Proyecto de Grado

Aplicación de Condiciones Semánticas, sobre documentos CDA.

InCo, Facultad de Ingeniería, Universidad de la República.

Estudiantes: Lorena Lambiaso – Lucía Grundel

Tutor: *Ana Erosa.* - ***Responsable****: Jacques Fauquex.*

Diciembre 2013.

RESUMEN

CDA *(Clinical Document Architecture*) es una especificación de arquitectura de documentos clínicos. Es el estándar de marcado, propuesto por **HL7** en su Versión 3, para definir la estructura y la semántica de documentos clínicos, que se requiere para intercambiar información clínica entre diversos sistemas.

Un conjunto de eventos clínicos representados en documentos CDA constituyen la base para la creación de la **Historia Clínica Electrónica** (HCE).

Una necesidad que deriva del intercambio de información clínica, es la de poder interoperar con otras instituciones u organismos centrales. Es así como además de interoperabilidad sintáctica, que garantiza el intercambio de información pero sin asegurar la correcta interpretación de los datos, surge el tema de la interoperabilidad semántica.

Recientemente **AGESIC** [1] en el marco del programa **Salud UY** ([2] - [3]) ha señalado como prioridad la creación del Banco Nacional de Historia Clínica, entendido como el lugar desde donde se podrá acceder a HCE de todos los pacientes del país. Dentro del mismo proyecto, el grupo de trabajo de AGESIC [4], ha definido que utilizará para la estandarización de los documentos la norma HL7 V3 CDA R2.

Es entonces en este marco, que consideramos de gran relevancia trabajar con documentos Clínicos CDA, junto con la aplicación de condiciones semánticas que permitan obtener información para el seguimiento y control del sistema de Salud a nivel nacional.

Se realiza en primer lugar un estudio sobre trabajos similares. Esta prospección nos permite identificar una línea de trabajo que converge en la generación de una ontología completa para HL7, la cual fue tomada como base del trabajo práctico, especializándola para CDA. De esta forma, se genera una ontología completa que provee todos los recursos conceptuales para **modelar cualquier tipo de documento CDA**, que denominamos **ontología matriz**. Conjuntamente a esta ontología, se desarrolla un **mecanismo de transformación** que permite traducir un documento CDA cualquiera, a lenguaje ontológico. El resultado de esta traducción, aporta los objetos concretos que son integrados a la mencionada ontología.

Con estos dos recursos (ontología matriz y transformación), se realizan pruebas prácticas de uso, mediante el desarrollo de un prototipo Java, que llamamos **Prot-PgSemCda**; y a través de la investigación y la utilización del servidor SPARQL denominado **Fuseki**.

En ambos ámbitos, sobre el modelo ontológico compuesto por la ontología matriz y las transformaciones de documentos CDA, se aplican consultas semánticas, que buscan filtrar información relevante y útil para el apoyo de toma de decisiones, detección de errores, estadística, entre otros.

Asimismo, se va incluyendo en el modelo ontológico, la nueva información que se obtiene del razonamiento sobre las instancias agregadas, de modo de favorecer la performance de la ejecución de consultas. También se utilizan mecanismos para agregar nuevos conceptos al modelo ontológico, lo cual permite enriquecer y potenciar el repositorio semántico.

El trabajo nos permite concluir que es posible generar un repositorio semántico de documentos CDA, con la potencialidad de extraer en cualquier momento, y de forma directa, la información que se requiera para su aplicación en los fines anteriormente detallados.

Queda como trabajo a futuro, el estudio de integración con otras ontologías existentes en el ámbito de la Salud. En particular, consideramos de gran interés, lograr la conexión semántica de la propuesta aquí presentada, con la ontología de SNOMED. Esto ofrecería un nivel de abstracción mayor en la construcción de consultas y en la capacidad de extraer información.

Índice de Contenido

CAPÍTULO 1

INTRODUCCIÓN

Un conjunto de eventos clínicos representados en documentos **CDA** *(Clinical Document Architecture*) constituyen la base para la creación de la Historia Clínica Electrónica (HCE). CDA es una especificación de arquitectura de documentos clínicos. Es el estándar de marcado, propuesto por **HL7** en su Versión 3, para definir la estructura y la semántica de documentos clínicos, que se requiere para intercambiar información clínica entre diversos sistemas.

Una necesidad que deriva del intercambio de información clínica, es la de poder interoperar con otras instituciones u organismos centrales. Es así como además de interoperabilidad sintáctica, que garantiza el intercambio de información pero sin asegurar la correcta interpretación de los datos, surge el tema de la interoperabilidad semántica.

Recientemente AGESIC [1] en el marco del programa Salud UY ([2] - [3]) ha marcado como prioridad la creación del Banco Nacional de Historia Clínica, entendido como el lugar desde donde se podrá acceder a HCE de todos los pacientes del país. "*Debe asegurarse que independientemente de dónde se realice la atención el médico y el paciente tengan acceso a la historia clínica. También es un lugar que puede resumir la HCE de un paciente, esto es muy importante a los efectos de gestión epidemiológica o estadística. Ahora hay que definir estándares, normas técnicas y crear de común acuerdo un modelo de historia clínica, para posteriormente trabajar con esto en todos los ámbitos de salud pública y privada."* [3].

Dentro del mismo proyecto, el grupo de trabajo de AGESIC en el [4], ha definido que utilizará para la estandarización de los documentos el estándar HL7 V3 - CDA R2.

Consideramos de gran relevancia trabajar con documentos Clínicos CDA, junto con condiciones semánticas que permitan gestión epidemiológica, así como seguimiento y control sobre documentos para aplicar sobre los documentos en forma automática.

El presente proyecto tiene como objetivo abordar el tema de la interoperabilidad semántica, y para esto el enfoque de ontologías aporta el elemento fundamental para estructurar la información contenida en los documentos.

Se realizará primero un estudio sobre trabajos similares, luego se planteará el modelo ontológico específico para los documentos, para finalmente demostrar la aplicabilidad de los resultados obtenidos mediante la realización de un prototipo.

En este primer capítulo, se presenta una descripción general del problema, los objetivos del proyecto, los resultados esperados y el alcance del mismo. Posteriormente, se esquematiza la organización del presente informe.

1.1 Descripción del Problema

Los documentos CDA son una importante fuente de datos clínicos. A partir de la información contenida en ellos se desea realizar consultas, para obtener resultados sobre qué documentos cumplen con los criterios establecidos.

Debido a que dentro de los documentos CDA se puede estructurar cualquier tipo de documento clínico, se hace necesario un mecanismo de consulta que permita obtener conclusiones en base a la información contenida, y analizar la toma de decisiones en función de las condiciones existentes; condiciones supeditadas a constantes cambios de acuerdo a nuevos criterios o a nuevos ejes de investigación.

1.2 Objetivos

- Estudiar el estado del arte de modelos de información semántica en la salud.
- Investigar, evaluar y proponer formas de especificar condiciones semánticas aplicadas a CDAs.
- Estudiar la aplicabilidad de estos mecanismos en el contexto nacional y realizar una prueba de concepto.

1.3 Resultados Esperados

- Un documento que presente una parte importante del estado del arte sobre los modelos de información semántica en la salud y sus aplicaciones, así como una propuesta de aplicación y/o adaptación de estos modelos al contexto nacional, teniendo en cuenta la toma de decisiones en base a condiciones semánticas.

- Un prototipo a nivel de prueba de concepto de las ideas expuestas en los documentos mencionados.

1.4 Alcance del proyecto

- Definir los componentes necesarios para la creación de estructuras semánticas sobre CDAs.

- Investigar y elegir las herramientas necesarias para poder realizar las consultas.

- Utilizar los conocimientos adquiridos en algún caso práctico real para evaluar la factibilidad de la solución.

1.5 Organización del Documento

Capítulo 1: Introducción al problema del proyecto y la motivación para realizarlo. Se indica el alcance y los objetivos del proyecto.

Capítulo 2: Descripción de los conceptos básicos relacionados.

Capítulo 3: Investigación de los antecedentes, y del estado del arte actual, así como trabajos que sirven como inicio del propuesto.

Capítulo 4: Proceso del modelado ontológico, realizado sobre los documentos CDA.

Capítulo 5: Presentación de los prototipos de aplicación realizados. Detallando las particularidades de cada uno.

Capítulo 6: Conclusiones del trabajo realizado así como los trabajos futuros.

CAPÍTULO 2

CONCEPTOS RELACIONADOS

En este capítulo se describirán los principales temas que sirven de sustento al proyecto. Comienza explicando temas generales de interoperabilidad, para luego enfocar en la Web Semántica y sus principales herramientas. Por último, se describe otro aspecto como es la informática médica, explicando los estándares actualmente utilizados.

2.1 Interoperabilidad y sus diferentes tipos.

Para comenzar con la explicación de los conceptos, consideramos relevante destacar las diferentes acepciones que el concepto de interoperabilidad tiene: para *IEEE* [5] se define como la habilidad de dos o más sistemas o componentes para intercambiar datos, y posteriormente utilizar dicha información intercambiada. Desde el punto de vista de la informática aplicada a la salud, el *Institute of Medicine of the National Academies* (IOM) usa la siguiente definición: *"Interoperabilidad es la habilidad de los sistemas para trabajar juntos, en general gracias a la adopción de estándares. La interoperabilidad no es solamente la habilidad de intercambiar información sanitaria, sino que requiere la habilidad de entender lo que se ha intercambiado"* (Institute of Medicine, 2004). [6]

Todas las definiciones tienen en común, la necesidad de intercambiar información para luego, de acuerdo al objetivo planteado, utilizar esta información.

Existen diferentes tipos de interoperabilidad, y es importante distinguir a qué refiere cada uno de ellos:

- ***Interoperabilidad sintáctica*:** Refiere a la capacidad de los Sistemas Información (SI) para leer datos procedentes de otros sistemas y obtener una representación que pueden usar. Esto significa que en todos los SI involucrados, tanto la codificación de los datos como los protocolos de acceso son compatibles. Desde luego, esto no implica que los SI procesen los datos de una manera consistente con su significado. La interoperabilidad sintáctica es alta cuando se encuentran disponibles analizadores sintácticos (parsers) y APIs para manipular los datos intercambiados, de manera que las organizaciones pueden usarlos para incorporar esos datos a sus SI. El grado más alto de interoperabilidad sintáctica se alcanza cuando dos SI cualesquiera intercambian datos cualesquiera sin ningún acuerdo previo y de manera completamente automática.

  El primer paso para conseguir la interoperabilidad sintáctica consiste en compartir algún tipo de formato de representación de datos para la información enviada. El segundo paso, reside en que los datos intercambiados compartan, además de un mismo formato, una misma estructura. [7]

- ***Interoperabilidad semántica:*** Es la capacidad de los SI para intercambiar información basándose en un significado común de los términos y expresiones que se usan. En otras palabras, denota la capacidad de los SI para intercambiar información consistente con el significado que se le supone. La interoperabilidad semántica no debe confundirse con la sintáctica. La sintáctica, refiere a la dificultad de procesar automáticamente los datos, no a su "contenido". La semántica, se refiere a la dificultad de relacionar los términos desconocidos que aparecen en los datos con los ya conocidos o definidos [7]

2.2 Web semántica.

La web semántica propone dotar de significado a la Web actual, de esta manera se enfoca en las limitaciones actuales mediante la introducción de descripciones explícitas del significado a los recursos, clasificando y dotando de estructura a los mismos, y ampliando los servicios disponibles en la red con este fin.

Los principales elementos de la web semántica, se describen en el siguiente diagrama:

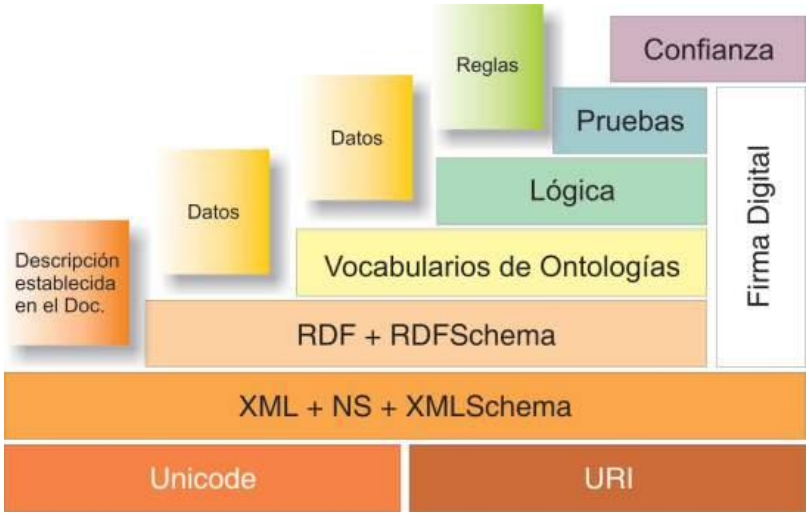


ILUSTRACIÓN 1 - ESTRUCTURA DE LA WEB SEMANTICA

Donde:

• *XML* (Extensible Markup Language). Es el lenguaje que actualmente se usa para intercambiar datos en la red.
• *RDF* (Resource Description Framework). Es un lenguaje que describe metadatos y fuentes de información (documentos, imágenes, etc.).
• *Ontologías.* Son conceptualizaciones que definen el significado de un conjunto de conceptos para un determinado dominio. Las ontologías se expresan en lenguajes formales como OWL.
• *Lógica*. El razonamiento lógico permite determinar si los datos son correctos e inferir conclusiones a partir de ellos.
• *Prueba*. Las pruebas explican y verifican los pasos de los razonamientos lógicos.
• *Confianza.* Se refiere a técnicas que aseguran la identidad y fiabilidad de los datos y servicios.

En las siguientes secciones, se explicarán los diferentes componentes descritos que utiliza la Web semántica, se hará especial hincapié en la capa correspondiente a RDF y Nivel ontológico, que son los fundamentales para el proyecto.

2.2.1 Nivel de Recursos – URI, UNICODE -

Corresponde al nivel base de la estructura, en él se incluye la identificación de recursos Web, estableciendo así la capital importancia que tiene definir el conjunto de recursos distribuidos por la red.

Los ***URIs*** (Uniform Resource Identifier) identifican de forma inequívoca un recurso introducido en la red, este identificador además cumplirá la función de identificador de objetos en el mundo real.

Unicode, es una norma de codificación del texto que permite utilizar los símbolos de diferentes idiomas sin observar caracteres extraños. Su objetivo es asignar a cada posible carácter de cada posible lenguaje un número y nombre único. Esto permite expresar información en la Web Semántica en cualquier idioma. [8]

2.2.2 Nivel Sintáctico

En este nivel se soluciona el problema de cómo definir distintos lenguajes de etiquetado para añadir contenido semántico a las páginas web.

El ***Extensible Model Language*** **(XML)** es un metalenguaje, es decir, un lenguaje para escribir lenguajes, donde el orden de los elementos es importante, su modelo representa un árbol. La característica que ha hecho de XML la elección más acertada es la capacidad de: formalizar y validar los documentos, la estructuración y la capacidad de ampliación. [9]

El XML es un formato simple basado en texto para representar información estructurada: documentos, datos, configuración, libros, transacciones y mucho más. Fue derivado de un formato estándar antiguo llamado SGML (ISO 8879), con el propósito de ser más adecuado para uso de la Web.

Es uno de los formatos más utilizados en la actualidad para el intercambio de información estructurada: entre los programas, sistemas y personas, tanto a nivel local como a través de redes.

Representa una primera aproximación a la web semántica, y aunque no está expresamente pensado para definir ontologías, es el estándar más extendido hoy día en las aplicaciones de esta línea pre- web semántica. XML permite estructurar datos y documentos en forma de árboles de etiquetas con atributos.

Dentro de las limitaciones fundamentales que se derivan del uso de XML, cabe destacar que no se puede lograr interoperabilidad completa. Sirve para otorgar de estructura y formato al documento, pero no impone una interpretación común de los datos del documento. Desde el punto de vista semántico, no permite reconocer ni los objetos ni las relaciones existentes en el dominio de interés; en realidad su aporte es solamente a nivel de interoperabilidad sintáctica [10].

Con **XML Schema (XMLS)** se pueden acordar de antemano las estructuras que se van a utilizar, así como manejar tipos de datos primitivos y derivados [11]

El propósito de un esquema XSD es definir y describir una clase de documentos XML mediante restricciones de la estructura y poder documentar el significado. Dichas restricciones afectan al uso y las relaciones de sus partes constituyentes: tipos de datos, elementos y su contenido, y los atributos y sus valores. Los esquemas también pueden proporcionar para la especificación de la información del documento adicional, tales como la normalización y el incumplimiento de los atributos y valores de los elementos.

Cualquier aplicación que consume XML bien formado puede utilizar el formalismo definido aquí para expresar restricciones sintácticas y estructurales. El formalismo XSD permite un nivel útil de la comprobación de restricciones que se describen y es implementado por un amplio espectro de aplicaciones XML. [12]

2.2.3 Nivel Semántico.

Si bien XML permite definir datos de una forma estructurada, de manera que puedan ser compartidos y procesados automáticamente, no permite especificar la semántica de dicha estructura, de una manera formal que pueda ser interpretada por un ordenador.

Además, el orden en el cual los elementos aparecen en un documento XML es significativo y muchas veces necesario. Sin embargo, el orden impuesto en XML no tiene tanto sentido en el mundo de los metadatos.

El ***Resource Description Framework (RDF)*** fue creado en agosto del 1997 bajo el auspicio del World Wide Web Consortium (W3C[1]), con la finalidad de crear un formato que posibilitara la compatibilidad entre diversos sistemas de metadatos.

RDF es un lenguaje formal para describir información estructurada y metainformación. El modelo es un grafo etiquetado y dirigido donde el orden no es relevante y tiene como fin proporcionar interoperabilidad entre aplicaciones que intercambian información legible por la máquina en la Web.

Veamos qué significa el nombre que se resume en la abreviatura R.D.F.: [13]

- ***(R - Resource)***: los recursos son un concepto central en la Web Semántica, pueden referir tanto a una página Web, una imagen, un video, así como también a una persona, un lugar, un dispositivo, un evento, una organización, un producto o un servicio. Más técnicamente hablando, todo lo que puede ser identificado por un URI puede ser considerado como un recurso.

- ***(D - Description)***: las descripciones de los recursos son esenciales para la comprensión y el razonamiento acerca de ellos. En el caso más general, una descripción es un conjunto de atributos, características, y las relaciones sobre el recurso.

- ***(F - Framework)***: significa que proporciona modelos, lenguajes y sintaxis de estas descripciones.

Las declaraciones se representan mediante tripletas, cada tripleta expresa una relación entre los recursos indicados por los nodos que conecta. Sus tres componentes son:

a) Un **sujeto** expresado por una URI o un nodo en blanco, corresponde también al recurso del enunciado.
b) Un **predicado** expresado por una URI, el cual representa una relación, y es denominado también como una propiedad.
c) Un **objeto** expresado por una URI, un literal o un nodo en blanco, corresponde al valor que tiene la propiedad.

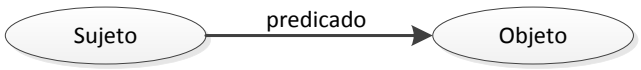


ILUSTRACIÓN 2: REPRESENTACION GRÁFICA DEL LAS TRIPLETAS.

Una tripleta RDF se escribe convencionalmente en el orden: sujeto (s), predicado (p), objeto (o).

La dirección del arco es significativo: siempre señala hacia el objeto. Los nodos de un grafo RDF son sus sujetos y objetos.

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

2.2.3.1 Elementos RDF

El modelo de datos de RDF se basa en tres elementos fundamentales:

- **recursos**: son todas las cosas descritas por expresiones RDF, desde una página web completa como un documento HTML "http://www.w3.org/Overview.html" o una colección completa de páginas, hasta una parte de una página web, por ejemplo, un elemento HTML o XML específico dentro de un documento fuente. Un recurso puede ser también un objeto que no sea directamente accesible vía Web, por ejemplo un libro impreso. Los recursos se designan siempre mediante URIs y la extensibilidad de estos, permite la introducción de identificadores para cualquier entidad imaginable.

- **propiedades**: una propiedad es un aspecto específico, característica, atributo o relación que puede utilizarse para describir un recurso. Cada propiedad tiene un significado específico, define sus valores permitidos y los tipos de recursos que puede describir.

- **declaraciones** (sentencias o enunciados): una declaración RDF es una propiedad más el valor de dicha propiedad para un recurso específico.

Así pues, una declaración o sentencia está compuesta por tres partes individuales:

a. *sujeto*: recurso
b. *predicado*: propiedad
c. *objeto*: valor de la propiedad (puede ser otro recurso -especificado por un URI- o un literal (una cadena simple de caracteres u otros tipos de datos primitivos definidos por XML).

El contenido de un literal no es interpretado por RDF en sí mismo y puede contener marcado XML adicional. Los literales se distinguen de los recursos en que el modelo RDF no permite que los literales sean sujeto de una declaración.

2.2.3.2 Ontologías Simples en RDF Schema

RDF Schema (RDFS) es una extensión semántica de RDF. Consecuentemente, cada documento RDFS es un documento RDF bien formado. Un documento RDFS es una especificación procesable por máquinas que describe conocimiento de algún dominio de interés.

Aunque RDFS es útil como lenguaje ontológico, también tiene sus limitaciones. RDFS es a veces categorizado como un lenguaje de representación para las llamadas ontologías ligeras.

2.2.3.3 Identificación de Recursos RDF

1. Nombres en RDF

RDF está orientado a identificar recursos utilizando identificadores Web, llamados *Uniform Resource Indentifiers* (URI's), como nombres que claramente distinguen cada recurso de los demás.

Son una generalización de *Uniform Resource Locators* (URL's). En numerosas aplicaciones, se necesita intercambiar información de una diversa variedad de objetos, no solamente de páginas web. Aunque las URI's puedan referir a recursos que no están localizados en la Web, igualmente se basan en un schema de construcción similar que las direcciones Web comunes.

No importa si algún documento pueda ser recuperado de esa ubicación, lo importante es que el recurso es relevante en el contexto dado.

2. Valores de datos en RDF

Los literales son nombres reservados para recursos RDF de un cierto tipo de datos. Los literales para los cuales no se especifica ningún tipo de datos, son interpretados como cadenas de texto (strings).

En la representación de un grafo RDF, los literales se incluyen en cajas rectangulares, y además, no pueden ser el origen de un arco. Esto significa que un literal puede ser el objeto de una declaración RDF, pero no el sujeto o el predicado.

3. Sintaxis para RDF

RDF es un modelo y no un formato en sí mismo, es por ello que para que pueda ser interpretable de forma automática necesitamos utilizar un lenguaje que sea procesable, lo que se conoce como serializaciones del RDF. Los grafos RDF son una colección de sus tripletas, dadas en un orden arbitrario. Existen diferentes formas de poder representarlos, de forma serializada, entre ellas: **XML** y **TURTLE**.

A modo de poder entender la diferencias entre ambas, y las particularidades de cada una, se ejemplificará en los dos formatos el siguiente grafo:

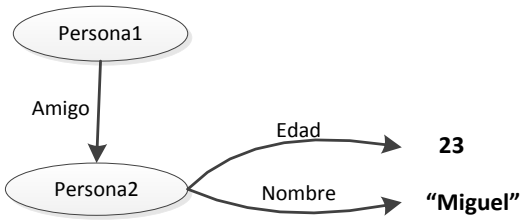


ILUSTRACIÓN 3: GRAFO DE PERSONAS

Serialización XML de RDF.

La serialización RDF/XML es que se encuentra estandarizada y publicada de forma oficial por el W3C en el *"RDF/XML Especificación de sintaxis en 2004"* Publicación disponible en: www.w3.org/TR/rdf-syntax-gram-mar/.

El objetivo principal de RDF/XML consiste en ser procesable por máquina y cumplir con el estándar de facto de formato de documentos Web, XML. Esto permite a los documentos RDF, ser fácilmente intercambiables entre muy diferentes tipos de sistemas y aplicaciones. Se debe tener en cuenta que RDF/XML puede encontrarse engorroso de leer, pues no se pretende para el consumo humano. [13]

Conceptualmente, RDF/XML se construye a partir de una serie de descripciones más pequeñas, cada una de las cuales traza un camino a través de un grafo RDF. Estas rutas de acceso se describen en términos de los nodos (sujetos) y los enlaces (predicados) que los conectan con otros nodos (objetos). [14]

Para representar la imagen (*Ilustración 3: Grafo de personas*) en formato **RDF/XML**:

```
(1) <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
(2)       <rdf:Description rdf:About=URI_Persona1>
(3)           <Amigo>
                  <rdf:Description rdf:About= URI_Persona2>
                            <Nombre>Miguel </Nombre>
                            <Edad>23</Edad>
                  </rdf:Description>
             </Amigo>
      </rdf:Description>
 </rdf:RDF>
```

Referencias anteriores en la especificación XML:

(1) Todos los documentos RDF/XML comienzan con el root *<rdf:RDF>* con el fin de declarar que es un documento XML para representar RDF. Los atributos del root, se utilizan para representar *prefijos* (URI's que se utilizarán).

(2) La etiqueta <rdf:Description> se utiliza siempre con un "id", o con un "about", junto con una URI, los cuales identifican el recurso del cual se les definirán los hijos (las propiedades para describirlos). Se entiende por: *"URI_Persona1*" la URI que corresponde al objeto que identifica a Persona1.

(3) A continuación se describen mediante etiquetas (en este caso <Amigo>, las propiedades que describirán al elemento principal -para este ejemplo: Persona1.

Por otra parte, entre las críticas a RDF/XML se destaca que existen varias posibilidades para expresar el mismo grafo RDF, o sea, que la representación no es única. También es un punto débil, la dificultad para lectura humana.

Serialización Turtle de RDF

Como contraposición a la dificultad de lectura, surge la representación *Turtle*, la cual propone en su formato varias abreviaciones que la hacen compacta y sencilla a la hora de comprender los elementos. La especificación se encuentra disponible en: www.w3.org/TeamSubmission/turtle/.

Para representar la imagen (*Ilustración 3: Grafo de personas*) en formato **Turtle**:

```
(1) @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
(2) URI_Persona1      : Amigo     URI_Person2.
(3) URI_Persona2      : Nombre    "Miguel";
                      : Edad      23.
```

Referencias anteriores en la especificación Turtle:

(1) La declaración **@prefix** permite representar todos los prefijos que se utilizarán. La sentencia termina con un punto (**.**)

(2) Se ingresan las sentencias que describen a los sujetos. Todas comienzan con la identificación del sujeto (URI), luego se describe la propiedad y el objeto. Cada línea contiene la tripleta en el siguiente orden: **<sujeto, propiedad, objeto>**

(3) Con el fin de poder abreviar cuando se agregan diferentes propiedades sobre un mismo sujeto, es que se separan cada una de las sentencias con un (**;**)

4. Limitaciones semánticas de RDF(S)

Las capacidades de modelado de RDF(S) están restringidas. Dependiendo de las características de la ontología que se aborda, puede ser una herramienta útil, siempre y cuando se requieran medios expresivos acotados a las posibilidades expuestas anteriormente. Sin embargo, posee ciertas limitantes:

- Impide definir propiedades de ámbito local, es decir, no podemos establecer restricciones de rango aplicables solo a determinadas clases.

- No permite expresar la disyunción de clases: establecer que una clase C no comparte ningún elemento con la clase D.

- No podemos definir clases en función de otras, como unión, complemento o intersección.

- No provee mecanismos para establecer cardinalidad de propiedades.

- Tampoco tiene forma de establecer propiedades específicas de las propiedades, por ejemplo, que una propiedad es una función, que es transitiva o que es inversa de otra. [15]

2.2.4 Nivel Ontológico

Una ***ontología*** es una jerarquía de conceptos con atributos y relaciones, que define una terminología consensuada para definir redes semánticas de unidades de información interrelacionadas. Una ontología proporciona un vocabulario de clases y relaciones para describir un dominio, poniendo el acento en el hecho de compartir el conocimiento y el consenso en la representación de éste. La idea es que la web semántica esté formada por una red de nodos tipificados e interconectados mediante clases y relaciones, definidas por una ontología compartida por sus distintos autores. [16]

Con las ontologías, los usuarios organizarán la información de manera que los agentes de software podrán interpretar el significado y, por tanto, podrán buscar e integrar datos mucho mejor que ahora. Gracias al conocimiento almacenado en las ontologías, las aplicaciones podrán extraer automáticamente datos de las páginas web, procesarlos y sacar conclusiones de ellos, así como tomar decisiones y negociar con otros agentes o personas. [8]

El **OWL**, lenguaje de ontologías para la web, se ha convertido en recomendación del W3C el 10 de febrero de 2004. Se encuentra disponible en: http://www.w3.org/TR/owl-features/ Es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web.

OWL se puede formular en RDF, por lo que se suele considerar una extensión de éste. OWL incluye toda la capacidad expresiva de RDF y la extiende con la posibilidad de utilizar expresiones lógicas. Asimismo, mejora muchas de las limitaciones de RDF.

Puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos (ontologías). Se trata de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. [8]

OWL se presenta en tres niveles del lenguaje, con el propósito de brindar al usuario la elección entre diferentes grados de expresividad, que se correlacionan con los distintos tipos de lógicas descriptivas existentes. De menos a más expresivos, tenemos:

- **OWL Lite**. Es suficiente para los casos en que se requieren posibilidades de clasificación en la jerarquía de clases de la ontología y restricciones simples.

- **OWL DL**. Es el lenguaje propicio para los casos donde se necesita la máxima expresividad pero sin perder completitud computacional, es decir, se garantiza que todas las conclusiones son computables; y decidibilidad, o sea, que todos los cálculos acaban en un tiempo finito. Incluye todos los constructores de OWL, pero sólo se pueden usar con restricciones. Es ampliamente soportado por la mayoría de las herramientas de software. Contiene a OWL Lite.

- **OWL Full**. Está pensado para las situaciones en que se requiere el máximo poder expresivo pero sin asegurar la decidibilidad. Es el único sublenguaje que contiene todo RDFS. Es semánticamente difícil de trabajarlo, y está presente en escasas herramientas de software. Contiene a OWL Lite y OWL DL.

Una ontología OWL es expresada básicamente en términos de clases y propiedades como fue visto para RDF(S), pero en OWL pueden ser descritas relaciones mucho más complejas entre estas clases y propiedades.

Todo documento OWL está formado por:

- Cabeceras de identificación para cada ontología.
- Un conjunto de clases, propiedades y restricciones, para declarar y definir las ontologías.

OWL no impone ninguna restricción con respecto al orden en que deben aparecer estos elementos

2.2.4.1 Clases, roles e individuos

Los constructores fundamentales de OWL son las clases y las propiedades, que ya son conocidas a partir de RDF(S), y los individuos, que en RDF son declarados como instancias de clases.

Las propiedades de OWL son también llamadas roles. Las clases son definidas en OWL usando **owl:Class**, la cual es una subclase de **rdfs:Class**. Existen dos clases predefinidas: **owl:Thing** es la clase más general, y tiene a todos los individuos como instancias; y **owl:Nothing**, la cual no tiene ninguna instancia por definición.

