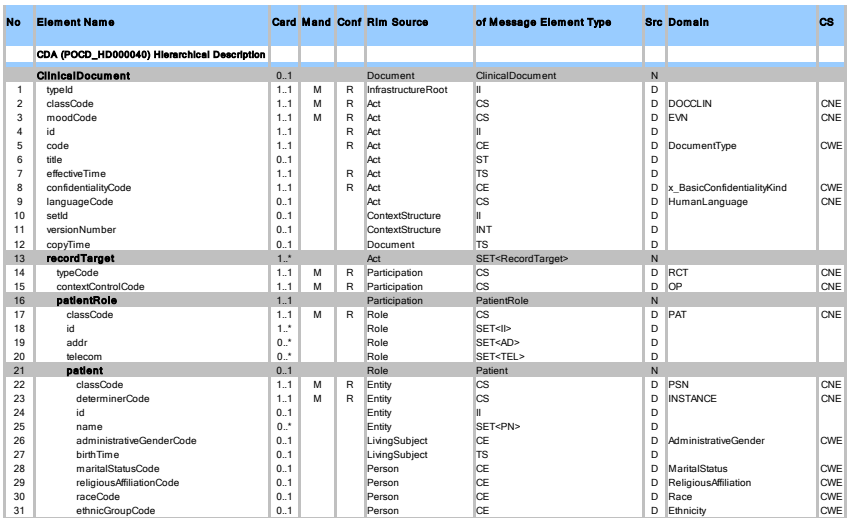


Figura 3 - Struttura dello standard RIM di HL7

Il modello R-MIM del CDA release 2 derivato dal RIM, secondo il processo formale descritto precedentemente, è uno standard che specifica la struttura e la semantica di documenti clinici per lo scambio nel dominio sanitario. Un documento CDA è un oggetto informativo strutturato in grado di contenere testi, immagini, suoni ed altri contenuti multimediali. Esso è composto da differenti blocchi informativi che veicolano informazioni relative ad esempio al paziente, al medico, alla struttura sanitaria, all'autore del documento, al firmatario del documento, agli eventi clinici, alle osservazioni o alle procedure mediche a cui il documento si riferisce.

La figura seguente mostra il documento HDM relativo alla strutturazione del documento CDA ottenuto secondo il processo di affinamento dal modello RIM HL7. Si notano gli elementi PatientRole e Patient ed i datatype specifici per ogni attributo.

Un documento CDA è composto da un header e un body. L'header del documento identifica e classifica il documento, fornisce informazioni sull'autenticazione, sul paziente, sull'evento di cura e sugli attori sanitari coinvolti. Il body del CDA contiene il report clinico e può alternativamente contenere un corpo non strutturato (ad es.: BLOB PDF) o un insieme di markup che ne descrivono il contenuto.



No	Element Name	Card	Mand	Conf	Rim Source	of Message Element Type	Src	Domain	CS
	CDA (POCD_HD000040) Hierarchical Description								
	ClinicalDocument	0..1			Document	ClinicalDocument	N		
1	typeId	1..1	M	R	InfrastructureRoot	II	D		
2	classCode	1..1	M	R	Act	CS	D	DOCCLIN	CNE
3	moodCode	1..1	M	R	Act	CS	D	EVN	CNE
4	id	1..1		R	Act	II	D		
5	code	1..1		R	Act	CE	D	DocumentType	CWE
6	title	0..1			Act	ST	D		
7	effectiveTime	1..1		R	Act	TS	D		
8	confidentialityCode	1..1		R	Act	CE	D	x_BasicConfidentialityKind	CWE
9	languageCode	0..1			Act	CS	D	HumanLanguage	CNE
10	setId	0..1			ContextStructure	II	D		
11	versionNumber	0..1			ContextStructure	INT	D		
12	copyTime	0..1			Document	TS	D		
13	**recordTarget**	1..*			Act	SET<RecordTarget>	N		
14	typeCode	1..1	M	R	Participation	CS	D	RCT	CNE
15	contextControlCode	1..1	M	R	Participation	CS	D	OP	CNE
16	**patientRole**	1..1			Participation	PatientRole	N		
17	classCode	1..1	M	R	Role	CS	D	PAT	CNE
18	id	1..*			Role	SET<II>	D		
19	addr	0..*			Role	SET<AD>	D		
20	telecom	0..*			Role	SET<TEL>	D		
21	**patient**	0..1			Role	Patient	N		
22	classCode	1..1	M	R	Entity	CS	D	PSN	CNE
23	determinerCode	1..1	M	R	Entity	CS	D	INSTANCE	CNE
24	id	0..1			Entity	II	D		
25	name	0..*			Entity	SET<PN>	D		
26	administrativeGenderCode	0..1			LivingSubject	CE	D	AdministrativeGender	CWE
27	birthTime	0..1			LivingSubject	TS	D		
28	maritalStatusCode	0..1			Person	CE	D	MaritalStatus	CWE
29	religiousAffiliationCode	0..1			Person	CE	D	ReligiousAffiliation	CWE
30	raceCode	0..1			Person	CE	D	Race	CWE
31	ethnicGroupCode	0..1			Person	CE	D	Ethnicity	CWE

Figura 4 - HDM del CDA

Le caratteristiche essenziali di un CDA sono:

- **Persistence**: Un documento clinico continua ad esiste in uno stato inalterato per un periodo di tempo definito da leggi o regolamenti locali.
- **Stewardship:** Gestione responsabile delle risorse, un documento clinico è mantenuto da una organizzazione responsabile.
- **Potential for authentication:** Un documento clinico è una raccolta di informazioni che possono essere autenticate giuridicamente.
- **Context:** Un documento clinico stabilisce il contesto predefinito per i suoi contenuti.
- **Wholeness:** L'autenticazione di un documento clinico viene applicata all'intero e non a porzioni del documento.
- **Human readability:** Un documento clinico è leggibile da un umano e non solo da una macchina.

Quest'ultima caratteristica è talvolta ritenuta secondaria, considerando principali le informazioni strutturate interpretabili dalla macchina. Lo standard prevede che il documento sia *"human readable"* e quindi in grado di essere visualizzato ed leggibile dal ricevente senza la necessità di conoscerne le specificità e le caratteristiche tecniche. Un documento CDA è composto da una serie di blocchi logici caratterizzati da uno specifico significato semantico. Lo standard CDA fornisce un modello astratto per la rappresentazione delle informazioni cliniche, assolutamente indipendente dalle modalità specifiche di realizzazione/serializzazione. La rappresentazione in formato XML è solo una delle possibili modalità di implementazione, o per meglio dire, l'unica per la quale HL7 abbia attualmente fornito una guida di implementazione.

Le specifiche CDA definiscono un'architettura multilivello, dove ogni livello deriva da un livello più basso. In particolare, esistono tre possibili livelli di astrazione, che specificano il grado di granularità del markup richiesto e non fanno riferimento alla granularità del contenuto.

- **Il livello 1:** rappresenta la radice della gerarchia e fornisce le specifiche più generali del documento. In questo livello, solo Header include semantica e la compliance a tale livello di astrazione assicura interoperabilità del contenuto solo tra persone.
- **Il livello 2:** definisce un insieme di vincoli sulla struttura e sulla semantica del documento basata sui template HL7. Specializzando il dominio applicativo del documento: referto di laboratorio, lettera di dimissione ospedaliera, etc. In tale livello di astrazione header del CDA è codificato come nel livello 1. I Templates sono sovrapposti al livello di astrazione 1 vincolando i tag di tale livello, senza introdurre nuovi markup.

- **Il livello 3:** aggiunge markup addizionale al documento in modo da esprimere formalmente il contenuto clinico. Nel livello 3 è necessario che la semantica dei documenti sia completa per una qualsiasi elaborazione automatica del documento. La specializzazione del livello deve essere coerente con il RIM e le strutture di livello 1 e 2 (header e body). L'obiettivo di tale livello di astrazione è quello di soddisfare le esigenze di elaborazione automatica dei documenti.

2 Costruzione di un'ontologia per la codifica di dati clinici a partire dal RMIM HL7 CDA

I concetti dello standard HL7 CDA Release 2.0 sono definiti per raffinamento dal modello RMIM-CDA, che consiste in un Object Model derivato dal RIM. A partire da tale modello, le HL7 XML Implementation Technology Specification definiscono le convenzioni XML usate dallo standard HL7 V3 e convertono algoritmicamente l'Object Model in uno schema XML. Nello specifico, HL7 fornisce sei file in formato W3C XML Schema Definition (XSD) per definire, in termini di vincoli, gli elementi e gli attributi costituenti documenti XML aderenti allo standard: quali elementi ed attributi possono apparire, in quale relazione reciproca, quale tipo di dati possono contenere, ed altro.

