Generación automática de ontologías en SKOS de clasificaciones estándar de productos: Common Procurement Vocabulary (CPV)

Jose María Alvarez Rodríguez
Fundación CTIC
Parque Científico Tecnológico
33203 Gijón, Asturias
josem.alvarez@fundacionctic.org

Emilio Rubiera Azcona
Fundación CTIC
Parque Científico Tecnológico
33203 Gijón, Asturias
emilio.rubiera@fundacionctic.org

Luis Polo Paredes
Fundación CTIC
Parque Científico Tecnológico
33203 Gijón, Asturias
luis.polo@fundacionctic.org

Resumen

En este artículo los autores se proponen ilustrar un método para derivar automáticamente una ontología SKOS Core de una clasificación estándar de productos y servicios, como es el caso del CPV, e@Class, UNSPSC, etc. A lo largo del artículo, los autores presentarán una método para la reconstrucción semántica de una clasificación de productos bajo el vocabulario SKOS Core y se definirá un conjunto de transformaciones, Tr_{skos}, para su generación automática mediante hojas de transformación XSL. Este método y su implementación se aplicará al caso concreto Common Procurement Vocabulary (CPV).

1. Introducción

En los últimos años, se ha despertado un creciente interés en el uso de las clasificaciones estándar de productos, PSCs[1] en adelante. Las PSCs son interesantes, en el ámbito de la Web Semántica, porque constituyen puntos de vista consensuados y adoptados como forma de intercambio de información entre distintos actores en procesos de negocio. El objetivo es conseguir que múltiples aplicaciones compartan un vocabulario de dominio común favoreciendo la comunicación automática en entornos B2G y B2B [2, 5, 1].

Las PSCs se utilizan actualmente en distintos escenarios debido a su gran flexibilidad y capacidad de especialización. Nos gustaría señalar dos importantes campos de aplicacion de las PSCs, como son: 1. Anotación de productos y servicios favoreciendo la mediación de datos y procesos. Esto es especialmente relevante en arquitecturas SOA, en escenarios de comercio electrónico y cadenas de logística; 2. Catalogación de productos o documentos para su explotación en entornos de búsqueda y recuperación de información: buscadores, catálogos virtuales, etc.

Tal y como se señala en [1], para que la comunicación automática entre distintas sistemas sea efectiva es necesario tener en cuenta dos factores: 1. un formato común de datos y 2. un modelo de conocimiento compartido entre los distintos sistemas que forman parte de los procesos. Nuestra propuesta es la siguiente:

Formato de datos: RDF (Resource Description Framework) [6] es el lenguaje propuesto por el W3C[2] para la representación y descripción de los recursos en la Web. Proporciona una infraestructura semántica común basada en un modelo de tripletas, fácilmente extensible, permitiendo que los datos se puedan relacionar, ligar o fusionar con cualquier otro recurso definido en la Web. La adopción de la

[1] Product Scheme Classification: PSC.

[2] http://www.w3c.org

sintaxis RDF/XML para la serialización del modelo RDF favorece el intercambio de información entre aplicaciones, aprovechando todas las ventajas inherentes al uso de la tecnología XML.

Modelo de datos: El vocabulario que utilizaremos para la representar uniformemente las PSCs es el SKOS-Core [7], iniciativa del W3C para representar esquemas conceptuales. El uso de SKOS Core nos permite reflejar la estructura base de cualquier PSC sin alejarnos de su semántica original, interpretándolas como vocabularios controlados. Consideramos que SKOS Core es un vocabulario web ideal para la reutilización las PSCs en el campo de la Web Semántica y para proporcionar un marco de interpretación común de las mismas.

En el presente artículo, presentaremos un método para representar las clasificaciones de productos utilizando SKOS Core. A su vez, definirán como parte de ese método un conjunto de transformaciones Tr_{skos} para automatizar el proceso de migración desde las fuentes originales hasta el formato de datos escogido para su representación semántica: RDF. Nuestra prueba de concepto consiste en la transformación automática del CPV a SKOS-Core: CPV_{skos}.

2. Características de las Clasificaciones de Productos

Las PSCs son instrumentos claves de estandarización que nacen con el fin de conseguir una clasificación común de productos y servicios en un determinado contexto de negocio [5, 2]. Estas pueden ser muy variopintas y obedecer a distintos intereses particulares, desde clasificaciones sectoriales, como puede ser el caso de e@Class[3] y RosettaNet[4], hasta clasificaciones globales de productos, como es el caso de la UNSPSC[5] o del CPV. Las diferencias entre las clasificaciones no sólo se limitan a cuestiones de alcance, perspectiva y cobertura sectorial de productos, también la granularidad o el nivel de profundidad de las clasificaciones difiere de unas a otras.