Tal como en RDF, los individuos pueden ser declarados como instancias de una clase. Esto es llamado asignación de clase.

Respecto a los roles, existen dos tipos en OWL. En primer lugar, los roles llamados abstractos, que conectan individuos con individuos, y se declaran mediante la expresión o**wl:ObjectProperty.** En segundo lugar, tenemos los roles concretos, que conectan individuos con valores, y que se establecen a través de la expresión **owl:DataProperty.**

Tanto **owl:ObjectProperty** como **owl:DatatProperty** son subclases de la clase **rdf:Property**.

El dominio y el rango de los roles puede ser declarado vía **rdfs:domain** y **rdfs:range** tal como en RDFS. Todos los tipos de datos de XML pueden ser en principio utilizados en OWL. Como en RDF, es posible declarar explícitamente dos individuos conectados por un rol, esto es llamado asignación de rol.

1. ***Relaciones de Clases***

Las clases de OWL pueden ser puestas en relación entre sí, mediante **rdfs:subClassOf**. Este constructo es transitivo tal cual lo es en RDFS, y por tanto nos permite hacer inferencias simples.

Cada clase es una subclase de **owl:Thing**, y **owl:Nothing** es una subclase de cualquier otra clase. Dos clases pueden ser declaradas disjuntas usando **owl:disjointWith**.

Dos clases pueden ser declaradas equivalentes usando **owl:equivalentClass.** De igual forma, esto puede ser logrado estableciendo que dos clases son subclases mutuas.

2. Relaciones entre Individuos

OWL nos permite declarar que dos individuos son de hecho el mismo, mediante **owl:sameAs**. De forma similar, es posible declarar que dos individuos son diferentes. **owl:differentFrom**. El lenguaje también provee un atajo para establecer que los elementos de una colección son todos diferentes entre sí, esto se realiza mediante las expresiones **owl:allDifferent** y **owl:distinctMembers**.

3. *Clases cerradas*

OWL provee mecanismos para establecer que una determinada clase contiene solamente determinados elementos, y no más. Es decir, que permite la definición de clases cerradas. Esto se logra mediante la expresión **owl:oneOf** de tipo colección, que se integra a la definición de una clase. Todos los individuos declarados dentro de esta colección, son los elementos constituyentes de la clase.

4. Constructores de Clases Booleanas

Con el propósito de poder expresar conocimiento más complejo, OWL brinda constructores que permiten combinar clases atómicas en clases complejas. Dichos constructores son los correspondientes a los conectores lógicos para la conjunción, disyunción y negación (or, and y not), y en el lenguaje OWL son expresados por **owl:unionOf**, **owl:intersectionOf**, y **owl:complementOf**, respectivamente. Los constructores booleanos de clases pueden ser anidados de forma arbitraria y profunda.

5. Restricciones de Roles

Por restricciones de roles entendemos otro tipo de constructores basados en la lógica, para clases complejas. Para marcar este tipo de restricciones, OWL ofrece la expresión **owl:Restriction**, como una subclase de **owl:Class**. Las restricciones de rol tienen la siguiente forma general:

```
<owl:Restriction>
      <owl:onProperty rdf:resource="(nombre_rol)" />
            (exactamente una restricción de valor o cardinalidad)
</owl:Restriction>
```

La primera restricción sobre roles está derivada del cuantificador universal de la lógica de predicados y define una clase como el conjunto de todos los objetos para los cuales el rol dado solamente alcanza valores de la clase dada. Esto se determina mediante la expresión **owl:allValuesFrom**.

La segunda restricción, se realiza mediante **owl:someValuesFrom**, el cual está íntimamente relacionado con el cuantificador existencial de la lógica de predicados. Por medio de **owl:allValuesFrom**, podemos decir algo acerca de todos los elementos del rango del rol. Con **owl:someValuesFrom**, podemos decir algo acerca de al menos uno de esos elementos.

De una forma similar, podemos también hacer restricciones respecto a la cardinalidad, mediante las expresiones **owl:maxCardinality**, **owl:minCardinality** y **owl:cardinality**.

La restricción **owl:hasValue** es un caso especial de **owl:someValuesFrom**, donde un individuo particular puede ser dado como un valor para el rol.

6. ***Relaciones de Roles***

Los roles pueden ser relacionados de varias formas. En particular, **rdfs:subPropertyOf** también puede ser utilizada en OWL. De forma similar es posible establecer que dos roles son de hecho idénticos. Esto se realiza utilizando **owl:equivalentProperty**.

Dos roles pueden ser uno el inverso del otro. Esto es declarado utilizando **owl:inverseOf**.

2.2.5 Inferencia y razonamiento.

Los razonadores semánticos, son herramientas útiles que permiten verificar consistencia, satisfactibilidad de los conceptos y determinar la clase más específica de un individuo. Los razonadores utilizan un motor de inferencias y un conjunto de reglas expresadas en un lenguaje semántico como OWL.

La complejidad del razonamiento en ontologías expresadas en OWL ha sido optimizada con "*Description Logic"* (DL) y actualmente existen razonadores como FACT++, Racer, Pellet, Flora-2, KAON-2. [17]

Para las máquinas actuales, el término "comprender" no debe entenderse en el sentido de la comprensión humana sino en el de inferir, deducir. Las máquinas son capaces de inferir conclusiones a partir de datos mediante procesos lógico-matemáticos. Estos datos incorporan información en los documentos en la forma de metadatos y por medio de un lenguaje formal, posibilitando que las máquinas puedan procesar la información mediante la aplicación de reglas lógicas y seguidamente extraer inferencias lógicas.

La Lógica resulta de suma relevancia en la Web Semántica por tres motivos, a saber:

1) permite desarrollar lenguajes formales para representar conocimiento;

2) proporciona semánticas bien definidas;

3) proporciona reglas de inferencia a partir de las cuales se pueden extraer consecuencias del conocimiento.

Lo que se conoce como "interpretación semántica automática de documentos" no pasa de ser la aplicación de reglas lógicas a datos presentados en algún lenguaje formal (como OWL o DAML+OIL).

Los lenguajes formales (OWL o DAML+OIL) se basan en lógica descriptiva. Ésta proporciona un formulismo para representar y expresar conocimiento humano, basándose en métodos de razonamiento bien establecidos y procedentes de un subconjunto de la lógica de predicados o lógica de primer orden. Los lenguajes naturales no resultan ser los más apropiados para expresar conceptos sin ambigüedad. Con la lógica que hay detrás de los lenguajes de la Web Semántica, las máquinas pueden realizar inferencias.

2.2.6 Lenguaje de Consulta: SPARQL

El ***Simple Protocol And RDF Query Language*** (**SPARQL**) es el lenguaje propuesto por W3C, para consultas de datos RDF publicados en la Web, el cual permite obtener información de grafos RDF y ontologías OWL.

SPARQL se describe mediante especificaciones, la última versión convertida en estándar en Marzo 2013 describe sus funcionalidades: consiste en un lenguaje de consulta, un formato para las respuestas, y un medio para el transporte de consultas y respuestas [18] - [19]:

- ***SPARQL Query Language***: Componente principal de SPARQL. Explica la sintaxis para la composición de sentencias y su concordancia.
- ***SPARQL Protocol for RDF***: Formato utilizado para la devolución de los resultados de las búsquedas (queries SELECT o ASK), a partir de un esquema XML.
- ***SPARQL Query Results XML Format****:* Describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores. Utiliza WSDL para definir protocolos remotos.

Este lenguaje de consulta permite [20]*:*

- Extraer información de URIs, literales y nodos vacíos; así como los valores de datos estructurados y semiestructurados.
- Explorar los datos, mediante consulta de relaciones, desconociendo la estructura interna.
- Realizar la unión de base de datos dispares, mediante consultas sencillas.

2.2.6.1 Estructura de consultas

Las querys más sencillas que podemos construir necesitan de la cláusula SELECT y de la cláusula WHERE. La cláusula SELECT identifica las variables que aparecen en el resultado de la Query. Las cláusulas que siguen al WHERE, están formada por tripletas que restringen los resultados obtenidos. [21]

El siguiente diagrama muestra un resumen de la estructura de las sentencias, que componen una consulta SPARQL:

Sintaxis básica de una consulta SPARQL	
Prologue (optional)	BASE <iri> PREFIX prefix: <iri> (repeatable)
Query Result forms (required, pick 1)	SELECT (DISTINCT)sequence of ?variable SELECT (DISTINCT)* DESCRIBE sequence of ?variable or <iri> DESCRIBE * CONSTRUCT { graph pattern } ASK
Query Dataset Sources (optional)	Add triples to the background graph (repeatable): FROM <iri> Add a named graph (repeatable): FROM NAMED <iri>
Graph Pattern (optional, required for ASK)	WHERE { graph pattern z}
Query Results Ordering (optional)	ORDER BY ...
Query Results Selection (optional)	LIMIT n, OFFSET m

TABLA 2: SINTAXIS BASICA DE UNA CONSULTA SPARQL

2.2.6.2 Sparql UPDATE

SPARQL / Update es un lenguaje para expresar cambios a un repositorio de datos RDF. Es un mecanismo estándar por el cual se pueden realizar cambios sobre tripletas RDF ya almacenadas. SPARQL / Update es un lenguaje que acompaña a SPARQL y está previsto para ser utilizado en conjunción con el lenguaje de consulta SPARQL. La reutilización de la sintaxis de SPARQL, tanto en estilo y detalle, reduce la curva de aprendizaje para los desarrolladores y reduce los costos de implementación.

SPARQL / Update ofrece las siguientes facilidades [22]:

- Insertar nuevas tripletas en un grafo RDF.
- Eliminar tripletas de un grafo RDF.
- Lleve a cabo un grupo de operaciones de actualización como una sola acción.
- Crear un nuevo grafo RDF.
- Eliminar un grafo RDF a partir de un grafo existente.

2.3 Estándares de Intercambio de Información de Salud

Debido al creciente de intercambio de información entre sistemas del dominio clínico, en la década de los 90, surge la necesidad de la incorporación de estándares que permitan colaborar entre las diferentes instituciones sanitarias al momento de compartir datos, que luego se transforman en información crucial en la toma de decisiones.

A continuación, se describen los principales conceptos relevantes, en el dominio de los estándares de intercambio de información utilizados en el proyecto. Inicialmente se describe el estándar **HL7v3**, sus principales características de forma general, para luego continuar con el dominio de **Documento Clínico CDA**.

La ***Arquitectura de Documento Clínico*** **–CDA**-, es una pieza fundamental del proyecto, ya que corresponde al conjunto de información (documento específico) que se incorpora a la ontología desarrollada.

Finalmente, se hará énfasis en la herramienta propuesta por HL7 para representación gráfica del modelo de CDA **–**el **RMIM**- , ya que constituyó un elemento de mucho apoyo al momento de profundizar y documentar el trabajo realizado.

2.3.1 HL7

Health Level Seven International (HL7) es una "*Organización de Desarrollo de Estándares*" (SDOs), para el ámbito de la salud. Opera a nivel internacional (con miembros en más de 55 países - [23]) y su misión es proveer estándares globales para los dominios: clínico, asistencial, administrativo y de procesos, con el fin de lograr una interoperabilidad real entre los distintos sistemas de información en el área de la salud.

Cuando se comienza a trabajar con el estándar, existe una confusión sobre el alcance de las definiciones que se encuentran dentro del estándar: HL7 no desarrolla software, simplemente provee especificaciones, utilizando un método formal para ello, para el intercambio de información clínica. [24]

2.3.1.1 Componentes Fundamentales de la Versión 3

En 2003, se aprobó una serie de estándares contenidos en la especificación de la versión 3 de HL7. Estos estándares emplean una metodología formal de modelado de casos de uso, basada en UML, y una sintaxis que emplea XML al momento del intercambio entre sistemas, con fin de aprovechar todas las ventajas de un lenguaje extensible, sencillo y versátil.

Las especificaciones definen casos de uso para llegar fácilmente a los requerimientos de implementación. Utilizar casos de uso, parte de la base de que los flujos de trabajo de las instituciones están unificados, y por tanto se pueden definir lo suficientemente genéricos como para poder aplicarlos.

La versión 3 de HL7 se apoya sobre un modelo de referencia arquitectónico propio conocido como ***Reference Information Model*** (RIM). [24]

1. ***Modelo de Referencia RIM***

El Modelo de Referencia (RIM), es uno de los componentes fundamentales de la Versión 3, es por esta razón que nos parece importante, presentar algunas características del mismo, ya que muchos de los conceptos luego desarrollados se basan en las clases fundamentales del RIM y la representación gráfica utilizada para representar el modelo.

El RIM, no es más que el modelo conceptual, que da soporte a todas las aplicaciones que se construyen respetando la especificación de HL7. Detalla las clases y tipos de datos que se necesitan para construir tanto estas aplicaciones, como los sistemas de mensajería que posibilitan el intercambio entre ellas.

Hay seis clases fundamentales en el RIM: ***Entity, Role, Participation, Act, ActRelationShip, RoleLink***. Cada una de ellas tiene asignada un color para identificarlas fácilmente, una serie de atributos y relaciones entre las mismas. Para cada una se especifica la cardinalidad, tal y como se representa en la siguiente figura:

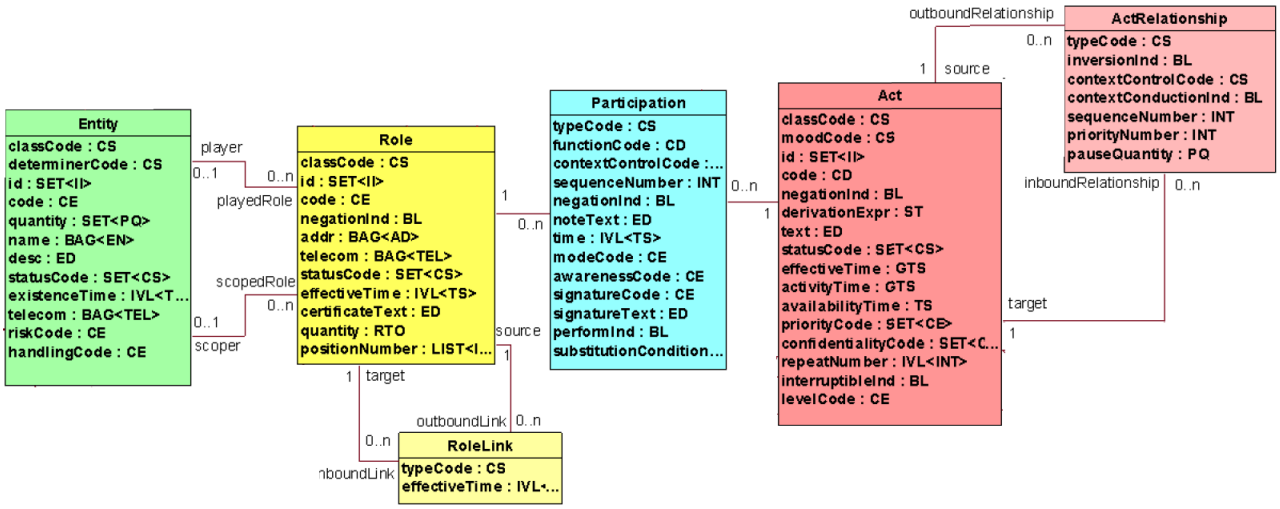


ILUSTRACIÓN 4: ESQUEMA DE LAS CLASES FUNDAMENTALES DEL RIM, SUS RELACIONES Y CARDINALIDAD.

Describiremos a continuación, los conceptos que existen detrás de cada uno de las clases [25]:

Entity: representa un agente o cualquier otro objeto de negocio que forme parte de una organización (organización, forma de vida – *personas* -, material, lugares, punto de actuación).

Role: responsabilidad o papel que puede jugar una entidad (paciente, empleado, médico de cabecera, médico de guardia, muestra de análisis,...).

Act: representa las acciones que han sucedido, está sucediendo o puede suceder en un futuro; gracias a la combinación de varias entidades (derivación, transporte, suministro, procedimiento, condición consentimiento, observación, medicación, acto clínico,...).

Participation: define cómo se involucra un Rol en una Actuación (autor, modificador, certificador, consultor, operador, habilitador, autorizador, beneficiario, autenticador, receptor, emisor,...). Un participante puede tener mas de un rol en un actuación.

ActRelationship:_asociación definida entre dos actuaciones. Permite establecer relaciones entre actos, por ejemplo, que uno sucedió antes que el otro; que una determinada *observación* (Act) es *consecuencia* (actRelationShip) de un determinado *procedimiento* (Act).
La siguiente imagen muestra una clase del modelo RIM, en este caso las clases Entity y Role, la cual describe también los atributos que la misma contiene. En la siguiente sección, se explicarán los tipos de datos que estos tienen como dominio.

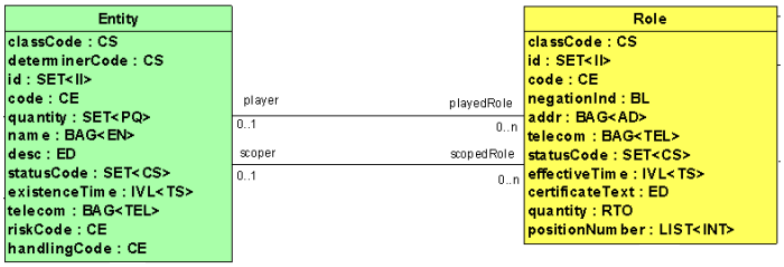


ILUSTRACIÓN 5: PARTE DEL RIM, CON ATRIBUTOS.

2. Tipos de Datos:

Los tipos de datos definen el formato estructural de la información contenida dentro de un atributo del RIM, e influencia el conjunto de valores permitidos que un atributo puede asumir.

Todo atributo contenido en cualquier clase del RIM, describe que tipo de datos tiene.

Los Data Types definidos se pueden clasificar de la siguiente forma:

a. *Tipos de Datos Básicos*

Corresponden a los tipos de datos elementales de cualquier sistema de información, entre ellos se encuentran: *String, Boolean, Integer.*

b. *Códigos e identificadores*

Corresponden a dos grandes grupos, los que identifican objetos, organizaciones o personas (II), y los que permiten agregar instancias de códigos (CS, CV, entre otros).

c. *Fechas y Tiempos*

Permiten asignar fechas (en formatos estándares) y horas o intervalos de tiempos a atributos.

d. *Nombres y Direcciones*

Corresponden a un conjunto de partes que forman el nombre o dirección y cada una describe a que corresponde y el valor.

Por ejemplo; el tipo *AD* se utiliza para representar direcciones, este contiene una lista de elementos *ADXP*, partes de la dirección. Cada *ADXP* contiene un elemento de la misma como puede ser: nombre de la calle, número, entre otras. La misma idea se utiliza para los nombres, de las personas u organizaciones, este tipo de datos se denomina EN.

e. *Colecciones, Conjuntos*

Se utilizan para representar diferentes conjuntos, por ejemplo: Set, List, IVL (intervalo).

3. Vocabularios Controlados

Otro elemento importante dentro de HL7v3, son los vocabularios controlados. Existen algunos atributos de tipo *Code*, que se define dentro de la norma, los valores posibles de códigos que pueden tomar. Por ejemplo en atributo de la clase *Act*, denominado *ClassCode*, tiene como dominio el Vocabulario HL7: "**ActClassCode**". Este determina qué tipo de Act corresponde, como valores posibles dentro del Vocabulario: [26]

Lvl	Type, Domain name and/or Mnemonic code	Mnemonic	Print Name	Definition/Description
1	**S: ActClassRoot** (ACT)	ACT	act	This concept represents a domain containing all possible values of the ActClass code. system
2	**S: ActClassContract** (CNTRCT)	CNTRCT	contract	An agreement of obligation between two or more parties that is subject to contractual law and enforcement.
3	**S: ActClassFinancialContract** (FCNTRCT)	FCNTRCT	financial contract	A contract whose value is measured in monetary terms.
4	L: (COV)	COV	coverage	A health care insurance policy or plan that is contractually binding between two or more parties.
2	**S: ActClassControlAct** (CACT)	CACT	control act	An act representing a system action such as the change of state of another act or the initiation of a query. All control acts represent trigger events in the HL7 context. ControlActs may occur in different moods.
3	L: (ACTN)	ACTN	action	Sender asks addressee to do something depending on the focal Act of the payload. An example is "fulfill this order". Addressee has responsibilities to either reject the message or to act on it in an appropriate way (specified by the
3	L: (INFO)	INFO	information	Sender sends payload to addressee as information. Addressee does not have responsibilities beyond serving addressee's own interest (i.e., read and memorize if you see fit). This is equivalent to an FYI on a memo.
3	L: (STC)	STC	state transition control	Sender transmits a status change pertaining to the focal act of the payload. This status of the focal act is the final state of the state transition. This can be either a request or a command, according to the mood of the control act.
2	**S: ActClassObservation** (OBS)	OBS	observation	Observations are actions performed in order to determine an answer or result value. Observation result values (Observation.value) include specific information about the observed object. The type and constraints of result

2.3.2 CDA – Clinical Document Arquitecture-

La documentación clínica se utiliza en el ámbito de la salud para describir la atención prestada a un paciente, comunicar información esencial entre los proveedores de salud y mantener un registro médico del paciente.

Un documento clínico es cualquier documento asociado a la historia clínica de un paciente. La definición utilizada por CDA para un documento clínico consiste en "documentación de observaciones y servicios clínicos".

CDA R2, es un estándar definido dentro de HL7v3 correspondiente a la Arquitectura de Documento Clínico (*Clinical Document Architecture)*, basado en XML para el marcaje de documentos que especifica la estructura y semántica de documentos clínicos con el propósito de facilitar su intercambio en un entorno de interoperabilidad. [27]

La versión de CDAR2, se publicó en Setiembre de 2005, y desde este momento no ha sufrido modificaciones lo que lo hace estable en el tiempo.

2.3.2.1 Características del CDA R2

Un documento clínico es ***el registro de las observaciones y servicios clínicos***, cuyas características más relevantes son las siguientes:

- ***Persistencia***: Un documento clínico debe continuar existiendo en estado inalterado, por un período de tiempo definido por requerimientos regulatorios locales.
- ***Administración*** Un documento clínico es mantenido por una organización que está involucrada con su custodia.
- ***Potencial de autentificación:*** Un documento clínico es un conjunto de información que debe ser autenticada legalmente. Firmado electrónicamente de ser necesario por las regulaciones nacionales.
- ***Plenitud:*** La autenticación de un documento clínico aplica para todo el documento y no aplica para porciones del documento sin el contexto completo del mismo.
- ***Legibilidad humana:*** Un documento clínico es legible por un humano. Mediante la aplicación de hojas de estilo, a los documento XML generados.
- ***Contexto:*** Un documento clínico establece el contexto por defecto para su contenido. Entendiéndose por contexto, *lugar donde se realizó el documento, confidencialidad,* entre otro parámetros.

2.3.2.2 Modelo de Referencia CDA: R-MIM CDA y estructura.

El modelo de referencia de R-MIM, utilizado para representar de forma gráfica la estructura del Documento CDA, nos fue de mucha utilidad para comprender mejor las clases involucradas, las relaciones entre ellas, así como los diferentes atributos y los dominios de Vocabularios específicos de cada uno.

Es por esta razón, y dado que se hará referencia en diferentes capítulos de la documentación, que lo explicaremos de forma resumida, detallando los diferentes elementos que contiene.

La siguiente imagen muestra de lo que refiere a la estructura del RMIM. Sigue la misma nomenclatura utilizada en la representación del RIM, en la referencia de colores y forma en que se describen los atributos y Data Types.

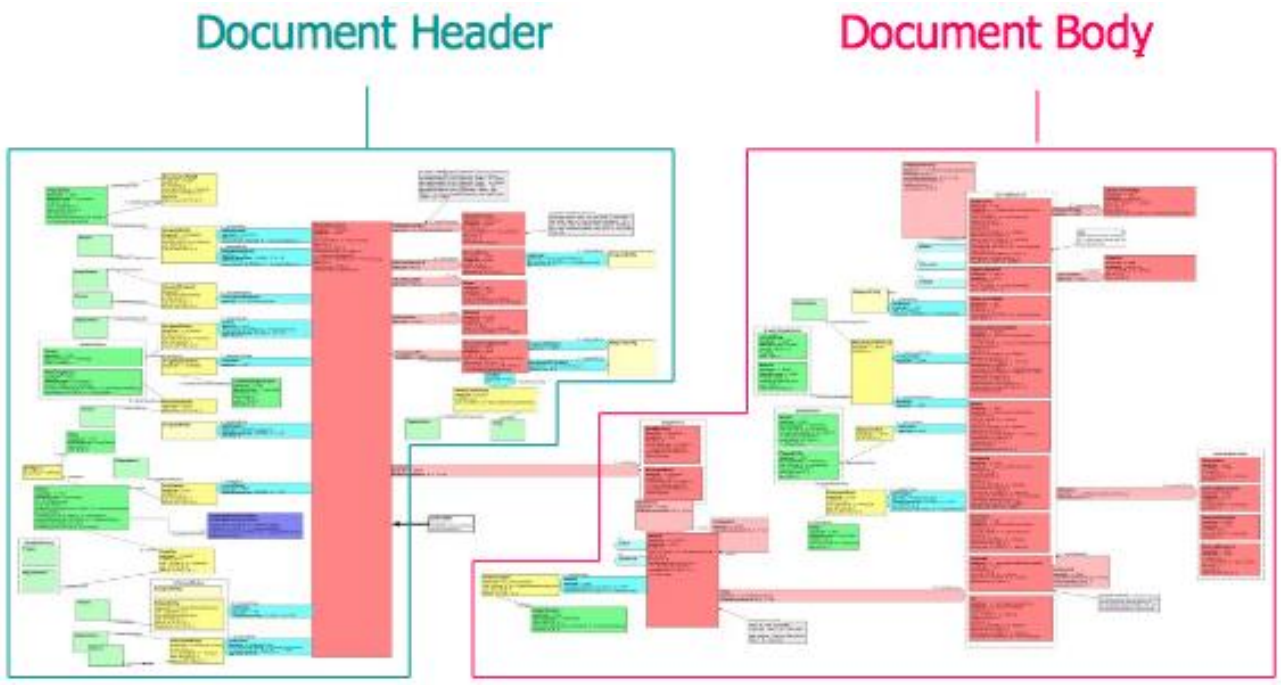


ILUSTRACIÓN 6: R-MIM DE CDA

La estructura se separa en el **Header** (o cabezal) y el **Body** (o cuerpo).

- El ***header***, contiene todo lo que refiere a la información contextual y contiene una serie de campos que son obligatorios, con el fin de contemplar las exigencias de completitud del documento. La idea de la completitud, refiere a que un documento, en si mismo debe contener la suficiente información, como para que por sí solo pueda ser entendido, independientemente de que el objetivo inmediato sea su transmisión o su almacenamiento.

Dentro de la estructura que representa el header se encuentra:

  - ***Información del documento****:* identificadores, código de descripción, fecha de creación, entre otros.
  - ***Información del paciente y los participantes***: dentro de los participantes, se encuentra el autor del documento, el responsable legal.
  - ***Actos relacionados***: descripciones de acto principal, ordenes, consentimientos.

- El ***body***, contiene la información clínica dentro del documento, todo lo referente al acto que se está documentando.
  - Existe la posibilidad del separar en diferentes secciones la información, pudiendo estar codificadas y estructurada de forma más específica según el tipo de acto que se esté documentando.

La siguiente imagen resume el contenido del documento [28]:

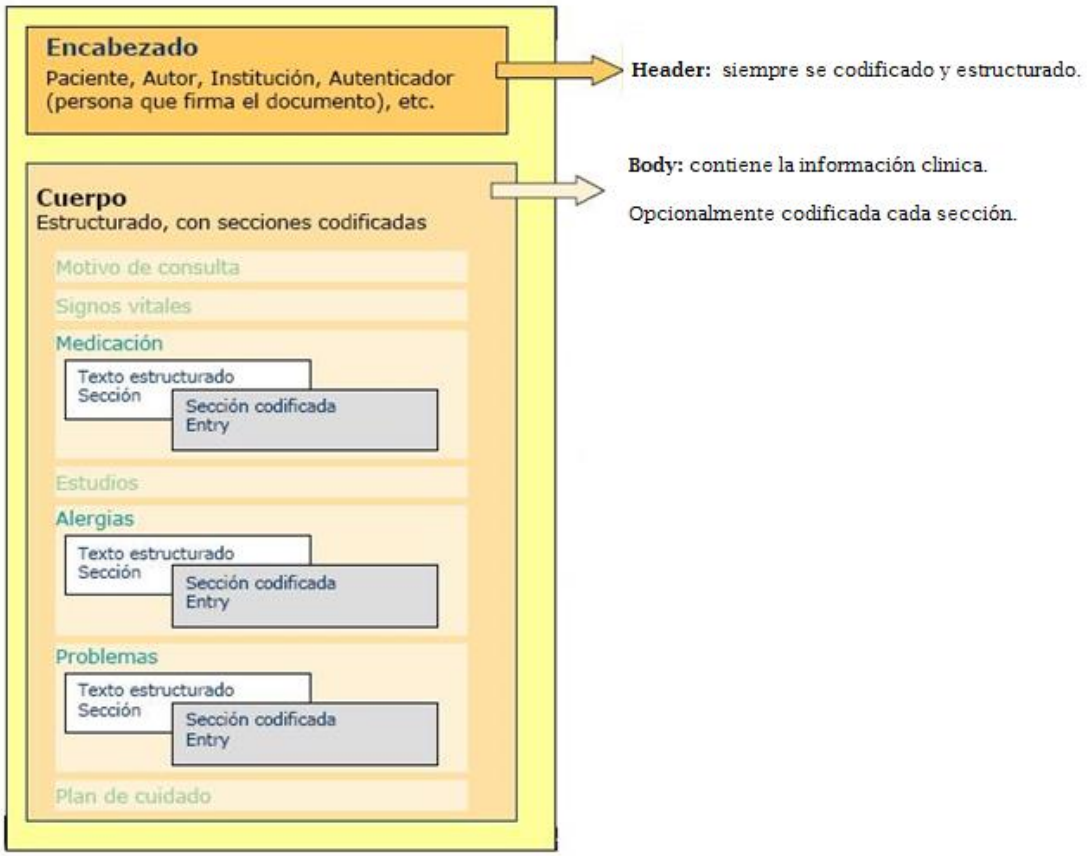


ILUSTRACIÓN 7: ESTRUCTURA GENERAL DE LA INFORMACIÓN DE UN CDA.

2.3.2.3 CDA Hoy.

La estructura de documento Clínico, se está utilizando actualmente en muchos proyectos entre los cuales destacan los de alcance nacional como: *Infoway* (Canadá [29]), Estados Unidos (NHIN), también en Finlandia, Grecia, España en proyectos de regiones. En Centros Médicos, es utilizado para intercambiar información con otras instituciones. [30]

En la siguiente tabla [31], se presentan los diferentes estándares que se utilizan en los diferentes proyectos gubernamentales, destacando que en ocho de doce países, se utiliza CDA.

