FACULTAD 1

Componente para la generación automática de ontologías de dominios

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Lázaro Carlos Martínez Odio

Tutores:

MSc. Aneyty Martín García

Ing. Leiny Amel Pons Flores

La Habana, junio 2018

Dedicatoria

Dedico el presente Trabajo de Diploma:

A mis padres por ser la luz que guía mi camino, por estar siempre a mi lado apoyándome y dándome su amor incondicional.

A mi hermana que la amo con todo mi corazón.

Declaración de autoría

Declaro por este medio que yo, Lázaro Carlos Martínez Odio, con carnet de identidad 94062304905, soy el autor principal del trabajo titulado Componente para la generación automática de ontologías de dominios, y reconocer a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos el presente documento a los __ días del mes de _____ del año 2018.

_____

Firma del Autor

Lázaro Carlos Martínez Odio

_____ _____

MSc. Aneyty Martín García

Firma del Tutor

Ing. Leiny Amel Pons Flores

Firma del Tutor

Resumen

La recuperación de información en World Wide Web se ha convertido en una de las áreas de investigación más importantes del mundo, como consecuencia del crecimiento acelerado de las fuentes de información. Dicho crecimiento ha dado paso a la necesidad de contar con mecanismos capaces de procesar y comprender dicha información y con ello resolver necesidades específicas. En ese sentido, el uso de motores de búsqueda es indispensable para poder encontrar contenido específico y relevante para el usuario. La plataforma de Contenidos Unificados para Búsqueda Avanzada (C.U.B.A), carece de una funcionalidad que permita desambiguar la información que contiene la misma.

En el documento se reflejan los resultados de la investigación realizada, describiéndose las principales características de las ontologías, metodologías y herramientas destinadas al desarrollo de las mismas. Se diseñó la arquitectura y diseño del componente propuesto, las herramientas y tecnologías utilizadas, y los artefactos generados en el proceso de desarrollo. El componente para la generación automática de ontologías de dominios permite a través de la desambiguación de la información definir mejor el criterio de búsqueda de los usuarios. De esta manera, se pretende apoyar el concepto de Web semántica cuya propuesta es clasificar, estructurar y anotar los recursos con semántica explícita para que puedan ser procesados por sistemas inteligentes.

Palabras claves: Ontología, conocimiento, búsqueda de información, generación.

Índice de Contenido

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

En la actualidad, con el desarrollo de las tecnologías y comunicaciones se genera un gran cúmulo de información en Internet. Las estadísticas del sitio Internet Live Stats muestran que cada minuto que pasa, los 2.700 millones de personas con acceso a Internet que se calcula que hay actualmente en el mundo envían más de 200 millones de correos electrónicos, realizan 2 millones de consultas a Google, suben 48 horas de vídeo a YouTube, escriben más de 100.000 mensajes en Twitter, publican casi 30.000 nuevos artículos en sitios como Tumblr o WordPress (InternetLiveStats, 2017). Internet facilita el acceso a la información generada desde cualquier parte del mundo. La administración de tal cantidad de información es una tarea cuya complejidad ha crecido durante los últimos años, determinado esto por la cantidad y complejidad de los documentos disponibles en la web y su constante cambio (Kuna, y otros, 2014).

La abundante información que se genera dificulta o imposibilita la búsqueda de forma manual. Los sistemas de recuperación de información o buscadores son las herramientas que facilitan el acceso a la información. Actividades como el almacenamiento, representación, análisis y mantenimiento de grandes volúmenes de datos son llevadas a cabo por herramientas de este tipo en diferentes contextos de aplicación (Kuna, y otros, 2016).

Los sistemas de recuperación de la información (SRI) están compuestos esencialmente por tres componentes principales:

- Subsistema de Recolección
- Subsistema de Indexación
- Componente de Visualización

La integración de estos componentes permite crear una herramienta cuyo objetivo es básicamente rastrear toda la web en busca de toda la información que se encuentra dispersa en la misma, posteriormente procede a su almacenamiento y luego brinda a través de interfaces o servicios los resultados solicitados por los usuarios (Rodríguez Leyva, y otros, 2016).

Los actuales buscadores web realizan la búsqueda de manera sintáctica, las búsquedas se realizan al tomar las palabras claves de la consulta y se obtiene como resultado una lista de documentos que incluyen dichas palabras en su contenido; luego, los resultados se muestran ordenando esta lista en base a estadísticas y nivel de visitas de las páginas pero sin considerar necesariamente el contexto buscando

en los documentos la coincidencia exacta de los términos ingresados en la cadena de búsqueda (Tello, 2012).

Los resultados de los buscadores en general son aceptables (cumplen de alguna manera los requerimientos de los usuarios), aunque en ocasiones, no existe relación con el objeto o sentido de la búsqueda. La información obtenida no pertenece a un mismo dominio, encontrándose así información personal, académica, comercial, entre otras, alejándonos así de nuestros intereses. La mayoría de la información que existe en la web está destinada para personas, no para máquinas, siendo incapaces las ultimas, de interpretar correctamente los datos. La información que se encuentra en textos y documentos necesita ser extraída y convertirla en un lenguaje procesable para las máquinas, lo que permite a las aplicaciones de software, utilizar la información para diversos propósitos (Tello, 2012).

Para promover una web donde sus páginas estén organizadas y codificadas de tal manera que los ordenadores sean capaces de efectuar inferencias y razonar a partir de sus contenidos, se desarrolla la web semántica. Lo que permite ahorrar tiempo y recursos que se invierten para la localización de la información.

La web semántica es considerada una rama de la inteligencia artificial y las tecnologías Web, que permite ordenar la información en el motor de búsqueda para evitar así la redundancia, el exceso de información, mejorar las búsquedas, reduce el costo y el tiempo que se invierte al consultar información útil en Internet, además, mantiene una búsqueda coherente que permite que las computadoras puedan procesar la información de manera inteligente. Es un área de influencia en la Inteligencia Artificial y las tecnologías web, que propone introducir descripciones explícitas sobre el significado de los recursos, para permitir que las propias máquinas tengan un nivel de comprensión de la web suficiente como para hacerse cargo de una parte del trabajo que actualmente es realizada manualmente por los usuarios que navegan e interactúan con la web. Dentro de la web, la necesidad de enfocarse al resultado de información útil ha permitido establecer la idea de implementar la semántica a las búsquedas y dejar de lado las búsquedas basadas puramente en sintaxis. La estructura de las búsquedas basadas en sintaxis tiene la desventaja que al momento de realizar la consulta pueden tomar cualquiera de los elementos que conforman esa estructura e interpretarlo de acuerdo con esa palabra (Rodríguez, y otros, 2017). La Web semántica impulsa la anotación de documentos o contenidos generales de recursos Web mediante la creación de metadatos, utilizando la información semántica de las ontologías de dominio (De Maio et al. 2014).