Como señala Hepp [4, 3] todos estos estándares reflejan una combinación de componentes variables. Sin embargo, podemos identificar una estructura común subyacente a todas ellas y que consideramos es fundamental señalar para proporcionar un modelo de datos uniforme de las PSCs. Utilizaremos las nociones tomadas de la teoría de grafos, como son los conceptos de *árbol* y *bosque*, para representar la semántica común de las PSCs.

Categorías de productos: las clasificaciones se dividen en categorías o clases de productos. Estas categorías agrupan los elementos de la PSC, Cat_{psc}, en distintos niveles de especialización semántica, desde términos genéricos, como «Productos alimenticios o bebidas», hasta productos altamente específicos y directamente identificables, «Salami». Las categorías de productos de una PSC cumplen las siguientes propiedades:

- Los elementos de la PSC se dividen varias categorías: $Cat_{psc} = \bigcup_{i=0}^{n} (Cat_{psc}^{i})$.
- Las categorías de la PSC se organizan jerárquicamente: $Cat_{psc}^{0} \succ Cat_{psc}^{1} \succ ... \succ Cat_{psc}^{n}$.
- Cada elemento de la PSC, t_{psc}, pertenece a una categoría de productos.
- Cada elemento de la PSC, t_{psc} pertenece sólo una categoría de productos. Es decir, las categorías son disjuntas: $\bigcap_{i=0}^{n} (Cat_{psc}^{i}) = \emptyset$.

Sectores de productos: El objetivo de una PSC es organizar y agrupar los productos en sectores verticales mediante algún tipo de criterio establecido por el organismo o consorcio que desarrolla el estándar. Son los elementos más genéricos, pertenecientes al nivel más alto de la jerarquía

[3] http://www.eclass.de
[4] http://www.rosettanet.org
[5] http://www.unspsc.org

de categorías, Cat^0_{psc}, los que determinan los sectores de productos que se van a especializar y desglosar en la PSC. La relación jerárquica entre las categorías de la PSC construye una estructura taxonómica de cada sector de productos. Formalmente, la estructura de un sector de productos podemos representarla con un **árbol**, T^k_{psc}, y concluyendo que una PSC no es más que un conjunto de sectores , $\mathbb{F}_{psc}$, que cumple las siguientes propiedades:

- Una PSC es un **bosque** de sectores de productos: $\mathbb{F}_{psc} = \bigcup_{i=0}^{m} (T^i_{psc})$.
- Cada elemento t_{psc} pertenece a un sector de productos.
- Cada elemento t_{psc} pertenece sólo a un sector de productos. Es decir, los sectores de productos son disjuntos: $\bigcap_{i=0}^{m} (T^i_{psc}) = \emptyset$.
- Cada sector de productos, T^k_{psc}, tiene como **raíz** un elemento $t^0_{psc} \in Cat^0_{psc}$.
- Cada elemento t^k_{psc} está relacionado con un elemento de nivel superior t^{k-1}_{psc}, menos el elemento raíz.

Estas son características genéricas de las clasificaciones de productos. Sin embargo, otras PSCs más sofisticadas incluyen un diccionario de propiedades estándar que se puede utilizar para describir productos con más detalle. Normalmente, estos diccionarios de propiedades también incluyen los tipos de datos que pueden ser valor de las mismas, así como su referencia con respecto estándares internacionales para establecer las unidades de medida. Este el caso de la clasificación de productos de e@Class. En otras ocasiones, se construyen clasificaciones multilingües para la expresión de los descriptores de cada elemento de la PSC. El caso extremo es el Common Procurement Vocabulary, donde se contempla un total de hasta 20 lenguas.

2.1. E-procurement y el Common Procurement Vocabulary

El Common Procurement Vocabulary[6] (en adelante CPV) es un catálogo de productos de obligada utilización en procesos de contratación pública según las Directivas de la Unión Europea[7]. Busca unificar el mercado europeo de contratación pública con el objetivo de minimizar costes de operaciones tanto para compradores como para suministradores y agilizar los procedimientos de Administración electrónica. El principal propósito del CPV es describir los objetos de contratación pública en los anuncios de licitación. La aplicación del CPV consiste en el etiquetado de documentos en dos escenarios complementarios con el doble objetivo de: 1. mejorar la calidad de búsqueda de contratos públicos 2. favorecer la gestión documental y de trámites en la Administración Electrónica .