TABLE 32.4 **Country-wise Usage of Standards**

Standards	Australia	Austria	Canada	Denmark	United Kingdom	Hong Kong	India[a]	Netherlands	Singapore	Sweden	Taiwan	USA
HL7 v2.x	Y	N	N	N	N	Y	Y	N	Y	N	N	Y
HL7 v3	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
CDA	Y	Y	Y	N	Y	Y	N	N	Y	N	Y	Y
ASTM CCR	N	N	N	N	Y	N	N	N	N	N	N	N
HL7/ ASTM CCD	N	N	N	N	N	N	Y	N	Y	N	N	Y
openEHR	Y	N	N	Y	N	N	N	N	Y	Y	N	N
IHE	N	Y	Y	N	N	N	N	N	Y	N	N	N
DICOM	N	Y	Y	N	N	N	Y	N	N	N	N	Y
EHRCom	N	N	N	Y	N	N	N	N	N	Y	N	N

[a]Recommended.

ILUSTRACIÓN 8: TABLA DE DESCRIPCION DE USO DE CDA.

Por otro lado, diferentes diversas herramientas on-line, para poder validar los contenidos de los documentos, así como descargar ejemplos de guías e implementaciones de otros Centros Médicos [32] – [33].

En el contexto nacional, con los recientes avances realizados por el ***Programa Salud Uy – Agesic - ,*** para el Proyecto de Historia Clínica Electrónica, se ha publicado un documento en el que se establece:

"En esta fase del proyecto el programa Salud.uy ha definido que utilizará para la estandarización de los documentos el estándar HL7 V3 CDA R2, resta la definición de los documentos que hacen parte de la HCEN. " [4].

CAPÍTULO 3

ESTADO DEL ARTE

Este capítulo se enfocará en los diferentes conceptos de las ontologías en el campo de la informática médica. Se expondrán también las que sirvieron como modelo inicial a nuestro trabajo práctico.

3.1 Ontologías Aplicadas

En la actualidad, el conocimiento formal de *estructuras ontológica*s se utiliza en diversas áreas y cumple un papel fundamental en la descripción de significado de los contenidos.

Los **Datos Enlazados (*Linked Data*)** es la forma que tiene la Web Semántica de vincular los distintos datos que están distribuidos en la Web o en distintas fuentes, de manera que se referencian de la misma forma que lo hacen los enlaces de las páginas web.

La Web Semántica no se trata únicamente de la publicación de datos en la Web, sino que éstos se pueden vincular a otros, de forma que las personas y las máquinas puedan explorar la web de los datos, pudiendo llegar a información relacionada que se hace referencia desde otros datos iniciales. [34]

La reutilización e incorporación de dicha información, es lo que se ve reflejado con la utilización de modelos ontológicos reutilizables por diversas fuentes.

3.1.1 Ontologías en el campo de la Informática Médica

En el contexto sanitario, las Ontologías constituyen uno de los mecanismos de formalización que más han crecido en los últimos años. Se ha demostrado la utilidad de la herramienta, debido a la posibilidad de reusar el conocimiento [35] . Esta herramienta se está manejando cada vez más para diversas tareas, como la de recuperación y extracción de información desde textos narrativos, para la clasificación de documentos de forma automática; entre otras.

En la actualidad existen diversas investigaciones que buscan recopilar y almacenar adecuadamente la información médica generada en consultas, tratamientos e investigaciones, así como generar conocimiento útil y aprovechable de los datos que se obtienen.

Existe una serie de proyectos de acceso libre, que como forma de contribuir a la expansión del conocimiento, distribuyen las ontologías que ya están desarrolladas en el campo de la informática médica. Dos portales que agrupan diversas ontologías, son "***The Open Biological and Biomedical Ontologies***" [36] de *OBO Foundry* y *"**Biomedical Ontology**"* [37], de *National Centers for Biomedical Computing*.

Destacamos estos portales, ya que en ellos se encuentran disponibles ontologías de áreas médicas concretas para el modelado del conocimiento. Cada uno tiene la descripción y caracterización del contenido de las ontologías, el fin con el que fueron creadas, las organizaciones que se encargan de desarrollarlas y mantenerlas.

3.1.2 Ontologías estudiadas relacionadas con el proyecto

En esta sección, se detallarán algunos de los artículos y trabajos relevados, principalmente los que aplican modelos ontológicos sobre HL7.

3.1.2.1 Modelado de ontología HL7 - por Helen Chen-

El proyecto realizado por *Helen Chen* en el año 2006, consistió en definir las bases de la representación en semántica OWL, aplicado al modelo del RIM de HL7. Este trabajo se encuentra publicado como recomendación en la W3C [38]. Principalmente se basó en un proyecto anterior realizado por ***Samson Tu*** de Stanford University [39] que desarrolló una ontología presentada en *Protégé* 2000 basada en *Frames.* Partiendo de la base de Samson Tu, Chen tomó los conceptos para realizar una nueva ontología basada en *Reglas*.

La metodología utilizada para la realización del modelo ontológico, fue la siguiente:

1- Se tomaron las clases principales del RIM y se modelaron como Clases ontológicas. Los atributos de las clases del RIM, se modelan también como propiedades.

2- Los Vocabularios controlados utilizados dentro de HL7, se representaron cada uno como clases ontológicas, y dentro de esta se definen instancias que representan cada uno de los valores del vocabulario.

El trabajo realizado, se encuentra publicado y es de libre acceso, el archivo principal ***RIMOWL.owl*** se descargó y se desplegó con la herramienta *Protégé*.

Luego de estudiado el trabajo, decidimos tomarlo como estructura inicial, extendiéndolo dentro del proyecto, reutilizando y completando solo los conceptos relevantes en el dominio de CDA. En el Capítulo 4 "*Ontología para el R-MIM e Instancias de CDA*" detallaremos los cambios realizados en el modelo.

Cabe aclarar que no encontramos trabajos publicados que extendieran esta ontología o que la utilizaran con algún fin particular, solamente se hace referencia a las pruebas de consistencia realizadas sobre la ontología utilizando los razonadores de *Pellet* y *RacerPro*.

3.1.2.2 Validación semántica de CDA utilizando SNOMED.

Dentro de las ontologías relevadas, nos parece interesante realizar un comentario más profundo sobre **SNOMED**, por la importancia actual que tiene y por los trabajos relacionados que investigamos que unifican los conceptos de CDA y SNOMED.

SNOMED

SNOMED CT es la más grande ontología clínica multilenguaje. Es utilizada en los sistemas de Historia Clínica Electrónica para la documentación clínica y para los análisis y reportes.

Su verdadero valor puede ser explotado cuando las aplicaciones clínicas son construidas en torno a él. SNOMED CT ha sido diseñada de manera que pueda ser utilizada directamente por el usuario y posteriormente en los procesos de gestión, vigilancia e investigación.

En lo que respecta a Uruguay, en 2012 [40] se realizó la suscripción del país a la IHSDO, organización que se encarga del mantenimiento de SNOMED. También, dentro del programa ***Salud.UY***, se está estudiando su uso dentro del proyecto de ***Historia Clínica Electrónica Nacional***.

Debido a la creciente difusión de esta terminología y los trabajos concretos que la utilizan, nos parece importante comentar los objetivos buscados y resultados obtenidos de dos trabajos relevantes relacionados con los conceptos de ontologías, utilizando SNOMED y la estructura de CDA.

Semantic Validation of the use of SNOMED CT in HL7 Clinical Document.

El primer trabajo estudiado se enfoca en validar mediante estructuras ontológicas, la información contenida en el *Body del CDA* [41]. Para esto, se utiliza una transformación de la información contenida en Body del CDA y luego se une con la ontología que contiene la información de SNOMED. Posteriormente, aplicando el razonador *Pellet*, se verifica la consistencia de la misma.

El siguiente diagrama (obtenido del Artículo) muestra cómo es la interacción de los diferentes componentes en el prototipo realizado:

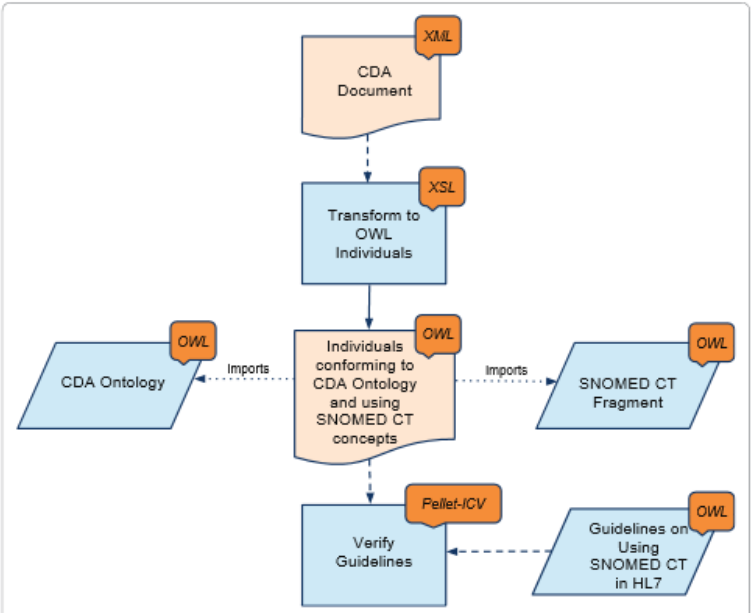


Figure 1 Used Method Verifying Guidelines SNOMED CT®. The figure describes a top level view of the used method to validate CDA documents using semantic technologies.

ILUSTRACIÓN 9: PROCESO DE VALIDACION DE ESTRUCTURA DE DOC.CDA CON SNOMED

El estudio tuvo como objetivo la validación de la estructura de las diferentes secciones del *Body de CDA*. Dentro de las mismas, se puede clasificar según el tipo de acto que se representa: *Observation, Supply, Substance Administration, Procedure*, entre otras. Para cada una de estas, de acuerdo al acto que corresponde, y los códigos que contenía cada sección, se creaban las instancias correspondientes dentro de la ontología, y luego con la información obtenida de SNOMED, se aplica el razonador para obtener los resultados de consistencias.

En base a estudios que se realizaron con varios tipos de CDA, se concluyó que algunos de ellos no conformaban la integridad con los conceptos representados en SNOMED, y que la metodología de validar los contenidos con ontologías sería aplicable para poder obtener información consistente en los documentos.

iSMART: Ontology-based Semantic Query of CDA Documents

El segundo trabajo [42] es el realizado por ***IBM China Reaseach Laboratory*** en 2011, que utiliza la información contenida en documentos CDA y los conceptos de SNOMED CT. En este caso, el estudio se basó en la realización de consultas sobre la información contenida en los documentos, junto con los nuevos conocimientos obtenidos de la información inferida aplicando razonadores.

Para mostrar los conceptos con ejemplos reales, se realizaron pruebas sobre más de cien millones de documentos CDA pertenecientes a un Hospital de China. Sobre ellos, se definieron “los pacientes candidatos” que son el resultado de las consultas con diferentes condiciones que se aplican a los documentos. Las pruebas

realizadas enfrentaron los resultados de consultas sin utilizar la inferencia y utilizándola luego. En la mayoría de los casos, el número de los pacientes candidatos obtenidos fue mayor al utilizar la inferencia con los aportes de SNOMED.

Dentro de los resultados obtenidos, se destaca la aplicabilidad del prototipo para obtener nuevos conocimientos en base a la información contenida. Por otra parte, las dificultades encontradas fueron en cuanto a las adecuaciones de los conceptos de los modelos documentos CDA y SNOMED CT, para interoperar entre sí.

3.1.3 Proyecto de Ontología HL7 a partir de los MIF

Otro trabajo relacionado, que fue investigado luego de avanzado el proyecto, es el **HL7OWL**, que se encuentra publicado en HL7 Gforge [43] un entorno de aplicaciones desarrolladas de forma libre, patrocinado por HL7, con el fin de promover y extender el uso de la norma por parte de las aplicaciones.

Se estableció un contacto vía e-mail con los desarrolladores activos del proyecto, consultando si existían aplicativos que utilizaran la ontologías obtenidas en este proyecto. La respuesta fue negativa: se han realizado pruebas de concepto pero no aplicativos concretos que lo utilicen.

CAPÍTULO 4

ONTOLOGÍAS PARA R-MIM E INSTANCIAS DE CDA

Introducción

En este capítulo se detalla uno de los ejes centrales del trabajo realizado. Se presenta lo que corresponde al modelado ontológico y sus diferentes componentes.

Nuestro trabajo consistió en desarrollar la ontología que contiene las definiciones de la información de documentos Clínicos CDA, luego incorporarle a ésta, las instancias de documentos concretos y finalmente; sobre el modelo ontológico resultante, poder realizar consultas o incorporación de nuevo conocimiento con el fin de obtener información que pueda resultar útil.

El siguiente diagrama presenta los diferentes componentes y como se relacionan entre ellos.

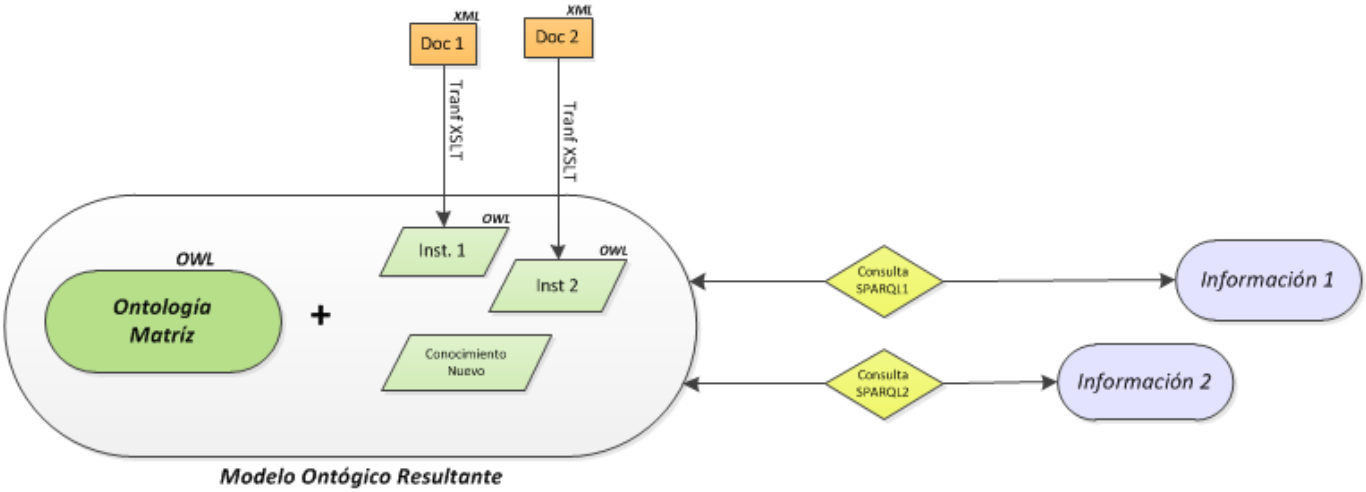


ILUSTRACIÓN 10: REPRESENTACION DE COMPONENTES INCLUIDOS EN LOS PROTOTIPOS.

Se describen los elementos presentados en el dibujo:

1. **Ontología Matriz**

   Corresponde a la estructura ontológica en la que se definen todas las *clases, subclases y relaciones* necesarias para lograr instanciar adecuadamente un documento CDA. En la sección 4.2 de este capítulo, se detalla el proceso de creación y las decisiones tomadas para su modelado.

2. **Transformaciones XML a Tripletas OWL**

   Las transformaciones de documentos CDA en formato XML a instancias OWL (*Inst1, Inst2,..*), se realiza mediante la aplicación de una transformación XSLT. El resultante consiste en un conjunto de tripletas en formato *turtle*, el cual junto con la ontología matriz, y los nuevos conocimientos, forman ***el Modelo Ontológico Resultante***. Se debió tener especial cuidado en dos aspectos: en la creación de las tripletas, según el elemento que corresponda dentro del CDA (en la Secc. *4.3.1* se muestra un Ejemplo de Instanciación) y la creación de identificadores únicos utilizando una nomenclatura definida.

3. **Conocimiento Nuevo**

Este elemento tiene como objetivo, mostrar que al modelo ontológico resultante, se le puede incorporar "nuevo conocimiento" agregado mediante tripletas *owl* representando nuevas clases o relaciones. También, se resume dentro de este componente, toda la información inferida sobre las tripletas instanciadas de los documentos CDA. Ver *Cap 2.2.5 "Inferencia y Razonamiento".*

4. **Modelo ontológico resultante**

Este modelo resultante, contiene lo detallado en el punto 1, 2 y 3: *las definiciones incluidas dentro de la ontología matriz, las instancias concretas de documentos CDA, y los nuevos conocimientos:* sobre este modelo se realizará el razonamiento para incorporar nuevas tripletas que se deriven de la información contenida. También, será sobre éste que se realizarán las consultas SPARQL. En el *Cap. 5*, se muestran los prototipos realizados en los que se obtiene este modelo.

5. **Realización de consultas SPARQL**

Las consultas SPARQL que se realizan sobre el modelo ontológico resultante, consisten en el mecanismo estudiado en el proyecto, para lograr obtener información útil sobre el modelo. En el *Cap. 5, secc: 5.3*, se muestran diferentes ejemplos de la aplicación de las mismas.

4.1 Terminología utilizada

Para evitar confusiones entre la terminología del lenguaje OWL, de la norma CDA, y el lenguaje XML, consideramos necesario comenzar realizando una aclaración correspondiente a ciertas expresiones, que se comparten en dichas terminologías, pero con significados distintos.

4.1.1 Clases y Propiedades

Este trabajo se basa en el concepto de que, dada la información de que dos entidades están interconectadas por una cierta propiedad, es posible obtener conclusiones acerca de las entidades en sí mismas. Dicho de otra forma, se puede inferir la pertenencia a ciertas clases de los individuos instanciados en la ontología, aplicando las reglas definidas mediante el razonamiento.

Los componentes fundamentales de OWL son las **clases** y las **propiedades** (o **roles**). Dentro del lenguaje OWL, denominamos **clase** a una cierta agregación de elementos. Asimismo, el término **propiedad** o **rol** describe una relación entre recursos e individuos, referenciados por sujeto y objeto de una tripleta OWL. Más detalle, ver *Cap. 2, Secc: 2.4.1.*

Por otra parte, la especificación formal del estándar HL7, representado en el Modelo de Información RIM, está basada en las llamadas "Clases Fundacionales". Más detalle, ver *Cap. 2, Secc: 2.4.1.*

Cuando se hable de clases, se estará haciendo referencia a las clases definidas dentro de la ontología, en el sentido del lenguaje ontológico OWL. En la ocasión que se requiera el uso del término "clase" en el sentido de la norma HL7, se utilizará la expresión completa "Clase Fundacional". Más detalle, ver *Cap. 2, Secc: 2.3.1*

En cuanto a la denominación de las relaciones entre recursos e individuos, estaremos utilizando el término "Propiedad". Por su parte, el término "Rol" se reservará para la clase fundacional "Role" de HL7.

4.1.2 Atributos.

Dado que también estaremos utilizando el lenguaje XML, como lenguaje de escritura de los documentos CDA, debemos aclarar, de qué estamos hablando cuando nos referimos a "atributos". En el contexto de un modelo de clases (como es el RIM), el término "atributo" es utilizado para representar propiedades de estas clases.

Por ejemplo, una clase que representa a un paciente, puede tener los datos de: nombre del paciente, sexo y fecha de nacimiento. Estos son considerados atributos del paciente en el modelo de clases. Los atributos de las clases HL7 pueden aparecer en XML como un elemento, o pueden aparecer también como un atributo XML.

En el contexto de este documento, siempre que hablemos de atributo, se deberá entender que nos estamos refiriendo a un atributo de clase. En el caso que debamos referirnos a un atributo XML, será debidamente aclarado.

4.1.3 Vocabularios

Un vocabulario es, tradicionalmente, un conjunto de símbolos, términos o palabras [44]. El estándar HL7, por su parte, define una serie de "Vocabularios", donde se agrupan y codifican valores posibles para determinados dominios, correspondientes a los distintos atributos a modelar.

Por ejemplo, HL7 define un **Vocabulario** para el contexto de las entidades Personas, correspondiente al atributo "Sexo", que tiene como valores posibles: *Masculino, Femenino.*

La diferencia con la noción usual de vocabulario, reside en que los vocabularios definidos por HL7 son jerárquicos, lo cual lleva a que se modelen en el desarrollo de nuestra ontología como clases y sus respectivas sub-clases. Adoptamos esta forma de modelar, de la ontología base de la cual partimos. Cuando hablamos de vocabulario, nos referiremos expresamente a los vocabularios de dominio definidos en HL7.

4.2 Modelado de la ontología matriz

En esta sección, explicaremos el modelado de la ***Ontología Matriz*** y en cada uno de los apartados, detallaremos los elementos que la componen.

Como punto de partida a la creación del modelo, se tomó la ontología desarrollada por Helen Chen, presentada en la sección: 3.3.1, la que llamaremos **Ontología Base**. La misma sirvió como modelo inicial, ya que definía unas pocas clases ontológicas y algunas propiedades del RIM.

Se tomó la decisión de seguir con la misma estructura jerárquica propuesta, extenderla y eliminar lo no aplicaba para el documento CDA. Así como realizar todas las nuevas clases, propiedades y restricciones que fueran necesarios para el modelado del CDA.

Las tres clases de las cuales se extiende toda la jerarquía son las siguientes:

- **HL7RIM**: Se modelan dentro de esta jerarquía, todas las clases que existen dentro del RIM que aplican al CDA. Se agrupan dentro de las cuatro clases fundamentales***: Act, Participation, Role, Entity.*** Para cada una, se determinan cuáles son los atributos que tienen, así como las restricciones sobre los mismos.
  Todos los atributos se modelan dentro de la ontología como *objectProperty* o *dataProperty*, según corresponda. Se detalla en cada caso según corresponda. En la Secc : 4.2.1 Modelado Ontológico de las Clases del R-MIM (HL7 RIM)

- **HL7Any**: Esta clase incluye a los tipos de datos que son definidos dentro del estándar HL7, que se utilizan para modelar CDAs y a los cuales refieren las propiedades de cada entidad en el RIM. Se decidió explicar en la Secc: *4.2.3 Modelado Ontológico de los Tipos de Datos (HL7 Any)*, solo algunos de los tipos modelados y el resto detallarlos en el Anexo 2 : "*Descripción de Tipos de Datos*. "

- **HL7Voc**: Se modelan todos los Vocabularios definidos en el estándar HL7, que aplican dentro de la especificación de CDA. Se describen en la Secc: *4.2.2 Modelado Ontológico de los Vocabularios (HL7 Voc)*, el modelado de los mismos.

  La idea central es que se utilizan para determinar los valores posibles que puede tomar un determinado atributo. Para cada valor del Vocabulario, se crea una instancia concreta que es luego la que se relaciona con el atributo cuando se quiere decir que un atributo tiene un determinado valor dentro del vocabulario.

4.2.1 Modelado Ontológico de las Clases del R-MIM (HL7 RIM)

Dentro de esta clase básica, se modelan las cuatro clases fundacionales de HL7 definidas en el modelo de Referencia RIM: *Act, Participation, Role* y *Entity*, y las clases que se derivan de estas.

4.2.1.1 Comentarios Generales

1. Se define una jerarquia de especializacion debajo de las cuatro clases fundamentales. Para determinar a que clase pertenece un determinado objeto, existen dos mecanismos:
   a. De acuerdo al valor de determinada propiedad (con rango en un vocabulario), o
   b. De acuerdo con el tipo de relaciones que tiene con otras clases.

   Se explicará en cada caso, cual es la propiedad que determina la pertenencia a la jerarquía.

2. Para representar los atributos de las clases básicas y sus jerarquías, se definen una serie de propiedades: ***DataProperty*** u ***ObjectProperty***, de acuerdo si toman valores dentro de los tipos básicos (*String, Boolean, entre otros*), o tipos de Datos definidos (HL7Any), respectivamente. Existe otro grupo de propiedades de gran relevancia, definidas como ***ObjectProperty***, que son las que corresponden a valores de Vocabularios definidos, para esto, se estudió cuáles de los atributos de las clases, toman valores dentro de algún Vocabulario en particular.

   Para cada clase que se presenta a continuación, listaremos cada uno de los tres grupos de propiedades, la nomenclatura utilizada es la siguiente: "*Nombre de la prop*": ***Dominio → Rango.*** En todos los casos, los nombres de las propiedades tienen como prefijo, según la clase a la cual apliquen: **act_; part_; role_; entity_.**

3. Existen diferentes restricciones que aplican tanto a las propiedades de las clases, como a las cardinalidades de las mismas. Para realizarlas, en los casos que refieren a cardinalidades, se optó por definir mediante ***restricciones sobre propiedades***, según se muestra en el siguiente fragmento de código:

```
:Act rdf:type owl:Class    rdfs:subClassOf :HL7RIM ,
  [ rdf:type owl:Restriction ;
    owl:onProperty :act_code ;
    owl:maxCardinality "1"^^xsd:nonNegativeInteger
  ] ,
```

   En lo respecta a propiedades que aplican a determinadas clases, se optó por declarar siempre de forma explícita, el rango y dominio de las mismas. Como se muestra en el siguiente fragmento de código turtle:

```
:act_moodCode rdf:type owl:FunctionalProperty ,  owl:ObjectProperty ;
        rdfs:domain :Act ;
        rdfs:range :VocActMood .
```

A continuación, se detallan las características particulares de los cuatro tipos básicos, así como sus subclases y las propiedades que tienen.

4.2.1.2 Act

Definición:

Representa algo que ha sucedido, está sucediendo o puede suceder en un futuro gracias a la combinación de varias entidades. Por ejemplo: *derivación, suministro, procedimiento, consentimiento, observación, medicación, acto clínico, entre otros.*

Características Particulares / Método:

En la ontología Base, se representan las subclases que corresponden a ACT. Al momento de instanciar un documento CDA, es determinante el valor de la propiedad *act_classCode* (con Dominio: *Act*, y Rango: *VocActClass*), ya que establece a que subclase de *Act* pertenece.

En la siguiente imagen se muestra la relación entre una instancia de "**Encuentro**", que luego toma el valor de **actEncounter** *(valor del vocabulario que determina que la clase corresponde a un Encuentro)* en la propiedad ***act_classCode*** *. En la* Sección: *4.2.2 Modelado Ontológico de los Vocabularios (HL7 Voc), se detalla el mecanismo de creación de valores de Voc.*

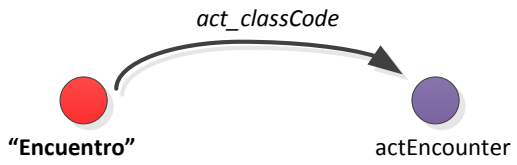


ILUSTRACIÓN 11: EJEMPLO DE INSTANCIACION DE PROPIEDAD ACT_CLASSCODE

Propiedades con Dominio Act.

Las propiedades que contienen la clase Act y las Subclases, se obtuvieron estudiando el modelo de Referencia para CDA, teniendo en cuenta todos los atributos que existen, para todos los tipos posibles de Act. Identificando el tipo de datos se obtiene el rango que la propiedad a incluir tendría.

Por ejemplo, el atributo "***id***" que es de tipo ***II –****que representa el identificador único del acto -* **,** se definió como la ***ObjetcProperty:*** act_id: Act → II. En cambio el title *-que aplica al documento, y representa el título del mismo-* se definió como ***DataProperty:*** act_title: ActClinicalDocument → string.

A continuación se presentan las propiedades definidas dentro de Act, divididas en tres grupos según el rango que las mismas puedan tomar.

1. ***Propiedades con rango en Vocabularios:***
   - act_classCode: Act → **VocActClass**
   - act_moodCode: Act → **VocActMood**
   - act_code: Act → **VocActCode**
   - act_statusCode: Act → **VocActStatus**
   - act_priorityCode: Act → **VocActPriority**
   - act_confidentialityCode: Act → **VocConfidentiality**

2. ***Propiedades con rango en tipos de datos HL7 (ObjectPropertys):***

- act_id: Act → II
- act_text: Act → ED
- act_effectiveTime: Act → IVL_TS
- act_languageCode : Act → CS
- act_setId: ActDocument → II

3. ***Propiedades con rango en tipos de datos básicos (DataPropertys):***

- act_negationInd: Act → boolean
- act_derivationExpr: Act → string
- act_repeatNumber: Act → int
- act_independentInd: Act → boolean
- act_versionNumber: ActDocument → int

Las propiedades que aplican a alguna de las subclases de Act, se muestran en el Anexo *"Propiedades definidas para las subclases del RIM"*.

4.2.1.3 Participation

Definición:

Representa cómo se involucra un Rol en una Actuación (*autor, modificador, certificador, consultor, operador, habilitador, autorizador, beneficiario, autenticador, receptor, emisor,...)*

Características Particulares / Método:

En el caso de las Participaciones, la ontología de base genera la subjerarquía de clases siguiendo el tipo de Participación, es decir, la clasificación de los distintos tipos de participaciones posibles según el vocabulario HL7 denominado ***ParticipationType***.

En todos los casos de participaciones presentes en un CDA, se infiere su tipo de dos formas posibles: o bien a través del valor de la propiedad ***part_typeCode***, cuyo rango está en el Vocabulario ***ParticipationType***, o bien a través de la relación que tenga la propiedad con un rol específico.

Propiedades con Dominio Participation.

Para este caso, las propiedades solo son de dos tipos, las que tienen como Rango un vocabulario definido (dentro de la clase HL7Voc) y las que son ***DataProperty*.**

1. ***Propiedades con rango en Vocabularios***

- part_typeCode: Participation → **VocParticipationType**
- part_contextControlCode: Participation → **VocContextControl**
- part_signatureCode: Participation → **VocParticipationSignature**
- part_functionCode: Participation → **VocParticipationFunction**
- part_awarenessCode: Participation → **VocTargetAwareness**
- part_modeCode: Participation → **VocModeCode**

2. ***Propiedades con rango en tipos de datos básicos (DataProperty)***

- part_time: Participation → datetime

4.2.1.4 Role

Definición:

Especifica la responsabilidad o papel que puede jugar una entidad (paciente, empleado, médico de cabecera, médico de guardia, muestra de análisis,...).

Características Particulares / Método:

La representación de los Roles en la ontología de base, sigue el mismo criterio explicado para Participation y para Act: se modelan las distintas subclases bajo la superclase Role, según la definición jerárquica dada en el vocabulario HL7 denominado RoleClass.

Esta estructura jerárquica, divide los roles según tres conceptos o dominios abstractos:

1) **RoleClassOntological**: es el dominio abstracto que recoge los roles en los cuales, la entidad que juega el rol, es definida o especificada por la entidad de alcance.
2) **RoleClassPartitive**: reúne los roles en los cuales la entidad que juega el rol es de alguna forma, parte de la entidad de alcance.
3) **RoleClassAssociative**: recoge todas las demás formas de asociación entre las entidades que se relacionan mediante un determinado rol

.

Propiedades con Dominio Role.

Las propiedades que tienen como dominio la clase Role, se agrupan en las que tienen rango en Vocabularios y las que tienen rango en tipos de datos.