El centro de Ideo- informática (CIDI), perteneciente a la Facultad 1 de la Universidad de las Ciencias Informáticas (UCI), desarrolla la plataforma de Contenidos Unificados para Búsqueda Avanzada (C.U.B.A). La plataforma C.U.B.A es un buscador cubano basado en software libre, brinda acceso a los contenidos publicados en la red cubana, logrando que dicha información sea accedida por los usuarios y responde a las búsquedas realizadas por ellos emitiendo un listado de los sitios que coinciden con los criterios introducidos. C.U.B.A está basada en la tecnología del motor de búsqueda Orión, desarrollado por la misma universidad. Su arquitectura está basada en componentes. La plataforma C.U.B. A indexa en su base de datos más de 6800 sitios bajo el dominio.cu (nic.cuba.cu, 2017). También los servicios que ofertan como Cartelera cultural “La Papeleta”, la plataforma de blogs “Reflejos” y la aplicación de mapas interactivos “El Andariego” entre otros.

La plataforma de Contenidos Unificados para Búsqueda Avanzada carece de una herramienta que elimine la ambigüedad de términos, ya que los motores de búsqueda no se enfocan en un modelo de dominio específico sino en cualquier fuente disponible a la que se tenga acceso. Lo cual puede generar que se obtenga como resultados documentos que contienen los términos de la consulta, pero en un contexto irrelevante para el usuario. Es por ello que se necesita un modelo de dominio sobre el cual trabajar. La desambiguación del conocimiento permite conocer mejor la intención de búsqueda del usuario. Tampoco es posible procesar semánticamente el conocimiento para representarlo en un lenguaje formal, con el fin de facilitar el almacenamiento de los metadatos relacionados.

Todos estos elementos conllevan a que se plantee como **problema de investigación:** ¿Cómo desambiguar la información, contenida en la plataforma de Contenidos Unificados para Búsqueda Avanzada?

Se define como **objeto de estudio** el proceso de generación de ontologías de dominio.

Se plantea como **objetivo general**: Desarrollar un componente que permita la desambiguación de la información en la plataforma de Contenidos Unificados para Búsqueda Avanzada mediante el uso de ontologías.

Se establece como **campo de acción:** el proceso de generación automática de ontologías de dominio.

Objetivos específicos que conducirán al logro del objetivo general**:**

1- Caracterizar los fundamentos teóricos relacionados a la generación automática de ontologías de

dominios.

2- Definir las tecnologías, herramientas y metodología de desarrollo para la implementación del componente para la generación automática de ontologías de dominios.

3- Identificar los requisitos funcionales y no funcionales del componente para la generación automática de ontologías de dominio desarrollar.

4- Diseñar e implementar las funcionalidades del componente de generación automática de ontologías de dominios para la plataforma C.U.B.A.

5- Validar el componente de generación automática de ontologías de dominios para la plataforma C.U.B.A.

Se plantea como **hipótesis de la investigación**: la implementación de un componente para la desambiguación de la información permitirá determinar mejor la intención de búsqueda del usuario.

De la hipótesis anterior se define la siguiente **variable independiente**: componente para la desambiguación de la información.

Como **variable dependiente**: determinar mejor la intención de búsqueda del usuario.

Entre los **Métodos de investigación** utilizados para darle cumplimiento a estas tareas se encuentran:

Métodos teóricos:

Analítico – sintético: se utilizó en la búsqueda y análisis de los elementos más relevantes para el estudio según las fuentes bibliográficas consultadas: libros, páginas web y artículos, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Análisis histórico – lógico: su uso permitió una mayor comprensión del estado y tendencias actuales de las ontologías, así como su evolución, también para determinar el comportamiento de las herramientas para el desarrollo de las ontologías.

Inductivo-Deductivo: para el estudio de las principales tendencias e ideas para la generación de ontologías y los algoritmos utilizados para lograrlo con el objetivo de determinar cuáles son las alternativas viables a incorporar en la presente investigación.

Modelación: Se utilizó para modelar los principales procesos de negocio en la plataforma C.U.B.A, y alcanzar un entendimiento común de los procesos con el menor esfuerzo posible.

Métodos empíricos:

Observación: se utilizó con la finalidad de caracterizar y analizar detalladamente cómo se realiza actualmente el proceso de búsqueda de información en la plataforma de Contenido Unificado para la Búsqueda Avanzada, identificando los mecanismos para desarrollar un componente acorde con las necesidades.

La presente investigación está estructurada en 3 capítulos, además de las secciones de Conclusiones, Recomendaciones, Bibliografía y Anexos.

Capítulo 1: "Estudio del estado teórico de las ontologías y tecnologías para su desarrollo": se realiza un análisis de los principales conceptos y elementos teóricos asociados a las ontologías, imprescindibles para la comprensión de la investigación. Además, se puede apreciar, una selección de ontologías publicadas en el mundo. También se describen las herramientas, tecnologías, conjunto de técnicas y metodología para el desarrollo del componente.

Capítulo 2. "Diseño del componente para la generación automática de ontologías de dominio": en este capítulo se expone la propuesta y características del componente. También se definen los diversos artefactos que especifica el proceso de desarrollo de software utilizado.

Capítulo 3: "Implementación y prueba del componente de generación automática de ontologías de dominios": se ejecuta la implementación y validación de la solución al problema planteado. Además, se realizan y describen las pruebas de software a la aplicación para garantizar la calidad requerida y que cumpla con los requisitos funcionales establecidos.

Capítulo 1: “Estudio del estado teórico de las ontologías y tecnologías para su desarrollo”

En el presente capítulo se hace alusión a las características y definiciones generales de las ontologías, que rigen la presente investigación. Se describen los conceptos fundamentales asociados al dominio del problema y el objeto de estudio, haciéndose un análisis de la situación actual. Se analizan las soluciones semejantes a la propuesta de solución; así como la fundamentación de la metodología, lenguajes, tecnologías y herramientas que se utilizarán para el desarrollo de la aplicación y la construcción de la solución.