D'altra parte, affinché la mole di dati contenuta in documenti CDA disponibili in ambito sanitario possa essere facilmente resa disponibile ed interrogabile in database semantici, essa deve essere affiancata da una modellazione ontologica ([GUARINO 1995]; [GUARINO 1998]) delle informazioni che possa giovare delle tecnologie di più alto livello del W3C per la costruzione e generazione di inferenza e interoperabilità semantica tra servizi e attori eterogenei. A tale scopo, in questa sezione è descritta la metodologia definita per la generazione automatica di modelli ontologici specificati mediante il Web Ontology Language (OWL) ([SCHNEIDER 2004]; [GRAU 2008]) a partire da file XSD contenenti la descrizione formale delle informazioni da modellare.

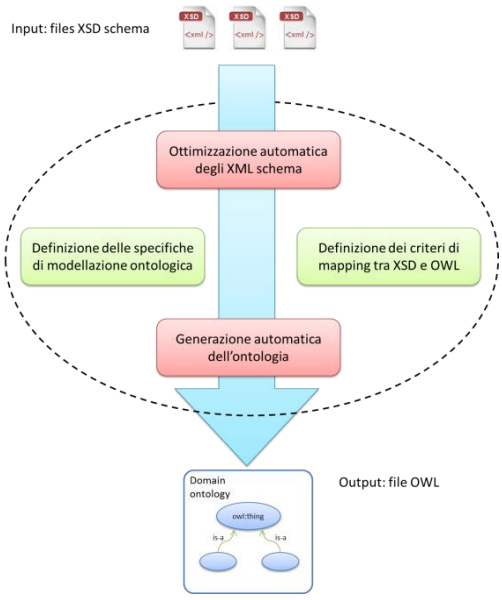


Figura 5 - Schematizzazione della metodologia definita

La metodologia di generazione definita, che risulta essere innovativa rispetto ad altri approcci esistenti ([BEDINI 2011]; [THUY 2012]; [BAKKAS 2014], [BOHRING 2005]) può essere suddivisa nei quattro passi fondamentali illustrati in figura 5:

1. Ottimizzazione automatica dei file XSD forniti in input allo scopo di determinare quali sono le informazioni effettivamente di interesse, e quali invece possono essere rimosse dagli schemi in quanto non utilizzate;

2. Definizione dei criteri di mapping fra i costrutti XSD e quelli OWL;

3. Definizione di eventuali specifiche di modellazione ontologica da tenere in considerazione;

4. Applicazione dei criteri e delle specifiche definite al fine di estrarre automaticamente un modello ontologico OWL associato agli XSD di partenza.

Nelle sezioni successive, sono forniti maggiori dettagli relativi alla metodologia definita e, infine, è presentato l'algoritmo di generazione automatica sviluppato.

2.1 Ottimizzazione automatica degli schemi XML

Lo standard HL7, come già riportato sopra, fornisce sei file in formato XSD. Ogni file XSD definisce un insieme di tipi di dati, specificando per ognuno di essi gli elementi da cui è formato, quali sono quelli necessari, e la struttura degli stessi. Ogni elemento definito all'interno di uno schema XSD può far riferimento ad altri tipi di dati definiti all'interno dello stesso schema, oppure importarli da altri file XSD all'interno dei quali essi sono definiti.

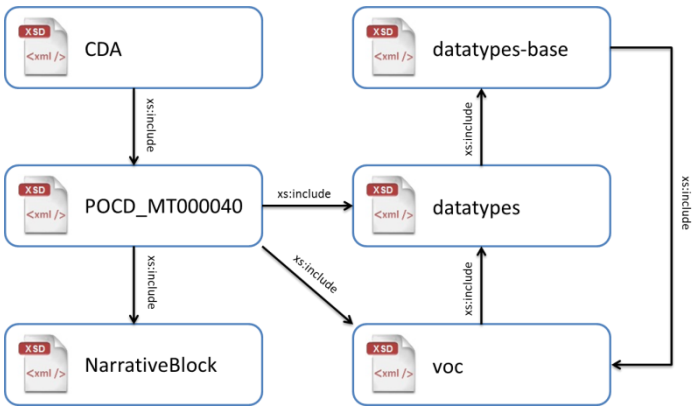


Figura 6 - I sei schemi XSD forniti dallo standard CDA

In figura 6, sono illustrati gli schemi XSD forniti dallo standard CDA e le relazioni d'inclusione (xs:include) di cui necessita ogni schema:

- **CDA**: è lo schema radice dello standard e rimanda allo schema POCD_MT000040 per la specifica effettiva degli elementi e dell'organizzazione di un documento CDA;
- **POCD_MT000040**: è lo schema in cui sono definiti effettivamente i tipi di dati che compongono un documento CDA e la loro struttura. Questo schema definisce le informazioni specifiche del dominio di applicazione che caratterizzano un documento CDA;
- **datatypes-base**: definisce tipi di dati HL7 che possono essere derivati direttamente dai tipi di dati standard del W3C XML Schema come, ad esempio, url internet e stringhe di caratteri formate da soli numeri o aventi limitazioni sul numero di caratteri, utili per la definizione di tipi di dati relativi a codici e/o identificatori;
- **datatypes**: definisce ulteriori tipi di dati HL7 a partire da quelli definiti nello schema datatypes-base;
- **voc**: come lo schema datatypes, definisce tipi di dati per descrivere le informazioni generiche interne a un documento;
- **NarrativeBlock**: definisce tipi di dati utili per la rappresentazione di elementi narrativi all'interno di un documento. Permette di definire la struttura di un campo di testo, in termini di allineamento e formattazione dello stesso.

Gli schemi XSD rilasciati dallo standard HL7 possono essere suddivisi in due gruppi fondamentali, gli schemi di dominio che definiscono elementi specifici di un documento CDA (CDA e POCD_MT000040), e gli schemi di dati (datatypes-base, datatypes, voc, e NarrativeBlock) che definiscono invece tipi di dati utili per la descrizione di un documento, ma che solo in parte vengono richiamati e utilizzati per la definizione di un documento CDA. Ciò comporta che, al fine di garantire la consistenza delle informazioni definite negli schemi XSD di dominio (CDA e POCD_MT000040), sono analizzati e processati una serie di dati e informazioni che non sono funzionali al documento CDA.

Tipicamente, gli approcci per la generazione di modelli ontologici a partire da schemi XSD prevedono la definizione di classi e proprietà ontologiche per ogni elemento definito negli schemi senza alcuna fase di pre-processing degli stessi.

Ciò comporta che ogni elemento definito negli schemi venga ricreato/convertito nell'ontologia generata senza alcuna analisi di quali siano gli elementi effettivamente necessari e quali invece che possano essere omessi poiché non funzionali alla definizione degli elementi di dominio.

In quest'ottica, è stata definita una procedura automatica di ottimizzazione degli schemi XSD che mira, da un lato, ad analizzare le dipendenze tra i file XSD per identificare gli schemi effettivamente utilizzati. D'altro lato, tali schemi verranno in seguito analizzati al fine di determinare, per ognuno di essi, quali sono gli elementi definiti e quali quelli necessari alla loro definizione.

L'obiettivo è quello di determinare, tra i file XSD inclusi dallo schema radice, quali elementi possano essere eliminati poiché non strettamente necessari alla definizione degli elementi costituenti lo schema radice.

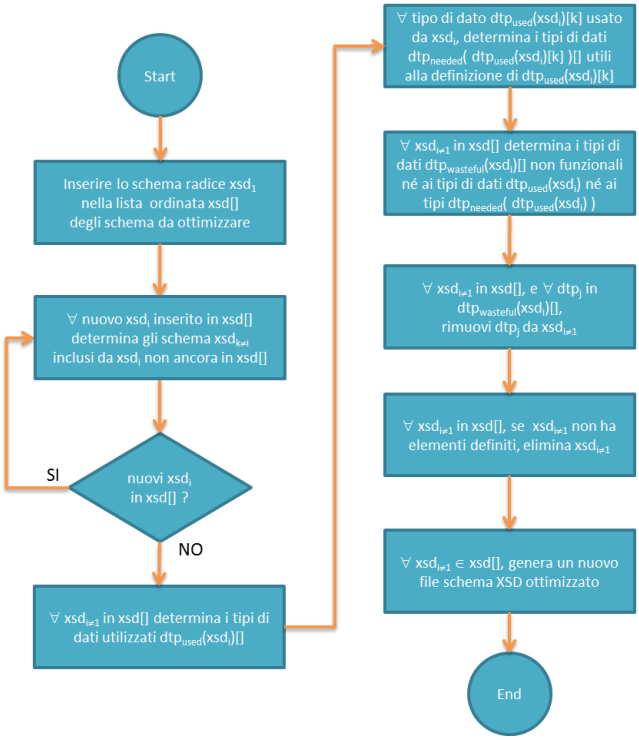


Figura 7 - La procedura automatica definita per l'ottimizzazione degli schemi XSD

In figura 7 è illustrato il flusso della procedura di ottimizzazione definita, evidenziando come vengono generati nuovi schemi XSD ottimizzati attraverso la cancellazione dei dati non funzionali agli elementi definiti e/o utilizzati nello schema radice. La procedura definita si basa sull'indicazione di uno schema XSD radice da cui poi andare ad identificare e analizzare eventuali altri schemi XSD inclusi. Gli schemi XSD inclusi dallo schema radice sono analizzati al fine di determinare i tipi di dati non funzionali alla definizione degli elementi nello schema radice. La procedura è legata al particolare schema XSD indicato come radice poiché esso guida la ricerca dei dati non funzionali. Per quanto riguarda i dati definiti all'interno dello schema radice, nel caso in cui alcuni di essi dovessero essere identificati come non funzionali ad altri elementi interni allo schema, questi non vengono eliminati poiché considerati dati di dominio a significatività maggiore rispetto ai dati degli schemi XSD inclusi. In pratica, poiché i dati di dominio descrivono concetti che possono essere utili alla verifica di documenti CDA, essi non sono eliminati anche se non utilizzati internamente allo schema radice.