1. ***Propiedades con rango en Vocabularios:***

   - role_classCode: Role → **VocRoleClass**
   - role_code: Role → **VocRoleCode**
   - role_statusCode: Role → **VocRoleState**

2. ***Propiedades con rango en tipos de datos HL7 (ObjectProperty):***

   - role_id: Role → II
   - role_effectiveTime: Role → IVL_TS
   - role_addr: Role → AD
   - role_telecom: Role → TEL

4.2.1.5 Entity

Definición:

Representa un agente, cualquier otro objeto de negocio que forme parte de una organización (*organización, forma de vida, material, lugares, punto de actuación).*

Características Particulares / Método:

En la ontología de base, la jerarquía que corresponde a la clase ***Entity*** se ajusta a la representación del modelo RIM. De ellas, se seleccionan únicamente las subclases que tienen presencia dentro del R-MIM, o sea validas para CDA, ellas son:

- Organization
- Person
- Device
- Place
- ManufacturedMaterial
- Entity

A su vez, la entidad Person sufre una diversificación en el R-MIM, siendo que Person contiene un subconjunto reducido de las propiedades originales para la entidad Person según el RIM. Asimismo, se define otra versión de la entidad Person, bajo el nombre de SubjectPerson, con propiedades adicionales a las que tiene Person. Y finalmente, se define la entidad Patient, la cual contiene todo el set de propiedades para la entidad Person que define el RIM.

Por otra parte, el R-MIM presenta agregaciones de entidades:

- *AuthorChoice*: Person | AuthoringDevice
- *GuardianChoice*: Person | Organization
- *DrugOrOtherMaterial*: LabeledDrug | Material
- *EntityChoice*: Device | PlayingEntity

Por tanto, sobre la subjerarquía de clases bajo la clase Entity, realizamos los siguientes cambios:

- Definimos las clases ***AuthorChoice, GuardianChoice y EntityChoice***, como unión disjunta de las clases que las componen.

En el siguiente fragmento de código turtle se muestra la restricción realizada para representar esto:

```
:EntityAuthorChoice rdf:type owl:Class ; rdfs:subClassOf :Entity ;
        owl:disjointUnionOf (
                :EntityAuthoringDevice
                :EntityPerson
        ) .
```

- Definimos la clase ***EntityDrugOrOtherMaterial***, como superclase de ***EntityLabeledDrug*** y de ***EntityMaterial***, ambas entidades de tipo Material Manufacturado, pero con una pequeña variante en las propiedades disponibles para cada una de ellas.

- Definimos la subclase ***EntityPatient*** y ***EntitySubjectPerson*** bajo la clase ***EntityPerson***.

- Definimos las subclases ***EntityAuthoringDevice*** bajo ***EntityDevice***, y ***EntityCustodianOrganization*** bajo EntityOrganization, de modo que queden representadas todas las variantes de Entidad que presenta el R-MIM.

Propiedades con Dominio Entity

Las propiedades que tienen como dominio la clase **Entity**, se agrupan en que tienen rango en Vocabularios, en tipos de datos HL7 (Object Property), y en tipos de datos básicos (DataProperty).

1- ***Propiedades con rango en Vocabularios***

- entity_classCode: Entity → **VocEntityClass**
- entity_determinerCode: Entity → **VocEntityDeterminer**
- entity_code: Entity → **VocEntityCode**
- entity_administrativeGenderCode: EntityLivingSubject → **VocAdministrativeGender**
- entity_maritalStatusCode: EntityPerson → **VocMaritalStatus**
- entity_religiousAffiliationCode: EntityPerson → **VocReligiousAffiliation**
- entity_raceCode: EntityPerson → **VocRace**
- entity_ethnicGroupCode: EntityPerson → **VocEthnicity**

2- Propiedades con rango en tipos de datos HL7 (ObjectProperty)

- entity_id: Entity → II
- entity_name: Entity → EN
- entity_telecom: Entity → TEL
- entity_quantity: Entity → PQ
- entity_desc: Entity → ED
- entity_addr: EntityOrganization | EntityPlace | EntityPerson → AD
- entity_standardIndustryClassCode: EntityOrganization → CE

3- Propiedades con rango en tipos de datos básicos (DataProperty)

- entity_manufacturerModelName: EntityDevice → string
- entity_softwareName: EntityDevice → string
- entity_birthTime: EntityLivingSubject → date
- entity_lotNumberText: EntityManufacturedMaterial → string

4.2.1.6 Propiedades entre las clases fundamentales.

Luego de explicar los elementos que componen las cuatro clases fundamentales: ***Act***, ***Participation***, ***Role*** y ***Entity***, es hora de detallar las propiedades que relacionan las diferentes clases.

Es oportuno, recordar el modelo de referencia RIM, para entender que no todas las clases se relacionan entre sí. La siguiente imagen muestra el sentido de las relaciones entre las cuatro clases fundamentales.

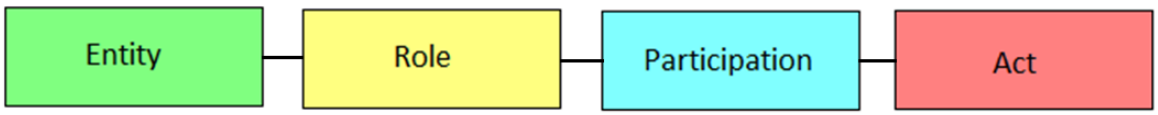


ILUSTRACIÓN 12: ASOCIACIONES DE CLASES FUNDAMENTALES DEL RIM

A continuación, describimos cada una de estas relaciones. Una aclaración oportuna: estas relaciones son ***ObjectProperty*** cuyo dominio y/o rango puede estar o bien, en la clase superior de la jerarquía, o bien en una subclase perteneciente a la especialización de la jerarquía. Por ejemplo: la relación ***guardian*** tiene dominio en EntityPerson y rango en RoleGuardian.

Para estos casos, se detallará cuáles son las clases involucradas.

1- ***Relaciones entre Acto y Participación***

Un Acto cualquiera, se relaciona con una Participación, mediante la propiedad **hasParticipation**, cuyo dominio está en la clase **Act** y cuyo rango está en la clase **Participation**.

- **hasParticipation:** Act → Participation

2- ***Relaciones entre Participation y Role***

Una Participación cualquiera, se relaciona con un Rol, mediante la propiedad **hasRole**, cuyo dominio está en la clase Participation y cuyo rango está en la clase Role.

- **hasRole:** Participation → Role

3- ***Relaciones entre Role y Entity***

Las relaciones entre los Roles y las Entidades, están definidas según el nombre que tienen en el R-MIM. Por ejemplo, como muestra la siguiente imagen, en lo que refiere a la relación entre el Role: *PatientRole* y la Entidad *Patient,* se modela como la relación **patient:** RolePatient → EntityPatient.

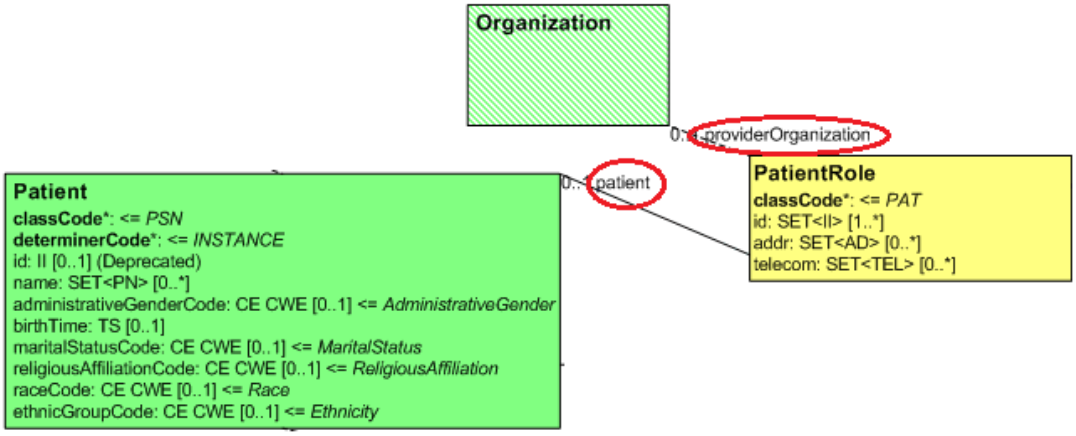


ILUSTRACIÓN 13: DESCRIPCIÓN DE ASOCIACION ENTRE ENTITY - ROLE

4- *Relaciones entre Entity y Role*

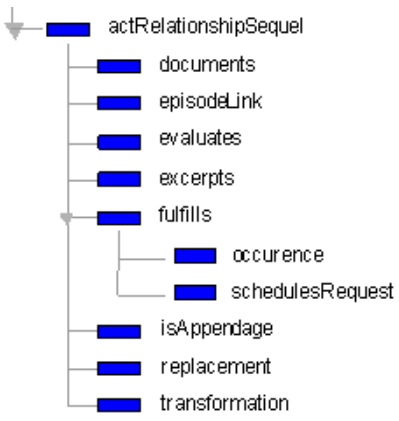
Las relaciones entre las Entidades y los Roles, están definidas según el nombre que tienen en el R-MIM ídem a lo explicado en el punto anterior. A continuación listamos las propiedades, dando el dominio y rango específico de cada una de ellas:

- **asOrganizationPartOf:** EntityOrganization → RoleOrganizationPartOf
- **birthplace:** EntityPlace → RoleBirthPlace
- **guardian:** EntityPatient → RoleGuardian

5- *Relaciones entre Act- Act*

Todas las relaciones necesarias para modelar vínculos entre Actos, son *subpropiedades* **ObjectProperty** de actRelationship, y tanto el dominio como el rango de todas estas propiedades, está en la clase Act. El único atributo relevante de estas relaciones, consiste en el denominado ***typecode*** en el RIM, que se verá reflejado en la subpropiedad que se utilice para enlazar dos Actos. Por ejemplo, un Acto de tipo ServiceEvent, estará relacionado con el Acto ClinicalDocument correspondiente al CDA, mediante la subpropiedad ***documents.***

Vemos a continuación un esquema de parte de la jerarquía de propiedades necesaria para representar los datos de un CDA, solo a modo de ejemplo del modelado realizado.



4.2.2 Modelado Ontológico de los Vocabularios (HL7 Voc)

En esta clase se modelan todos los vocabularios definidos en el estándar HL7, que aplican dentro de la especificación de CDA. En este caso, en la ontología de base, solamente se modelaban cinco tipos básicos: *VocActMood, VocActState, VocAdministrativeGender, VocRoleState, VocEntityState*.

Para el resto de los vocabularios necesarios dentro del modelado, se definió la metodología en tres pasos que se explica a continuación.

4.2.2.1 Método de incorporación de un nuevo Vocabulario

Paso 1: Creación de Clase Base del Vocabulario.
Se crea la clase para modelar el vocabulario nuevo, con el nombre que tiene definido en la norma añadiéndole el Prefijo **VOC.** Se define siempre como subclase directa de la clase *HL7VOC*.

Paso 2: Creación de Clases de la Jerarquía del Vocabulario.
Se seleccionan exclusivamente los términos del vocabulario que aplican al CDA, para estos se crea una nueva clase, por cada valor del vocabulario existente. Modelamos la subjerarquía de clases siguiendo la jerarquía del vocabulario HL7.

Paso 3: Incorporación de Individuos para cada una de las clases.
Una vez creadas todas las clases Vocabulario, se definió un individuo (o instancia) perteneciente a dicha subclase.

4.2.2.2 Ejemplo de Instanciación de Vocabulario

En el presente ejemplo de instanciación, se detalla cómo se utiliza un vocabulario concreto.
Cuando se instancia un documento CDA y se quiere establecer el sexo del paciente, el individuo que representa a la entidad paciente, se relacionará mediante la propiedad ***admistrativeGenerCode*** **–**sexo- asociado al individuo perteneciente a la subclase correspondiente del vocabulario *VocAdministrativeGender*. De esta forma, estamos diciendo que el paciente tiene un determinado sexo.
Vemos un esquema gráfico de esta relación:

El círculo verde ( ) representa al individuo “paciente”, el círculo violeta ( ) representa al individuo “masculino”. A través de esta relación llamada ***administrativeGenderCode***, podemos entonces inferir que un determinado paciente, tiene sexo masculino.

En formato *turtle* el ejemplo anterior quería de la siguiente forma:

```
:paciente        RIMV3OWL_CDA: administrativeGenderCode        RIMV3OWL_CDA:entityGenMasc .
```

Siguiendo esta metodología, definimos entonces las siguientes subclases adicionales de Vocabularios:

- VocActClass
- VocActCode
- VocActPriority
- VocActSite
- VocConfidentiality
- VocContextControl
- VocEntityClass
- VocEntityCode
- VocEntityDeterminer
- VocEntityNameUse
- VocEthnicity
- VocMaritalStatus
- VocMaterialForm
- VocMediaType
- VocObservationInterpretation
- VocParticipationFunction
- VocParticipationMode
- VocParticipationSignature
- VocParticipationType
- VocPostalAddressUse
- VocRace
- VocReligiousAffiliation
- VocRoleClass
- VocRoleCode
- VocRouteOfAdministration
- VocTargetAwareness
- VocTelecommunicationAddressUse
- VocURLScheme

4.2.3 Modelado Ontológico de los Tipos de Datos (HL7 Any)

El concepto de esta clase básica y toda la subjerarquía que se desprende de ella, es modelar la definición de tipos de datos que se establece en la norma HL7 y que constituyen el rango de propiedades definidas en el RMIM. De esta forma, realizamos una selección de los tipos de datos que tienen utilidad en el modelado de documentos CDA. Los tipos de datos seleccionados son los que mantenemos de la ontología base, los demás fueron eliminados.

Nos pareció oportuno listar de todos los elementos definidos dentro de esta jerarquía de Tipos de Datos:

- ***II****: Corresponde a identificadores únicos,* es utilizado para identificar distintas instancias de un tipo de entidad. Es utilizado extensivamente en el modelo de CDA para identificar personas, lugares, cosas, acciones, roles.

- ***CD****:* Permite representar códigos, tanto codificaciones externas como LOINC o SNOMED, como para codificaciones definidas dentro de HL7 (llamado Vocabularios).

- ***TEL:*** El tipo de datos TEL (*Telecommunication Address*) permite modelar números de teléfono, direcciones de correo electrónico.

- ***PQ****:* El tipo de datos PQ (*Physical Quantity*) se utiliza para expresar una cantidad dimensionada, que se obtiene como resultado de una medida.

- ***IVL:*** El tipo de datos IVL ofrece la forma de representar un intervalo, sea de tiempo (cuando marcas temporales *DataTime*) o de cantidades cuando se utiliza PQ.

- ***AD****:* Representa la Dirección o ubicación física, se compone de diferentes partes (ADXP).

- ***EN****: Representa Nombres, de organización, personas, etc. Al igual que las direcciones se compone de partes (ENXP).*

4.2.3.1 Comentarios Generales

Nos parece relevante comentar aspectos generales que se tienen en cuenta al momento de crear los diferentes tipos de datos.

4.2.3.2 Método de incorporación de un nuevo Tipo de Datos.

Paso 1: Creación de Clase del Tipo de Datos.

Se crea la clase para modelar el tipo de Datos, con el nombre que tiene en la especificación de la Norma. Se lo ubica dentro de jerarquía según corresponda.

Paso 2: Creación de Propiedades relacionadas al Tipo.

Luego de tener la clase, se crean las propiedades (atributos dentro del tipo de datos) que aplican al tipo específico. Cada una de estas propiedades se las modela como ***OWL DataProperty*** con Dominio en tipo de Datos recientemente creado y rango en tipos de datos básicos (string, boolean, int).

4.2.3.3 Instanciación de un tipo concreto.

Algunas de las propiedades que modelan atributos de clases HL7, tienen su rango en un tipo de datos definido dentro de la norma, que se representa en la ontología como alguna subclase de la clase HL7Any. Para instanciar los datos concretos que llegan en el CDA, sobre individuos pertenecientes a dichas subclases, utilizamos nodos auxiliares, en los casos que necesitamos agrupar la información, que serán individuos de una de esas clases, y sobre ellos se aplican las OWL DataProperties con rango en tipos de datos básicos.

A modo de ejemplo, se presentan tres tipos incluidos en la ontología, II, CS, AD. En cada uno se muestra un aspecto diferente que se tuvo en cuenta. Los demás tipos de datos, se detallan en el Anexo*: "Descripción de Tipos de Datos"*

4.2.3.4 Tipo de Datos II

Definición:

El tipo de datos II es utilizado para identificar distintas instancias de un tipo de entidad. Es utilizado extensivamente en el modelo de CDA para identificar personas, lugares, cosas, acciones, roles.

Atributos que contiene:

Su definición en el estándar HL7, se realiza mediante cuatro atributos, éstos son:

- **root**: identificador único que garantiza la unicidad global del identificador de instancia.
- **extensión**: cadena de texto como un identificador único dentro del alcance de la raíz (root).
- **assigningAuthorityName**: nombre de la Autoridad que asigna el valor del identificador.
- **displayable**: especifica si el identificador está destinado a ser entendido y manipulado por seres humanos (displayable = true), o si de lo contrario está ideado solamente para interoperabilidad entre máquinas (displayable = false).

Definimos una serie de OWL DataProperties, para modelar la representación HL7 de este tipo de datos:

- ii_root: II → string
- ii_extension: II → string
- ii_assigningAuthorityName: II → string
- ii_displayable: II → boolean

Ejemplos en diferentes formatos:

1- Un ejemplo de codificación II dentro de un documento CDA es - representación XML-:

```
<id root="2.16.840.2131" extension="4.0.123.412-3" assigningAuthorityName="DNIC" displayable ="True"/>
```

2- En la instanciación de un objeto de tipo II, se crea un individuo auxiliar, al cual se le asignan todas las propiedades con dominio en este tipo. Este individuo no se define explícitamente como de tipo II, luego de aplicar el razonamiento es que se obtiene el resultado la clasificación correspondiente, de acuerdo las propiedades que tiene.
La siguiente imagen muestra un esquema de las relaciones que aplican, se define el nodo auxiliar nombrándolo en este caso ***id.*** Corresponde al mismo caso que se encuentra en formato XML en la presente sección punto 1.

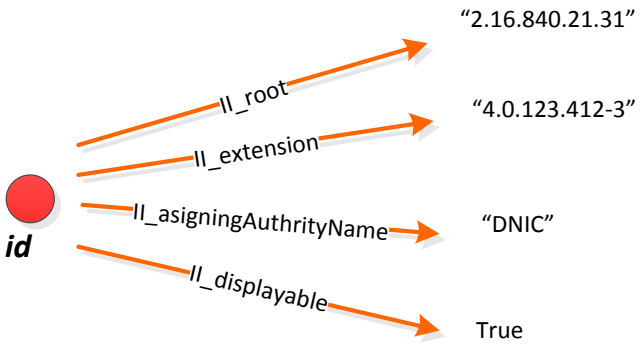


ILUSTRACIÓN 14: REPRESENTACIÓN DE INSTANCIA DE TIPO DE DATOS II.

En formato *turtle* el ejemplo anterior quería de la siguiente forma:

id	RIMV3OWL_CDA:ii_**root**	"2.16.840.21.31"^^xsd:string .
id	RIMV3OWL_CDA:ii_ **extension**	"4.0.123.412-3"^^xsd:string .
id	RIMV3OWL_CDA:ii_ **assigningAuthorityName**	"DNIC"^^xsd:string .
id	RIMV3OWL_CDA:ii_ **displayable**	"True"^^xsd:boolean .

4.2.3.5 Tipo de Datos Code.

Definición:

El tipo de datos CD (Concept Descriptor) representa un código en algún sistema de codificación determinado. Puede corresponder a codificaciones externas estándares como LOINC, CIE10, o códigos internos definidos por el usuario o dentro de la Norma.

Atributos que contiene:

Para representar este código, HL7 define una serie de atributos:

- **code** → un identificador único, correspondiente al código en el sistema de codificación determinado.
- **codeSystem** → el UID del sistema de codificación determinado
- **codeSystemName** → el nombre común de este sistema de codificación
- **codeSystemVersion** → si corresponde, un descriptor de versión definido específicamente para el sistema de codificación dado
- **displayName** → un nombre o título para el código

Definimos una serie de OWL DataProperties, para modelar la representación HL7 de este tipo de datos:

- cd_code: CD → string
- cd_codeSystem: CD → string
- cd_codeSystemName: CD → string
- cd_codeSystemVersion: CD → string
- cd_displayName: CD → boolean

Ejemplos en diferentes formatos:

1- Un ejemplo de codificación CD que podemos encontrar dentro de un documento CDA:

```
<code code="26436-6" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="LABORATORY STUDIES"/>
```

2- Al momento de instanciar este **code** concreto, se crea un individuo auxiliar.

Para cada propiedad con rango en CD, y cuya codificación sea una codificación externa a HL7, definiremos un individuo que se inferirá perteneciente a la clase ontológica “CD”. A este individuo, se le asignarán las propiedades más arriba detalladas, conteniendo los valores concretos que se encuentren en el documento CDA.

La siguiente imagen muestro un esquema de las relaciones que aplican, se define el nodo auxiliar nombrándolo en este caso ***code.*** Corresponde al mismo caso que se encuentra en formato XML en la presente sección punto 1.

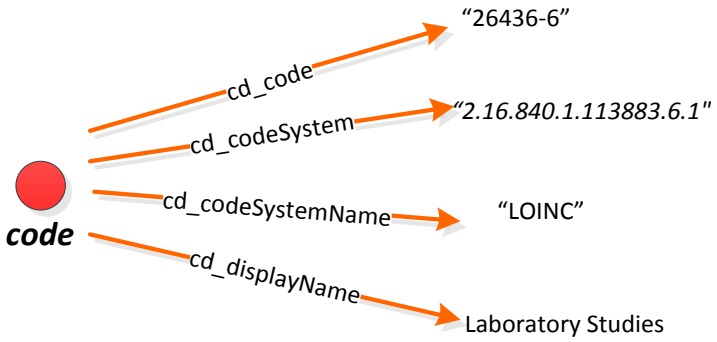


ILUSTRACIÓN 15: REPRESENTACIÓN DE INSTANCIA DE TIPO DE DATOS CODE.

En formato *turtle* el ejemplo anterior quería de la siguiente forma:

code	RIMV3OWL_CDA:cd_**code**	"26436-6"^^xsd:string .
code	RIMV3OWL_CDA:cd_ **codeSystem**	"2.16.840.1.113883.6.1"^^xsd:string.
code	RIMV3OWL_CDA:cd_**codeSystemName**	"LOINC"^^xsd:string .
code	RIMV3OWL_CDA:cd_ **displayName**	"Laboratory Studies"^^xsd:string.

4.2.3.6 Tipo de Datos AD – Direcciones.

Definición

El tipo de datos **AD** (*Postal Address*) se define en el estándar HL7 para modelar todo tipo de direcciones u ubicaciones físicas. Estas direcciones se definen como una secuencia de “partes de dirección” (address parts en inglés – **ADXP**-), como por ejemplo: calle, ciudad, país, etc.

En HL7, este tipo de datos acarrea una cierta complejidad, siendo que cada “parte de dirección” debe ser uno de los términos definidos dentro del vocabulario ***AddressPartType***, y a su vez, cada “parte de dirección”, que está asociado a este término, contiene un valor (texto) que refiere al dato concreto. Por ejemplo, el nombre específico de la calle o del país.

Además, el conjunto de información dado por todas las “partes”, puede estar asociado a otro dato, que se consigna como un atributo XML de nombre “use”. El valor que contiene este atributo, debe pertenecer a otro vocabulario, denominado ***PostalAddressUse***, y que define el uso que se le da a la dirección: *domicilio particular, de trabajo, temporario, etc.*

Atributos que contiene:

Para modelar dentro de la ontología esta información, generamos las siguientes propiedades:

- Una OWL *ObjectProperty*, de nombre **ad_use**, cuyo dominio es AD y su rango es el vocabulario **VocPostalAddressUse**.

- ad_use: AD → VocPostalAddressUse

- Una OWL DataProperty por cada término del vocabulario **VocAddressPartType**, con dominio en AD y rango en string. De este modo, se puede representar todas las partes de la dirección.

  Algunas de estas propiedades son:
  - adxp_country: AD → string
  - adxp_city: AD → string
  - adxp_streetName: AD → string
  - adxp_postalCode: AD → string
  - adxp_buildingNumber: AD → string

Ejemplos en diferentes formatos:

Un ejemplo de la representación de este tipo de datos en un documento CDA podría ser:

```
<addr use='HP'>
        <streetAddressLine>Calle 1356</streetAddressLine>
        <city>Montevideo</city>
        <postalCode>11400</postalCode>
        <country>Uruguay</country>
</addr>
```

La siguiente imagen muestro un esquema de las relaciones que aplican, se define el nodo auxiliar nombrándolo en este caso ***code.*** Corresponde al mismo caso que se encuentra en formato XML de la presente sección punto 1.

En formato *turtle* el ejemplo anterior quería de la siguiente forma:

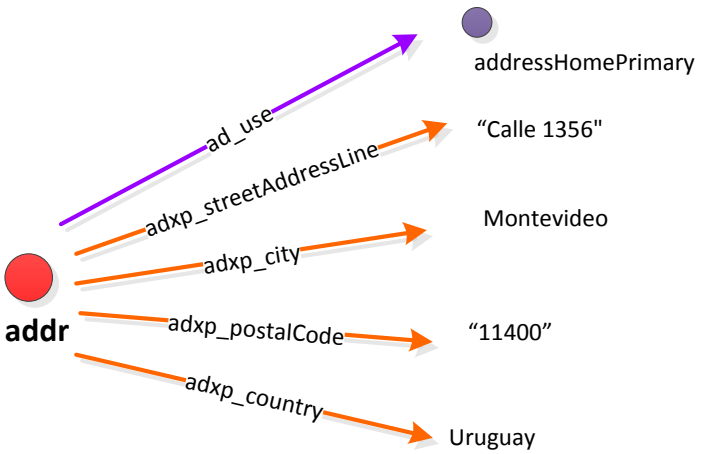


ILUSTRACIÓN 16: REPRESENTACIÓN DE INSTANCIA DE TIPO DE DATOS AD.

En el dibujo, se diferencia la asociación de la propiedad **ad_use,** ya que esta corresponde a un elemento dentro de un vocabulario.

En formato *turtle* el ejemplo anterior quería de la siguiente forma:

:addr	RIMV3OWL_CDA:ad_**use**	RIMV3OWL_CDA : **addressHomePrimary**
:addr	RIMV3OWL_CDA:adxp_**streetAddressLine**	"Calle 1356 "^^xsd:string .
:addr	RIMV3OWL_CDA:adxp_**city**	"Montevideo"^^xsd:string .
:addr	RIMV3OWL_CDA:adxp_**postalCode**	"11400"^^xsd:string .
:addr	RIMV3OWL_CDA:adxp_**country**	"Uruguay"^^xsd:string .

4.3 Transformación CDA a OWL

Un elemento fundamental que se tuvo que dedicar tiempo consistió en realizar la traducción de documentos CDA escritos en formato XML, a instancias OWL serializadas en formato Turtle.

Se desarrolló una transformación mediante lenguaje XSLT, que denominamos **CDAtoOWL.xsl,** y que puede ser ejecutada en cualquier procesador XSLT.

XSLT es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. Una hoja XSLT realiza la transformación del documento utilizando una o varias reglas de plantilla, llamadas ***templates***. XSLT permite formatear los elementos fuente basados en relaciones de ancestro/descendiente, posición y unicidad. El documento original se traduce a un árbol que representa dicho documento. La conversión se realiza a través de los templates, compuestos de un patrón mediante la obtención de elementos con *Xpath*, que especifica a qué elementos del árbol fuente se aplica, y una acción, que indica cómo se traduce un subárbol en otro.

Comentamos a continuación los principales aspectos de la metodología utilizada para el diseño de la transformación CDAtoOWL.xsl.

- Se modelan **propiedades genéricas** en templates, que sirven para crear un individuo nuevo, y para representar cualquier DataPropery u ObjectProperty.

- Se genera un template por cada **propiedad** que posea **rango sobre un determinado Vocabulario**. Cabe acotar, que para algunas de estas propiedades que modelan atributos opcionales de la estructura del CDA, el template correspondiente no está completo.

- Se generan templates que corresponden a secciones del R-MIM, y se establece una secuencia de ejecución para ellos, según se muestra en la *"Ilustración 17: Estructura de invocacion de la CdaTOowl.xsl"*. Aquí se van invocando los templates mencionados en los dos puntos anteriores, que se reutilizan cada vez que deba generarse una propiedad similar.

Se cubre así, la estructura completa del R-MIM, pudiendo convertir cualquier clase HL7 presente en un CDA, ya sea perteneciente al *header* o al *body* del mismo.

En la sección 4.4 se explica con mayor nivel de detalle, el proceso conceptual de traducción de un CDA.

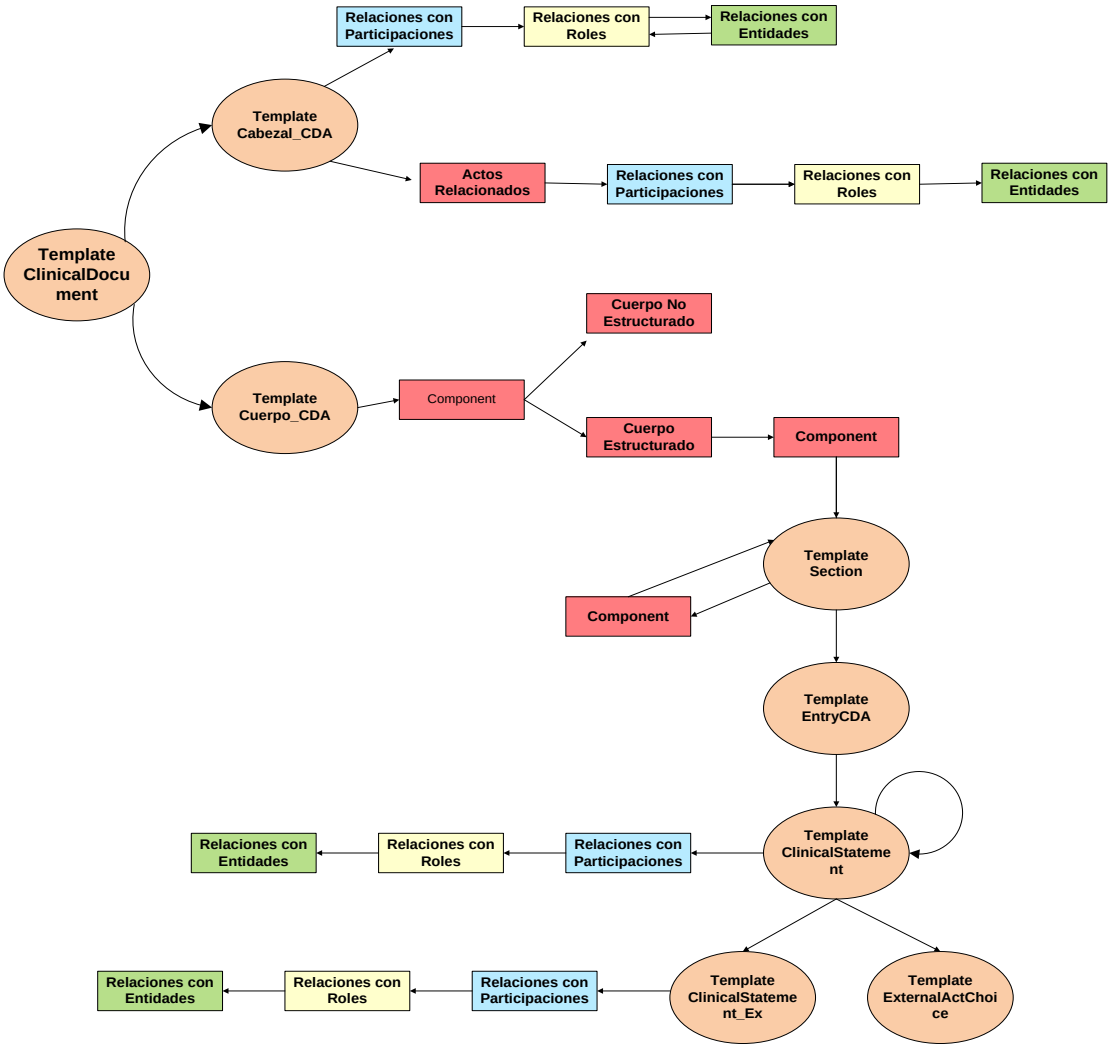


ILUSTRACIÓN 17: ESTRUCTURA DE INVOCACION DE LA CDATOOWL.XSL

4.4 Instanciación de un CDA en la ontología.

Luego de realizar todo el modelado la ontología matriz, el siguiente paso es, partiendo de diferentes documentos CDA (en formato XML), crear las instancias de los mismos para incluir en la ontología resultante.