1.1 Marco conceptual

Cada día el usuario necesita dedicar más tiempo para filtrar las páginas web que devuelve una búsqueda, es decir, la calidad de la información es cada vez peor. Unido al incremento sustancial de información en Internet y a la necesidad de aprovechar al máximo la gran cantidad de recursos disponibles, han dado a la luz proyectos como la Web Semántica, según (Codina, y otros, 2006) se define como “un conjunto de iniciativas destinadas a promover una futura Web cuyas páginas estén organizadas, estructuradas y codificadas de tal manera que los ordenadores sean capaces de efectuar inferencias y razonar a partir de sus contenidos.”

El valor social de la Web es que permite la comunicación humana, el comercio y las oportunidades de compartir conocimiento. Uno de los principales objetivos del W3C es hacer que estos beneficios estén disponibles para todas las personas, independientemente de su hardware, software, infraestructura de red, idioma nativo, cultura, ubicación geográfica o capacidad física o mental (W3C, 2017).

Entre los objetivos de la Web semántica se encuentra la posibilidad de que sea posible sostener una interacción entre un usuario y un agente de software mediante el cual el primero pueda ir expresando y perfilando sin ambigüedad puntos como los siguientes: objetivos de la búsqueda, géneros documentales pertinentes, punto de vista, granularidad esperada en la respuesta, etc. Se espera que el agente de software sea capaz de elaborar una estrategia de búsqueda según su propia iniciativa (la del agente de software) que involucre el uso de lenguajes documentales, metadatos y ontologías para responder con eficacia y rapidez al usuario (Codina, y otros, 2006).

La Web Semántica se basa en estándares que permitirán gestionar computacionalmente la información de un dominio de conocimiento. Se parte de la idea de definir una ontología en términos computables, con el

objetivo de que las páginas web que se puedan asociar a esa ontología permitan hacer búsquedas inteligentes. Se entiende por búsquedas inteligentes, aquellas que puedan hacer inferencias con lógica descriptiva (Fernández, 2009).

Existen diferentes definiciones de lo que son las ontologías, según (Breuker, 1999), ¨Una ontología es una representación explícita de una conceptualización cognitiva, es decir, la descripción de los componentes de conocimiento relevantes en el ámbito de la modelización. También se encuentra la caracterización desde el punto de vista práctico, como herramienta, de acuerdo Weingand, una ontología es una base de datos que describe los conceptos generales o sobre un dominio, algunas de sus propiedades y cómo los conceptos se relacionan unos con otros (Weingand, 1997).

Por lo anterior expuesto el autor de esta investigación puede concluir que:

Las ontologías son una estructura conceptual sistematizada donde son representados los términos y las relaciones que existen entre los mismos, pertenecientes a un área o dominio del conocimiento en particular.

Según (GRUBER, 1993), las ontologías se componen de:

- **conceptos**: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.
- **relaciones**: representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.
- **funciones**: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como: asignar-fecha, categorizar-clase, etc.
- **instancias**: se utilizan para representar objetos determinados de un concepto.
- **reglas de restricción o axiomas**: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición B1, A es C", etc. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

Algunas de las características de las ontologías son:

- Pueden existir ontologías múltiples: si el propósito de una ontología es hacer explícito algún punto de vista, en algunos casos, necesitamos combinar dos o más ontologías. Cada ontología introduce conceptualizaciones específicas.

- Se pueden identificar distintos niveles de abstracción estableciendo una topología de ontologías: se puede caracterizar una red de ontologías usando multiplicidad y abstracción. Al no poder realizar una descripción completa del mundo, se puede pensar una estrategia de construcción gradual que vaya de abajo hacia arriba.

- Multiplicidad de la representación: un concepto puede ser representado de muchas formas, por lo que pueden coexistir múltiples representaciones del mismo concepto.

- Mapeo de ontologías: se pueden establecer las relaciones entre los elementos de una o más ontologías para establecer generalizaciones, especializaciones, conexiones, etc.

Según el ámbito del conocimiento al que se apliquen se pueden clasificar en tres tipos: Ontologías generales, son las ontologías de nivel más alto ya que describen conceptos generales (espacio, tiempo, materia, objeto, etc.). Ontologías de dominio, describen el vocabulario de un dominio concreto del conocimiento. Ontologías específicas, son ontologías especializadas que describen los conceptos para un campo limitado del conocimiento o una aplicación concreta (Lapuente, 2013).

Una ontología es un sistema para la representación del conocimiento que resulta de seleccionar un dominio o ámbito del conocimiento, y aplicar sobre él un método con el fin de obtener una representación formal de los conceptos que contiene y de las relaciones que existen entre dichos conceptos. Además, una ontología contiene definiciones que nos proveen del vocabulario para describir a un dominio. Estas definiciones están sujetas al tipo de lenguaje que se utilice, para referirse a dicho dominio. Todas las conceptualizaciones (definiciones, categorizaciones, jerarquías, propiedades, herencia, etc.) de una ontología pueden ser interpretadas por máquina. Las aplicaciones de las ontologías son diversas, desde servir de herramientas de referencia en la construcción de sistemas de bases de conocimiento que aporten consistencia, fiabilidad y falta de ambigüedad a la hora de recuperar información, hasta establecer modelos normativos que permitan la creación de la semántica de un sistema y un modelo para poder extenderlo y transformarlo entre diferentes contextos.

1.2 Estudio de sistemas homólogos existentes en el ámbito internacional

En la actualidad existen diferentes herramientas que se utilizan para generar aplicaciones basadas en el uso de ontologías. Fueron analizados XML2OWL, X2OWL, JANUS y la desarrollada por N. Yahia, S. Mokhtar y A Ahmed. Su selección se debió a que son algunas de las que trabajan con documentos XML como parámetros de entrada. La Web semántica se basa en la capacidad de XML para definir esquemas de etiquetas a medida y en la aproximación flexible de RDF para representar datos. El lenguaje XML es ampliamente conocido en muchos ámbitos para estructurar información. Permite la definición de vocabularios de naturaleza semántica para la descripción del contenido (Mora, y otros, 2015).