2.2 Definizione di criteri di mapping tra costrutti XSD e OWL

La seconda fase della metodologia consiste nella definizione di un modello ontologico che descrive, in termini di classi e proprietà OWL, gli schemi XSD ottimizzati. Lo scopo è di definire un mapping diretto tra i costrutti degli schemi XSD e quelli ontologici, così che i dati e le informazioni degli schemi acquisiscano espressività semantica e possano supportare meccanismi inferenziali.

I tipi di dati descritti in uno schema XSD sono di due tipi:

- xs:simpleType
- xs:complexType

Il tipo xs:simpleType è utilizzato per definire tipi di dati semplici in termini di range e valori ammessi per quel particolare tipo di dati. Generalmente sono costruiti e derivati dai tipi standard come, ad esempio, xs:string, xs:boolean, e xs:integer.

Il tipo xs:complexType è utilizzato per definire strutture dati di tipo complesso composte da attributi ed elementi di diversa tipologia, che possono essere di tipo xs:simpleType o altri xs:complexType.

I costrutti xs:simpleType e xs:complexType utilizzati in uno schema XSD possono essere modellati in un'ontologia tramite due classi ontologiche tipicamente utilizzate proprio per modellare tipi di dati e concetti strutturati.

Il tipo xs:simpleType può essere modellato in OWL tramite la classe rdfs:Datatype. Un rdfs:Datatype è tipicamente utilizzato per modellare i tipi di dati associati ai valori di una proprietà ontologica. In OWL, ogni istanza della classe rdfs:Datatype è sottoclasse anche della classe rdfs:Literal dei valori letterali. Tali valori letterali sono usati per modellare valori di dati quali stringhe, numeri, e date.

Il tipo xs:complexType può essere modellato in OWL tramite la classe owl:Class che modella concetti ontologici complessi caratterizzati da attributi e relazioni con altri concetti del dominio. In OWL, owl:Class fornisce un meccanismo per raggruppare risorse del domino aventi caratteristiche simili. In particolare, una classe di tipo owl:Class può essere specificata tramite restrizione, enumerazione, unione, complemento o intersezioni di altre classi e risorse del dominio.

Modellazione ontologica di xs:simpleType

La scelta di convertire i tipi xs:simpleType e xs:complexType in classi rdfs:Datatype e owl:Class di OWL, rispettivamente, è già stata utilizzata in letteratura in quanto soluzione di modellazione più appropriata per rappresentare gli elementi descritti in uno schema XSD. Tipicamente, però, le soluzioni esistenti non permettono di rappresentare la complessità dei vincoli e delle restrizioni che si possono trovare in uno schema XSD e, in particolare, negli schemi rilasciati dallo standard CDA.

Consideriamo, ad esempio, lo schema XSD riportato in figura 8. I due tipi "age" e "password" definiti come xs:simpleType derivati dai tipi standard xs:integer e xs:string, si differenziano da questi ultimi per alcune restrizioni sui valori ammessi rispetto allo spettro completo dei valori originali.

Al fine di modellare in OWL i tipi xs:simpleType definiti tramite xs:restriction, la tecnica proposta prevede di utilizzare i costrutti OWL (in particolare OWL 2) per la definizione formale di rdfs:Datatype tramite la restrizione dei valori ammessi di altri rdfs:Datatype.

A tale scopo, OWL prevede i seguenti costrutti:

- owl:onDatatype – per la restrizione di rdfs:Datatype;
- owl:withRestrictions – per specificare l'elenco delle restrizioni effettive;
- owl:equivalentClass – per definire classi in funzione di altre, tramite ad esempio operatori di restrizione o insiemistici;

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://xsd_example"
    elementFormDefault="qualified" targetNamespace="http://xsd_example">

    <xs:simpleType name="age">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0" />
            <xs:maxInclusive value="100" />
        </xs:restriction>
    </xs:simpleType>


    <xs:simpleType name="password">
        <xs:restriction base="xs:string">
            <xs:minLength value="5" />
            <xs:maxLength value="8" />
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

Figura 8 - Esempio di definizione di tipi di dati tramite xs:simpletype e xs:restriction

Bisogna notare che, per definire una classe oggetto di restrizioni, il linguaggio OWL prevede che essa venga dichiarata equivalente, mediante il costrutto owl:equivalentClass, ad una classe anonima in cui le restrizioni vengono definite e specificate.

Partendo dalle considerazioni sopra esposte, è stato definito il seguente mapping XSD-OWL:

Tipo di dato XSD	Concetto ontologico
xs:simpleType ▪ nome *sTypeName*	rdfs:Datatype ▪ nome (rdf:about) *sTypeName*
xs:simpleType ▪ nome *sTypeName* ▪ xs:restriction definita sul tipo *sTypeParent* (attributo base della xs:restriction)	rdfs:Datatype ▪ nome (rdf:about) *sTypeName* ▪ equivalente ad una classe anonima, di tipo rdfs:Datatype, contenente le restrizioni specificate sul tipo *sTypeParent*

Nella figura 9, è riportata l'ontologia generata automaticamente applicando il mapping descritto all'esempio considerato.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://xsd_example#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdfs:Datatype rdf:about="http://xsd_example#age">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:withRestrictions rdf:parseType="Collection">
          <rdf:Description>
            <xsd:minInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
            >0</xsd:minInclusive>
          </rdf:Description>
          <rdf:Description>
            <xsd:maxInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
            >100</xsd:maxInclusive>
          </rdf:Description>
        </owl:withRestrictions>
        <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
  <rdfs:Datatype rdf:about="http://xsd_example#password">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:withRestrictions rdf:parseType="Collection">
          <rdf:Description>
            <xsd:minLength rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >5</xsd:minLength>
          </rdf:Description>
          <rdf:Description>
            <xsd:maxLength rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >8</xsd:maxLength>
          </rdf:Description>
        </owl:withRestrictions>
        <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
</rdf:RDF>
```

Figura 9 - Ontologia generata per lo schema XSD riportato in figura 8

Per quanto riguarda la modellazione ontologica, OWL prevede l'uso dei seguenti costrutti per la definizione di tipi di dati tramite unione di altri tipi esistenti:

- owl:unionOf – per la definizione di una risorsa, in questo caso un rdfs:Datatype, come unione di altri rdfs:Datatype;
- owl:equivalentClass – per definire classi in funzione di altre classi;

Come discusso per la modellazione ontologica delle restrizioni, anche per la definizione di rdfs:Datatype tramite unione di altri rdfs:Datatype, bisogna prima definire un rdfs:Datatype anonimo come unione dei tipi di interesse, e poi dichiarare tale classe anonima equivalente al rdfs:Datatype oggetto della modellazione.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://xsd_example"
    xmlns:mif="urn:hl7-org:v3/mif" elementFormDefault="qualified"
    targetNamespace="http://xsd_example">

    <xs:simpleType name="jeans_size">
        <xs:union memberTypes="sizebyno sizebystring">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="normale" />
                    <xs:enumeration value="forte" />
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
    </xs:simpleType>

    <xs:simpleType name="sizebyno">
        <xs:restriction base="xs:positiveInteger">
            <xs:maxInclusive value="42" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="sizebystring">
        <xs:restriction base="xs:string">
            <xs:enumeration value="small" />
            <xs:enumeration value="medium" />
            <xs:enumeration value="large" />
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

Figura 10 - Esempio di definizione di tipi xs:simpletype tramite xs:union e xs:enumeration

Il tipo xs:simpleType può essere definito anche tramite unione di altri tipi già esistenti. Si consideri, ad esempio, il tipo *jeans_size*, riportato in figura 10, definito come xs:simpleType unione dei due xs:simpleType "*sizebyno"* e "*sizebystring"* definiti altrove e di un xs:simpleType anonimo definito localmente.

Inoltre, il tipo "sizebystring" mostrato in figura 10 rappresenta un esempio di definizione di tipi di dati tramite enumerazione di valori. A tale scopo si utilizza il costrutto xs:enumeration che permette di definire un tipo di dato tramite enumerazione diretta dei valori ammessi a partire dai valori previsti per il tipo di dato padre da cui viene derivato.

OWL prevede inoltre l'uso dei seguenti costrutti per la definizione di tipi di dati tramite enumerazione diretta dei valori costituenti:

- owl:oneOf – per la definizione di un rdfs:Datatype tramite la collezione di valori letterali che il rdfs:Datatype può assumere;
- owl:equivalentClass – per la definizione di classi in funzione di altre classi.

Partendo dalle considerazioni sopra esposte, è stato definito il seguente mapping XSD-OWL per la modellazione di xs:simpleType definiti tramite xs:union, e tramite xs:enumeration:

Tipo di dato XSD	Concetto ontologico
xs:simpleType ▪ nome *sTypeName* ▪ xs:union dei tipi $sType_i$ (dichiarati tramite l'attributo "memberTypes" o definiti localmente alla union)	rdfs:Datatype ▪ nome (rdf:about) *sTypeName* ▪ equivalente ad una classe anonima, di tipo rdfs:Datatype, definita come unione degli rdfs:Datatype associati agli $sType_i$
xs:simpleType ▪ nome *sTypeName* ▪ xs:restriction del tipo *sTypeParent* (attributo base della xs:restriction), definita tramite xs:enumeration dei valori ammessi di *sTypeParent*	rdfs:Datatype ▪ nome (rdf:about) *sTypeName* ▪ equivalente ad una classe anonima, di tipo rdfs:Datatype, restrizione del rdfs:Datatype *sTypeParent*, tramite enumerazione diretta owl:oneOf dei valori ammessi

Nella figura 11, è riportata l'ontologia generata automaticamente applicando il mapping descritto all'esempio mostrato

in figura 10.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://xsd_example#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdfs:Datatype rdf:about="http://xsd_example#sizebyno">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:withRestrictions rdf:parseType="Collection">
          <rdf:Description>
            <xsd:maxInclusive rdf:datatype="http://www.w3.org/2001/XMLSchema#positiveInteger">42</xsd:maxInclus
          </rdf:Description>
        </owl:withRestrictions>
        <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger"/>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
  <rdfs:Datatype rdf:about="http://xsd_example#jeans_size">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:unionOf rdf:parseType="Collection">
          <rdfs:Datatype rdf:about="http://xsd_example#sizebyno"/>
          <rdfs:Datatype rdf:about="http://xsd_example#sizebystring"/>
          <rdfs:Datatype rdf:about="http://xsd_example#jeans_size_AnonymousMember"/>
        </owl:unionOf>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
  <rdfs:Datatype rdf:about="http://xsd_example#jeans_size_AnonymousMember">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">forte</rdf:first>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">normale</rdf:first>
        </owl:oneOf>
        <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
  <rdfs:Datatype rdf:about="http://xsd_example#sizebystring">
    <owl:equivalentClass>
      <rdfs:Datatype>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">large</rdf:first>
            </rdf:rest>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">medium</rdf:first>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">small</rdf:first>
        </owl:oneOf>
        <owl:onDatatype rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </rdfs:Datatype>
    </owl:equivalentClass>
  </rdfs:Datatype>
</rdf:RDF>
```