Denominamos instancia, a la traducción de un documento CDA a tripletas OWL en formato Turtle.

En esta sección, detallaremos el proceso que se realiza al momento de crear las tripletas con la información de los documentos. En una primera parte se explicarán las pautas generales respecto a cómo se van generando las tripletas según la estructura de la ontología; y luego se describe un caso práctico a modo de ver más claramente los conceptos antes explicados.

Se debedió tener especial cuidado en la creación de los identificadores de los elementos a incorporar en la ontología, el proceso completo de creación, se describe en el Anexo *"Detalles de la Nomenclatura utilizada."*

4.4.1 Creación de instancias

Para cada elemento, que represente una de las cuatro clases fundamentales – *Act, Participation, Role y Entity* - se le asocia un individuo único dentro de la ontología. Estos individuos se generan dentro de la superclase de toda la jerarquía de clases de la ontología (*Thing*), es decir, que no se generan dentro de la clase ontológica específica a la que pertenecen. Como se mencionaba anteriormente, un individuo se reconocerá como miembro de cierta clase, a través de las inferencias que se realicen en función el valor de sus propiedades o de las relaciones que tenga con otras clases.

Por ejemplo, para representar acto principal, *docclin*, se creará una tripleta de la siguiente forma:

```
DocClin_id    rdf:type    owl:Thing.
```

Donde "DocClin_id" es el identificador único de este acto.

A este individuo que representa un Act (Docclin), se le asignan las propiedades que describen sus atributos, por ejemplo el título del documento:

```
DocClin_id    RIMV3OWL_CDA:act_title  "Titulo"^^xsd:string .
```

También, se le asignan las propiedades que vinculan este Act con las diversas Participation que correspondan. Por ejemplo:

```
DocClin_id    RIMV3OWL_CDA:hasParticipation RIMV3OWL_CDA:author1  .
```

Este mecanismo se repite para cada Act, Participation, Role y Entity presentes en el documento CDA, teniendo en cuenta que un Act solamente se puede relacionar, o bien con otro Act, o bien con una Participation. A su vez, una Participation se puede relacionar exclusivamente con un Role, un Role solamente puede relacionarse con una Entity. Y la Entity, es el elemento final que refiere a una Organización, Persona, Paciente, Dispositivo, Material Manufacturado o Lugar. En algún caso particular definido en el R-MIM, una Entity puede a su vez relacionarse con un Role.

En la Ilustración siguiente *(Ilustración 18*), se representa el cabezal del CDA en el modelo ontológico, donde cada círculo de color representa un individuo generado en la ontología, cuya pertenencia a las clases fundacionales se deduce de sus relaciones con los demás individuos. En este gráfico se mantienen los colores que se utilizan en el R-MIM para identificar a las clases fundacionales.

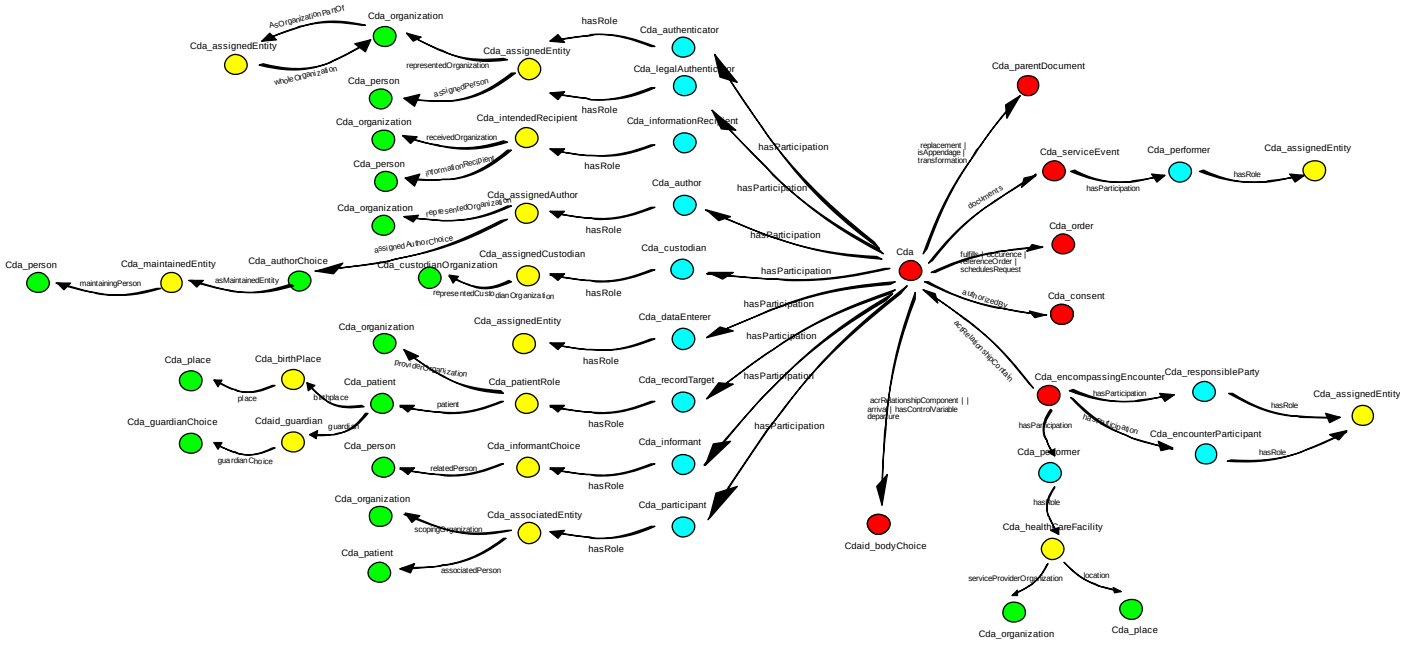


ILUSTRACIÓN 18: HEADER DEL CDA, EN EL MODELO ONTOLÓGICO.

4.4.2 Ejemplo de instanciación

Veamos un fragmento de CDA donde se definen los atributos del Acto ClinicalDocument, y de la Participación recordTarget :

```
<ClinicalDocument … >

    <!-- Identificación de la instancia de documento -->
    <id root="AA" extension="111"/>

    <!-- Código del tipo de documento -->
     <code code="51851-4"codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="Administrative note"/>

    <!-- Título del documento -->
    <title>LICENCIA MEDICA</title>

    <!-- Fecha del documento [Fecha del Acto Médico] AAAAMMDDHHMMSS -->
    <effectiveTime value="20110627210938-0300"/>

       <!-- ****** SET DE DATOS DEL REGISTRO DE PACIENTE ****** -->
    <recordTarget typeCode="RCT" contextControlCode="OP">

    <!-- Nivel de confidencialidad Very Restricted-->
    <confidentialityCode code="V" codeSystem="2.16.840.1.113883.5.25"/>


        <patientRole classCode="PAT">

        <!-- Identificación del paciente -->
        <id root="2.16.858.60" extension="910969" assigningAuthorityName="DNIC"/>
              <addr use="HP">
                    <streetAddressLine>Calle 1356</streetAddressLine>
                    <city>Montevideo</city>
                    <country>Uruguay</country>
                    <postalCode>11400</postalCode>
              </addr>

              <telecom use="H" value="tel:+29130366"/>

              <patient classCode="PSN">

                        <!-- Nombre del paciente -->
                        <name>
                            <given>ADRIANO</given>
                            <given>B.</given>
                            <family>ALFA</family>
                        </name>
                        <administrativeGenderCode code="M" codeSystem="2.16.840.1.113883.5.1"/>
                    <birthTime value="19811217"/>
```

```
                </patient>
            </patientRole>
        </recordTarget>
…
…
</ClinicalDocument >
```

Estamos mostrando el fragmento de un documento CDA, que corresponde al siguiente tramo del R-MIM:

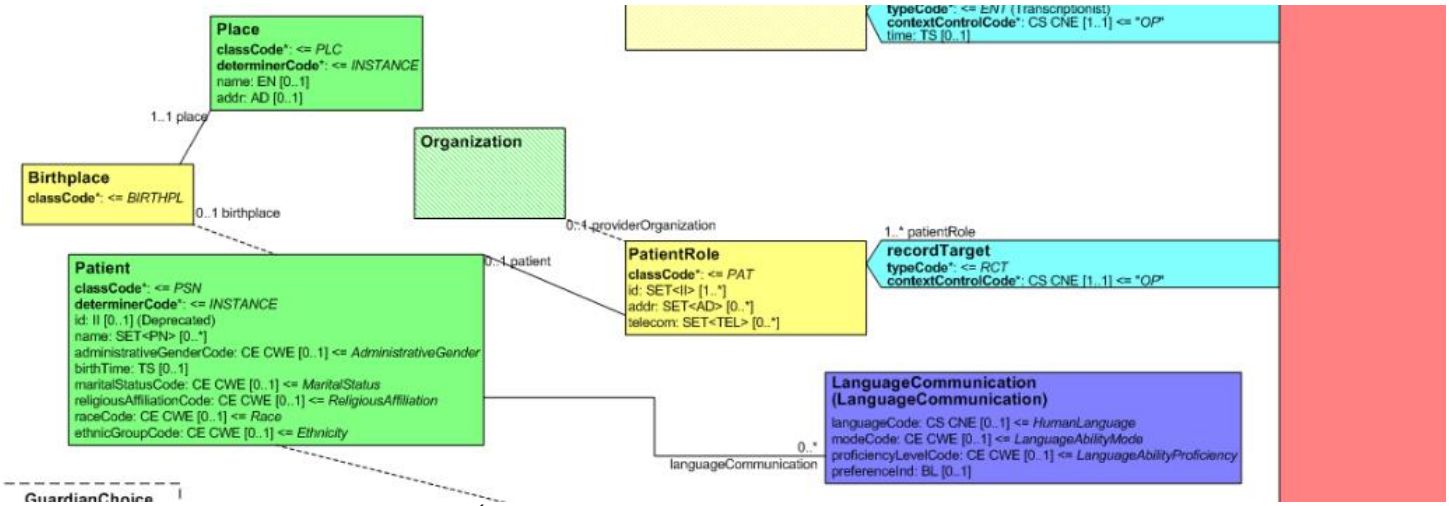


ILUSTRACIÓN 19: REPRESENTACION RMIM DEL PACIENTE.

Este fragmento se instancia dentro de la ontología de la siguiente forma:

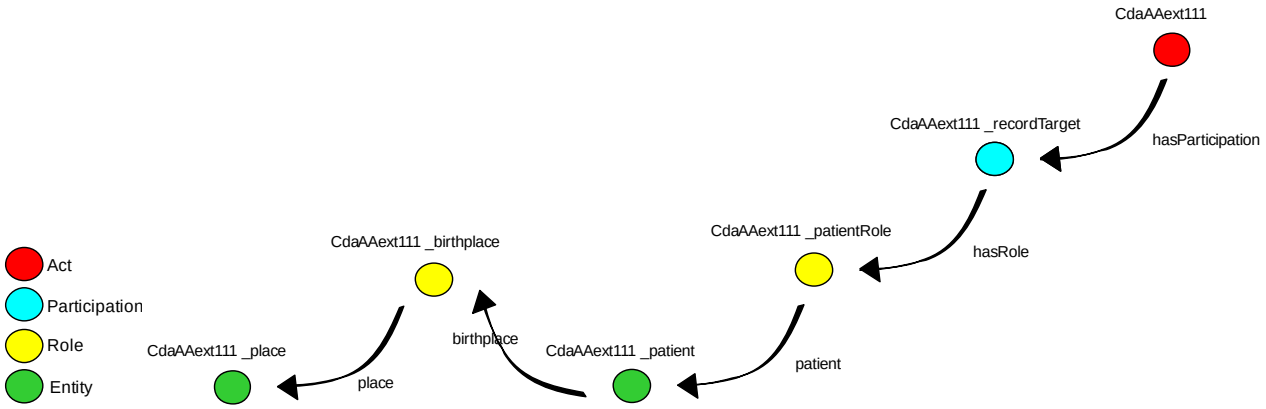


ILUSTRACIÓN 20: REPRESENTACION ONTOLÓGICA DEL PACIENTE.

Cada círculo representa un individuo, los colores se adoptan del modelo R-MIM.

Los individuos generados se describen a continuación:

- **CdaAAext111** → es el individuo que modela el Acto correspondiente al Documento Clínico, es la base de la instanciación de cualquier CDA.
- **CdaAAext111_recordTarget** → es el individuo que modela la participación recordTarget
- **CdaAAext111_patientRole** → es el individuo que modela el rol patientRole
- **CdaAAext111_patient** → es el individuo que modela la entidad Patient
- **CdaAAext111_birthplace** → es el individuo que modela el rol birthplace

- **CdaAAext111_place** → es el individuo que modela la entidad Place

Las tripletas correspondientes a la instanciación en formato turtle, mostrada en el esquema anterior, son las siguientes:

:cdaAAext111	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:hasParticipation	:cdaAAext111_recordTarget .
:cdaAAext111 _recordTarget	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:hasRole	:cdaAAext111_patientRole .
:cdaAAext111_patientRole	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:patient	:cdaAAext111_patient .
:cdaAAext111_patient	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:birthplace	:cdaAAext111_birthplace .
:cdaAAext111_birthplace	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:place	:cdaAAext111_place .
:cdaAAext111_place	rdf:type	owl:NamedIndividual , owl:Thing.

Vemos ahora cómo se modelan los atributos que figuran en el R-MIM para cada una de las clases involucradas.

Individuo CdaAAext111:

1) Atributo **classCode** → El valor que contiene este atributo, es uno de los valores posibles del Vocabulario HL7 denominado ActClass, y definido en la ontología dentro de la clase *VocActClass*, como subclase de HL7VOC. Para declarar que el individuo CdaAAext111 posee un código de clase determinado, utilizamos la propiedad **act_classCode**, con dominio en **Act** y rango en **VocActClass**. Para este caso, el valor del atributo es "DOC", por lo que toma el valor del vocabulario "*actDocClin*."

:cdaAAext111	RIMV3OWL_CDA:act_classCode	RIMV3OWL_CDA:actDocClin .

2) Atributo **id** → Comprende una serie de datos que son recogidos en las propiedades correspondientes al tipo de datos HL7 denominado *II*. Utilizamos por tanto, un nodo auxiliar, identificado como **CdaAAext111_id.** Asignamos a este nodo auxiliar, las siguientes propiedades, cuyo dominio reside en la clase *II* que representa el tipo de datos homónimo de la norma HL7: **ii_root, ii_extension, ii_displayable, ii_assigningAuthorityName**, según lo detallado en la Secc: "*4.2.3.4 Tipo de Datos II*" del presente capítulo.

   En todos los casos, el rango de estas propiedades es un valor de cadena de texto.
   El nodo auxiliar **CdaAAext111_id,** corresponde conceptualmente a la identificación del documento CDA. El individuo **CdaAAext111** se relaciona con éste nodo auxiliar mediante la propiedad **act_id**, cuyo dominio está en la clase **Act** y su rango en la clase **II**.

:cdaAAext111_id	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:ii_root	"AA"^^xsd:string ;
	RIMV3OWL_CDA:ii_extension	"111"^^xsd:string.
:cdaAAext111	RIMV3OWL_CDA:act_id	:cdaAAext111_id .

3) Atributo **code** → Comprende una serie de datos que son recogidos en las propiedades correspondientes al tipo de datos HL7 denominado CE. El mismo deriva del tipo de datos CD, para el cual se definen una serie de propiedades que son heredadas por CE. Utilizamos un nodo auxiliar, identificado como **CdaAAext111_code**, al cual asignaremos las siguientes propiedades, con dominio en la clase CD de la ontología, que modela el tipo de datos CD de HL7: **cd_code, cd_codeSystem, cd_codeSystemName, cd_codeSystemVersion, cd_displayName, cd_originalText**, según se detalla en la Secc: " *4.2.3.5 Tipo de Datos Code."*

En todos los casos, el rango de estas propiedades es un valor de cadena de texto.
El nodo auxiliar **CdaAAext111_code,** corresponde conceptualmente a la codificación del tipo de documento CDA. El individuo **CdaAAext111** se relaciona con éste nodo auxiliar mediante la propiedad **act_code**, cuyo dominio está en la clase **Act** y su rango en la clase **CD**. Por lo general, la codificación de tipo de documento se brinda bajo el sistema **LOINC**, codificación externa a HL7.

:cdaAAext111 _code	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:cd_code	"51851-4"^^xsd:string ;
	RIMV3OWL_CDA:cd_codeSystem	"2.16.840.1.113883.6.1"^^xsd:string ;
	RIMV3OWL_CDA:cd_codeSystemName	"LOINC"^^xsd:string ;
	RIMV3OWL_CDA:cd_displayName	"Administrative note"^^xsd:string .
:cdaAAext111	RIMV3OWL_CDA:act_code	:cdaAAext111_code .

4) Atributo **title** → Tiene un valor de cadena de texto, que corresponde al título del documento CDA. Este valor lo relacionamos con el individuo **CdaAAext111,** mediante la propiedad **act_title**, cuyo dominio es Act y su rango es un valor de cadena de texto. Está modelada entonces como Dataproperty de OWL. Esta es una modificación que realizamos sobre la ontología base, ya que originalmente esta propiedad venía modelada como ObjectProperty. De esta forma, hubiéramos tenido que generar un nuevo individuo: **CdaAAext111_title,** y a él aplicarle otra Dataproperty que permitiera guardar el valor textual. Esta forma de modelar que había sido asumida en la ontología base, se justifica en la intención de mantener el mayor apego posible con la estructura del modelo de información RIM. Dado que el atributo **title** definido para la clase fundacional HL7 Acto, tiene como rango el tipo de datos ST, es la razón por la cual la propiedad **act_title** estaba definida como una ObjectProperty con dominio Act y con rango ST. Consideramos, en el contexto de este trabajo, que no aporta al reconocimiento de la semántica del documento, esta generación de un nodo auxiliar para instanciar el título del documento.

:cdaAAext111	RIMV3OWL_CDA:act_title	"LICENCIA MEDICA"^^xsd:string.

5) Atributo **effectiveTime** → Contiene un valor de fecha y hora que es recogido en una propiedad correspondiente al tipo de datos HL7 denominado IVL_TS. Utilizamos un nodo auxiliar, identificado como **CDAAext111_effectiveTime**, al cual asignamos la propiedad **ivl_ts_value**, con dominio en la clase

IVL_TS de la ontología, que modela el tipo de datos IVL_TS de HL7. Por más detalle, el tipo de datos IVL_TS se explica en el Anexo*: "Descripción de Tipos de Datos."*

:cdaAAext111 _effectiveTime	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:ivt_ts_value	"20110627210938-0300"^^xsd:datetime.
:cdaAAext111	RIMV3OWL_CDA:act_effectiveTime	:cdaAAext111_effectiveTime .

El rango de la propiedad **ivl_ts_value** es un valor del tipo de datos básico datetime.
El nodo auxiliar **CdaAAext111_effectiveTime**, corresponde conceptualmente al valor de fecha y hora exactas de creación del documento CDA. El individuo **CdaAAext111** se relaciona con éste nodo auxiliar mediante la propiedad **act_effectiveTime**, cuyo dominio está en la clase **Act** y su rango en la clase **IVL_TS**.

6) Atributo **confidentialityCode** → El valor que contiene este atributo, es uno de los valores posibles del Vocabulario HL7 denominado Confidentiality, y definido en la ontología dentro de la clase **VocActConfidentiality,** como subclase de HL7VOC. Este código indica el grado de confidencialidad del documento. Para declarar que el individuo CdaAAext111 posee un código de confidencialidad determinado, utilizamos la propiedad **act_confidentialityCode**, con dominio en **Act** y rango en **VocActConfidentiality**.

:cdaAAext111	RIMV3OWL_CDA:act_confidentialityCode	RIMV3OWL_CDA:actConfVeryRest .

Mostramos gráficamente los resultados de la instanciación de las propiedades del individuo **CdaAAext111**:

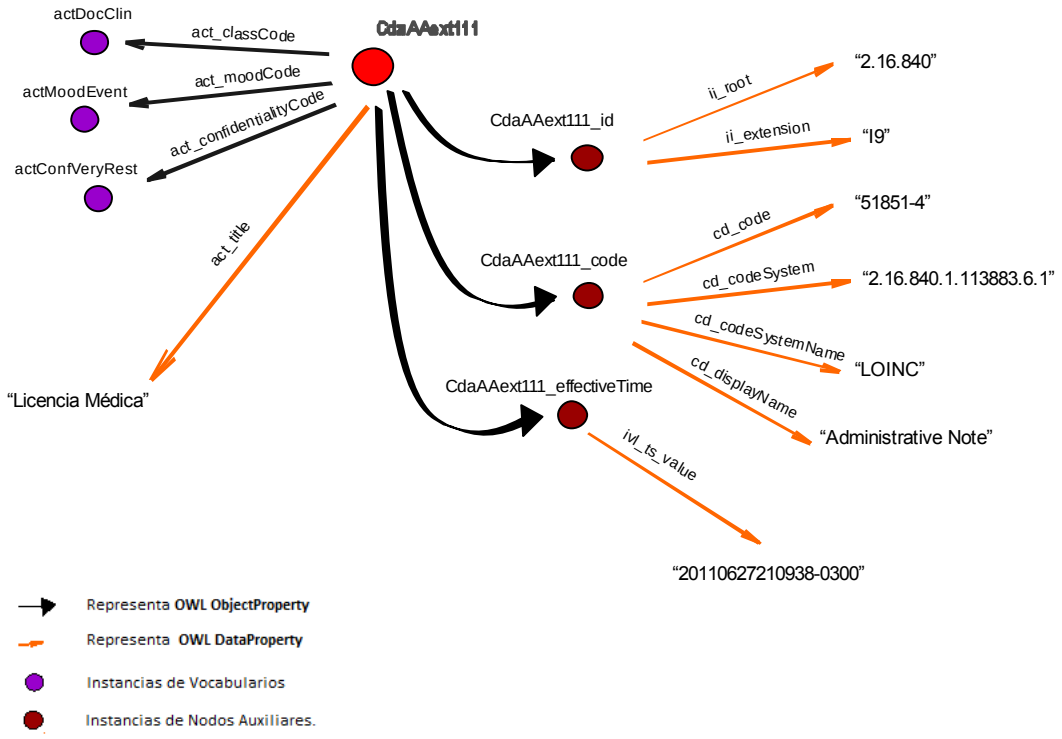


ILUSTRACIÓN 21: REPRESENTACIÓN ONTOLÓGICA DEL DOCCLIN - ACTO PRINCIPAL-

Individuo CdaAAext111_recordTarget:

7) Atributo **typeCode** → El valor que contiene este atributo, es uno de los valores posibles del Vocabulario HL7 denominado ParticipationType, y definido en la ontología dentro de la clase VocParticipationType, como subclase de HL7VOC. Para declarar que el individuo CdaAAext111_recordTarget posee un código de tipo determinado, utilizamos la propiedad **part_typeCode**, con dominio en Participation y rango en **VocParticipationType**.

```
:cdaAAext111_recordTarget RIMV3OWL_CDA:part_typeCode RIMV3OWL_CDA:partTRecordTarget .
```

8) Atributo **contextControlCode** → El valor que contiene este atributo, es uno de los valores posibles del Vocabulario HL7 denominado ContextControl, y definido en la ontología dentro de la clase VocContextControl, como subclase de HL7VOC. Para declarar que el individuo CdaAAext111_recordTarget posee un código de contexto determinado, utilizamos la propiedad **part_contextControlCode**, con dominio en Participation y rango en **VocContextControl**.

```
: cdaAAext111_recordTarget RIMV3OWL_CDA:part_contextControlCode
RIMV3OWL_CDA:partOverridingPropagating .
```

Mostramos gráficamente los resultados de la instanciación de las propiedades del individuo **CdaAAext111_recordTarget**:

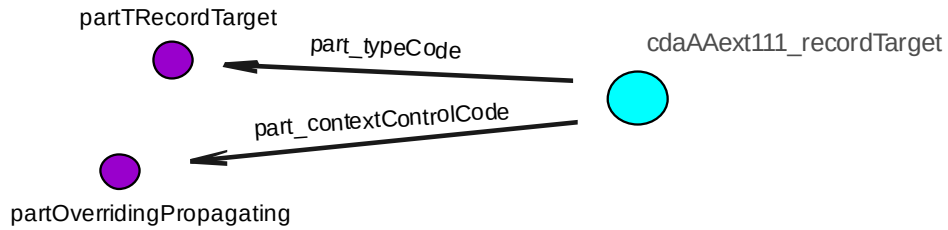


ILUSTRACIÓN 22: REPRESENTACIÓN ONTOLÓGICA DEL RECORD TARGET

Individuo CdaAAext111_patientRole:

9) Atributo **classCode** → El valor que contiene este atributo, es uno de los valores posibles del Vocabulario HL7 denominado RoleClass, y definido en la ontología dentro de la clase VocRoleClass, como subclase de HL7VOC. Para declarar que el individuo CdaAAext111_patientRole posee un código de clase determinado, utilizamos la propiedad **role_classCode**, con dominio en **Role** y rango en **VocRoleClass**.

```
:cdaAAext111_patientRole RIMV3OWL_CDA:roleClassCd RIMV3OWL_CDA:rolePatient .
```

10) Atributo **id** → Corresponde a la identificación del paciente. Se modela de la misma forma que el atributo id del individuo correspondiente al Acto ClinicalDocument. Se genera un nodo auxiliar CdaAAext111_patientRole_id, al cual se le asignan las propiedades del dominio II. El individuo CdaAAext111_patientRole se relaciona con el nodo auxiliar a través de la propiedad **role_id**, cuyo dominio reside en la clase **Role** y su rango en la clase **II**.

```
:cdaAAext111_patientRole_id     rdf:type                                  owl:NamedIndividual,
                                                                          owl:Thing;
                                RIMV3OWL_CDA:ii_root                      "2.16.858.60"^^xsd:string ;
                                RIMV3OWL_CDA:ii_extension                 "910969"^^xsd:string ;
                                RIMV3OWL_CDA:ii_assigningAuthorityName    “DNIC”^^xsd:string .

:cdaAAext111_patientRole        RIMV3OWL_CDA:role_id        :cdaAAext111_patientRole_id .
```

11) Atributo **addr** → Porta la dirección postal del paciente. El atributo *use*, es el que indica un valor posible del vocabulario PostalAddressUse, definido en la ontología dentro de la clase VocPostalAddressUse, como subclase de HL7VOC.

La dirección postal se estructura con varias partes, tal como fue explicado en la Secc: “*4.2.3.6 Tipo de Datos AD – Direcciones.*” del presente capítulo. Para modelar este atributo de la clase fundacional Role, generamos un nodo auxiliar que representa la dirección postal, que llamamos **CdaAAext111_patientRole_addr**. En principio, a este nodo se le asocia la propiedad **ad_use**, con dominio en la clase **AD** (representación del tipo de datos AD de HL7), y rango en la clase **VocPostalAddressUse**.
Seguidamente, se le asignan las propiedades: **adxp_streetAddressLine, adxp_city, adxp_state, adxp_postalCode** y **adxp_country.**

El individuo **CdaAAext111_patientRole**, se vincula con el nuevo individuo addr, mediante la propiedad **role_addr**, con dominio en la clase Role y rango en la clase **AD**.

```
:cdaAAext111 _patientRole_addr   rdf:type                                  owl:NamedIndividual ,
                                                                           owl:Thing;
                                 RIMV3OWL_CDA:ad_use
                RIMV3OWL_CDA:addressPrimaryHome ;
                                 RIMV3OWL_CDA:adxp_streetAddressLine       "Calle 1356"^^xsd:string ;
                                 RIMV3OWL_CDA:adxp_city                    "Montevideo"^^xsd:string ;
                                 RIMV3OWL_CDA:adxp_state                   “Montevideo”^^xsd:string ;
                                 RIMV3OWL_CDA:adxp_postalCode              “11400”^^xsd:string ;
                                 RIMV3OWL_CDA:adxp_country                 “Uruguay”^^xsd:string .

:cdaAAext111_patientRole         RIMV3OWL_CDA:role_addr                    :cdaAAext111_patientRole_addr .
```

12) Atributo **telecom** → Comprende información de contacto con el paciente. En el fragmento de ejemplo, se ven 3 elementos de tipo telecom. Cada uno de ellos, tiene un cualificador bajo el atributo *use*, cuyo valor corresponde a alguna expresión del vocabulario HL7 denominado TelecommunicationAddressUse, y que está definido en la ontología dentro de la clase **VocTelecommunicationAddressUse**, como subclase de HL7VOC.

A su vez, bajo el atributo *value*, se obtiene una cadena de texto con el dato correspondiente. Modelamos este atributo de la clase fundacional Role, mediante tres nodos auxiliares que llamamos **CdaAAext111_patientRole_telecom1, CdaAAext111_patientRole_telecom2 y CdaAAext111_patientRole_telecom3.** Cada uno de estos nodos auxiliares, representa uno de los elementos *telecom* definidos dentro del rol patientRole. Asociamos los datos mediante dos propiedades. La primera de ellas, **tel_use**, tiene su dominio en la clase **TEL** y su rango en el vocabulario **VocTelecommunicationAddressUse**, y es la encargada de recoger el cualificador del atributo telecom.
La segunda propiedad, **tel_value**, tiene su dominio en la clase **TEL** y su rango en el tipo de datos **string**, aquí se guarda el valor del dato concreto.