XML2OWL[1]

Bohring y Auer han desarrollado una herramienta, llamada xml2owl, para crear una ontología OWL a partir de un esquema XML y convertir datos XML a instancias OWL que cumplan con la ontología creada. Técnicamente, han desarrollado cuatro instancias de Transformación de lenguaje de hojas de estilo extensible (XSLT) para transformar archivos XML a OWL, sin ninguna otra intervención en semántica y estructuras durante la transformación. Finalmente, el archivo OWL (ontología de lectura) se genera automáticamente. Este método también se aplicó a la plataforma Ontowiki [2]. Los enfoques de Bohring y Ferdinand son similares a los que se basan en un conjunto de reglas de transformación de Esquema XML a OWL. Propusieron que los tipos complejos con nombre XML Schema Definition (XSD) con contenido complejo (es decir, una combinación de atributos y secuencia de elementos) se transformaran en clases OWL.

Cuando un elemento contiene otro elemento, se crea una propiedad de objeto OWL entre sus correspondientes clases OWL. Las propiedades de tipo de datos OWL surgen de los atributos XML y del elemento que contiene solo un literal y sin atributos. Las restricciones de aritmética del esquema XML, como minOccurs o maxOccurs, se asignan a las restricciones de cardinalidad equivalentes en OWL, minCardinality y maxCardinality. Durante la creación de la ontología desde un esquema XML, el usuario no tiene control sobre el proceso. El usuario no tiene control sobre la nueva ontología creada que captura

[1] H. Bohring, and S. Auer, "Mapping XML to OWL Ontologies", In Leipziger Informatik-Tage, vol. 72, 2005, pp. 147–156. Society, Washington, DC, USA, (2007).
[2] S. Auer, S. Dietzold, and T. Riechert, "OntoWiki – A Tool for Social, Semantic Collaboration", 5th International Semantic Web Conference, Nov 5th-9th, Athens, GA, USA. In I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273, 2006, pp. 736–749.

la semántica implícita existente en la estructura del esquema XML. Por lo tanto, las ontologías creadas son bastante primitivas, es decir, no son semánticamente más ricas que los esquemas XML asignados. Además, no mencionan cómo aplicarlo en múltiples documentos fuente. Los enfoques de Ferdinand y Bohring introducen una buena base de reglas para crear ontologías OWL a partir de XML Schema. Sin embargo, abordan solo casos simples y no se refieren a casos complejos que surgen de la reutilización de tipos y elementos globales. Además, no mencionan cómo especificar asignaciones entre múltiples orígenes de datos XML y ontologías OWL generadas (Yahia, y otros, 2012).

X2OWL[3]

Diseñada por R. Ghawi y N. Cullot es una herramienta, que tiene como objetivo construir una ontología OWL a partir de una fuente de datos XML. Este método se basa en el esquema XML para generar automáticamente la estructura ontológica, posee un conjunto de puentes de mapeo entre las entidades de la fuente de datos XML y la ontología creada. Los puentes de mapeo contribuyen a la traducción de consultas entre OWL y XML. Este enfoque aborda casos simples y casos complejos que surgen de la reutilización de tipos y elementos globales que se utilizan para crear un esquema XML. Los esquemas XML se pueden modelar utilizando diferentes estilos, algunos de ellos usan un único elemento global (elemento raíz) y otros usan múltiples elementos globales, algunos estilos usan tipos globales y otros usan solo tipos locales. Sin embargo, el método de mapeo debe hacer frente a todos los patrones de diseño posibles [11]. El método presentado también incluye un paso de refinamiento que permite limpiar los puentes de mapeo y re-estructurar la ontología generada, aunque el nuevo enfoque propuesto tendrá en cuenta cómo construir ontología a partir de múltiples fuentes de datos XML (Yahia, y otros, 2012).

JANUS[4]

I. Bedini y N. Benjamin propusieron un marco llamado Janus que genera una ontología a partir de un gran corpus fuente de esquemas XML. Presentan un conjunto de patrones que permiten la transformación

[3] R. Ghawi, and N. Cullot, "Building Ontologies from XML Data Sources", In 1st International Workshop on Modelling and Visualization of XML and Semantic Web Data (MoViX '09), held in conjunction with DEXA'09 (Linz, Austria, September 2009).

[4] I. Bedini, G. Gardarin, and B. Nguyen, "Transforming XML Schema to OWL Using Patterns", University of Versailles, France. 5th IEEE International Conference on Semantic Computing (ICSC), Palo Alto: United States (2011).

directa y automática de XML Schema en OWL, permitiendo la integración de mucha información XML en la Web Semántica. Se centran en una representación lógica avanzada de componentes de esquema XML y presentan una implementación que permite extraer fuentes de esquemas XML para extraer el conocimiento suficiente para crear ontologías semánticamente correctas con considerable expresividad. Se demostró que su implementación era más completa que otras, y en particular mejora enormemente el número de patrones de transformación complejos con respecto al enfoque de Bohring. Este enfoque no aborda la cuestión de cómo crear la ontología OWL, si no hay ningún esquema XML disponible (Yahia, y otros, 2012).

TABLA 1: COMPARACIÓN ENTRE SISTEMAS

Sistema	Documento XML de entrada	Descripción del esquema XML	Método de mapeo	Lenguaje de Ontología
XML2OWL	Documento XML único	El esquema XML contiene solo casos simples	Basado en XML esquema usando archivos XSLT	OWL
X2OWL	Documento XML único	El esquema XML contiene casos simples y complejos	Basado en XML Schema y XSG usando puentes de mapeo y archivos Xpath	OWL
JANUS	Documento XML no disponible	Los esquemas XML heterogéneos fusionados contienen casos simples y complejos	Mapeo Surjective basado en técnicas de minería XML	OWL
N. Yahia , S. Mokhtar and A Ahmed	Documentos XML heterogéneos	Los esquemas XML heterogéneos contienen casos simples y complejos	Basado en XML Schema y XSG	OWL

Para la comparación se definieron los criterios de comparación:

1. Documento XML de entrada

En el modelo de web semántica definido por Berners-Lee en el 2001 define los archivos XML como la base de la semántica ya que proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.

2. Descripción del esquema XML

Es un lenguaje que se utiliza para restringir la estructura de los documentos XML, además de para ampliar XML con tipos de datos.

3. Método de mapeo

Es necesario definir un método de extraer e identificarlos, para poder obtener los atributos y elementos contenidos en los archivos.