Figura 11 - Ontologia generata per lo schema xsd riportato in figura 10

Modellazione ontologica di xs:complexType

Per la modellazione ontologica di tipi xs:complexType, la metodologia proposta prevede la generazione di classi owl:Class, che possono essere caratterizzate da proprietà ontologiche al fine di modellare attributi ed elementi costituenti un xs:complexType. Un tipo xs:complexType può contenere i seguenti costrutti:

- xs:element – un elemento xs:element è caratterizzato da un nome e da un tipo che specifica come deve essere interpretato il testo dell'elemento xs:element. Il tipo di dato può essere scelto tra quelli standard (boolean, string, date, etc.), oppure tra quelli definiti nello schema XSD analizzato (xs:simpleType o xs:complexType).

- xs:attribute – sono tipi semplici utilizzati per definire gli attributi degli elementi. Un tipo semplice non può contenere altri attributi. Un tipo xs:attribute è caratterizzato da un nome e da un tipo che specifica come deve essere interpretato il valore dell'attributo. Il tipo può essere scelto tra quelli standard (boolean, string, date, etc.), oppure tra quelli xs:simpleType definiti nello schema XSD analizzato.

Per i tipi xs:complexType, la metodologia definita prevede che gli elementi e gli attributi che lo caratterizzano siano modellati come proprietà owl:DatatypeProperty e owl:ObjectProperty a seconda del tipo dell'elemento xs:element (xs:attribute) modellato. In particolare, detto aType il tipo di un xs:attribute, se aType deriva da un tipo di dato xs:simpleType, allora viene instaurata una relazione di tipo owl:DatatypeProperty tra la classe del xs:complexType modellato e il rdfs:Datatype aType. Invece, se un tipo xs:attribute è di tipo xs:complexType, allora viene creata una relazione di tipo owl:ObjectProperty tra la classe del xs:complexType modellato e l'elemento xs:complexType dell'attributo.

Nel caso di attributi o elementi globali, e quindi non definiti internamente a un tipo xs:complexType, essi vengono convertiti nell'ontologia come rdfs:Datatype e owl:Class, rispettivamente.

Inoltre, un xs:complextType $cType_1$ può essere definito come estensione o restrizione di un altro xs:complexType $cType_2$, tramite il costrutto xs:complexContent. Nel caso di una restrizione, $cType_1$ viene modellato nell'ontologia come una owl:Class sotto classe della owl:Class associata al tipo $cType_2$.

Infine, nel caso un xs:complexType contenga un xs:element con vincoli di cardinalità sulla sua occorrenza o meno all'interno del complex type, tali vincoli si traducono in vincoli di cardinalità ontologica definiti sulla rdf:Property rappresentante l'elemento vincolato.

Ad esempio, consideriamo il complex type "fullpersoninfo" riportato in figura 12.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://xsd_example"
    xmlns:mif="urn:hl7-org:v3/mif" elementFormDefault="qualified"
    targetNamespace="http://xsd_example">

    <xs:simpleType name="cityName">
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z]" />
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="personinfo">
        <xs:sequence>
            <xs:element name="firstname" type="xs:string" />
            <xs:element name="lastname" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="fullpersoninfo">
        <xs:sequence>
            <xs:element maxOccurs="1" minOccurs="1" name="name" type="personinfo" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="address" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="city" type="cityName" use="optional" />
        <xs:attribute name="country" type="xs:string" />
    </xs:complexType>