El individuo **CdaAAext111_patientRole**, se vincula con los nuevos individuos telecom creados, mediante la propiedad **role_telecom**, con dominio en la clase **Role** y rango en la clase **TEL**.

:cdaAAext111_patientRole_telecom1	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:tel_use	RIMV3OWL_CDA:addressHomeAddress ;
	RIMV3OWL_CDA:tel_value	"tel:+29130366"^^xsd:string .
:cdaAAext111_patientRole	RIMV3OWL_CDA:role_telecom	:cdaAAext111_patientRole_telecom1.
:cdaAAext111_patientRole_telecom2	rdf:type	owl:NamedIndividual , owl:Thing;
	RIMV3OWL_CDA:tel_use	RIMV3OWL_CDA:addressMobile ;
	RIMV3OWL_CDA:tel_value	"tel:+096609558"^^xsd:string .
:cdaAAext111_patientRole	RIMV3OWL_CDA:role_telecom	:cdaAAext111_patientRole_telecom2.
:cdaAAext111_patientRole_telecom3	rdf:type	owl:NamedIndividual, owl:Thing;
	RIMV3OWL_CDA:tel_use	RIMV3OWL_CDA:addressDirect ;
	RIMV3OWL_CDA:tel_value	"mailto://aa@gmail.com"^^xsd:string .
:cdaAAext111_patientRole	RIMV3OWL_CDA:role_telecom	:cdaAAext111_patientRole_telecom3.

Vemos un esquema gráfico de la instanciación del individuo **CdaAAext111 _patientRole**

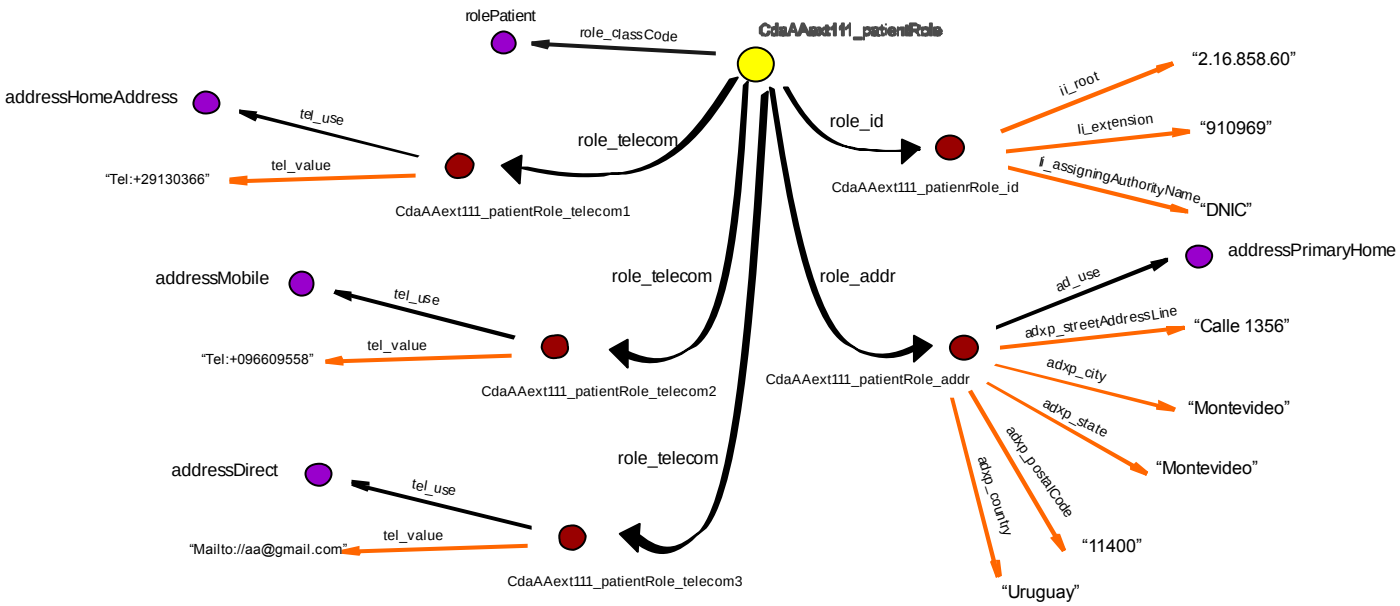


ILUSTRACIÓN 23: REPRESENTACIÓN ONTOLÓGICA DEL PACIENTE.

Individuo CdaAAext111 _patient:

13) Atributo **name** → Registra el nombre del paciente. Este nombre, puede estar compuesto por varias partes, de la misma forma que vimos para el atributo addr de la clase fundacional HL7 Role, y lo modelamos de una forma similar.
El cualificador *use*, es el que indica un valor posible del vocabulario **EntityNameUse**, definido en la ontología dentro de la clase **VocEntityNameUse,** como subclase de HL7VOC.
Para modelar este atributo de la clase fundacional Entity, generamos un nodo auxiliar que representa el nombre, denominado **CdaAAext111_patient_name**. En principio, a este nodo se le asocia la propiedad **en_use**, con dominio en la clase **EN**, que modela el tipo de datos EN de HL7, y rango en la clase **VocEntityNameUse**. En el presente ejemplo, no se declara el uso del nombre, por lo cual no utilizamos esta propiedad.
Se le asignan las siguientes propiedades, correspondientes a las diferentes partes del nombre: **enxp_given** y **enxp_family.**

El individuo **CdaAAext111_patient**, se vincula con el nuevo individuo creado para representar el atributo name, mediante la propiedad **entity_name**, con dominio en la clase Entity y rango en la clase EN.

```
:cdaAAext111_patient_name   rdf:type                        owl:NamedIndividual ,
                                                            owl:Thing;
                            RIMV3OWL_CDA:enxp_given         "ADRIANO"^^xsd:string ;
                            RIMV3OWL_CDA:enxp_given         "B."^^xsd:string ;
                            RIMV3OWL_CDA:enxp_family        “ALFA”^^xsd:string .

:cdaAAext111_patient        RIMV3OWL_CDA:entity_name        :cdaAAext111_patient_name .
```

14) Atributo **administrativeGenderCode** → Indica el sexo del paciente. Según la codificación registrada en el atributo XML *code*, se estará indicando uno de tres posibles valores, correspondientes al vocabulario HL7 AdministrativeGender, modelado en la ontología dentro de la clase VocAdministrativeGender, como subclase de HL7VOC. Se asocia entonces, al individuo **CdaAAext111_patient,** mediante la propiedad **entity_administrativeGenderCode** con dominio en EntityLivingSubject y rango en VocAdministrativeGender, el individuo correspondiente de dicho vocabulario que refiere al tipo de sexo indicado.

```
:cdaAAext111_patient    RIMV3OWL_CDA:entity_administrativeGenderCode :RIMV3OWL_CDA:entityGenMasc .
```

15) Atributo **birthtime** → Indica la fecha de nacimiento del paciente. En el atributo XML “value” se indica dicha fecha. Al individuo **CdaAAext111_patient** se le asocia la fecha de nacimiento mediante la propiedad **entity_birthTime**, con dominio en **EntityLivingSubject** y rango en el tipo de datos **date**.

```
:cdaAAext111_patient    RIMV3OWL_CDA:entity_birthTime    “1981-12-17”^^xsd:date.
```

Esquema gráfico de la instanciación de los atributos del individuo **CdaAAext111_patient:**

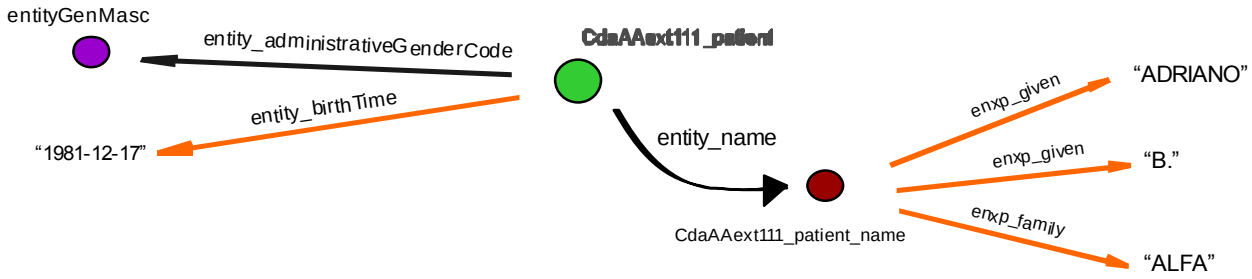


ILUSTRACIÓN 24: REPRESENTACIÓN ONTOLÓGICA DE LA ENTIDAD PACIENTE.

CAPÍTULO 5

PROTOTIPOS

En este capítulo, se explicarán los dos prototipos prácticos realizados para mostrar la utilidad del estudio. En primer lugar, se describirá la aplicación desarrollada en Java que contiene todas las funcionalidades que se detallan en los requerimientos; luego, se muestra el servidor de Fuseki, utilizando lo referente a la ontología creada e instancias.

En ambos casos, se utiliza la herramienta **Jena**, consistente en un framework Java para construir aplicaciones basadas en ontologías. Provee dos variedades de propuestas de almacenamiento para persistir tripletas RDF, bajo los componentes **SDB** y **TDB**. SDB gestiona la persistencia de la información semántica bajo una **base de datos relacional**, mientras que TDB lo hace en un **sistema de archivos**.

La ***aplicación Prot-PgSemCDA***, desarrollada en Java, es un ejemplo de una solución que integra todos los componentes generados y provee una forma práctica de incluir información semántica a la ontología matriz y razonar sobre ella. Utiliza el **componente SDB** de **Jena**, con lo cual las tripletas se persisten en una base de datos relacional. Se construye un modelo ontológico, partiendo de la ontología matriz. Luego, es posible agregar una o varias instancias de documentos CDA, así como aplicar modificaciones al repositorio semántico mediante operaciones SPARQL Update.

Sobre este repositorio semántico, es posible ejecutar consultas SPARQL, extrayendo información según las condiciones semánticas que se apliquen.

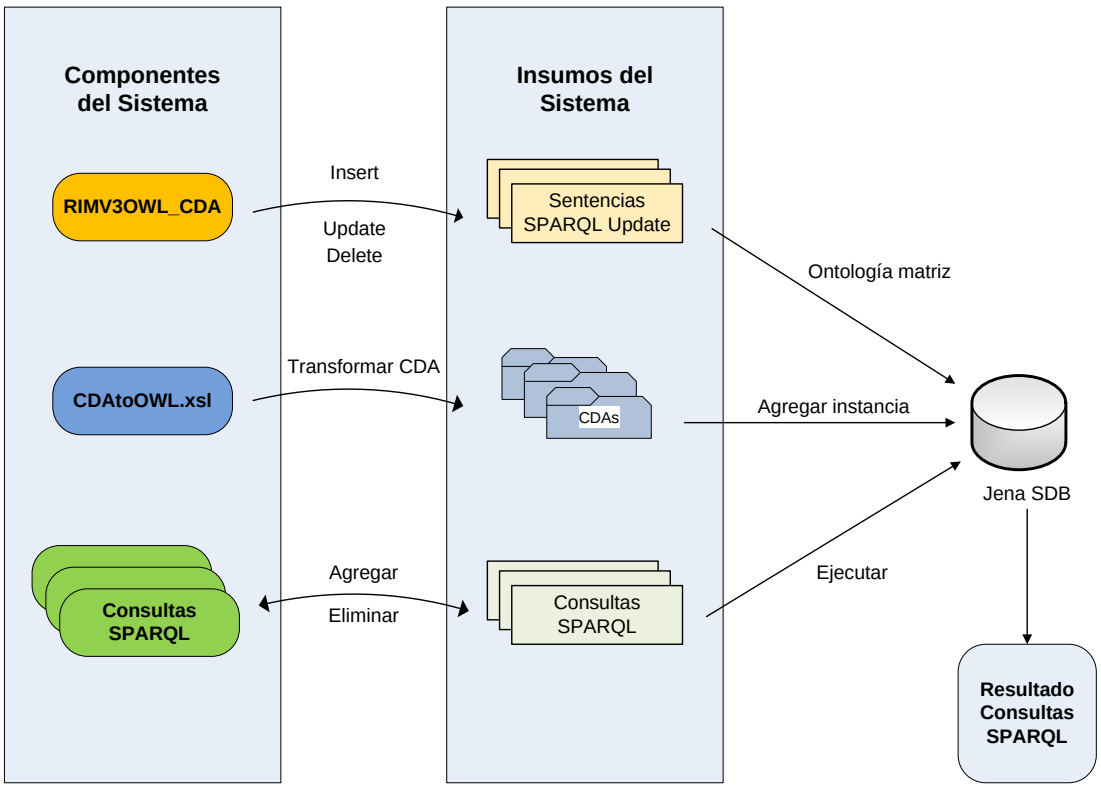


ILUSTRACIÓN 25 - ESQUEMA DE LA APLICACIÓN PROT-PGSEMCDA

Por otra parte, la utilización del ***servidor Fuseki*** brinda la posibilidad de manipular la ontología y razonar, sin la dependencia de una aplicación específica, pudiendo realizar consultas mediante el protocolo SPARQL sobre HTTP (SOH – ***Sparql Over HTTP)***.

Dicho protocolo es un conjunto de scripts de línea de comandos para trabajar con SPARQL 1.1, es independiente del servidor y funciona con cualquier sistema compatible con SPARQL 1.1 ofreciendo acceso HTTP. Fuseki es un servidor SPARQL sobre HTTP con soporte TDB y SDB.

Para complementar la investigación realizada sobre estos componentes, y dado que el SDB fue el utilizado en la aplicación Java, nuestra propuesta sobre la plataforma Fuseki hace uso del **componente TDB** – que utiliza una base transaccional.

Un servidor Fuseki puede ser iniciado mediante un archivo de configuración, donde se define un grafo RDF mediante un cierto modelo. Estos archivos de configuración, usualmente son denominados archivos assembler.

Nuestra propuesta sobre Fuseki, se resume a un archivo assembler que inicia el servidor creando un modelo a partir de la ontología matriz, y conectando con el componente TDB, de modo que los datos quedan persistidos en un sistema de archivos local. Ya habiendo levantado el servidor, se accede mediante un navegaro web, con una URL definida. Desde allo, se pueden agregar las instancias de CDAs que se requiera (documentos CDAs traducidos a OWL en formato Turtle mediante la transformación CDAtoOWL.xsl), además de agregar, modificar o quitar restricciones, clases o propiedades al repositorio semántico mediante operaciones SPARQL Update, y ejecutar consultas mediante SPARQL Query.

La siguiente imagen, tiene como objetivo, mostrar gráficamente la interacción de los diferentes componentes del Servidor Fuseki. Por un lado se encuentran los equipos que utilizan los servicios provistos por el Fuseki*: SPARQL Query, Update.* Por otro, se encuentra el Servidor, que guarda los grafos de las ontologías mediante el componente Jena TDB, inicialmente se carga ya la ontología Matriz: RIMV3OWL_CDA.

El agregado de nuevos documentos, es lo que se pretende mostrar con las carpetas de CDA1, CDAn..

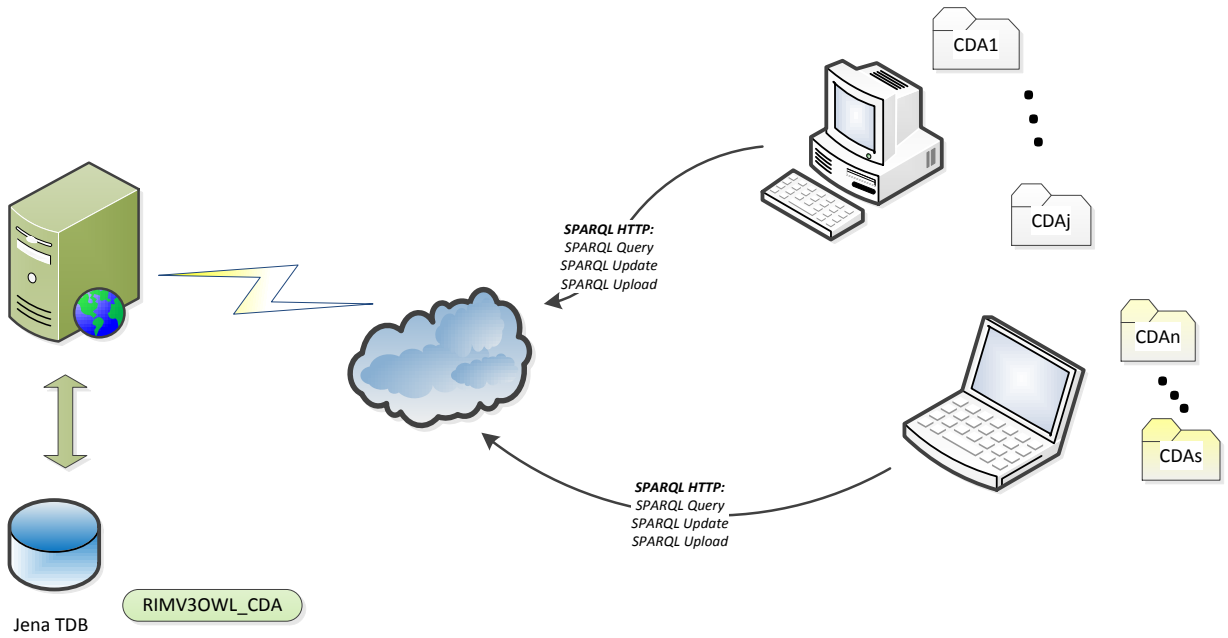


ILUSTRACIÓN 26 - ESQUEMA DEL USO DEL SERVIDOR FUSEKI

Finalmente, se muestra el potencial de uso de estas herramientas, mediante la ***generación de consultas SPARQL*** que permitan obtener información del Modelo Ontológico Resultante y de esta forma, apoyar la toma de decisiones, la detección de errores de registro de la información, y la posible conexión con otras bases semánticas. Justamente, la riqueza del trabajo radica en este potencial: a través de la correcta formulación de consultas SPARQL, es posible extraer datos específicos de la semántica de los documentos CDA.

5.1 Sistema "Prot – PGSemCDA"

El prototipo construido consiste en un sistema que muestra las potencialidades del trabajo con la ontología generada y la transformación de un CDA a tripletas RDF/OWL que se integran a la ontología matriz. El propósito del mismo, es dar cuenta de lo investigado en el contexto del Proyecto de Grado y brindar una herramienta que permita, de forma práctica, utilizar estos conocimientos y recursos generados.

Utilizamos la ontología ya desarrollada con la herramienta *Protégé*: ***RIMV3OWL_CDA*** *(*llamada *Ontología Matríz)*, la cual extrae la semántica que debe contener un documento CDA, según su modelo de información RIM. La misma, fue explicada en detalle en el capítulo 4.

Para instanciar un documento CDA concreto, generamos la transformación ***CDAtoOWL.xsl***. Ésta transformación convierte un documento CDA bien formado, en un documento OWL serializado bajo formato *Turtle*.

Ahora bien, nos proponemos integrar estos dos recursos en una solución informática que permita obtener de forma automática, la semántica de cualquier documento CDA válido, y razonar sobre él.

5.1.1 Alcance

La herramienta generada, que llamaremos **Prot-PgSemCDA**, cumple los siguientes objetivos:

- Integrar instancias de documento CDA a la Ontología Matriz diseñada en el contexto de este proyecto de grado.
- Persistir la información semántica generada. Esto es, la extracción de la semántica de un documento CDA debe quedar disponible para posteriores consultas o usos.
- Razonar y/o consultar sobre la información semántica disponible.
- Agregar nuevo conocimiento a la información semántica.

5.1.2 Interfaces del Sistema

Nuestro sistema en principio no se encuentra relacionado con ningún otro sistema de información externo, por lo cual no aplican interfaces.

5.1.3 Interfaces del Usuario

El sistema cuenta con una interfaz gráfica. Consiste en una ventana de aplicación, desde la cual se brinda el acceso al siguiente menú de opciones:

a. Configuración: Permite cambiar las variables de conexión a la base de datos, así como los directorios de sistema predefinidos. Esta información es almacenada en un archivo de propiedades del sistema.
   1) Ubicación por defecto de los CDAs a agregar.
   2) Directorio donde se guarda la colección de consultas disponibles.
   3) Directorio donde se guardan los resultados de la ejecución de consultas

b. Ontología: Este menú da acceso a las funcionalidades que permiten crear la base matriz a partir de la ontología matriz, de agregar CDAs a esta base, así como de modificar la información semántica recopilada a través de sentencias SPARQL Update.

c. **Consultas**: Permite agregar y eliminar consultas a la colección de consultas disponibles, y posibilita la validación y ejecución de dichas consultas contra el repositorio semántico.

En la imagen siguiente, se muestra como se accede a cada una de estas opciones:

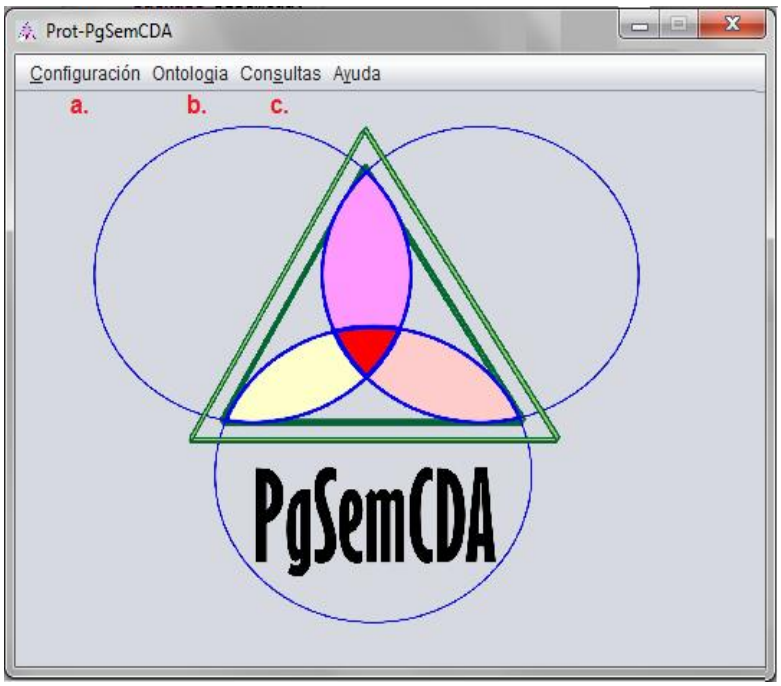


ILUSTRACIÓN 27- VENTANA PRINCIPAL DE LA APLICACIÓN PROT-PGSEMCDA

A modo de ejemplo, se presentan otras dos pantallas relevantes de la aplicación, la que permite agregar nuevos documentos CDA y luego donde se encuentra la ejecución de las consultas SPARQL. Por más detalles: Anexo "Manual de Usuario de Prototipo ***PgSemCDA***", donde se presenta el manual de usuario.

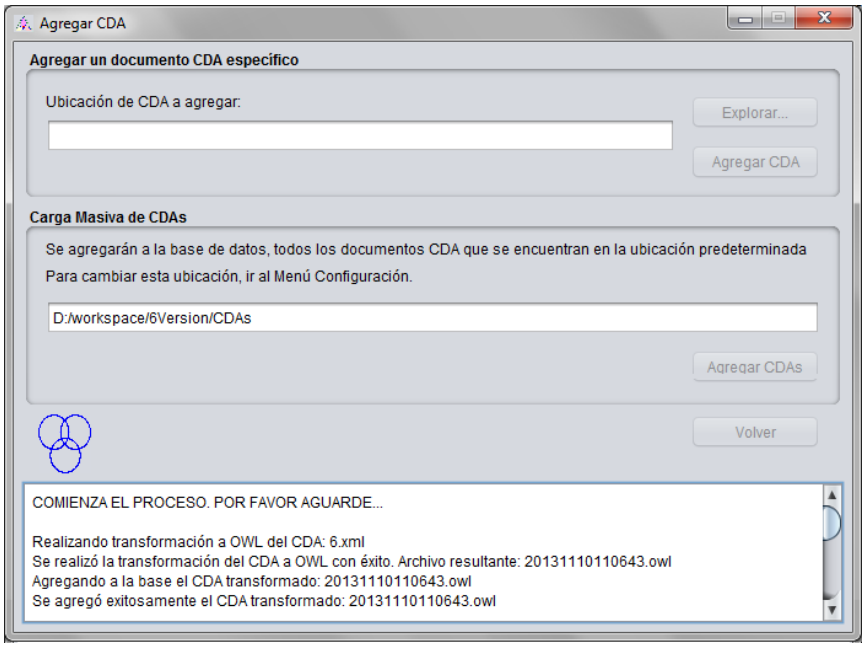


ILUSTRACIÓN 28 - FUNCIONALIDAD DEL SISTEMA QUE PERMITE AGREGAR NUEVAS INSTANCIAS DE DOCUMENTOS

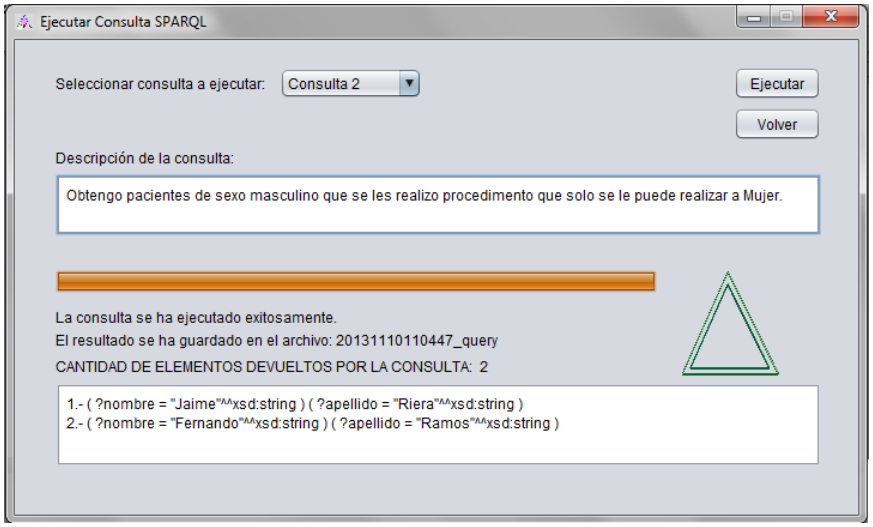


ILUSTRACIÓN 29 - FUNCIONALIDAD DEL SISTEMA, QUE PERMITE EJECUTAR CONSULTAS SPARQL Y VISUALIZAR EL RESULTADO OBTENIDO

5.1.4 Funciones del producto

a. **Crear base matriz** - El sistema crea una nueva base de datos, eliminando alguna versión anterior de existir, que permite persistir en forma de base de datos relacional, la información semántica contenida en la ontología matriz (RIMV3OWL_CDA).

b. **Agregar CDA** - El sistema pide la ruta a uno o varios documentos CDA bien formados, les aplica la transformación CDAtoOWL.xsl, guarda su resultado en un archivo interno del sistema (único por cada CDA), y a su vez persiste esta información en la base de datos.

c. **Modificar Datos** – Es posible modificar la información semántica recopilada, mediante una sentencia SPARQL/Update. Esto permite agregar, modificar o borrar tripletas de la base.

d. **Consultar** - Se puede seleccionar una consulta específica para ejecutarse sobre la información semántica recopilada. El sistema ejecuta dicha consulta y guarda su resultado en un identificado de la siguiente forma: xxxxyyzzhhmmss_query.txt, siendo: xxxx año - yy mes - zz día - hh hora - mm minutos - ss segundos. A su vez muestra el resultado por pantalla.

e. **Agregar Consulta** - El sistema permite al usuario, generar una nueva consulta semántica que, luego de ser verificada sintácticamente, queda habilitada para ser ejecutada sobre la información semántica disponible.

f. **Eliminar Consulta** – La inversa de la función Agregar, quita una determinada consulta de la colección de consultas disponibles.

5.1.5 Herramientas Utilizadas

Es de interés describir cuales fueron las diferentes las herramientas, librerías y frameworks que aportaron al desarrollo del prototipo *Prot-PgSemCDA* y especificar cada elemento utilizado de las mismas.

5.1.5.1 IDE Eclipse

Eclipse es un entorno de desarrollo integrado (IDE -*Integrated Development Environment*-) de código abierto y multiplataforma. Si bien soporta varios lenguajes de programación, con el que mejor se integra es con el lenguaje Java. Eclipse brinda el entorno de desarrollo solamente, además es necesario para el caso del lenguaje Java, disponer del Java Development Kit (JDK), para lograr compilar y ejecutar las aplicaciones desarrolladas.

El IDE de Eclipse emplea módulos de extensiones (*plugins*) que pueden ser integrados para proporcionar funcionalidades específicas o extensiones del ambiente. Esto es una ventaja importante frente a los entornos monolíticos, que incluyen todas las funcionalidades, sean necesarias o no.

5.1.5.2 Saxon 9

Saxon es un procesador transformaciones XSLT, que cuenta con versiones de código abierto y versiones comerciales. La línea de desarrollo actual, Saxon 9, implementa las especificaciones de XSLT 2.0. Esta es la razón fundamental por la que elegimos este procesador para ejecutar nuestra transformación XSLT, la cual contiene una evaluación de expresión regular que solamente es válida para la versión 2.0 del lenguaje XSLT. Esta expresión regular, permite convertir satisfactoriamente formatos de fecha escritos en los documentos CDA, a un formato válido para el tipo de datos datetime.

5.1.5.3 Jena-SDB-1.3.5

Jena es un framework desarrollado en Java por HP Labs, de código abierto, para construir aplicaciones de Web Semántica.

Esta plataforma permite analizar y manipular modelos basados en grafos RDF, así como la incorporación de un razonador que sea capaz de inferir relaciones entre sus elementos que previamente no se hayan hecho explícitas.

Jena contiene los siguientes componentes [45]:

- **a.** API para trabajar con grafos RDF programando en código Java: generar, leer y cargar grafos, serializarlos en diferentes formatos (RDF/XML, Turtle, N-Triples), navegar los grafos.
- **b.** API para trabajar con ontologías en distintos lenguajes (RDF Schema, OWL)
- **c.** Capacidad de procesamiento de consultas SPARQL
- **d.** Motores de inferencia y conectores a motores externos
- **e.** Mecanismos de almacenamiento para RDF

Los modelos consisten en un conjunto de sentencias RDF, y proveen un mecanismo para el almacenamiento y el acceso a la información semántica. Jena es un sistema bastante complejo en cuanto a la diversidad de modelos y las formas de construirlos. Describimos a continuación las funcionalidades más relevantes dentro de los modelos.

1. **Creación de un modelo simple**: La forma más simple de crear un modelo es mediante el método *createDefaultModel()* del constructor *ModelFactory*. Esto genera un modelo RDF que se almacena en memoria principal y que no realiza inferencias.