4. Lenguaje de Ontología

Es el lenguaje que se utiliza para la creación de la ontología y describir formalmente el significado de la terminología usada en los documentos Web.

Después de realizado el estudio del entorno actual de los sistemas se puede concluir que, a pesar de que las soluciones estudiadas presentan algunas dificultades como: la incapacidad de analizar XML con casos complejos, como consecuencia, se reduce la diversidad de documentos a analizar. También algunos enfoques apuntan más a un mapeo general entre XML y RDF que otros apuntan a mapear XML Schema a OWL. OWL y RDF son casi lo mismo. Es importante señalar que a pesar que todas las herramientas mencionadas anteriormente, utilizan el lenguaje OWL, no es una desventaja con respecto a usar lenguaje RDF, ya que OWL es un lenguaje más fuerte con una mayor capacidad de interpretación de la máquina que RDF. El lenguaje para el desarrollo de ontologías OWL viene con un vocabulario más amplio y una sintaxis más fuerte que RDF. OWL puede ser usado para representar explícitamente el significado de términos en vocabularios y las relaciones entre esos términos. OWL tiene mayor capacidad para expresar significado y semántica que XML, RDF, y RDF-S, y, de este modo, OWL va más allá de estos lenguajes en su capacidad para representar contenido interpretable por un ordenador en la Web.

El sistema desarrollado por N. Yahia, S. Mokhtar and A Ahmed es el más completo debido a:

1. El método de mapeo que emplean, basado en XML schema y XSG, es el más efectivo ya que permite el análisis de XML simples y también complejos.

2. Utilizan lenguaje el lenguaje para el desarrollo de ontologías OWL que es un lenguaje más fuerte con una mayor capacidad de interpretación de la máquina que RDF.

3. Acepta como entrada documentos XML heterogéneos, es decir los que tengan tipos de variables complejas.

1.3 Estudio de sistemas homólogos existentes en el ámbito nacional

En el ámbito nacional referente al área de la generación de ontologías se tiene las investigaciones de Manuel E. Puebla-Martínez, José M. Perea-Ortega y Alfredo Simón Cuevas titulada “Integración de técnicas browsing con ontologías en un modelo de recuperación de información geográfica” del 2015 y la presentada en el XV Congreso Internacional de Información del 2018 denominada “Generación Semiautomática de una Ontología Geográfica a partir de Fuentes Heterogéneas”. Ambas enmarcadas al estudio Recuperación de Información Geográfica.

1.4 Herramientas para el desarrollo de ontologías

En el proceso de construcción de ontologías intervienen varios pasos como investigar el dominio, localizar los conceptos importantes del dominio, crear la representación de los objetos y relaciones, para luego realizar la captura de la ontología, codificación de la misma y la integración con otras ontologías. El componente no se le asociará un dominio en específico.

La selección de las herramientas para la creación y el desarrollo de ontologías es importante. Por ello se pueden clasificar en dos categorías (Singh, et al., 2013).

1. **Herramientas para el desarrollo:** Son editores de ontología que permiten a los usuarios definir nuevos conceptos, relaciones, e instancias. Por lo general estas herramientas poseen la capacidad de importar y exportar otras ontologías. Protégé 2000, OntoEdit, OilEd, WebODE, y Ontolingua, son ejemplos de herramientas de desarrollo.

2. **Herramientas para mapeo, alineación y fusión de ontologías:** Estas son las herramientas de mapeo ontológico que ayudan a los usuarios a encontrar similitudes y diferencias entre las ontologías de origen. Las herramientas de mapeo identifican las correspondencias potenciales automáticamente o proporcionar el entorno para que los usuarios las detecten y define estas correspondencias, o ambas.

Cartografía las herramientas a menudo son extensiones de herramientas de desarrollo. PROMPT, ONION, Chimaera etc.

En la actualidad existen diversos sistemas para la construcción de ontologías, como Protégé, Ontolingua Server y OntoEdit Ontoprise. A continuación, se detallan algunas de las características de estos sistemas.

Ontolingua Server: El servidor de Ontolingua[5] fue la primera herramienta de ontología creada y desarrollado a principios de los años noventa en Knowledge Systems Laboratorio (KSL) en la Universidad de Stanford. Fue construido para facilitar el desarrollo de ontologías Ontolingua con base en formularios aplicaciones. La creación de una nueva ontología se vuelve más fácil por la flexibilidad de incluir partes de ontologías existentes de un repositorio. Este repositorio consiste en una gran cantidad de ontologías de diferentes campos. Después de la finalización de nuevo ontología generada, se puede agregar al repositorio para posible reutilizar. El editor de ontologías también incluye traductores a los idiomas como Prolog, CORBA's, y Loom, etc. Los editores remotos pueden explorar y editar ontologías y aplicaciones remotas o locales para acceder a cualquiera de las ontologías en la biblioteca de ontología con Protocolo OKBC.

Protégé

Esta herramienta se ha desarrollado por Stanford Informática Médica (SMI). Es una aplicación de código abierto, autónoma con una arquitectura extensible. El módulo principal está enfocado a la edición de la ontología y sostiene biblioteca de *plugins* eso agregan más funcionalidades al ambiente. Protegido 4.2 es la última versión en la línea del protegido de herramientas soltada en 2013. Esta versión tiene varios nuevo *plugins,* incluso una la herramienta de diferencia de ontología, el apoyo para la explicación reforzada, y el apoyo para la generación del código.

Apache Jena[6]

Jena se desarrolló en HP Labs en el 2000, en 2009 HP cedió el proyecto a la fundación Apache que decidió adoptarlo en noviembre de 2010. Jena es un marco de Java para construir aplicaciones de Web Semántica. Proporciona una extensa biblioteca de Java para ayudar a los desarrolladores a desarrollar código que maneja RDF, RDFS, RDFa, OWL y SPARQL de acuerdo con las recomendaciones publicadas

[5] http://ontolingua.stanford.edu/
[6] https://jena.apache.org.

del W3C. Jena incluye un motor de inferencia basado en reglas para realizar un razonamiento basado en ontologías OWL y RDFS, y una variedad de estrategias de almacenamiento para almacenar RDF se triplica en la memoria o en el disco. El framework tiene varios razonadores internos y el razonador Pellet (un razonador Java OWL-DL de código abierto) puede configurarse para funcionar en Jena.