El CPV es un vocabulario controlado multilingüe que alberga más de 8000 términos en cada idioma y cuya estrutura taxonómica consta de 4 niveles de jerarquía: 1. $Cat^0_{cpv} = Division$; 2. $Cat^1_{cpv} = Grupo$; 3. $Cat^2_{cpv} = Clase$; 4. $Cat^3_{cpv} = Categoria$

El CPV está traducido actualmente a 20 lenguas, y se están preparando versiones en lenguas de países que ya han sido admitidos en la UE, como Rumanía. Se encuentra disponible en distintos formatos: HTML, Microsoft Excel y PDF. Estos formatos, sin embargo, están orientados a presentación y consulta, más que a procesamiento automático de la información.

2.2. Interpretación Semántica de las PSCs

El desafío básico más importante que hay que afrontar cuando se deriva una ontología de una PSC, es cómo interpretar la semántica original de la taxonomía. No existe una definición formal de las relaciones taxonómicas que construyen cada T_{psc} de la clasificación y es tentador utilizar la propiedad de herencia de un vocabulario de ontologías, como *rdfs:subClassOf* en el caso de RDF Schema, para intentar representar estas relaciones semánticas.

[6] http://simap.eu.int
[7] Reglamento (CE) nº 2195/2002

Sin embargo, esta suposición es errónea. Como señala [3], esta relación de jerarquía entre los elementos no se puede considerar equivalente a una relación de subclase. En primer lugar, consideremos el siguiente ejemplo. El elemento «Partes y accesorios de bicicleta» del CPV (34442000-7) tiene como antecesor a «Bicicletas» (34440000-3), donde la relación semántica entre los dos elementos no es de herencia o subclase: una **parte de una bicicleta** no es una **bicicleta**. Más bien, debería modelarse como una relación de composición o agregación. En este sentido, ocurre igual con la relación entre «Tubos de resina Epoxi»(25271100-2) y «Resinas Epoxi» (25271000-1). Es difícil justificar de nuevo una relación de herencia clásica entre el primero y el segundo elementos, ya que en ningún caso podemos considerar que el continente y el contenido de un objeto descrito tengan el mismo estatus en una ontología de dominio. En otras palabras, un **tubo** no es un tipo de **resina**.

En cualquier caso, no sólo es complicado interpretar correctamente las relaciones semánticas que codifica la taxonomía de una PSC. Desde el punto de vista de las ontologías como modelos de conocimiento de dominio, es complicado considerar algunos elementos como tales. Consideremos, por ejemplo, un elemento del CPV como «Barras, varillas, perfiles y alambre de estaño» (27623100-9) que es más una colección artificial de productos que una clase estructural, en la que sus instancias comparten algún tipo de propiedad común[8].

Consideramos nuestra postura más funcional. Desde nuestro punto de vista, las PSCs fueron construidas para solucionar problemas de comunicación, proporcionar una forma de organizar tipos de productos y agruparlos acuerdo a unos conceptos y definiciones que funcionasen *de facto* como un estándar en determinados entornos de actividad comercial e industrial. Las PSCs no fueron diseñadas como modelos conceptuales de dominio, en el sentido actual que tiene el término «ontología» como conceptualizacón de dominio, sino como una forma de estructurar la terminología y de imponer un criterio a la hora de clasificar los productos. De ahí, que interpretemos las PSCs como simples esquemas conceptuales en el que la relación taxonómica que jerarquiza los distintos elementos de cada T^k_{psc} no es equivalente a una relación de subclase, sino a una relación de mayor o menor especificidad en el alcance del significado de sus elementos. Resumiendo, consideramos las semántica original de las PSCs se corresponde con la de los vocabularios controlados como veremos en la siguiente sección.

3. PSC_{skos}: Adaptación de las PSCs a SKOS Core

Desde este enfoque, las PSCs son vocabularios controlados y la adopción de una ontología como SKOS Core como metamodelo de las mismas está completamente justificada[9]. SKOS Core es una ontología RDF/OWL que gira en torno a una clase principal *skos:Concept*.