</xs:schema>
```

Figura 12 - Esempio di xs:complexType contenente sia elementi che attributi

Gli attributi "city" e "country", essendo di tipo xs:simpleType o derivati da essi, essi vengono modellati ontologicamente tramite delle owl:DatatypeProperty tra la owl:Class associata al complexType e il simpleType dell'attributo. Per quanto riguarda gli elementi "name" e "address", il primo diventa una owl:ObjectProperty poiché di tipo complexType, mentre il secondo una owl:DatatypeProperty in quanto di tipo xs:string.

Inoltre, per la definizione delle restrizioni sulla cardinalità delle rdf:Property, OWL mette a disposizione i seguenti costrutti:

- owl:Restriction – per definire classi OWL come restrizioni di altre classi;

- owl:onProperty – per specificare la proprietà oggetto della restrizione;
- owl:allValuesFrom – per specificare il tipo di dato ammesso per quella proprietà;
- owl:minCardinality, owl:cardinality, owl:maxCardinality – per specificare la cardinalità della proprietà.

Infine, la metodologia definita prevede anche la codifica di dominio (rdfs:domain) e range (rdfs:range) per ogni rdf:Property. In pratica, utilizzando il costrutto ontologico owl:unionOf, per ogni attributo/elemento contenuto in diversi xs:complexType, viene creata una rdf:Property avente come rdfs:domain l'unione delle classi associate ai xs:complexType che lo contengono, e come rdfs:range l'unione delle owl:Class/rdfs:Datatype rappresentanti i tipi degli attributi/elementi.

Partendo dalle considerazioni sopra esposte, è stato definito il mapping per la gestione delle restrizioni tramite enumerazione e per le unioni di tipi di dati mostrato nella tabella successiva.

Tipo di dato XSD	**Concetto ontologico**
xs:complexType ▪ nome c*TypeName*	owl:Class ▪ nome (rdf:about) c*TypeName*
xs:complexType ▪ nome c*TypeName* ▪ xs:restriction del tipo cTypeParent (attributo base della xs:restriction), tramite il costrutto xs:complexContent	owl:Class ▪ nome (rdf:about) c*TypeName* ▪ *rdfs:subClassOf* della owl:Class associata al tipo c*TypeParent* ▪ xs:complexContent modellato secondo il mapping dei xs:complexType
xs:complexType ▪ nome c*TypeName* ▪ xs:extension del tipo cTypeParent (attributo base della xs:extension), tramite il costrutto xs:complexContent	owl:Class ▪ nome (rdf:about) c*TypeName* ▪ *rdfs:subClassOf* della owl:Class associata al tipo c*TypeParent* ▪ xs:complexContent modellato secondo il mapping dei xs:complexType
xs:complexType ▪ nome c*TypeName* ▪ xs:attribute di nome *aName* e definito di tipo aType	owl:Class ▪ nome (rdf:about) c*TypeName* ▪ *rdfs:subClassOf* delle classi generiche su cui è definita una owl:DatatypeProperty di nome *aName*, i cui valori sono (owl:allValuesFrom) di tipo aType ▪ c*TypeName* aggiunta al rdfs:domain di *aName*, e l' rdfs:Datatype associato a aType viene aggiunto al range di *aName*
xs:complexType ▪ nome c*TypeName* ▪ xs:element di nome ele*Name* e definito di tipo eleType ▪ eleType di tipo xs:complexType	owl:Class ▪ nome (rdf:about) c*TypeName* ▪ *rdfs:subClassOf* delle classi generiche su cui è definita una owl:ObjectProperty di nome ("has"+ele*Name)*, i cui valori sono (owl:allValuesFrom) di tipo eleType ▪ c*TypeName* aggiunta al rdfs:domain di ele*Name*, e l' owl:Class associata a eleType viene aggiunta al range di ele*Name*
xs:complexType ▪ nome c*TypeName* ▪ xs:element di nome ele*Name* e definito di tipo eleType ▪ eleType di tipo xs:simpleType	owl:Class ▪ nome (rdf:about) c*TypeName* ▪ *rdfs:subClassOf* delle classi generiche su cui è definita una owl:DatatypeProperty di nome ele*Name*, i cui valori sono (owl:allValuesFrom) di tipo eleType

	▪ c*TypeName* viene aggiunta al rdfs:domain di ele*Name*, e l' rdfs:Datatype associato a eleType viene aggiunto al range di ele*Name*
xs:element ▪ globale non definito all'interno di un xs:complexType	owl:Class ▪ definita e caratterizzata applicando il mapping dei xs:complexType
xs:attribute ▪ globale non definito all'interno di un xs:complexType	rdfs:Datatype ▪ definito e caratterizzato applicando il mapping dei xs:simpleType

Si noti che il nome di una proprietà di tipo owl:ObjectProperty, viene composto aggiungendo il prefisso "has" al nome dell' xs:element in oggetto, per richiamare la relazione ontologica tra concetti di dominio (owl:Class) che instaura una owl:ObjectProperty all'interno dell'ontologia.

2.3 Specifiche di modellazione dell'ontologia

Il mapping tra costrutti XSD e OWL definito nella sezione precedente permette di modellare le dipendenze di basso livello che esistono tra i tipi definiti nello schema XSD, ma soltanto a livello di xs:simpleType e xs:complexType senza discernere o evidenziare eventuali differenze esistenti tra diverse tipologie di xs:complexType. In pratica, applicando il mapping descritto, ogni xs:simpleType viene modellato nell'ontologia come un tipo di dato (rdfs:Datatype) mentre ogni xs:complexType è modellato come concetto ontologico strutturato (owl:Class), così da rispettare l'effettiva natura di xs:simpleType e xs:complexType.

Inoltre, può essere di interesse andare a rappresentare delle distinzioni concettuali/gerarchiche tra i diversi xs:complexType di uno schema XSD e poter distinguere tra informazioni di dominio e informazioni relative al tipo di dati. Infatti, alcuni xs:complexType fanno riferimento ad informazioni specifiche che si possono ritrovare all'interno di un documento sanitario aderente allo standard HL7 CDA, che quindi è possibile considerare come effettivi xs:complexType di dominio. Altri xs:complexType, invece, modellano tipologie di informazioni relative ai tipi di dato.

Partendo da queste considerazioni, la metodologia definita prevede oltre al mapping di basso livello tra costrutti XSD e OWL, anche una distinzione di alto livello tra Schema XSD di dominio e Schema XSD di dati.