Su arquitectura incluye:

- API para trabajar (leer, procesar, escribir) ontologías RDF y OWL
- Motor de inferencia para razonar sobre ontologías RDF y OWL
- Estrategias de almacenamiento flexible para almacenar tripletas RDF en memoria o fichero
- Motor de consultas compatible con especificación SPARQL

OntoEdit Ontoprise

Se ha desarrollado por AIFB en la Universidad de Karlsruhe. Desarrollada de manera extensible y flexible basada en plugins que proporcionan las funcionalidades para hojear y revisar las ontologías. Esta herramienta está disponible en dos versiones, es decir Gratuitamente y Profesional. Incluye el plugins para exportar e importar las ontologías en los formatos diferentes. La herramienta está basada en una plataforma flexible. Primeramente, permite extenderse la funcionalidad a la manera modular. La interfaz del plugin se abre a terceras partes que les permiten a los usuarios que extiendan OntoEdit fácilmente con las funcionalidades adicionalmente requeridas. En segundo lugar, teniendo un juego de plugins disponible como un léxico del dominio, plugin de la inferencia y algunos plugins de importación y de exportación, permiten al usuario el personalizar la herramienta de manera amistosa para diferentes escenarios. También proporciona el desarrollo colaborativo de ontologías para el tejido semántico.

TABLA 2: COMPARACIÓN ENTRE LAS HERRAMIENTAS

Criterios	**Ontolingua Sever**	**Protégé**	**OntoEdit Ontoprise**	**Apache Jena**
Políticas de Privacidad	Código abierto	Código abierto	Libre distribución y licencias	Código abierto

Arquitectura	Cliente/Servidor	Cliente/Servidor	Autónomo	Autónomo
Almacena-miento	Archivos	Archivos	Archivos/DBMS	Archivos
Formato de importación	Ontolingua IDL KIF	XML, RDF(S), XML Schema	XML, RDF(S), FLogic, DAML+ OIL	RDF(S), SPARQL
Formato de exportación	KIF-3.0, CLIPS CML, LOOM, OKBC Syntax	XML, RDF(S) ,XML Schema, FLogic, CLIPS, Java, HTML	XML, RDF(S), FLogic, DAML+OIL,SQL-3	OWL, SPARQL, RDF(S)

Existen diversas herramientas para el desarrollo de ontologías, algunas para propósitos específicos y otras con propósitos generales. Las mismas tienen diferentes características en cuanto a las políticas de privacidad, las cuales pueden ser de código abierto o mediante licencias y distribución libre, debido a las circunstancias de nuestro país el uso de herramientas de código abierto permite el mantenimiento de las aplicaciones desarrolladas. Siendo un factor clave en la selección de la herramienta. Luego de realizado el análisis se seleccionó para el desarrollo de las ontologías a la herramienta Apache Jena en su versión 3.0 debido a:

- Es una herramienta de código abierto.
- Posee un motor de inferencia para razonar sobre ontologías RDF y OWL.
- Ostenta una extensa biblioteca de Java.

1.5 Metodologías y métodos

Para el diseño de cualquier ontología es necesario seleccionar una metodología específica. Son muchas las propuestas existentes. De entre ellas podemos destacar algunas como la metodología CYC, publicada por Lenat y Guha desde 1990 (D.B.t, y otros, 1990), en la que divulgaron algunos pasos generales para la construcción de ontologías. El primero paso consiste en extraer manualmente el conocimiento común que está implícito en diferentes fuentes para después, cuando se tenga suficiente conocimiento en la

ontología, adquirir nuevo conocimiento común usando herramientas de procesamiento de lenguaje natural o aprendizaje computacional.

Metodología de GRÜNINGER Y FOX

Surge en 1995, dentro de esta metodología las cuestiones de competencia hacen referencias a consultas a las cuales la ontología debería responder. En esta metodología se proponen los siguientes pasos: Definición de los escenarios motivadores, consiste en identificar intuitivamente las aplicaciones posibles en las que se usará la ontología. Luego la formulación de preguntas en lenguaje natural, a las que se les denomina cuestiones de competencia, esto con la finalidad de determinar el ámbito de la ontología. Se usan estas preguntas para extraer los conceptos principales, sus propiedades, relaciones y axiomas, los cuales se definen formalmente en Prolog. Posteriormente la especificación de la terminología, es decir en base a las preguntas realizadas en el paso anterior, se define conceptos principales, relaciones, propiedades, etc. Formalización de las interrogantes. Especificación de axiomas formales. Por último, la verificación de la ontología.

Metodología METHONTOLOGY

Esta es una de las propuestas más completas ya que toma la creación de ontologías como un proyecto informático. Así, además de las actividades propias de la generación de la ontología esta metodología abarca actividades para la planificación del proyecto, la calidad del resultado, la documentación, etc. Además, permite construir ontologías totalmente nuevas o reutilizar otras ontologías. El entorno incluye la identificación del proceso de desarrollo de la ontología donde se incluyen las principales actividades (evaluación, conceptualización, configuración, integración, implementación, etc.), un ciclo de vida basado en prototipos evolucionados y la metodología propiamente dicha, que especifica los pasos a ejecutar en cada actividad, las técnicas usadas, los productos a obtener y su forma de evaluación. Esta metodología está parcialmente soportada por el entorno de desarrollo ontológico WebODE y propone las siguientes etapas: (1) especificación, (2) conceptualización, (3) formalización, (4) implementación y (5) mantenimiento (A.Gomez-Perez, y otros, 1996).

Metodología basada en SENSUS

En 1997, un nuevo método fue propuesto para construir ontologías, este estaba basado en la ontología de SENSUS. La cual constituye un enfoque top-down para derivar ontologías específicas del dominio a partir

de grandes ontologías. En esta metodología se identifican un conjunto de términos semilla que son relevantes en un dominio particular. Tales términos se enlazan manualmente a una ontología de amplia cobertura. Los usuarios seleccionan automáticamente los términos relevantes para describir el dominio y acotar la ontología Sensus. Consecuentemente, el algoritmo devuelve el conjunto de términos estructurados jerárquicamente para describir un dominio, que puede ser usado como esqueleto para la base de conocimiento (B.Swartout, y otros, 1997).