La correspondencia entre la estructura de las PSCs y los vocabularios controlados, permite interpretar cada elemento t_{psc} como un *skos:Concept* de manera directa y sin provocar incoherencias. Todos los elementos de una PSC son considerados instancias de *skos:Concept*. Entendemos que todos ellos son recursos conceptuales que codifican algún tipo de significado y que se expresan mediante algún término en una lengua natural. Denominaremos mediante PSC_{skos} cualquier PSC representada bajo el modelo de datos SKOS Core. Consideremos el caso particular del Common Procurement Vocabulary (CPV).

Para agrupar todos los elementos del CPV bajo un paraguas común, construimos una clase contenedora del conjunto Cat_{cpv} de todos sus elementos: *CpvConcept*, y la declaramos

[8] Estrictamente hablando, para interpretar correctamente el elemento «Barras, varillas, perfiles y alambre de estaño» como una clase, creemos que se debería definir como la unión de varias clases: por ejemplo, «Barras», «Varillas» o «Alambre de estaño»

[9] Para la representación de tripletas RDF utilizaremos la sintaxis N3 y para la declaración de los axiomas en CPV_{skos} usaremos directamente fórmulas en Lógica Descriptiva. Para abreviar los ejemplos, utilizaremos los siguientes prefijos: **skos** para el espacio de nombres: *http://www.w3.org/2004/02/skos/core*; y **dc** para *http://purl.org/dc/elements/1.1/*.

como subclase del concepto $skos : Concept$:

$$\#CpvConcept \sqsubseteq skos : Concept \quad (1)$$

Por un lado, por cada categoría de productos, Cat^k_{cpv}, construimos una nueva clase y la declaramos subclase de *CpvConcept*,

$$Cat^0_{cpv} \rightsquigarrow \#Division \sqsubseteq \#CpvConcept \quad (2)$$
$$Cat^1_{cpv} \rightsquigarrow \#Grupo \sqsubseteq \#CpvConcept \quad (3)$$
$$Cat^2_{cpv} \rightsquigarrow \#Clase \sqsubseteq \#CpvConcept \quad (4)$$
$$Cat^3_{cpv} \rightsquigarrow \#Categoria \sqsubseteq \#CpvConcept \quad (5)$$

Para completar la semántica original de la clasificación añadimos los siguientes axiomas siguiendo el modelo de datos de las PSCs analizado en la sección 2.1, perfectamente expresable en OWL-DL:

$$\#CpvConcept \equiv \#Division \sqcup \quad (6)$$
$$\#Grupo \sqcup \#Clase \sqcup \#Categoria$$

$$\#division \sqcap \#grupo \sqcap \quad (7)$$
$$\#clase \sqcap \#categoria \equiv \bot$$

Por otro lado, el segundo factor importante es la estructura taxonómica de cada T^k_{cpv}, que como vimos, tenía una difícil interpretación como conjunto de relaciones de subclase. Sin embargo, bajo el paradigma SKOS Core se contemplan tres relaciones semánticas: *skos:broader*, *skos:narrower* y *skos:related*. Para reconstruir la taxonomía original del CPV sólo son necesarias las dos primeras. Estas propiedades se utilizan para declarar que el significado de un concepto es más genérico que otros, *skos:broader*, o para declarar que el significado de un concepto es más específico que otros, *skos:narrower*. Consideremos de nuevo estos ejemplos bajo el prisma de los vocabularios controlados. Lo único que se asegura es que la jerarquía de elementos de una clasificación: $t^0_{cpv} \succ t^1_{cpv} \succ ... \succ t^n_{cpv}$, implica una cadena de conceptos desde los más genéricos hasta los más específicos. Así, por ejemplo, «Partes y accesorios de bicicleta» (34442000-7) sí puede tener como concepto más genérico (*skos:broader*) «Bicicletas» (34440000-3).

Mediante estas propiedades reconstruimos la estructura taxonómica de cada sector de productos, T^k_{cpv}, de forma que podemos recorrerla de forma ascendente (*skos:broader*) y descendente (*skos:narrower*). Además, el uso de un razonador de Lógica Descriptiva, como es el caso de Pellet o Racer, nos ayuda a mejorar el tratamiento de este tipo de ontologías basadas en SKOS Core. Ambas propiedades son transitivas y están declaradas como inversas. Un razonador DL es capaz de, a partir del conjunto de relaciones en un sentido, por ejemplo, *skos:narrower*, deducir el conjunto de relaciones inversas, *skos:broader*[10].

```
#34440000-3 skos:broader #34442000-7
->
#34442000-7 skos:narrower #34440000-3
```