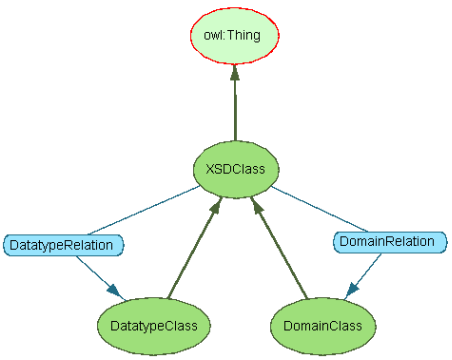


Figura 13 - Distinzione tra classi e proprietà riguardanti xs:complexType di dominio e di dati

L'obiettivo è di ottenere una gerarchia tra le owl:Class generate all'interno dell'ontologia che rispecchi la distinzione tra owl:Class che provengono da dati definiti nello schema di dominio e le owl:Class generate dai dati definiti negli schemi di supporto. In Figura 13 è mostrata la gerarchia di classi ontologiche che si vuole realizzare. DomainClass rappresenta la classe ontologica dei tipi xs:complextType che rappresentano concetti di dominio e DatatypeClass la classe ontologica dei tipi xs:complexType che rappresentano concetti esterni al dominio.

Allo stesso modo, anche tra le owl:ObjectProperty può essere instaurata una gerarchia di proprietà, in modo da distinguere tra diverse tipologie a seconda che esse agiscano su dati di dominio, oppure su dati non di dominio. L'idea è di definire due distinte owl:ObjectProperty, di cui sono sotto proprietà (rdfs:SubProperty) tutte le owl:ObjectProperty che vengono generate nell'ontologia per modellare le relazioni instaurate dagli xs:element presenti nei xs:complextType.

Detta "DomainRelation" la owl:ObjectProperty generica che può essere instaurata tra un xs:complexType (di dominio o di dati), e un xs:complexType di dominio, allora tutte le owl:ObjectProperty che hanno come rdfs:domain classi di "XSDClass" e come rdfs:range classi di "DomainClass" vengono definite sotto proprietà (rdfs:SubProperty) di "DomainRelation".

Viceversa, detta "DatatypeRelation" la owl:ObjectProperty generica che può essere instaurata tra un xs:complexType (di dominio o di dati), e un xs:complexType di dati, allora tutte le owl:ObjectProperty che hanno come rdfs:domain classi di "XSDClass" e come rdfs:range classi di "DatatypeClass" vengono definite sotto proprietà (rdfs:SubProperty) di "DatatypeRelation".

La gerarchia di classi e proprietà mostrata in Figura 13 viene definita e instaurata automaticamente all'interno dell'ontologia generata, analizzando gli schemi forniti in ingresso e domini e range delle proprietà ontologiche generate applicando il mapping XSD-OWL definito nella sezione precedente.

L'idea su cui si basa la generazione automatica della gerarchia discussa è quella di dare la possibilità all'utente di segnalare, tra gli Schema XSD da analizzare, quale sia lo Schema radice di dominio e quale sia lo Schema radice dei dati. In questo modo, automaticamente, la procedura di generazione dell'ontologia è in grado di distinguere tra informazioni provenienti da Schema di dominio e da Schema di dati, così da costruire la gerarchia di classi mostrata in Figura 13.

2.4 Generazione di un'ontologia owl da schemi xsd

In questa sezione viene descritta l'applicazione della metodologia discussa nelle sezioni precedenti, al fine di generare automaticamente un modello ontologico rappresentante le informazioni definite all'interno degli schemi XSD.

L'algoritmo di generazione dell'ontologia si basa su 5 passi fondamentali:

- identificazione degli XSD di dominio e di quelli di dati;
- ottimizzazione degli schema XSD;
- creazione modello ontologico iniziale;
- analisi dei tipi di dati definiti negli XSD e applicazione del mapping opportuno;
- applicazione delle specifiche di modellazione alle classi e proprietà ontologiche create.

Identificazione degli XSD di dominio e di quelli di dati

Tra gli schemi XSD forniti in ingresso vengono indicati lo schema radice di dominio e lo schema radice dei dati. Tali schemi vengono analizzati e navigati per identificare tra gli schemi restanti quali fanno parte del dominio e quali invece non ne fanno parte. Tale procedura di identificazione si basa sull'ipotesi che gli schemi siano stati predisposti in modo da avere schemi di dominio contenenti solo i dati che si vogliono modellare come dati di dominio, e schemi di dati contenenti solo i dati che si vuole modellare come dati non di dominio.

Ottimizzazione degli schema XSD

Gli schemi XSD forniti in ingresso vengono analizzati e processati automaticamente come descritto nella sezione 2.1.1. L'obiettivo è l'eliminazione dai file XSD di tutti i tipi di dati che non sono funzionali alla definizione dei dati di dominio. A partire dallo schema radice di dominio vengono analizzati tutti i tipi di dati utilizzati in modo da determinare quali tipi di dati inclusi vengano effettivamente utilizzati. Quindi, solo per gli schemi di dati, vengono eliminati tutti i tipi di dati non utilizzati.

Creazione modello ontologico iniziale

Si procede alla creazione del modello ontologico che andrà a contenere le classi e le proprietà associate ai vari costrutti degli schemi ottimizzati. In particolare, nel modello vengono definiti i namespace necessari in funzione di quelli dichiarati all'interno degli schemi in input.

Analisi dei tipi di dati definiti negli XSD e applicazione del mapping opportuno

In questa fase si procede alla enumerazione di tutti i tipi di dati definiti all'interno degli schemi XSD e, per ognuno di essi si analizza la tipologia di dato e si identifica il mapping associato. Di conseguenza, nuove asserzioni ontologiche vengono inserite nel modello a seconda del mapping applicato.

Applicazione delle specifiche di modellazione ontologica alle classi e alle proprietà del modello

In questa fase vengono stabilite le gerarchie di classe e proprietà indicate nelle specifiche di modellazione. In particolare, si procede all'analisi dei rdfs:range definiti per ogni owl:ObjectProperty creata, così da identificare per ogni proprietà se essa debba essere classificata come relazione avente valori tra i dati di dominio, o se come relazione con valori attesi nei dati non di dominio.