Metodología TERMINAE

TERMINAE la cual aporta tanto una metodología como una herramienta para la construcción de ontologías a partir de textos. Se basa en un análisis lingüístico de los textos, el cual se realiza mediante la aplicación de diferentes herramientas para el procesamiento del lenguaje natural. En particular se usan dos herramientas: (1) Syntex para identificar términos y relaciones; y (2) Caméléon para identificar roles o relaciones. La metodología funciona como sigue. Mediante la aplicación de Syntex obtenemos una lista de posibles palabras y frases del texto y algunas dependencias sintácticas y gramaticales entre ellas. Estos datos se usan como entrada para el proceso de modelado junto con el texto original (Corcho, y otros, 2003).

Metodología ONTOLOGY DEVELOPMENT 101

Propuesta por la Universidad de Stanford EEUU, en donde sus principales recomendaciones radican en: (1) Determinar el dominio y ámbito de la ontología, (2) Determinar la intención de uso de la ontología, (3) Reutilizar ontologías o vocabularios controlados existentes. (4) Enumerar los términos importantes del dominio (5) Definir jerarquía de clases. (6) Crear las instancias (Yahia, y otros, 2012).

Metodología Ding y Foo

Realizan un repaso a cerca de los métodos más empleados: (1) Datos fuente: Vocabularios controlados, corpus de sentencias, extracción de texto libre, preguntas a usuarios. (2) Métodos para la extracción de conceptos: las diferentes técnicas empleadas para la extracción de información (análisis sintáctico, procesamiento del lenguaje natural, implicación humana, etc.) (3) Métodos para la extracción de relaciones: puede ser de forma automática o basándose en algoritmos que en ocasiones se aplican de forma manual. (4) Reutilización de ontologías: puede ser habitual utilizar otros instrumentos

terminológicos. (5) representación de la ontología, que va desde la estructura jerárquica, pasando por la lógica de descripción hasta los grafos conceptuales y el XML (Ding, y otros, 2002).

Metodología de M. Ferdinand

Ferdinand propuso asignaciones directas de XML Schema a OWL, así como describieron las asignaciones de gráficos XML a RDF, pero estas asignaciones son independientes entre sí, es decir, las instancias generadas no necesariamente respetan la ontología creada a partir del Esquema XML. El proceso de asignación de esquema XML a OWL se basa en un conjunto de reglas de interpretación y transformación desde XML Schema hasta OWL. XML Schema complexType, las definiciones de grupos de modelos y las definiciones de grupos de atributos se asignan a las clases OWL. La declaración de elemento y atributo se asigna a una propiedad OWL Más precisamente, los elementos de simpleType y todos los atributos se asignan a OWL: DatatypeProperty; elementos de complexType son mapeado a un OWL: ObjectProperty. El Esquema XML admite dos tipos de herencia: herencia por restricción y herencia por extensión. Ambos se asignan al único mecanismo de herencia en OWL: rdfs: subClassOf. Las asignaciones propuestas en este enfoque están destinadas a ser aplicadas para la ingeniería de aplicaciones web. En particular, para mejorar los lenguajes y herramientas XML tradicionales por las capacidades de los razonadores OWL. Este enfoque no aborda la cuestión de cómo crear el modelo OWL, si no hay ningún esquema XML disponible (Ferdinand, y otros, 2004).

Metodología a utilizar

El método definido por Yahia, y otros, (2012) utiliza las mismas anotaciones definida por R. Ghawi y N. Cullot Ghawi, y otros, (2009) , con algunas modificaciones para aplicar en múltiples fuentes de datos XML. El nuevo enfoque propuesto se basa en el esquema XML para construir la ontología. Si el esquema no existe, puede generarse automáticamente a partir del documento XML de origen, este método resuelve todos los casos complejos posibles que surjan de diferentes estilos de diseño de esquema XML.

La generación de ontología OWL a partir de fuentes de datos XML podría describirse en 4 pasos que son:

1. El documento XML se transforma a XML-Schema
2. El XML-Schema se analiza utilizando XML-Schema Object Model (XSOM)
3. La salida de XSOM se utiliza como entrada para el marco de Java Universal Network / Graph (JUNG).

4. Se utiliza XSG como entrada para generar entidades OWL.

1.6 Selección del entorno de desarrollo para la construcción de la solución

Para el desarrollo de la propuesta de solución se hace indispensable la definición de las tecnologías, herramientas y metodología que se utilizarán, disponibles para la realización del objetivo general de la investigación. Esta selección se hizo en función de las herramientas que se utilizan en la plataforma y fueron escogidas por el arquitecto del proyecto.

1.6.1 Lenguaje de desarrollo

Un lenguaje de programación es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (Lutz, 2010).

Se pueden clasificar de dos formas diferentes; de acuerdo a la arquitectura Cliente/Servidor, los lenguajes del lado del servidor son aquellos reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él; por otro lado, los lenguajes del lado del cliente son aquellos que pueden ser directamente comprendidos por el navegador y no necesitan un pre tratamiento. A continuación, se presentan los lenguajes utilizados por parte del cliente.

Para el lado del cliente

Java

Es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. El modelo de Java para la gestión de la memoria, los procesos múltiples y la gestión de excepciones lo convierte en un lenguaje eficaz para los desarrolladores nuevos y para los más experimentados. Es una de las plataformas de aplicaciones más populares que existen y proporciona un interesante ecosistema de desarrolladores impulsado por herramientas eficaces, libros, bibliotecas, muestras de código, entro otros aspectos [[7]].

Se decidió el uso de Java como lenguaje de programación, ya que, posee un lenguaje multi-plataforma, es decir, el código que es escrito en java es leído por un intérprete, por lo que el programa se ejecutará en

[7] Oracle y Java - Características | Tecnologías | Oracle ES. 2014. dirección:http://www.oracle.com/es/technologies/java/features/index.html.

cualquier plataforma, no necesitas conocer a priori el tipo de ordenador o el sistema operativo para el cual se programará. Otra ventaja del uso de Java es que el manejo de la memoria se hace automáticamente. El lenguaje Java es orientado a objetos. El paradigma de programación orientada a objetos supuso un gran avance en el desarrollo de aplicaciones, ya que es capaz de acercar la forma de programar a la forma de pensar del ser humano.

Una de las características que más eficacia aporta al lenguaje Java es que viene acompañado de una serie de librerías estándar para realizar multitud de operaciones comunes a la hora de programar.

1.6.2 Herramientas

Entorno de Desarrollo Integrado. (NetBeans)

Es un proyecto de código abierto dedicado a proveer un sólido desarrollo de software, dirigido fundamentalmente a las necesidades de los desarrolladores y los usuarios; dotándolos de una herramienta para el desarrollo rápido, eficiente y fácil de productos de software [8].