Figura 1: Inferencia de relaciones inversas en SKOS Core

La transitividad se puede aprovechar además para inferir las cadenas conceptuales entre los elementos de cada T^k_{cpv}, ya que a partir de las relaciones explícitas de *skos:narrower* y *skos:broader* se pueden obtener todas las dependencias implícitas entre los elementos de un mismo sector de productos. Si consideramos por ejemplo la cadena «Resinas»$\succ$«Resinas Epoxi»$\succ$«Tubos de resina Epoxi», se puede inferir lo siguiente en la ontología CPV_{skos}:

Todas las PSCs identifican a sus elementos de forma unívoca mediante un código alfanumérico. Sin embargo, SKOS Core no contempla directamente ninguna propiedad que pueda ser utilizada en este sentido. Gracias a la flexibilidad y extensibilidad de RDF podemos

[10] Esta característica es relevante para la generación automática de PSCs_{skos}, ya que sólo haría falta generar a partir de T^k_{psc} el conjunto de relaciones explícitas en un sentido. Un razonador DL se encargará de cerrar el modelo y deducir el conjunto de relaciones inversas.

```
#25271100-2 skos:broader #25271000-1
#25271000-1 skos:broader
->
#25271100-2 skos:broader #25270000-4
```

Figura 2: Transitividad de las relaciones semańticas en SKOS Core

combinar distintos vocabularios y ontologías para la descripción de un recurso. Hemos optado por la propiedad por la propiedad del vocabulario Dublin Core, *dc:identifier*, para este fin.

```
#35111000-5 dc:identifier '35111000-5'
#35111000-5 skos:prefLabel 'Buques de guerra'@es
#35111000-5 skos:prefLabel 'Warships'@en
#35111000-5 skos:prefLabel 'Navires de guerre'@fr
#35111000-5 skos:prefLabel 'Krigsfartyg'@sv
```

Figura 3: Ejemplo de uso de descriptores e identificadores.

De la misma manera, los elementos de las PSCs se asocian con un término descriptor que describe en mayor o menor grado su significado. SKOS Core está especialmente diseñado para el tratamiento de esta información, que suele ser común para la mayor parte de los vocabularios controlados: desde taxonomías hasta tesauros. En este documento, sólo nos centraremos en los descriptores de conceptos, que deben ser expresados mediante la propiedad *skos:prefLabel* indicándose mediante *xml:lang* cuál es el idioma origen. Es decir, para cada concepto tendremos tantos *skos:prefLabel* como lenguas tenga la PSC en cuestión.

Para la aplicación de las PSCs$_{skos}$ como vocabularios controlados debemos ser capaces de relacionar los recursos que queremos describir (productos, servicios o documentos) con los elementos correspondientes de la clasificaciones. Mediante la propiedad *skos:subject* podemos describir el significado de un recurso web apuntando a un elemento de vocabulario en SKOS Core, en este caso, a términos de las PSCs$_{skos}$. Por ejemplo, podemos referenciar los objetos de contratación en contratos públicos y anuncios de licitación con este método. Además, para mejorar la recuperación de recursos utilizando la estructura de un vocabulario controlado, SKOS Core combina las propiedades de *skos:broader* y *skos:subject* con la regla conocida como *Subject Generality Rule*[7]. De esta forma, un recurso se relaciona con el sector de productos relevante para su definición y no sólo con el término que representa su significado en el contexto de una PSC determinada:

```
(?resource skos:subject ?x) (?x skos:broader ?y)
->
(?resoure skos:subject ?y)
```

Figura 4: Subject Generality Rule

3.1. Método para transformar automáticamente PSCs en vocabularios SKOS-Core

Nuestra propuesta para construir PSCs$_{skos}$ se basa en el método genérico para reutilizar estándares definido en [1]. Las tareas de este método son válidas como guía para la adaptación de un estándar a una convención determinada, en nuestro caso a SKOS Core, y para obtener un modelo normalizado de representación de las PSCs partiendo de fuentes heterógeneas. Estas tareas nos han servido como procedimiento de actuación para definir PSCs$_{skos}$, aunque no son aplicables directamente a nuestra implementación, por ejemplo, en el caso de la CPV. Por eso, hemos adaptado este método para el caso concreto de las PSCs, ajustándolo al modelo uniforme que nos proporciona SKOS Core.

La primera parte de nuestro método consiste en expresar el modelo conceptual de la PSC bajo la estructura de SKOS Core. Ejemplificaremos los pasos seguidos para la contrucción de CPV$_{skos}$.