I vari passi dell'algoritmo descritto, sono riportati con maggior dettaglio in Figura 14.

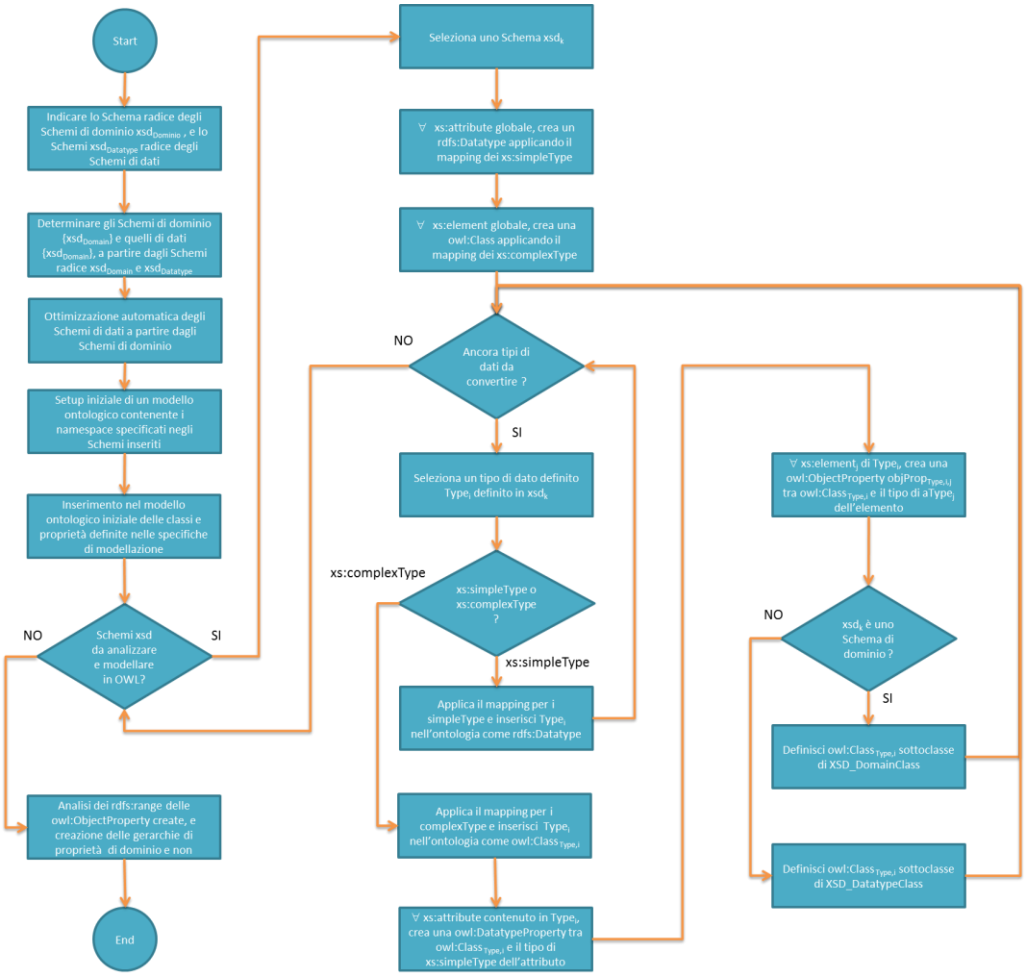


Figura 14 - L'algoritmo definito per la generazione automatica di un'ontologia owl

3 Applicazione della metodologia al caso di studio

Questa sezione descrive l'applicazione della metodologia definita per la generazione automatica di modelli ontologici a partire dagli schemi XSD forniti dallo standard HL7. Come discusso nelle sezioni precedenti, lo standard CDA fornisce sei schemi XSD per la specifica e verifica di documenti CDA:

- **CDA**: Schema radice per documenti CDA, che rimanda allo Schema POCD_MT000040;
- **POCD_MT000040**: Schema relativo a intestazione e corpo di un documento CDA;
- **datatypes, datatypes-base, voc, NarrativeBlock**: costituiscono i core-schemas rilasciati dallo standard HL7, i quali contengono informazioni generali quali definizioni di tipi di dati, lessici e definizioni per parti di testo narrative.

Di questi, dopo una prima analisi, si è deciso di non includere lo Schema NarrativeBlock all'interno del processo di creazione dell'ontologia, in quanto atto a definire le modalità per la rappresentazione di testo narrativo, che esula dagli interessi di questo lavoro. Infatti, l'obiettivo è creare una ontologia atta a favorire la generazione di database semantici in grado di supportare efficacemente inferenza e interrogazioni intelligenti su dati estratti da documenti CDA, e quindi solo parte strutturata del documento va considerata e non la parte testuale non codificata. Inoltre, tra tutti i 397 tipi di dati definiti negli altri Schema, soltanto uno utilizza un tipo definito nel NarrativeBlock. In particolare, soltanto l'elemento xs:element "text" interno al tipo complesso "POCD_MT000040.Section" necessita di "StrucDoc.Text" per la sua definizione. Inoltre, l'eliminazione dell'elemento "text" per il tipo di dato "POCD_MT000040.Section" comporta la non inclusione dell'intero Schema NarrativeBlock la cui presenza nell'ontologia comporta soltanto un'eccessiva complicazione della stessa per descrivere informazioni di formattazione e allineamento testo che non incidono in alcun modo sul potere semantico del documento.
In quest'ottica si è deciso di applicare la metodologia di generazione soltanto a cinque degli schemi rilasciati dallo standard **HL7: CDA, POCD_MT000040, datatypes, datatypes-base, voc.**

Al fine di applicare l'algoritmo di generazione automatica descritto nella sezione precedente, gli schemi sono stati suddivisi in due gruppi, in modo che la procedura automatica possa creare anche una gerarchia di classi e proprietà che metta in risalto la distinzione tra classi e relazioni di dominio e classi e relazioni di dati non di dominio. Allo scopo sono stati selezionati i seguenti schemi radice:

- Schema radice dei dati di dominio ($XSD_{Dominio}$): **CDA**;
- Schema radice dei dati non di dominio ($XSD_{Datatype}$): **datatypes**.

Per quanto riguarda l'ottimizzazione automatica degli schemi XSD, la procedura ha individuato più di sessanta tipi di dati (circa il 15% del totale) contenuti negli schemi non di dominio che possono essere eliminati al fine di semplificare e ottimizzare l'ontologia da generare, in quanto dati non funzionali alla dichiarazione dei dati di dominio.

In dettaglio, in seguito si riportano i tipi di dati eliminati da ogni Schema analizzato:

- datatypes.xsd
  - tipi definiti: 38
  - tipi eliminati: 31
    1. PIVL_TS
    2. EIVL_TS
    3. PPD_TS
    4. PPD_PQ
    5. PIVL_PPD_TS
    6. SXCM_PPD_TS
    7. IVL_PPD_TS
    8. IVXB_PPD_TS
    9. EIVL_PPD_TS
    10. IVL_PPD_PQ
    11. SXCM_PPD_PQ
    12. IVXB_PPD_PQ

13. SXPR_TS
14. SXCM_CD
15. SXCM_MO
16. SXCM_REAL
17. IVL_REAL
18. IVXB_REAL
19. IVL_MO
20. IVXB_MO
21. HXIT_PQ
22. HXIT_CE
23. BXIT_CD
24. BXIT_IVL_PQ
25. SLIST_PQ
26. SLIST_TS
27. GLIST_TS
28. GLIST_PQ
29. RTO_MO_PQ
30. UVP_TS
31. list_int

- datatypes-base.xsd:
  - tipi definiti: 89
  - tipi eliminati: 10
    1. ANYNonNull
    2. BN
    3. CO
    4. TN
    5. REAL
    6. MO
    7. RTO
    8. EIVL.event
    9. RTO_QTY_QTY
    10. Probability

- voc.xsd
  - tipi definiti: 182
  - tipi eliminati: 20
    11. Classes
    12. CalendarCycle
    13. CalendarCycleOneLetter
    14. CalendarCycleTwoLetter
    15. GregorianCalendarCycle
    16. Currency
    17. MediaType
    18. ApplicationMediaType
    19. AudioMediaType
    20. ImageMediaType
    21. ModelMediaType
    22. MultipartMediaType
    23. TextMediaType
    24. VideoMediaType
    25. ProbabilityDistributionType
    26. TimingEvent
    27. URLScheme

28. CommunicationFunctionType
29. RoleLinkType
30. RelatedLinkType

4 Analisi dell'ontologia prodotta

L'ontologia generata automaticamente dalla procedura discussa, è costituita da due classi principali "DomainClass" e "DatatypeClass" che raggruppano tutte le classi generate distinguendo tra informazioni relative a dati di dominio, e informazioni relative a dati non di dominio.

In dettaglio, Figura 15 riporta un frammento della gerarchia di classi generata, così come visualizzata all'interno di Protegé, uno dei strumenti informatici più utilizzati per l'editing di ontologie.

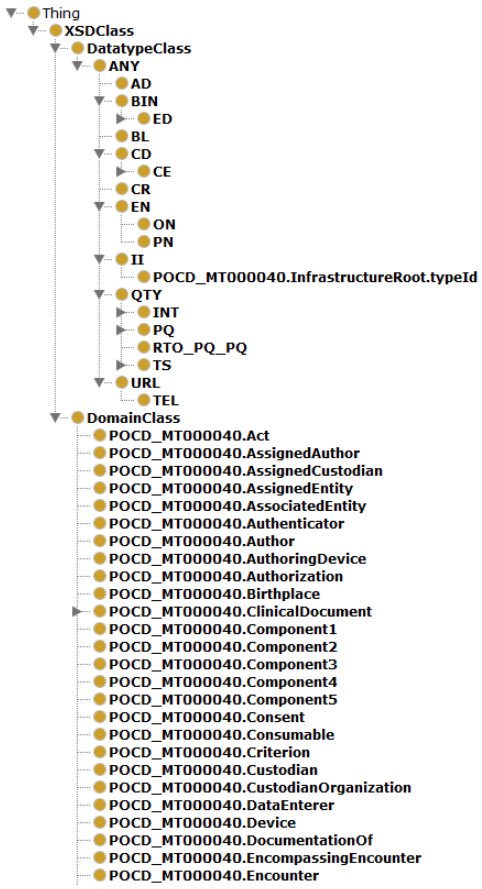


Figura 15 - Le classi dell'ontologia generata

Ogni classe dell'ontologia modella un particolare xs:complexType descritto negli schemi dello standard HL7. Per ogni elemento e attributo definito all'interno di un xs:complexType, la metodologia definisce un'appropriata proprietà (owl:ObjectProperty o owl:DatatypeProperty) a seconda della tipologia di elemento (xs:complexType o xs:simpleType). In particolare, la metodologia definita specifica per ogni classe ontologica (xs:complexType) un insieme di restrizioni relative alle proprietà definite per quella classe, che vanno a modellare le restrizioni e i vincoli di cardinalità associati agli elementi costituenti il xs:complexType modellato.