2. **Creación de un modelo de base de datos**: Jena ofrece dos componentes que permiten almacenar tripletas RDF en bases de datos y recuperarlas mediante consultas SPARQL [46]:

    1) **TDB**: Es un motor de almacenamiento que persiste las tripletas RDF como un grupo de datos en un solo directorio, en un sistema de archivos, llamado *Dataset*.

    2) **SDB**: Es un motor de almacenamiento que funciona como adaptador de un motor de base de datos relacional. Soporta los siguientes motores, a partir de la versión indicada:

    - PostgreSQL v8
    - MySQL 5.0.22
    - Apache Derby 10.2
    - Oracle 10gR2
    - Microsoft SQL Server 2005
    - IBM DB2
    - HSQLDB 1.8
    - H2 1.0.73

Para crear un repositorio persistente en una base de datos relacional utilizando Jena, se necesita crear un modelo mediante el componente *SDBFactory*, asociado a un objeto Store encargado de realizar la conexión con un esquema de una base de datos. Luego de establecida dicha relación, los cambios que se efectúen sobre el modelo se consolidan automáticamente en la base de datos.

3. **Creación de modelos de inferencia**: Una característica importante de Jena, es que soporta varios tipos de inferencia sobre modelos basados en RDF. Los modelos de inferencia son construidos aplicando razonadores a un modelo de base. Las sentencias deducidas por el razonador a partir del modelo de base aparecen en el modelo inferido junto a las propias sentencias del modelo de base.

4. **Creación de modelos ontológicos**: Un modelo ontológico consiste en un modelo que presenta un grafo RDF como una ontología. Jena soporta ontologías RDFS y OWL a través de perfiles ontológicos. Para generar el modelo ontológico, se establece una relación entre un modelo de base y un perfil ontológico. El modelo de base contiene las sentencias RDF, mientras que el perfil ontológico permite establecer la

funcionalidad que tendrá el modelo ontológico. Por ejemplo, el perfil *"OntModelSpec.OWL_DL_MEM"* dice que el modelo se corresponde con la descripción de una ontología con la especificación OWL DL y que es almacenado en memoria, pero que no realiza ningún razonamiento sobre el modelo. Por otra parte, el perfil *"PelletReasonerFactory.THE_SPEC"* establece que se utilizará el razonador Pellet para efectuar las inferencias sobre el modelo ontológico.

Utilizamos el término "**inferencia**" para referirnos al proceso abstracto de derivar información adicional, y el término "**razonador**" para referir al objeto de código específico que realiza esta tarea.

Dentro de la distribución de Jena, se incluyen los siguientes razonadores predefinidos:

1. *Razonador transitivo:* Provee soporte para almacenar y consultar jerarquías de clases y propiedades.
2. *Razonador RDFS:* Implementa un subconjunto configurable de la funcionalidad ofrecida por RDFS.
3. *Razonadores OWL, OWL Mini, OWL Micro:* Implementación incompleta de OWL Lite.
4. *Razonador genérico basado en reglas:* utiliza algoritmos de tipo forward y backward chaining.

Además, Jena se ha diseñado pensando en favorecer la inclusión de nuevos razonadores externos, como por ejemplo Pellet. [47]

El componente de Jena que interpreta las consultas SPARQL es ARQ. Para cada consulta que se ejecuta, se realizan las siguientes acciones:

- *Convierte la cadena de texto de contiene la consulta a un objeto Query.*
- *Traduce el objeto Query a una expresión de álgebra SPARQL.*
- *Optimiza esta expresión.*
- *Determina qué plan de acción ejecuta.*
- *Ejecuta el plan elegido.*

Una funcionalidad que no se utilizó de Jena, es la posibilidad de realizar consultas SPARQL a repositorios remotos, vimos que también es soportada por el Framework.

5.1.5.4 PostgreSQL

PostgreSQL es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor. Comenzó a desarrollarse hace más de 16 años, basándose en las características de estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. En el prototipo realizado, se decidió utilizar el almacenamiento en este sistema [48].

5.1.5.5 Pellet-2.3.0

Pellet, es una propuesta de Clark & Parsia, para el razonamiento e inferencia sobre ontologías OWL DL, disponible para su descarga en http://clarkparsia.com/pellet/download. Este razonador implementa las mejores técnicas de optimización, lo cual hace que su desempeño sea bueno.

Su principal función consiste en la inferencia de conocimiento explícito a través de las meta propiedades definidas sobre una ontología.

5.1.6 Arquitectura

La arquitectura del prototipo *Prot-PgSemCda,* se puede esquematizar en tres niveles interdependientes. Estos niveles se inician en uno o varios recursos, los cuales se procesan, y alimentan o extraen información de la base de datos.

5.1.6.1 Nivel de la Ontología Matriz

El punto de partida consiste en el recurso **RIMV3OWL_CDA**, es decir, la ontología matriz generada y serializada en XML. Se levanta la información de este recurso, y con ella se inicializa la base de datos: se cargan las tripletas en las tablas correspondientes. Para realizar esa manipulación de los datos, se utiliza Jena SDB embebido en código Java.

5.1.6.2 Nivel de Instancias

Este nivel se alimenta de dos recursos, uno de ellos, es la transformación XSLT desarrollada para la conversión de CDAs a tripletas en formato Turtle, denominada **CDAtoOWL.xsl**. El otro recurso, consiste en cualquier documento CDA bien formado. La transformación XSLT es aplicada al documento CDA, utilizando el componente Saxon como procesador XSLT.

El resultado de dicha transformación, es un documento CDA serializado en Turtle, el cual denominamos "**Instancia CDA**". Este documento es el que se integra a la información consignada en la base de datos. Se aporta conocimiento al repositorio semántico, por dos vías, la primera de ellas consiste en las tripletas de la instancia CDA; la segunda, mediante las inferencias que se aplican a la nueva versión de la base de conocimiento, con la instancia CDA integrada.

5.1.6.3 Nivel de Consultas

En este nivel, se parte de una cadena de texto que contiene una consulta SPARQL sintácticamente correcta. Mediante los componentes de Jena SDB que posibilitan la ejecución de este tipo de consultas, se extrae la información requerida de la base de datos.

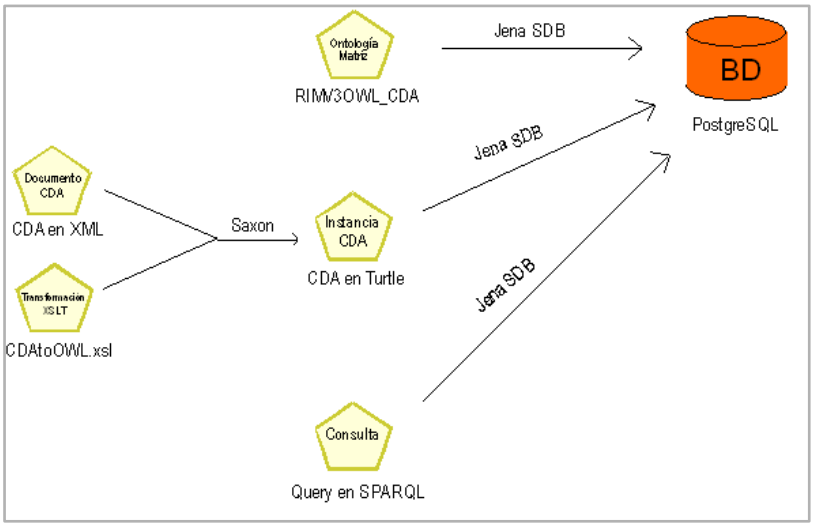


ILUSTRACIÓN 30 - COMPONENTES DE LA ARQUITECTURA DEL PROTOTIPO

5.1.7 Principales aspectos de la implementación

5.1.7.1 Definición de modelos

Cuando se inicia la aplicación, se realizan las siguientes acciones:

1) Se generan los objetos **store_ont** y **store_inf**, que permiten establecer la conexión con la base de datos relacional. Si la base no existe, la crea.

2) Se crean los siguientes modelos:
   - Modelos de base de datos **model_schema** y **model_inf**, asociados a los objetos **store_ont** y **store_inf** respectivamente.
   - Modelo ontológico **ont_schema**, asociado al modelo de base de datos **model_schema,** y al perfil ontológico OWL_MEM. Utilizado para crear la base matriz y para agregar inferencias a la misma. Cualquier cambio que se realice sobre este modelo, impacta en la base de datos, debido a que su modelo de base es model_schema y éste es un modelo de base de datos.
   - Modelo de inferencia **ont_inf**, cuyo modelo de base es model_inf, y su perfil ontológico asocia al razonador Pellet. A partir de este modelo se realizará la carga de instancias de documentos CDA, se aplicarán las operaciones SPARQL Update, y se ejecutarán las consultas SPARQL Query. De la misma forma que ont_schema, cualquier cambio realizado en este modelo impactará en la base de datos.
   - Modelo simple **model_instancias**, modelo auxiliar que se utilizará para agregar instancias a la ontología.

Los modelos se mantienen durante toda la ejecución de la aplicación, se eliminan al salir de la misma.

Esquema de los modelos creados al inicio de la aplicación:

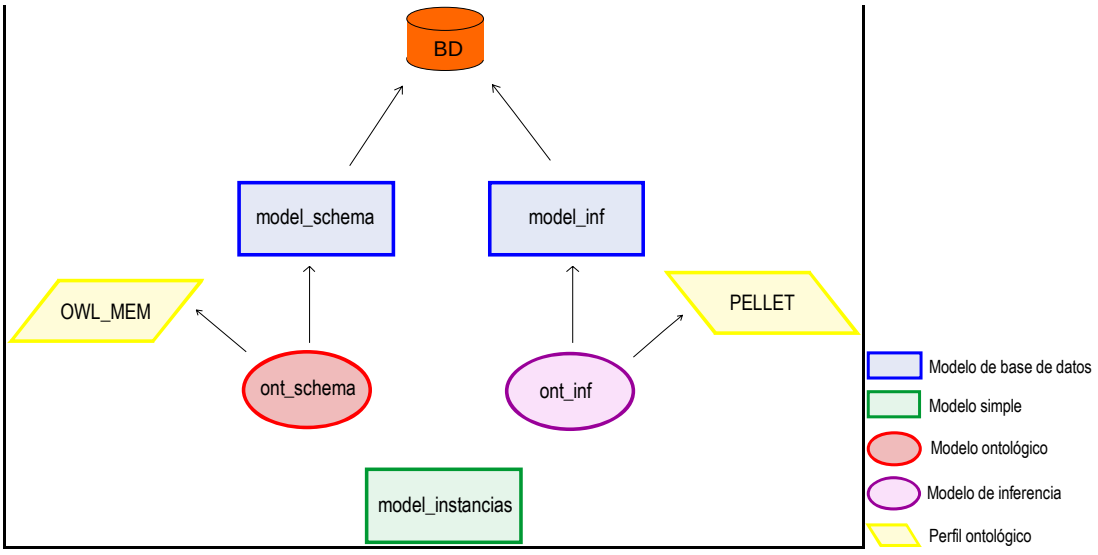


ILUSTRACIÓN 31 - ESQUEMA DE MODELOS JENA CREADOS

5.1.7.2 Uso de los modelos

Funcionalidad Crear base matriz: Se elimina toda la información existente en la base, y seguidamente se hace una lectura sobre el archivo que contiene la ontología matriz, volcada al modelo *ont_schema.*

ILUSTRACIÓN 32 - USO DE MODELOS EN FUNCIONALIDAD CREAR BASE MATRIZ

Funcionalidad Agregar CDA: Se vacía el modelo *model_instancias*, y se realiza una lectura del archivo que contiene la instancia correspondiente a un CDA transformado, volcado sobre el modelo *model_instancias*. Luego, se agrega el contenido de *model_instancias* al modelo *ont_inf*.

Cuando se realiza la carga masiva de CDAs, se repite esta operación por cada CDA individual que se esté cargando.

Una vez que finaliza la carga de CDAs, se agregan las inferencias al modelo ontológico, es decir, se agregan explícitamente todas las sentencias que se obtienen de aplicar el razonador sobre la ontología con la nueva carga de información. Para realizar la carga de inferencias, se generan dos nuevos modelos: extractor e inferencias.

El modelo extractor se crea mediante el componente *ModelExtractor* aplicado sobre el modelo de inferencia *ont_inf*. Este componente provee una forma conveniente de extraer un subconjunto específico de inferencias. El modelo inferencias, se obtiene de aplicar el método *extractModel* sobre el modelo extractor, permitiendo así extraer el conjunto predeterminado de inferencias [49].

Luego, el contenido del modelo inferencias es agregado al modelo ontológico.

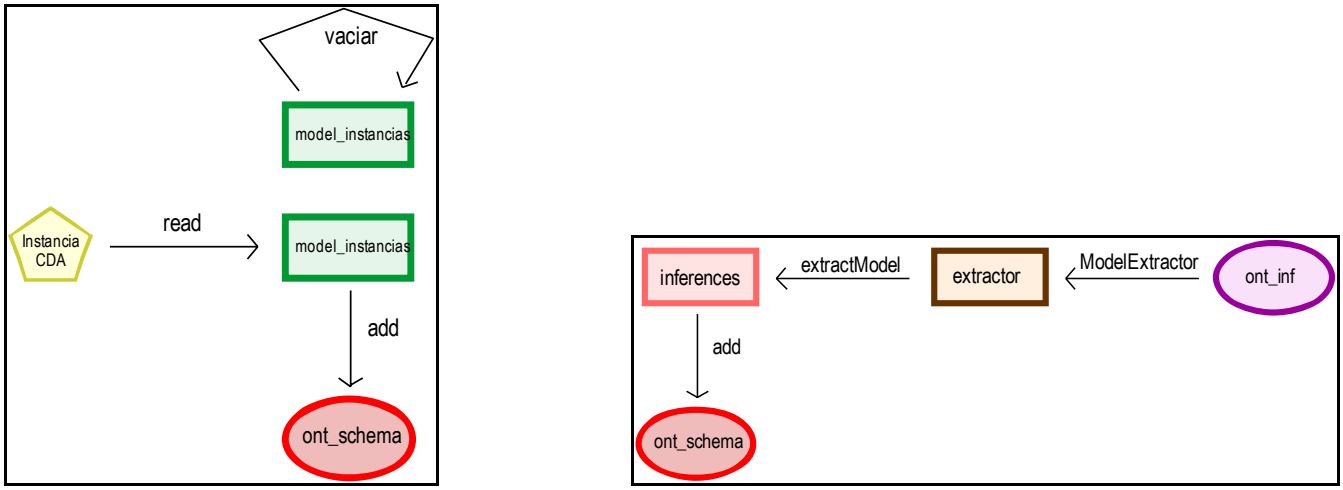


ILUSTRACIÓN 33 - USO DE MODELOS EN FUNCIONALIDAD AGREGADO DE INSTANCIAS

Funcionalidad Ejecutar Consulta: Sobre el modelo de inferencia ont_inf se realiza la ejecución de consultas, mediante el componente *QueryExecutionFactory*.

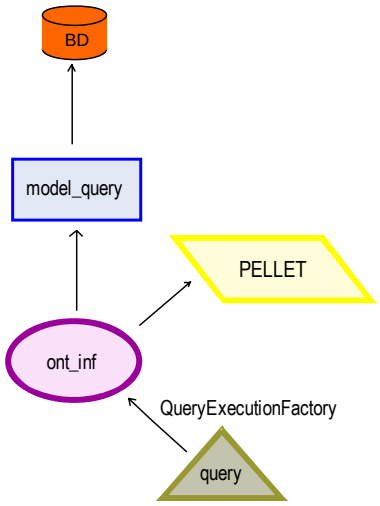


ILUSTRACIÓN 34 - USO DE MODELOS EN FUNCIONALIDAD EJECUTAR CONSULTA

5.1.8 Dificultades Encontradas

5.1.8.1 Documentación y estado del componente SDB

Debemos destacar que la documentación del componente SDB de Jena, para ser utilizado embebido en código Java en el desarrollo de una aplicación, nos resultó bastante escasa y poco profunda, cuando iniciamos la investigación de Jena. La documentación más abundante al respecto, sobre todo en lo referente a ejemplos de uso, fue información obsoleta respecto al componente que se utilizaba para conectar a la base de datos. A la fecha, esta situación ha cambiado y si bien la documentación disponible hace referencia al componente actualizado, a partir de Junio de 2013 SDB queda solamente en estado de mantenimiento, o sea que no se seguirá desarrollando. Se recomienda como contrapartida el uso del componente TDB, argumentando que es más rápido, más escalable y tiene mejor soporte que SDB. [50]

5.1.8.2 Procesador XSLT

Al iniciar el desarrollo de la aplicación, comenzamos utilizando el procesador XSLT Xalan, incorporado por defecto al entorno de desarrollo. Cuando debimos hacer frente a la adecuada conversión del formato de una fecha, desde un documento CDA al tipo de datos requerido para ser integrado en el modelo Jena, la solución encontrada para realizar esta conversión fue implementar el uso de una expresión regular, válida en la versión 2.0 del lenguaje XSLT. En este punto, debimos abandonar el uso de Xalan, y en su lugar incorporar el procesador XSLT Saxon, el único que encontramos soporta XSLT 2.0.

5.1.8.3 Elección del Razonador

En las primeras versiones de la aplicación, probamos el razonador OWL que se provee en la distribución Jena. Este razonador, resulta ser bueno en cuanto a performance asociado a un modelo que se mantiene en memoria principal, pero no era nada bueno su desempeño, cuando es asociado a un modelo de base de datos. Las consultas ejecutadas con este razonador, tardan promedio unos 10 minutos en ser resueltas. Razón por la cual, decidimos la incorporación del razonador Pellet, del cual encontramos buena documentación de guía para su uso con Jena, y cuya performance es satisfactoria.

5.1.8.4 Decisiones respecto al uso de modelos

Debido a la gran complejidad de Jena respecto a la variedad de modelos que ofrece, es que dedicamos buena parte del desarrollo de la aplicación, a la investigación de este aspecto, incluido el ensayo y error en diversas pruebas respecto al modelado de la información y su uso. Comprobamos que, para ejecutar consultas SPARQL, necesariamente se requería un modelo de inferencia asociado al razonador Pellet. Por otra parte, cualquier operación que se realice sobre el modelo de inferencia, que conlleve una modificación de la información (creación de base matriz, agregado de instancias CDA o de inferencias), implica un tiempo extenso de procesamiento. Razón por la cual se mantiene también el modelo ont_schema, que facilita la creación de la base matriz y el agregado de inferencias. De todas maneras, la inclusión de instancias CDA a la base ontológica, se realiza a través del modelo ont_inf, debido a que no encontramos la forma de que dicho modelo contemple la información que se agrega a la base "por fuera" de él. Dicho de otra forma, si las instancias CDA se agregan mediante el modelo ont_schema, las consultas SPARQL ejecutadas no devuelven la información procedente de

las instancias agregadas. Una solución posible pudiera haber sido generar un modelo de inferencia aparte, cada vez que se fuera a ejecutar una nueva consulta SPARQL. Lo cual ciertamente le resta mucha eficiencia al proceso de ejecución de consultas, dado que a la ejecución en sí misma, debe sumársele el tiempo de generación del modelo. Por estas razones, decidimos llegar a una solución de compromiso que favorezca la performance del nivel de consultas, en detrimento de imputar un mayor tiempo de procesamiento al nivel de instancias.

5.1.8.5 Pruebas y limitaciones de memoria

Durante el desarrollo de la aplicación, se fueron realizando las pruebas unitarias de cada funcionalidad, y las pruebas de integración correspondientes. Llegamos a obtener un resultado satisfactorio del funcionamiento de la aplicación en su integridad, con pruebas sobre pocos documentos CDA.

Para efectuar una prueba con mayor cantidad de instancias, recopilamos una serie de documentos CDA de varias instituciones de diferentes países, descargada de: http://www.ringholm.com/download/CDA_R2_examples.zip.

De allí tomamos 172 documentos CDA, y los agregamos a la base matriz, con la carga de inferencias correspondiente. Al querer incluir otro conjunto de documentos CDA, cuando la aplicación iba a proceder a la carga de inferencias, ocurre un error de memoria heap insuficiente (java.lang.OutOfMemoryError: Java heap space). La memoria heap es el espacio de memoria donde se almacenan los objetos y arreglos. Si los objetos que están siendo creados exceden el tamaño de la memoria heap, entonces ocurre un error como el mencionado. De esta forma, debimos aumentar el tamaño por defecto de esta memoria, de 128M a 2Gb. Esta limitación es dependiente del equipo donde se ejecute la aplicación, y de la cantidad de datos que se vayan sumando a la base ontológica.

5.2 Prototipo Fuseki

Fuseki es un servidor SPARQL, que provee la posibilidad de cargar una ontología e instancias, y aplicar a esta base de conocimiento, tanto consultas de selección como modificación de datos mediante sentencias SPARQL Update.

Mediante esta herramienta, es posible leer y procesar información semántica de diversos CDAs de forma abierta y sin dependencia de aplicaciones específicas. Basta con correr el servidor Fuseki, levantando la ontología matriz *RIMV3OWL_CDA.owl.* A su vez, transformando los CDAs que se requieran mediante cualquier procesador XSLT, se pueden ir cargando todas estas instancias de CDAs. Y sobre esta base de conocimiento, se puede aplicar consultas y operaciones SPARQL.

5.2.1 Comentarios Generales.

1. **Componente TDB**

El prototipo que utiliza Fuseki, como servidor SPARQL, se decidió que guardara la información mediante el componente TDB, para contrarrestar lo antes realizado con el prototipo **PgSemCDA** desarrollado en Java al utilizar SDB.

Cuando se utiliza TDB, la tripletas pertenecientes a las ontologías se persisten como archivos en disco, guardando así los grafos correspondientes.

2. **Razonadores.**

En las pruebas realizadas, encontramos que el razonador ***Pellet***, fue el que respondió mejor al momento de realizar todas las inferencias. Para poder configurar el mismo, se modifica el archivo del que toma la configuración al momento de levantar el servidor, realizando un ajuste de las variables de entorno del sistema (classpath). El tiempo de respuesta al momento de realizar las consultas es aceptable.

Otros razonadores probados, no realizaban las inferencias que necesitábamos, por ejemplo; *RDFSExptRuleReasoner, OWLMicroFBRuleReasoner, OWLMiniFBRuleReasoner*, son algunos de los probados, con estos no obteníamos las clasificaciones correspondientes de las tripletas.

5.2.2 Funcionalidades del Producto

El servidor fuseki, puede descargarse de http://jena.apache.org/download/index.html. Luego de descargarlo, debimos modificar los archivos de configuración, archivos llamados de especificación assembler, que contienen una descripción RDF de cómo construir un modelo y sus recursos asociados: como razonadores, prefijos y contenidos iniciales. El vocabulario Assembler está dado en el esquema Assembler, disponible en: REF http://jena.apache.org/documentation/assembler/assembler.ttl

Adecuaciones de configuración para RIM_OWL:

En primer lugar, declaramos los prefijos que serán utilizados. Luego, definimos el **servicio**, configurando cuales serían las funcionalidades que se aceptan: ***query, update, upload.***

Posteriormente, se declara el grafo a utilizar, generado como un **modelo de inferencia**, que combina un **modelo de base de datos** asociado al componente **TDB** y un **modelo ontológico** donde se levanta la ontología matriz. A este modelo de inferencia, se le asocia el razonador **Pellet**.

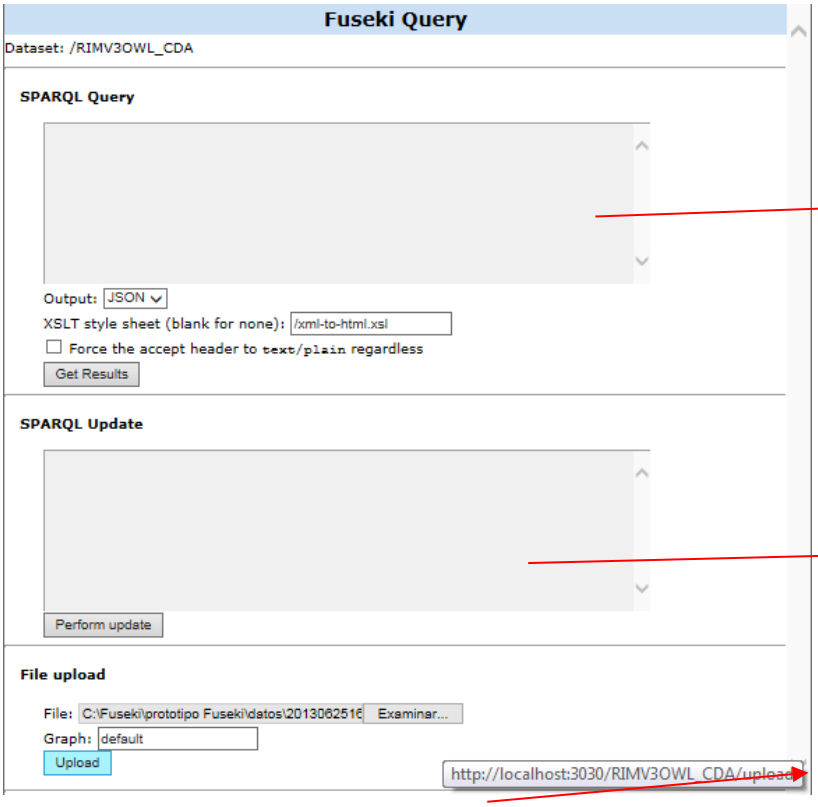
Se elabora entonces, un modelo en memoria que se replica en los archivos de persistencia generados por el TDB de Jena, que se ubican dentro de la carpeta donde se encuentran los contenidos. Cada vez que se realiza algún cambio sobre la base de conocimiento (agregar instancias, sentencias de update, delete o insert de SPARQL), esta información se agrega a dichos archivos.

Se crearon dos archivos de procesamiento por lotes, uno para Windows y otro para Unix, que configuran las variables de entorno y que inician el servidor tomando como archivo de configuración el anteriormente descrito.

Utilidades de Fuseki:

Una vez levantado el servidor Fuseki mediante la consola de comandos, se debe acceder a la dirección: ***http://localhost:3030/*** en un navegador Web. De esta forma, queda habilitado el ingreso al panel de control de Fuseki.

La página principal en la cual se puede interactuar con el servidor es la siguiente:



Permite agregar consultas en formato SPARQL, y devolver los resultados en diferentes formatos.

Permite realizar ejecuciones de update, agregando nueva información a la ontología.

Lugar donde se agregan nuevos archivos en formato ttl. Aquí agregamos las instancias de los Documentos CDA.

ILUSTRACIÓN 11- PANEL DE CONTROL DE FUSEKI

a. Agregado de nuevos documentos:

El agregado de nuevas instancias, se realiza mediante la funcionalidad FileUpload, para esto se debe contar con el resultado de la aplicación de la transformación del documento CDA(XML) original. Luego de agregar un documento, se visualiza la cantidad de tripletas agregadas.

b. Consultas Sparql

En el SPARQL Query, se agregan los contenidos de las consultas. El resultante se visualiza en pantalla con las tripletas que cumplen los criterios establecidos.

c. Update o Creación de nuevas Clases/relaciones:

El servicio SPARQL Update, permite realizar operaciones de modificación sobre un modelo RDF. Provee las siguientes funciones: [51]

- **a.** Insertar nuevas tripletas a un grafo RDF
- **b.** Borrar tripletas de un grafo RDF
- **c.** Ejecutar un grupo de operaciones de modificación como una sola acción
- **d.** Crear un nuevo grafo RDF
- **e.** Borrar un grafo RDF

En particular, al insertar nuevas tripletas a un grafo, estamos agregando conocimiento al repositorio semántico. Entendemos por nuevo conocimiento a las nuevas clases o restricciones que pueden ser agregadas en etapas posteriores. En la complejidad de cómo se generen estas nuevas clases, estará el beneficio de esta utilidad.

Por ejemplo, agregaremos mediante una operación SPARQL Update, algunas tripletas al modelo:

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dctype: <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

INSERT DATA {
        dctype:acto1        rdf:type       dctype:Act .
        dctype:actoagreg1   rdf:type       dctype:ActNuevo .
        dctype:ActNuevo     rdfs:subClassOf  dctype:Act .
}
```

También se permite agregar nuevos conceptos o relaciones de equivalencia entre clases, mediante **File Upload.** Un ejemplo de esto, se muestra en la siguiente sección de resultados.

Por más detalle de las funcionalidades, ver Anexo: *"Descripción de Funcionalidades de Fuseki."*.

5.3 Resultados

Con el objetivo de mostrar el potencial de los resultados que se pueden obtener mediante consultas SPARQL, se realizaron una serie de pruebas sobre los documentos existentes en la Ontología Resultante: *Cap 4, secc 4.1*.

Se pretende que las pruebas sean diversas y con diferentes fines. Las consultas se probaron tanto en el prototipo ***PGSemCDA***, como ejecutando las mismas sobre el servidor ***Fuseki***. Obteniendo los mismos resultados en ambos casos.

Los casos buscan información contenida en cuatro tipos de documentos CDA: *Resultado de Laboratorio, Descripción Operatoria, Egreso Hospitalario, Consulta Ambulatoria*. Para las pruebas referentes a agregado de nuevo conocimiento, se consideran todos los tipos de documentos que existan en el repositorio semántico.

El primer caso - ***Caso 1***- , busca detectar niñas con Anemia (tomando el valor de hemoglobina dentro de su análisis de Laboratorio); el segundo y tercer caso - ***Caso 2 – y - Caso 3***- examina los documentos de descripciones operatorias, uno en el que se busca errores: pacientes de sexo masculino que se le realizo un procedimiento que se le debe indicar a una mujer. El otro, permite obtener los anestesistas actuantes en los procedimientos quirúrgicos realizados a niños menores de 2 años y que la duración supera las 3hs.

El último caso - ***Caso 4*** -, une los documentos de Egreso y Consulta ambulatoria de un paciente, en los que la consulta ambulatoria, detecto un paciente diabético y luego tuvo un episodio de internación por complicaciones de la enfermedad.

Cada caso, muestra un aspecto diferente, de la información contenida en los mismos.

Otra prueba de concepto realizada, fue agregar nuevas clases ontológicas que se realicen equivalencias con algunas de las ya existentes, con restricciones sobre los valores de las propiedades.

Para este caso - ***Caso 5 -***, se plasmó en la creación de una clase, llamada “Pacientes_Canarios” definida como los pacientes de *sexo masculino*, que viven en el departamento de *Canelones*. Luego sobre el grupo de pacientes, se listaron los tipos de documentos que tienen. Este última prueba, busca mostrar la extensibilidad que permite la ontología resultante, tanto vía el agregado de nuevas clases o relaciones, como consultando a otras fuentes de conocimiento externas (otras ontologías publicadas).

5.3.1 Detalle de las pruebas realizadas

Para cada caso, se describe la consulta, luego se presenta el formato **Sparql**, aclarando los diferentes filtros aplicados, y por último, se muestran los resultados obtenidos.

5.3.1.1 Caso 1 - Niñas con Anemia.

a. Descripción:

Busca los análisis de laboratorio realizados a niñas, donde en su resultado de hemograma, el valor de Hemoglobina fue menor que 20. Busca detectar niñas (paciente de sexo femenino de menos de 12 años) con Anemia, como paramentos de salida, devuelve el país, cuidad y calle de donde viven las pacientes.

b. Formato Consulta SPARQL.

En la siguiente sección se muestra la consulta SPARQL

```
SELECT    ?pais ?ciudad ?calle

WHERE  {

  //Enganche del doc
        ?docClin          RIMV3OWL_CDA:actRelationshipComponent        ?strBody.
        ?docClin          rdf:type                                     RIMV3OWL_CDA:ActClinicalDocument.
        ?strBody          RIMV3OWL_CDA:actRelationshipComponent        ?secc.
        ?secc             RIMV3OWL_CDA:actRelationshipComponent        ?obs.