1. Asociar un espacio de nombres mediante una URI al CPV.

2. Construir la clase *CpvConcept* en ese espacio de nombres y declararla como subclase de *skos:Concept*.

3. Identificar las categorías o niveles de jerarquía, Cat^n_{cpv}, y construir respectivas clases declarándolas subclases de *CpvConcept*.

La segunda parte de nuestro método consiste en un conjunto de transformaciones, Tr_{skos} que, a partir de la fuente original de datos, convierten la informacion y la semántica original del CPV en tripletas RDF bajo la estructura conceptual construida en la primera parte del método. Los objetivos de la transformación son:

1. Generar a partir del código de cada elemento del CPV, recursos web identificables mediante URIs: tr^{URI}_{skos}.

2. Establecer la categoría o tipo de cada elemento de acuerdo al CPV, $t^k_{cpv} \in Cat^k_{cpv}$: tr^{type}_{skos}.

3. Establecer relaciones semánticas generando un subgrafo acíclico dirigido con la propiedad *skos:broader* reconstruyendo cada sector de productos, T^k_{cpv}, en sentido ascendente[11]: tr^{broad}_{skos}.

4. Añadir identificadores para cada elemento del CPV utilizando el código de cada elemento: tr^{id}_{skos}

5. Añadir términos descriptores para cada elemento del CPV teniendo en cuenta el idioma: tr^{label}_{skos}

La fuente original del CPV son cadenas de texto que el conjunto de transformaciones Tr_{skos} deben saber interpretar para generar correctamente el grafo RDF esperado. Por un lado, *id* hace referencia a la cadena que representa el código de identificación de cada elemento del CPV y, por otro, *term* a su término descriptor. Por ejemplo, para un elemento t^k_{psc} tenemos: $id(t^k_{psc}) = 01112210 - 0$ y $term(t^k_{psc}) =$ 'Leguminosas secas'.

[11]En el siguiente apartado se explicará por qué sólo generamos el árbol en sentido ascendente. En cualquier caso, hay que recordar que el uso de un razonador DL permitiría reconocer las relaciones inversas *skos:narrower*

```
<Row>
  <Cell><Data>01112210-0</Data></Cell>
  <Cell><Data>Leguminosas secas</Data></Cell>
</Row>
```

Figura 5: Formato XML del CPV tras exportarlo desde una hoja de cálculo

Tr_{skos}	Args.	Grafo RDF (N3)
tr^{URI}_{skos}	id	id
tr^{type}_{skos}	id	id *rdf:type* Cat^k_{psc}
tr^{broad}_{skos}	id_1, id_2	id_1 *skos:broader* id_2
tr^{id}_{skos}	id	id *dc:identifier* $'id'$
tr^{label}_{skos}	$id, term$	id *skos:prefLabel* $'term'$

Cuadro 1: Conjunto de transformaciones Tr_{skos}

4. Implementación de Tr_{skos} mediante XSL

Las clasificaciones suelen estar disponibles en distintos formatos, aunque no todos ellos son aptos para el tratamiento automático. En cualquier caso, la mayor parte de las clasificaciones sí están disponibles en hojas de cálculo, lo que facilita su exportación a un formato como XML, formato estructurado y procesable mediante tecnología robusta como son las hojas de transformación XSL.

Desde nuestro punto de vista, XSL es la tecnología que se ajusta perfectamente a nuestros requisitos para las implementaciones Tr_{skos} por las siguientes razones:

- Permite generar el modelo RDF deseado, PSC_{skos}, al poder serializar este último mediante sintaxis RDF/XML a la salida del proceso de transformación

- Es un lenguaje sencillo, funcional y especializado para la realización de plantillas y encaje de patrones (XSLT) en XML y un potente lenguaje de selección de nodos y procesamiento de valores (especialmente literales cadena) como XPath.

- Asegura la corrección sintáctica tanto del árbol de entrada como del de salida

- Es una tecnología ampliamente asentada (muchas implementaciones) con soporte tanto para distintas plataformas y lenguajes de programación. En nuestro caso, el entorno de ejecución escogido es la plataforma Linux y el procesador de XSL-XSLTProc (implementación especialmente eficiente)

Hay señalar que la implementación de las funciones podría variar según el tipo de PSC, la entrada muy probablemente sería distinta, según el organismo encargado de su desarrollo (gustos particulares de nomenclatura, etc.), pero en ningún caso romperá con la definición teórica realizada.