A tale scopo, la Figura 16 riporta le restrizioni definite per la classe ontologia "II" associata al xs:complexType "II" che, come descritto nelle sezioni precedenti, in OWL si specificano come dichiarazioni di sottoclasse della classe su cui vigono le restrizioni di interesse. Si notino, ad esempio, i vincoli sulle relazioni (owl:DatatypeProperty) "root", "extension", "assigningAuthorityName", "displayable" e "root" generate per modellare i corrispondenti xs:attribute costituenti il tipo "II". Inoltre, si può notare che essendo il tipo complesso "II" definito tramite xs:extension di "ANY", la classe ontologica "II" è stata generata come sottoclasse della classe "ANY".

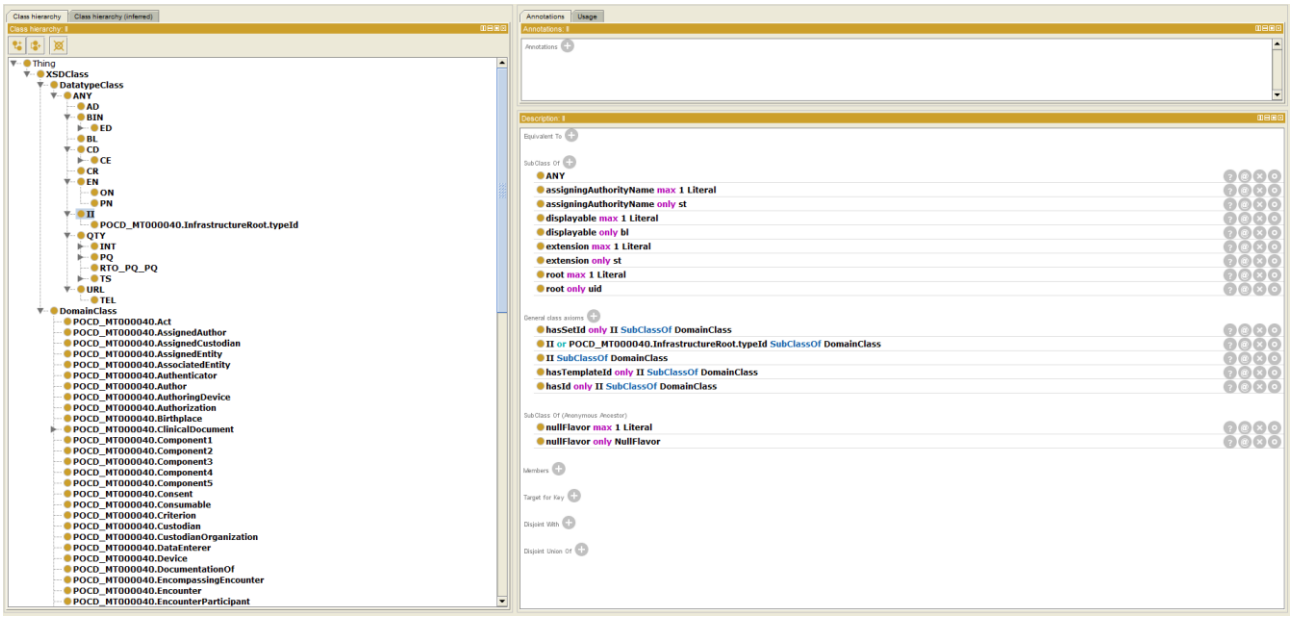


Figura 16 - La classe ontologica "ii" e le restrizioni definite su di essa

L'informazione sui valori ammessi per ogni proprietà modellata, intesi come classi ontologiche che appartengono al dominio e al range di una determinata proprietà, sono specificate nell'ontologia creata e possono essere analizzate e navigate tramite Protégé. A tale scopo si veda, ad esempio, la specifica di dominio e range per la owl:DatatypeProperty "assigningAuthorityName", riportata in Figura 17.

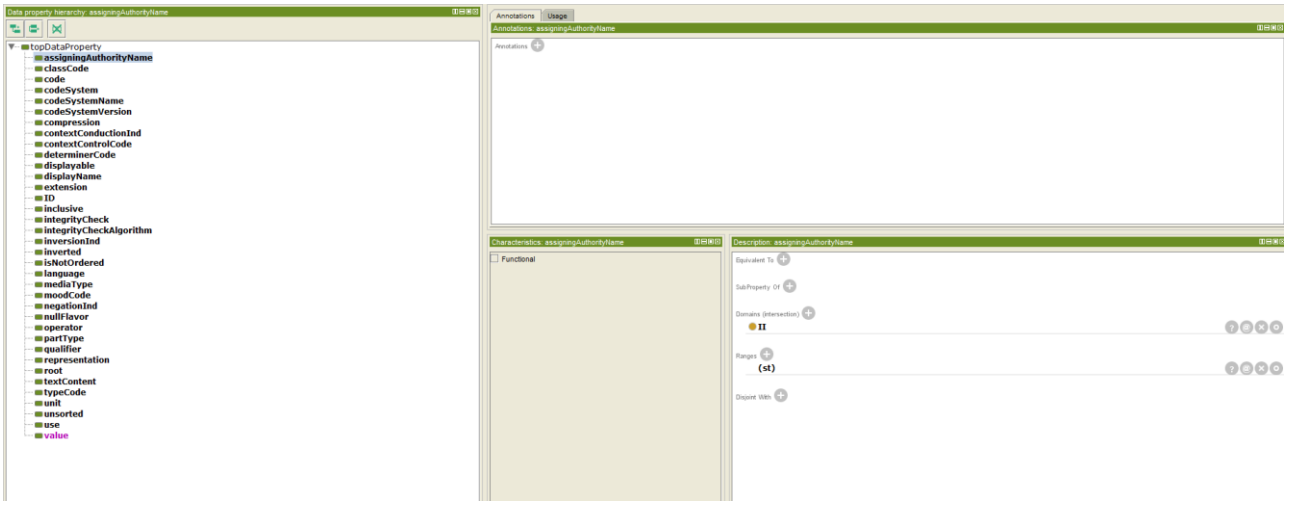


Figura 17 - Dominio e range per la proprietà assigningauthorityname

In Figura 18 sono riportate alcune delle owl:ObjectProperty generate per modellare xs:element interni a xs:complexType, e la cui tipologia faccia riferimento ad altri complex:Type. In particolare, differentemente dalle owl:DatatypeProperty, il nome dato alle owl:ObjectProperty è stato generato tramite il prefisso "has" per differenziarle dalle owl:DatatypeProperty e per mettere in risalto il concetto di relazione tra concetti. Inoltre, tre le varie owl:ObjectProperty definite, è stata generata una gerarchia di proprietà per mettere in evidenza la diversa tipologia di classi su cui agisce la proprietà, andando a distinguere tra classi di dominio e classi non di dominio secondo la gerarchia di classi discussa precedentemente.

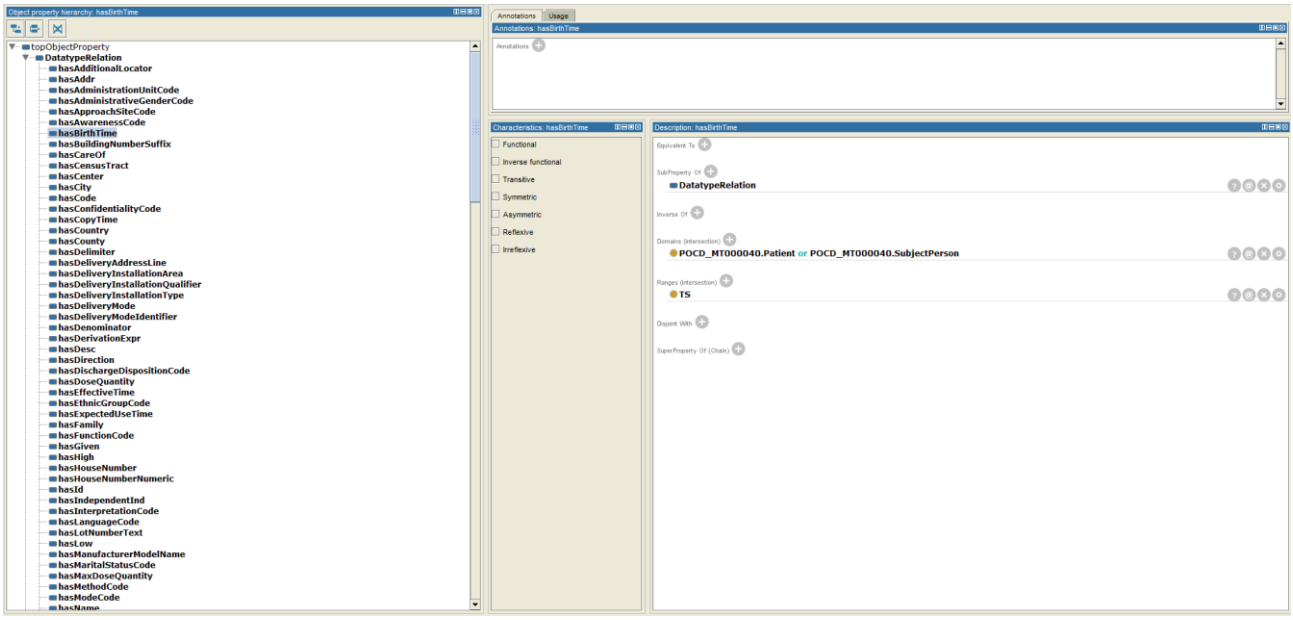


Figura 18 - Le owl:ObjectProperty generate dalla procedura automatica definita

Infine, la Figura 19 riporta alcuni degli rdfs:Datatype generati per modellare gli elementi xs:simpleType.
In particolare, si noti il tipo “ActClassExtract” definito come restrizione del tipo “cs”.

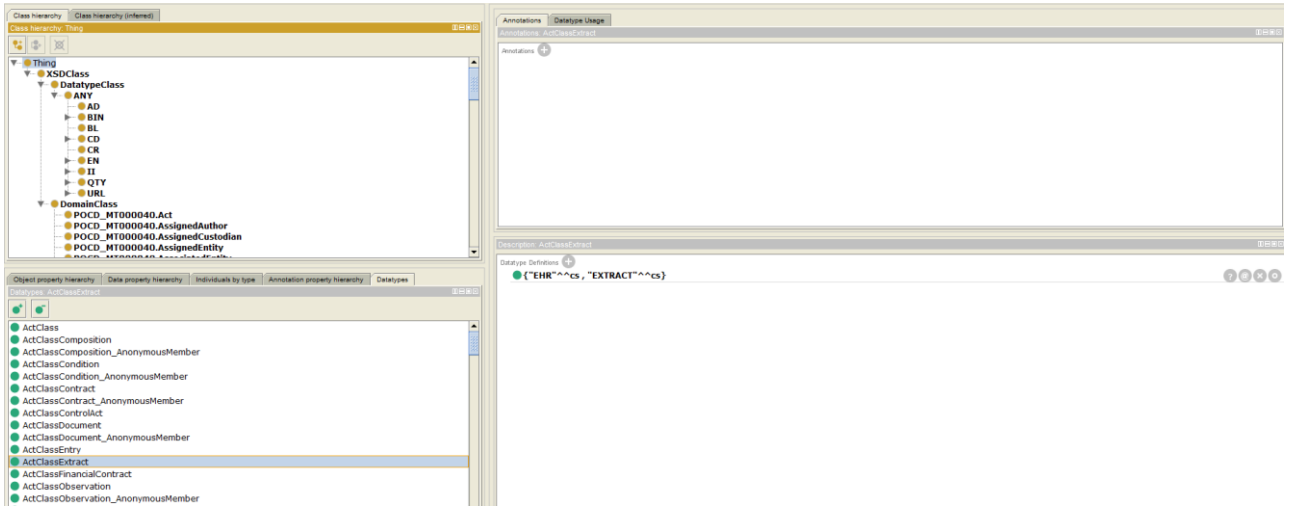


Figura 19 - rdfs:Datatype generati per modellare gli xs:simpleType degli Schema analizzati

5 Conclusioni

In questo lavoro è stato presentato un algoritmo di modellazione per la definizione automatica di ontologia a partire da XML Schema, a differenza di altri lavori noti in letteratura, questo algoritmo prevede una fase di pre-processamento degli XML Schema per ottimizzare gli stessi e ottenere una ontologia notevolmente semplificata. Inoltre, l’algoritmo aggiunge semantica all’ontologia prodotta andando a rappresentare delle distinzioni concettuali/gerarchiche tra i diversi oggetti di uno schema XSD per poter distinguere tra informazioni di dominio e quelle relative al tipo di dati.

Infine, questo lavoro ha permesso la definizione di una ontologia di riferimento dello standard HL7 CDA 2, infatti quelle disponibili in letteratura presentano una serie di incongruenze di carattere concettuale e tecnico che sono facilmente riscontrabili mediante un qualsiasi programma di editing delle ontologie come ad esempio Protegé.

La realizzazione dell’algoritmo apre ad una nuova fase di sperimentazione, testing e misurazione di performace dell’algoritmo, che sarà svolta nell’ambito di attività future.

6 Riferimenti bibliografici

BAKKAS 2014. (2014). *Restructuring of XML Documents in the Form of Ontologies*. World Academy of Science, Engineering and Technology International Journal of Computer, Information, Systems and Control Engineering Vol 8 No 2. J. Bakkas, M. Bahaj, A. Soklabi.

BEDINI 2011. (2011). *Transforming XML Schema to OWL Using Patterns*. 2011 Fifth IEEE International Conference on Semantic Computing (ICSC), pp.102-109. I. Bedini, C. Matheus, P. F. Patel-Schneider; Aidan Boran and Benjamin Nguyen.

BOHRING 2005. (2005). *Mapping XML to OWL ontologies*. In Leipziger Informatik-Tage, volume 72 of LNI, pp. 147–156. H. Bohring and S. Auer.

GRAU 2008. (2008). *OWL 2: The next step for OWL*. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 6 (4), 309–322. B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider and U. Sattler.

GUARINO 1995. (1995). *Formal ontology, conceptual analysis and knowledge representation*. International Journal of Human-Computer Studies, 43(5/6), pp. 625-640. N. Guarino.

GUARINO 1998. (1998). *Understanding, building and using ontologies*. International Journal of Human-Computer Studies 46, pp. 293-310. N. Guarino.

HL7CDA2 2009. (2009). *ISO/HL7 27932:2009 - Data Exchange Standards - HL7 Clinical Document Architecture, Release 2, http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7*

SCHNEIDER 2004. (2004). *OWL Web Ontology Language Semantics and Abstract Syntax*. Technical report, W3C Recommendation. P.F. Patel-*Schneider*, P. Hayes, I. Horrocks.

THUY 2012. (2012). *S-Trans: Semantic transformation of XML healthcare data into OWL ontology*. Knowledge-Based Systems 35, 349-356. P. T. T. Thuy, Y. Lee, S. Lee.

XMLSCHEMA 2004. (2004). W3C Recommendation - XML Schema ver. 1.1, http://www.w3.org/XML/Schema