   //Condiciones sobre la Hemoglobina.
        ?obs      rdf:type                    RIMV3OWL_CDA:ActObservation.
        ?obs      RIMV3OWL_CDA:act_code       ?cd.
        ?cd       RIMV3OWL_CDA:cd_code        "718-7"^^<http://www.w3.org/2001/XMLSchema#string>.
        ?obs      RIMV3OWL_CDA:act_value      ?valor.

        FILTER (?valor > "20").

  //Condiciones sobre el sexo y edad.
        ?docClin  RIMV3OWL_CDA:hasParticipation   ?rt.
        ?rt       RIMV3OWL_CDA:hasRole            ?patient.
        ?patient  RIMV3OWL_CDA:patient            ?entityRecdTarg.

        ?entityRecdTarg     RIMV3OWL_CDA:entity_administrativeGenderCode    RIMV3OWL_CDA:entityGenFem

        ?entityRecdTarg     RIMV3OWL_CDA:entity_birthTime          ?z.

        BIND (year(?z) AS ?an).
        BIND ((2013 - ?an) AS ?ed).
        FILTER (?ed < 12).

 //Obtener la dirección.
        ?patient  RIMV3OWL_CDA:role_addr                ?adPat.

        ?adPat    RIMV3OWL_CDA:adxp_country             ?pais;
                  RIMV3OWL_CDA:adxp_city                ?ciudad;
                  RIMV3OWL_CDA:adxp_streetAddressLine   ?calle.
}
```

5.3.1.2 Caso 2 - Documento de Descripción Operatoria con Errores.

a. Descripción:

Se obtienen los documentos de Descripción Operatoria, en los que el paciente es de sexo masculino y el procedimiento realizado solo corresponde a una mujer. Para el caso presentado el procedimiento es el de Cesárea, representado con la codificación SNOMED.

b. Formato Consulta SPARQL.

En la siguiente sección se muestra la consulta SPARQL

```
PREFIX RIMV3OWL_CDA: <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT    ?nombre ?apellido
WHERE {
          ?docClin   RIMV3OWL_CDA:documents                ?servEv.
          ?servEv    RIMV3OWL_CDA:act_code                 ?cd.
          ?cd        RIMV3OWL_CDA:cd_code                  "872637015"^^<http://www.w3.org/2001/XMLSchema#string>;
                     RIMV3OWL_CDA:cd_codeSystem            "2.16.840.1.113883.6.96"^^<http://www.w3.org/2001/XMLSchema#string>.

          ?docClin              RIMV3OWL_CDA:hasParticipation                    ?rt.
          ?rt                   RIMV3OWL_CDA:hasRole                           ?patient.
          ?patient              RIMV3OWL_CDA:patient                           ?entityRecdTarg.
          ?entityRecdTarg       RIMV3OWL_CDA:entity_name                       ?en;
                                RIMV3OWL_CDA:entity_administrativeGenderCode   RIMV3OWL_CDA:entityGenMasc.

          ?en        RIMV3OWL_CDA:enxp_given    ?nombre;
                     RIMV3OWL_CDA:enxp_family   ?apellido.
}
```

5.3.1.3 Caso 3 - Anestesistas en Procedimientos Quirúrgicos

a. Descripción:

Busca los anestesistas actuantes en intervenciones quirúrgicas realizadas a niños menores de 2 años, que superan las 3 horas de duración.

b. Formato Consulta SPARQL.

En la siguiente sección se muestra la consulta SPARQL

```
PREFIX RIMV3OWL_CDA: <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT          ?nombre ?apellido ?hs
WHERE {

// Datos de los participantes de Anestesia.
      ?docClin   RIMV3OWL_CDA:documents              ?servEv.
      ?servEv    RIMV3OWL_CDA:hasParticipation       ?anest.
      ?anest     RIMV3OWL_CDA:part_functionCode      RIMV3OWL_CDA:partFuncAnesthesist.

      ?anest     RIMV3OWL_CDA:hasRole                ?assEnt.
      ?assEnt    RIMV3OWL_CDA:assignedPerson         ?pers.
      ?pers      RIMV3OWL_CDA:entity_name            ?en.
      ?en        RIMV3OWL_CDA:enxp_given             ?nombre.
      ?en        RIMV3OWL_CDA:enxp_family            ?apellido.


// Duración del procedimiento mayor a 3hs.
                 ?servEv    RIMV3OWL_CDA:act_effectiveTime      ?efTime.
      ?efTime    RIMV3OWL_CDA:ivl_ts_low             ?low.
      ?efTime    RIMV3OWL_CDA: :ivl_ts_high          ?high.

      BIND ((?high - ?low) AS ?hs).
      FILTER (?hs  > "PT3H"^^xsd:duration).
```

```
// Filtro sobre la edad del niño.
      ?docClin   RIMV3OWL_CDA:hasParticipation                    ?rt.
      ?rt        RIMV3OWL_CDA:hasRole                 ?patient.
      ?patient   RIMV3OWL_CDA:patient                 ?entityRecdTarg.
                 ?entityRecdTarg      RIMV3OWL_CDA:entity_name   ?en.

      ?en        RIMV3OWL_CDA:enxp_given    ?nombre;
                 RIMV3OWL_CDA:enxp_family   ?apellido.

      ?entityRecdTarg RIMV3OWL_CDA:entity_birthTime ?z.
      BIND (year(?z) AS ?an).
      BIND ((2013 - ?an) AS ?ed).
      FILTER (?ed  < 2).
}
```

5.3.1.4 Caso 4 - Pacientes diabéticos con internaciones por descompensaciones.

a. Descripción:

Busca los pacientes que son diabéticos, y que a su vez han tenido algún episodio de internación por complicaciones producidas por la enfermedad. Se toman los documentos de Consulta Ambulatoria donde se registra que el paciente es diabético, junto a documentos de Egreso de pacientes que han tenido un episodio de internación debido a descompensaciones de la diabetes. El resultado es la intersección de ambos conjuntos.

b. Formato Consulta SPARQL.

En la siguiente sección se muestra la consulta SPARQL

```
PREFIX RIMV3OWL_CDA: <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT  ?ii_root ?ii_ext

WHERE {
  DOCUMENTO DE Consulta Ambulatoria

  //Tipo de Documento
     ?docClin        rdf:type                        RIMV3OWL_CDA: ActClinicalDocument.
     ?docClin        RIMV3OWL_CDA:act_code           ?doc_code.
     ?doc_code       RIMV3OWL_CDA:cd_code            " 34108-1" ^^<http://www.w3.org/2001/XMLSchema#string>.
     ?doc_code       RIMV3OWL_CDA:cd_codeSystem      "2.16.840.1.113883.6.1" ^^<http://www.w3.org/2001/XMLSchema#string>.

  //Enganche del doc
          ?docClin   RIMV3OWL_CDA:actRelationshipComponent      ?strBody.
          ?strBody   RIMV3OWL_CDA:actRelationshipComponent      ?secc.
          ?secc      RIMV3OWL_CDA:actRelationshipComponent      ?obs.

  //Condiciones sobre problemas de diabetes.
          ?obs            rdf:type                    RIMV3OWL_CDA:ActObservation.
          ?obs            RIMV3OWL_CDA:act_code       ?obs_code.
          ?obs_code       RIMV3OWL_CDA:cd_code        "250"^^<http://www.w3.org/2001/XMLSchema#string>.
          ?obs_code       RIMV3OWL_CDA:cd_codeSystem  "2.16.840.1.113883.6.2" ^^<http://www.w3.org/2001/XMLSchema#string>.

DOCUMENTO DE Egreso Hospitalario
```

```
//COND 2- Documentos Notas de Consultas – Control de paciente Diabetico.

  //Tipo de Documento

        ?docClin2       rdf:type                        RIMV3OWL_CDA: ActClinicalDocument.
        ?docClin2       RIMV3OWL_CDA:act_code           ?doc_code2.
        ?doc_code2      RIMV3OWL_CDA:cd_code            "18842-5" ^^<http://www.w3.org/2001/XMLSchema#string>.
        ?doc_code2      RIMV3OWL_CDA:cd_codeSystem      "2.16.840.1.113883.6.1" ^^<http://www.w3.org/2001/XMLSchema#string>.

  //Enganche del doc
        ?docClin2  RIMV3OWL_CDA:actRelationshipComponent    ?strBody1.
        ?strBody2  RIMV3OWL_CDA:actRelationshipComponent    ?secc2.
        ?secc2     RIMV3OWL_CDA:actRelationshipComponent    ?obs2.


  //Condiciones sobre problemas de diabetes.
    ?obs2          rdf:type                    RIMV3OWL_CDA:ActObservation.
    ?obs2          RIMV3OWL_CDA:act_code       ?obs_code2.
    ?obs_code2     RIMV3OWL_CDA:cd_code            "2889093016"^^<http://www.w3.org/2001/XMLSchema#string>.
    ?obs _code2    RIMV3OWL_CDA:cd_codeSystem      "2.16.840.1.113883.6.96 " ^^<http://www.w3.org/2001/XMLSchema#string>.

CONDICIONES SOBRE LOS 2 DOCS.

//COND 3- Sean del mismo paciente (el doc de egreso y nota de consulta) – Cond: Mismo root y extensión

        ?docClin   RIMV3OWL_CDA:hasParticipation            ?rt.
        ?rt        RIMV3OWL_CDA:hasRole             ?patient.
        ?patient   RIMV3OWL_CDA:patient             ?entityRecdTarg;
                   RIMV3OWL_CDA:role_id             ?id.

        ?id        rdf:type                         RIMV3OWL_CDA:II.
        ?id        RIMV3OWL_CDA:ii_root             ?ii_root.
        ?id        RIMV3OWL_CDA:ii_extension        ?ii_ext.

        ?docClin2  RIMV3OWL_CDA:hasParticipation    ?rt2.
        ?rt2       RIMV3OWL_CDA:hasRole             ?patient2.
        ?patient2  RIMV3OWL_CDA:patient             ?entityRecdTarg2;
                   RIMV3OWL_CDA:role_id             ?id2.

        ?id2       rdf:type                         RIMV3OWL_CDA:II.
        ?id2       RIMV3OWL_CDA:ii_root             ?ii_root2.
        ?id2       RIMV3OWL_CDA:ii_extension        ?ii_ext2.

        FILTER  ( sameTerm(?ii_ext,?ii_ext2) && sameTerm(?ii_root,?ii_root2) && !(?id = ?id2) )

// COND 4- La fecha de ambos doc, no difiere más de 1 mes.

        ?docClin   RIMV3OWL_CDA:act_effectiveTime   ?effTime.
        ?effTime   RIMV3OWL_CDA:ivl_ts_value  ?val1.

        ?docClin2  RIMV3OWL_CDA:act_effectiveTime   ?effTime2.
        ?effTime2  RIMV3OWL_CDA:ivl_ts_value  ?val2.

        BIND ((?val2 - ?val1) AS ?diff).

        FILTER (?diff  < "P50DT1H"^^xsd:duration).

        FILTER (?diff   >"-P1D"^^xsd:duration).

}
```

5.3.1.5 Caso 5 - Nuevo concepto: "Pacientes_Canarios" y consulta sobre documentos.

a. Descripción:

El enfoque de este caso el diferente, primero se agrega a la *Ontología Resultante*, la clase: **Pacientes_Canarios,** que busca englobar a todos los pacientes de sexo masculino que viven en Canelones, luego se realiza una consulta sobre los tipos de documentos que tienen este grupo de pacientes.

b. Agregado de Clase Equivalente.

Para esto, se agrega la nueva clase definida de la siguiente forma, mediante definiciones de equivalencias y restricciones sobre propiedades. Se utilizan dos clases auxiliares: **AD_Canelones** *(representa direcciones en la ciudad de Canelones)* y **Hombre** (*representa pacientes de sexo masculino*), que luego se usan como restricciones en la nueva clase: **Pacientes_Canarios.**

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix RIMV3OWL_CDA: <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#> .

:AD_Canelones rdf:type owl:Class ;
          owl:equivalentClass
                                [ rdf:type owl:Restriction ;
                                owl:onProperty :adxp_city ;
                        owl:hasValue "Canelones" ].

:Hombre  rdf:type owl:Class ;
         owl:equivalentClass
                                [ rdf:type owl:Restriction ;
                                owl:onProperty :entity_administrativeGenderCode ;
                                owl:hasValue :entityGenMasc] .


:Pacientes_Canarios rdf:type owl:Class ;
          owl:equivalentClass
                                 [ rdf:type owl:Restriction ;
                                   owl:onProperty : role_addr;
                          owl:allValuesFrom :AD_Canelones ]
                                ,
                                [ rdf:type owl:Restriction ;
                                  owl:onProperty :patient ;
                                  owl:allValuesFrom :Hombre ] .
```

Para los dos prototipos, el razonamiento se realiza al momento de incorporar la nueva clase, de esta forma, se clasifican a todas las instancias de pacientes según corresponda. Luego para futuras consultas, solo se pregunta por la clase agregada.

c. Consulta SPARQL sobre documentos.

La consulta correspondiente, pretende mostrar los códigos que representan el tipo de documento que tiene como paciente a los antes definidos.

```
PREFIX RIMV3OWL_CDA: <http://www.fing.edu.uy/inco/grupos/csi/ontologies/RIMV3OWL_CDA#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT  ?docClin

WHERE {
        ?docClin        RIMV3OWL_CDA:hasParticipation       ?rt.
        ?rt             RIMV3OWL_CDA:hasRole                ?patient.
        ?patient        rdf:type                            RIMV3OWL_CDA:Pacientes_Canarios.
        ?patient        RIMV3OWL_CDA:patient                ?entityRecdTarg.

}
```

CAPÍTULO 6

CONCLUSIÓN – TRABAJOS A FUTURO

6.1 Resumen

El presente trabajo de grado se planteó como objetivos generales, por una parte, el estudio del estado de situación -o bien, estado del arte- en relación a modelos de información semántica en la salud, y por otra la investigación y propuesta en relación a especificar condiciones semánticas aplicadas a documentos CDAs.

Se realizó una prospección amplia del estado del arte en el contexto sanitario, habiendo encontrado la línea de trabajo de *Helen Chen* [38] y sus predecesores. Su trabajo converge en la generación de una ontología completa para HL7, la cual tomamos como base para nuestro trabajo práctico, especializándola para CDA. Así, generamos un recurso denominado **RIMV3OWL_CDA.owl**, escrito en lenguaje OWL serializado bajo XML.

De esta forma, generamos una propuesta de ontología para representar documentos CDA, y por ende, para disponer de la información semántica de las historias clínicas electrónicas, entendidas como conjuntos de CDAs asociados a cada paciente.

En sí mismo, el desarrollo de esta ontología, que denominamos ***ontología matriz***, consiste en un desafío importante, que fue logrado en la primera instancia práctica de este proyecto de grado. Conjuntamente a la ontología matriz, desarrollamos un mecanismo de transformación, que permite traducir cualquier documento CDA, a lenguaje ontológico en formato *Turtle*. El resultado de esta traducción, que llamamos instancias CDA, aporta los objetos concretos de los conceptos definidos en la ontología matriz. Esta conversión, se efectúa mediante la aplicación de transformación XSLT denominada **CDAtoOWL.xsl**.

Una vez logrados estos dos recursos (ontología matriz y transformación), se realizaron pruebas prácticas de uso, mediante el desarrollo de una aplicación Java, que llamamos **Prot-PgSemCda**; y a través de la investigación y la utilización del servidor SPARQL denominado **Fuseki**. Con el *modelo ontológico resultante* compuesto por ontología matriz y las transformaciones de documentos CDA, se ejecutaron consultas del ámbito sanitario donde se pudieron evaluar los resultados empleando ambas herramientas.

6.2 Resultados Obtenidos

1. Una de las fortalezas de nuestro trabajo es haber logrado el **desarrollo de una ontología completa** para el modelo de información RIM **específico para CDA** (R-MIM). Este fue el insumo inicial para continuar con las etapas posteriores del proyecto. Acoplado a esta ontología, se desarrolló la **transformación XSLT** que sirve para generar las instancias de *CDAs* dentro del modelo ontológico resultante.
   Ambos recursos: *ontología matriz y transformación*, quedan disponibles para su reutilización en posibles proyectos futuros.

2. En **Prot-PgSemCda**, nos parece relevante destacar la integración del proyecto Jena dentro de un proyecto Java. La documentación disponible, en la etapa del proyecto en que desarrollamos la aplicación, contaba con ejemplos puntuales sobre cada aspecto de la manipulación de una ontología en **Jena embebido en código Java**. El desafío consistió en integrar todas las acciones posibles sobre una ontología, aplicando el potencial de Jena en lo referente a **modelos**, sumándole la complejidad de la **persistencia** de los datos.

3. En la utilización del Servidor SPARQL –**Fuseki**- logramos realizar de manera satisfactoria, las mismas funcionalidades que en el *Prot-PgSemCDA*, el agregado de instancias de documentos, la ejecución de consultas SPARQL, así como la modificación de datos ya persistidos. El razonamiento e inferencias se configuró con el razonador *Pellet* integrado.

4. En los prototipos, se investigaron e implementaron las dos formas de persistencia de datos propuestas por Jena. Estos son los componentes **SDB** -*Persistencia en Base de Datos Relacional*- y **TDB** -*Base de Datos Transaccional*-.

5. Se logró **ejecutar consultas** que aplican **condiciones semánticas** sobre la información registrada en documentos CDA, y de esta forma obtener **nueva información** que brinde soporte a la **toma de decisiones**.

6. Mediante los mecanismos de **introducción de nuevas clases, propiedades o restricciones** al modelo ontológico con operaciones **SPARQL Update**, se logra la **simplificación** de la escritura de **consultas SPARQL**. Una consulta SPARQL puede alcanzar un alto grado de complejidad, en función de la cantidad de filtros que deban aplicarse para extraer la información que se requiera, lo cual resulta directamente proporcional al tiempo de ejecución de dicha consulta, y al recurso de memoria principal necesario para completar la operación. La creación de clases que cumplan con alguna de las condiciones de dichos filtros, permite, mediante el razonamiento sobre el repositorio semántico, la **clasificación de individuos** dentro de estas nuevas clases, y la **persistencia de esta nueva información**. Esto facilita la ejecución de aquellas consultas con algún filtro aplicado a los individuos de las nuevas clases, dado que esa información queda disponible directamente, sin necesidad de efectuar razonamiento alguno.

7. Se utilizaron para las pruebas realizadas, **documentos CDA de más de 50 tipos diferentes**, entre ellos: asignación de *diagnóstico radiológico, admisión hospitalaria, resultados de exámenes preoperatorios, informe de alta hospitalaria, informe de clínica de rehabilitación, análisis de laboratorio, certificación médica, prescripciones de medicamentos*, entre otros. Corresponden a CDAs creados por instituciones de todo el mundo. Esto es una muestra de que la solución aplica a cualquier tipo de documento. Todos se transformaron en instancias OWL, se agregaron a la ontología resultante, y se pudieron ejecutar consultas sobre sus contenidos. Potencialmente, la solución propuesta en este proyecto de grado, es **capaz de recibir y procesar cualquier tipo de documento CDA**, sin necesidad de realizar ninguna modificación estructural de ningún componente.

8. En ambos prototipos, se realiza el **razonamiento** cuando se cargan las instancias de CDAs, de forma que la **información inferida** ya queda **persistida** y queda disponible a la hora de ejecutar consultas.

6.3 Conclusiones

Con los resultados obtenidos en el presente trabajo, demostramos que **es posible crear un repositorio semántico de documentos CDA**, bajo un modelo ontológico equivalente al modelo de información restringido para CDA (R-MIM).

Según los lineamientos del Proyecto Salud Uy, las Historias Clínicas Electrónicas se conformarán como un conjunto de documentos CDA, las cuales se reunirán en un **Banco Nacional de HCE**. El aporte de este proyecto de grado, en esta tendencia estratégica, consiste en mostrar que es posible y necesario, trabajar con los contenidos semánticos de dicho Banco. La **riqueza** de generar y gestionar una base nacional de HCE de estas características, consiste no solamente en persistir los documentos CDA en sí mismos, sino que, sobre todo, reside en **construir un repositorio semántico**, que ofrezca la **potencialidad de extraer** en cualquier momento, y de forma directa, **información tanto a nivel micro como a nivel macro del sistema de atención de Salud a nivel nacional**, que pueda utilizarse con fines epidemiológicos, estadísticos, detección de errores, seguimiento de enfermedades poco estudiadas, entre otros.

Por otra parte, con la experiencia adquirida en el transcurso del proyecto, entendemos que la solución ideal debería estandarizar la comunicación con el repositorio semántico, mediante la independencia que facilita SPARQL sobre HTTP, de esta forma, se aprovecharía la potencialidad de las herramientas existentes como Fuseki. Dicho de otra forma, consideramos que la **evolución de este trabajo** consiste en generar **una aplicación Web que utilice un endpoint SPARQL**, entendido como una herramienta que permite realizar consultas SPARQL sobre modelos ontológicos , que en nuestro caso representa "*modelo ontológico resultante*".

Por la vía de la estandarización de la comunicación entre repositorios semánticos, se debería proyectar hacia la **consulta de información en diversas y distribuidas fuentes**.

6.4 Trabajos Futuros.

Para aumentar significativamente la riqueza semántica de la propuesta del presente proyecto de grado, consideramos que sería de gran interés investigar la **integración con otras ontologías existentes en el ámbito de la Salud**.

Debido al rol que cumple la ontología de **SNOMED** en el *Proyecto Salud Uy*, y al invalorable conocimiento que podría aportar, sería fundamental lograr la realización de consultas más completas y complejas, uniendo la información contenida en dicha ontología con la información semántica recopilada de los documentos CDA. Entendemos que ello aportaría un **nivel de abstracción** mayor en la construcción de consultas, dado que SNOMED provee una **extensa red de conceptos relacionados**.

Bibliografía

[1] AGESIC, «AGESIC,» [En línea]. Available: http://agesic.gub.uy/innovaportal/v/19/1/agesic/que_es_agesic.html.

[2] Presidencia, «Programa Salud.uy avanza hacia la historia clínica electrónica.,» [En línea]. Available: http://presidencia.gub.uy/comunicacion/comunicacionnoticias/programa-salud-uy. [Último acceso: 24 11 2013].

[3] AGESIC, «Acuerdo para programa de salud.uy,» [En línea]. Available: http://agesic.gub.uy/innovaportal/v/2325/1/agesic/acuerdo_para_programa_de_saluduy.html. [Último acceso: 09 11 2013].

[4] P. S. Uy, «Estructura Mínima Nac.l del documento clínico HL7 V3 CDA-R2 para uso en el Dom de Salud .,» -, 2013.

[5] I. o. E. a. E. Engineers, «Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.,» 1990.

[6] Cepal, «Interoperabilidad.,» de *Manual de Salud Electronica*, 2012, pp. 317-328.

[7] Gerson Villa , Luis Hernandez, «Limitaciones de la Web Actual, Futuro de la Web.,» 14 07 09. [En línea]. Available: http://www.cicataqro.ipn.mx/es/tecnologa/V2N3A2.pdf. [Último acceso: 08 07 2012].

[8] M. I. L. &. H. T. O. C. -. U. O. d. C.-. Codesido, « LENGUAJES DE CONSULTA PARA DOCUMENTOS RDF.,» 2006.

[9] W3c, «XML Essentials.,» [En línea]. Available: http://www.w3.org/standards/xml/core. [Último acceso: 23 06 12].

[10] A. L. Gil, «Estado del Arte de los lenguajes de marcación para la Web Semántica.,» Universidad Rey Juan Carlos. [En línea]. [Último acceso: 12 07 2013].

[11] V. G. L. M. a. H. S. M. C. Gerson, «Limitaciones de la Web Actual Futuro de la Web,» 2010.

[12] W3C, «W3C XML Schema Definition Language (XSD).,» [En línea]. Available: http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/. [Último acceso: 24 6 12].

[13] F. L. a. K. R. a. H. S.-K. a. T. I. Gandon, «Semantic Annotation and Retrieval: RDF,» de *Handbook of Semantic Web Technologies*, Springer Berlin Heidelberg, 2011, pp. 117-155.

[14] T. Segaran, C. Evans y J. Taylor, «Just Enough RDF,» de *PROGRAMMING THE SEMANTIC WEB*, O'Reilly Media, Inc., 2009.

[15] P. H. a. M. K. a. S. Rudolph, Foundations of Semantic Web Technologies, Chapman & Hall, 2009.

[16] P. Castells, «La Web Semántica - Sistemas Interactivos y Colaborativos en la Web -,» 2003.

[17] I. J. Flores, «Introduccion al Razonamiento sobre ontologías.,» 04 2011. [En línea]. Available: www.ciens.ucv.ve/escueladecomputacion/documentos/archivo/117.

[18] E. P. Lee Feigenbaum, «SPARQL by Example -A tutorial-,» 30 05 2013. [En línea]. Available: http://www.cambridgesemantics.com/semantic-university/sparql-by-example. [Último acceso: 26 07 2013].

[19] B. DuCharme, «SPARQL Queries: A Deeper Dive,» de *Learning SPARQL- Querying and Updating with SPARQL 1.1*, O'Reilly Media, 2011.

[20] W3C, «SPARQL Query Language for RDF,» 26 03 2013. [En línea]. Available: http://www.w3.org/TR/rdf-sparql-query/. [Último acceso: 26 07 2013].

[21] E. D. a. C. S. Valle, «Querying the Semantic Web: SPARQL,» de *Handbook of Semantic Web Technologies*, Springer Berlin Heidelberg, 2011, pp. 299-363.

[22] W3C, «SPARQL Update - A language for updating RDF graphs,» 15 07 2008. [En línea]. Available: http://www.w3.org/Submission/SPARQL-Update/. [Último acceso: 26 07 2013].

[23] H. Co., «HL7,» [En línea]. Available: www.hl7.com.

[24] T. Benson, Principles of Health Interoperability HL7 and SNOMED, Springer, 2010.

[25] HL7 España, «Guia de Elementos Minimos CDA.,» [En línea]. Available: http://www.hl7spain.org/documents/comTec/cda/GuiaElementosMinimosCDA.pdf . [Último acceso: 11 07 2013].

[26] H. Co., «HL7 Vocabulary Domains,» 2009.

[27] K. W. Boone, The CDA Book, Springer, 2011.

[28] P. d. Toledo, «IHE y la historia clínica compartida: XDS,» Madrid, 2009.

[29] I. -. Canada., «Interoperability Solution,» [En línea]. Available: https://www.infoway-inforoute.ca/index.php/programs-services/interoperability-solutions. [Último acceso: 12 11 2013].

[30] H. Co., «CDA® Release 2,» [En línea]. Available: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7. [Último

acceso: 10 11 2013].

[31] G. S. B. M. D. Pradeep K. Sinha, «Electronic Health Record,» de *Standards, Coding Systems, Frameworks, and Infrastructures.*, 2013, p. 322.

[32] N. I. o. S. a. Tecnology, «CDA Guide Validation,» [En línea]. Available: http://xreg2.nist.gov/cda-validation/downloads.html. [Último acceso: 19 11 2013].

[33] Nictiz, «CDA XSL Test,» [En línea]. Available: https://decor.nictiz.nl/CDA/. [Último acceso: 12 10 2013].

[34] W3c, «Guia Breve de Linked Data,» [En línea]. Available: http://www.w3c.es/Divulgacion/GuiasBreves/LinkedData. [Último acceso: 10 11 2013].

[35] A. A. L. H.Vasquéz, «Ontologías Médicas: Una revisión.,» [En línea]. Available: http://revista.iutet.edu.ve/v11_21_1.pdf. [Último acceso: 2 11 2013].

[36] «The Open Biological and Biomedical,» [En línea]. Available: http://www.obofoundry.org/. [Último acceso: 02 11 2013].

[37] «Bioportal,» [En línea]. Available: http://bioportal.bioontology.org/ontologies. [Último acceso: 02 11 2013].

[38] H. Chan, «HL7 RIM V3 OWL Ontology for Semantic Web - W3C,» [En línea]. Available: http://www.w3.org/wiki/images/2/2e/HCLS$$ClinicalObservationsInteroperability$$HL7CDA2OWL.html$notes.html.

[39] A. Iqbal, «An OWL-DL Ontology for the HL7 Reference,» 2007.

[40] «IHTSDO,» [En línea]. Available: http://www.ihtsdo.org/members/uruguay/ .

[41] M. M. J. P. Stijn Heymans, «Semantic validation of the use of SNOMED CT in HL7 clinical documents,» 2011. [En línea]. Available: http://link.springer.com/article/10.1186%2F2041-1480-2-2#page-2.

[42] IBM China Research Laboratory, «iSMART: Ontology-based Semantic Query of CDA Documents,» [En línea]. Available: http://researcher.watson.ibm.com/researcher/files/cn-XIEGUOT/iSMART_AMIA_final_revision.pdf.

[43] «HL7 GForce - HL7OWL,» [En línea]. Available: http://gforge.hl7.org/gf/project/hl7owl/scmsvn/?action=ScmCommitDetail&scm_commit_id=7926.

[44] RAE, «Real Academia Española,» [En línea]. Available: http://lema.rae.es/drae/srv/search?key=vocabulario. [Último acceso: 19 11 2013].

[45] A. Jena. [En línea]. Available: http://jena.apache.org/about_jena/about.html.

[46] J. A. M. Gimenez, «Tesis "Entorno para la gestión semántica de información biomédica en investigación",» 2012. [En línea]. Available: http://www.tdx.cat/bitstream/handle/10803/92299/TJAMG.pdf?sequence=1.

[47] J. Inferencia., «Reasoners and rule engines: Jena inference support,» [En línea]. Available: http://jena.apache.org/documentation/inference/index.html. [Último acceso: 11 11 2013].

[48] R. M. G. Postgresql, «PostgreSQL en español,» [En línea]. Available: http://www.postgresql.org.es/sobre_postgresql. [Último acceso: 12 11 2013].

[49] C. &. Parsia. [En línea]. Available: http://clarkparsia.com/pellet/faq/extract-all-inferences/. [Último acceso: 12 11 2013].

[50] J. Apache, «Jena Apache - SDB,» [En línea]. Available: http://jena.apache.org/documentation/sdb/ . [Último acceso: 12 11 2013].

[51] W3c, «W3c - SPARQL 1.1 Update,» [En línea]. Available: http://www.w3.org/TR/sparql11-update/. [Último acceso: 26 11 2013].

Tabla de Ilustraciones