4.1. Generación automática del CPV$_{skos}$

Como vimos en la sección 2.1, el CPV consta de 4 niveles de jerarquías. Se realiza a través de un código único de 8 cifras, que sigue un patrón para definir a cada una de las categorías de la clasificación. El noveno dígito se utiliza como número de chequeo para verificar que el identificador del elemento es correcto. El patrón de cada uno de los elementos permite determinar su categoría Cat^k_{cpv} y las relaciones jerárquicas entre los elementos de un sector de productos.

Para realizar una interpretación completa de la jerarquía del CPV hay que tener en cuenta que el nivel **Categoría** no es uniforme, sino que existen relaciones jerárquicas que se deben mantener para aprovechar toda la información disponible en la taxonomía. De esta forma, como se puede observar en el cuadro 2, existen 4 nuevos niveles adicionales de jerarquía que no aparecen reflejados como **categorías de productos**.

Categoría	**Id**	**Ejemplo**
Division	XX000000-y	01000000-7
Grupo	XXX00000-y	01100000-8
Clase	XXXX0000-y	01110000-1
Cat(L0)	XXXXX000-y	01112000-5
Cat(L1)	XXXXXX00-y	01112200-7
Cat(L2)	XXXXXXX0-y	01112210-0
Cat(L3)	XXXXXXXX-y	01112211-7

Cuadro 2: Jerarquía CPV

La transformación de cada árbol, T^k_{cpv}, a CPV$_{skos}$ se genera en sentido ascendente utilizando la propiedad *skos:broader*. Dado cualquier elemento del CPV, sabemos cómo generar sus ascendientes procesando su identificador alfanumérico. De esta manera, para un elemento t^k_{cpv}, "01112000-5", sabemos que el elemento correspondiente t^{k-1}_{cpv} con el que está relacionado se identifica con la cadena "01110000", con lo que ya podemos calcular el dígito control. Con este procedimiento somos capaces de generar los ascendientes de cualquier elemento T^k_{cpv} con certeza y con una sola operación.

Por el contrario, dado el elemento t^k_{cpv}, "01112000-5", obtener los identificadores de sus descendientes (relaciones *skos:narrower*) conllevaría generar al menos 10 nuevos identificadores (01112[0-9]00) sin saber ni siquiera si existen en la clasificación. Si extendemos esta situación para todos los niveles y elementos del CPV, reconstruir la jerarquía de T^k_{cpv} en sentido descendente implicaría un doble proceso de generar y comprobar la existencia de cada uno de los nodos generados, ya que no tenemos certeza ni de su existencia ni del número real de descendientes. Esto generaría un árbol multigrado con la posibilidad de 10 hijos por nodo, y con un número de máximo de nodos estimado en 11111094 (División: 99 + Grupo: 999 + Clase: 9999 + Categoría(L0) 99999 + Categoría(L1) 999999 + Categoría(L2) 9999999). Por tanto, dado que el procedimiento de generación descendente conlleva una dificultad extra de implementación y teniendo en cuenta que las relaciones *skos:narrower* se pue-

den obtener, como ya se ha comentado, a través de un razonador DL mediante la relación *skos:broader*, consideramos que queda totalmente justificada la elección de generación en un solo sentido, el ascendente.

Nuestra implementación del método propuesto en la sección 3.1 para construir la CPV_{skos} ha sido la siguiente:

1. La implementación de la primera parte del método se realiza a través de un fichero OWL en el que se importa la ontología SKOS Core y se declaran los axiomas necesarios para la definición de *CpvConcept*, como vimos en el apartado anterior.

2. A continuación se ejecutan las transformaciones Tr_{skos} implementadas en XSL. Como muestra, ponemos los siguientes ejemplos de código:

   a) Función tr_{skos}^{URI}:

```
<xsl:template name="format-id">
 <xsl:param name="id"/>
 <xsl:choose>
   <xsl:when test="contains($id,'(')">&
        pscNS;<xsl:value-of select="concat
        ('cpv-',substring-before($id,'('))
        "/></xsl:when>
   <xsl:otherwise>&pscNS;<xsl:value-of
        select="concat('cpv-',$id)"/>
   </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

   b) Función tr_{skos}^{type}:

```
<xsl:when test="substring($id,5,4)='0000'">
  <xsl:call-template name="format-id">
    <xsl:with-param name="id" select="'
         clase'"/>
  </xsl:call-template>
</xsl:when>
```

   c) Función tr_{skos}^{broad}:

```
<xsl:when test="substring($id,6,3)='000'">
  <xsl:call-template name="generate-broader
       ">
    <xsl:with-param name="id" select="$id"/
         >
    <xsl:with-param name="start" select="1"
         />
    <xsl:with-param name="end" select="4"/>
    <xsl:with-param name="suffix" select="
         '000'"/>
  </xsl:call-template>
</xsl:when>
```

   d) Función tr_{skos}^{id}:

```
<xsl:element name="dc:identifier">
 <xsl:value-of select="$id"/>
</xsl:element>
```

   e) Función tr_{skos}^{label}:

```
<xsl:element name="skos:preLabel">
   <xsl:attribute name="xml:lang">
        <xsl:value-of select="$inputLang"/>
    </xsl:attribute>
   <xsl:value-of select="translate($
        description,'.','')"/>
</xsl:element>
```

Para el total de la CPV_{skos} en un solo idioma, los resultados han arrojado algunas estadísticas relevantes 1. Tamaño de archivo de generación: 3,9MB; 2. Número de tripletas: 41568

El procedimiento de transformacion del CPV que hemos descrito a lo largo de esta sección ha sido utilizado en la confección del tesauro del buscador del Boletín Oficial del Principado de Asturias desarrollado por la Fundación CTIC en el proyecto Esperteyu[12]. Este buscador combina técnicas híbridas de búsqueda semántica y sintáctica enriqueciendo la consulta inicial del usuario mediante terminología adecuada al vocabulario de los textos oficiales del Boletín. El buscador se centra en tres dominios concretos: empleo público, subvenciones y licitaciones. El uso de CPV_{skos} es facilita las búsquedas cuando el usuario está buscando anuncios o convocatorias de licitaciones: tanto concursos públicos como subastas.

[12] http://idi.fundacionctic.org/esperteyu/

5. Conclusiones y Trabajo Futuro

A lo largo de este artículo se ha justificado la interpretación de las PSCs como vocabularios controlados y se ha utilizado el vocabulario web SKOS Core para la representación de un metamodelo que englobe la semántica original común a todas las PSCs: Categorías jerarquizadas y sectores de productos. Además hemos presentado un método y un conjunto de transformaciones Tr_{skos} que permite generar automáticamente las pertinentes representaciones de las PSCs en RDF: PSC_{skos}. Hemos demostrado la validez de este método con la transformación de la CPV en CPV_{skos}.

Consideramos que, en el futuro, este trabajo puede continuarse en varias direcciones:

- Aplicar el método a otras PSCs importantes, como es el caso de la UNSPSC o e@Class. En este punto, también habrá que estudiar y analizar cómo se puede extender en SKOS Core para dar cuenta de los diccionarios de propiedades que algunas PSCs más sofisiticadas utilizan para un intercambio más rico de información.

- Generalizar el modelo de PSC_{skos} en una ontología que permita subsumir el resto de clasificaciones de productos bajo SKOS Core. La sencillez del método descrito en este artículo favorece la migración del resto de clasificaciones a este modelo de datos.

- Definir correspondencias entre las diferentes PSC_{skos} para integrar las distintas visiones contextuales que cada PSC tiene del mercado en un vocabulario común. Consideramos que su uso puede tener un gran impacto en el descubrimiento automático de nuevos servicios y en la interoperabilidad de sistemas en el ámbito de la Web Semántica.

Referencias

[1] Oscar Corcho and Asunción Gómez Pérez. Integrating e-commerce standards and initiatives in a multi-layered ontology.

[2] Dieter Fensel, Ying Ding, and Borys Omelayenko. Product data integration in b2b e-commerce. *IEEE-Intelligent E-Business*, pages 54–59, 2001.

[3] Martin Hepp. A methodology for deriving owl ontologies from products adn services categorization standards. (1):72–99, 2006.

[4] Martin Hepp. The true complexity of product representacion in the semantic web. In *Proceedings of the 14th European Conference on Information System (ECIS 006)*, Gothenburg, Sweden, 2006.

[5] Joerg Leukel, Volker Schmitz, and Frank-Deiter Dorloff. Exchange of catalog data in b2b relationships - analysis and improvement.

[6] Frank Manola and Eric Miller. RDF Resource Description Language primer. W3C recommendation, W3C, 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[7] Alistair Miles and Dan Brickley. SKOS Simple Knowledge Organisation Systems. W3C recommendation, W3C, 2005. http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102.