NetBeans es una herramienta modular de desarrollo para un amplio rango de tecnologías para el desarrollo de aplicaciones. Incluye un avanzado editor para varios lenguajes, editor de perfiles y un detector de errores, además de herramientas para el control de versiones y el desarrollo colaborativo. Proporciona aplicaciones de muestra en forma de plantillas de proyectos para todas las tecnologías que soporta, así como una amplia variedad de plugins para todos los tipos de desarrollo. NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring. También puedes ver todas las variables del programa en tiempo de ejecución. Los errores son bastante descriptivos, ayuda mucho las indicaciones que te da. La Plataforma NetBeans como es una base modular y extensible usada como una estructura de integración permite la Reutilización del Módulos.

Spring Boot

Spring Boot es un framework de desarrollo de código libre para la plataforma Java. Su aspecto modular lo hace flexible y configurable para cualquier tipo de aplicación. Se compone de múltiples módulos que abarcan una gran variedad de servicios que facilitan la implementación de elementos de una aplicación empresarial (Faraoni, 2015).

[8] NetBeans IDE - Base IDE Features. 2013. https://netbeans.org/features/ide/index.html.

Spring Boot provee configuraciones por defecto para Spring y otra gran cantidad de librerías, además provee un modelo de programación parecido a las aplicaciones java tradicionales que se inician en el método main.

Los objetivos de Spring Boot son:

- Proveer una forma muy sencilla de arrancar desarrollos Spring
- Ofrecer funcionalidad out-of-the-box pero permitir incorporar las peculiaridades del proyecto
- Proporcionar una serie de características no funcionales comunes a los proyectos (por ejemplo, servidores embebidos, seguridad, indicadores, configuración externalizada)

Apache Jena

Es un framework Java libre y de código abierto para la construcción de Web Semántica y de aplicaciones Liked Data (usado para conectar datos distribuidos en la web)
La principal funcionalidad es que posee varios razonadores internos de owl haciendo uso de la api instalada desde algún entorno de desarrollo java. La api Apache Jena nos es más que un conjunto de librerías que deben ser instaladas en el entorno de desarrollo de su preferencia y a partir de estas hace uso de ella para poder crear clases y métodos para hacer razonamiento en base a la libería ModelFactory usando una ontología estructura Jena.

Lenguaje OWL

OWL Web Ontology Language o Lenguaje de Ontologías para la Web es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del W3C, y 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). En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste. 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. OWL surge como una revisión al lenguaje DAML-OIL y es mucho más potente que éste. Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de

expresividad que se precise y a los distintos tipos de aplicaciones existentes (motores de búsqueda, agentes, etc.)[9].

Herramientas CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadora o *Computer Aided Software Engineering*) son aplicaciones informáticas destinadas al incremento de la productividad en el proceso de desarrollo de software. Reduciendo el costo de desarrollo de las aplicaciones en cuanto a términos de tiempo y dinero (SLIDESHARE, 2008). Se utiliza para una rápida construcción de aplicaciones. Permite construir diagramas como el diagrama de proceso del negocio y el diagrama de despliegue. Se considera muy completa y fácil de usar, con soporte multiplataforma (Díaz, 2009).

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información a la hora de construir software. El uso de las mismas permite:

- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

Visual Paradigm

Herramienta que soporta el ciclo de vida completo de desarrollo de software. Soporta el lenguaje unificado de modelado (UML). Sencillo de utilizar, fácil de instalar y actualizar. Permite la generación de código a partir de diagramas para varios lenguajes como .Net, Java, PHP. Posibilita la representación gráfica de diagramas como: componentes, despliegue, secuencia, casos de uso; clase, actividad, estado, entre otros (LARMAN, 1999). Permite la representación de bases de datos, conversión de diagramas, mapeos de entidades y objetos, gestión de requisitos y modelación de diagramas de negocio entre otras opciones (KEFALAKIS 2015).

Entre algunas características que posee la herramienta se encuentran:

[9] http://www.w3.org/TR/owl-features/

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación. Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene. UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos.[10]

Solr

Es un servidor de índice empresarial, que actúa como una base de datos no SQL. Una plataforma de búsqueda de código que funciona como un "servidor de búsquedas". Solr es escalable, permitiendo realizar búsquedas distribuidas y replicación de índices. Utiliza la biblioteca Java de búsqueda en su base para la indexación de texto completo y de búsqueda, y tiene como REST HTTP / XML y JSON APIs que hacen que sea fácil de utilizar desde prácticamente cualquier lenguaje de programación. Potente configuración externa de Solr permite que sea adaptado a casi cualquier tipo de aplicación Java sin codificación, simplemente hay que utilizarlo con peticiones GET para realizar las búsquedas en el índice, y POST para agregar documentos (Apache Solr, 2015).

[10] https://www.lucidchart.com/pages/es/qué-es-el-lenguaje-unificado-de-modelado-uml

La principal característica de Solr (o al menos la más útil) es su API estilo REST, ya que en vez de usar drivers o APIs programáticas para comunicarnos con Solr podemos hacer peticiones HTTP y obtener resultados en XML o JSON.

Las ventajas de usar esta interfaz universal (y no propia de un lenguaje) son varias:

- Es agnótico del lenguaje porque usa XML y JSON, que hoy en día pueden ser interpretados por casi cualquier cosa. La métrica generalmente es JavaScript: si podemos interpretarlo con JavaScript dentro de todas las limitaciones que impone un navegador, entonces lo podemos interpretar en cualquier otro lugar. Por supuesto, JavaScript tiene soporte nativo para JSON y XML.
- Es agnóstico de los tipos de datos, ya que HTTP sólo transmite textos. Los lenguajes dinámicos como PHP tienen éxito porque su protocolo básico no tiene tipos estrictos.
- Más o menos es un protocolo estándar (aunque la representación de datos no lo sea): si alguien inventa una base de datos y publica su propio protocolo binario de comunicación, tendríamos muchas más librerías.

Uno de los factores que incidieron en la selección de esta herramienta fue su característica del utilizar archivos XML.

1.7 Metodología de desarrollo

La metodología de desarrollo del software constituye un proceso donde se definen técnicas y procedimientos para llevar a cabo el desarrollo de software. No existe hasta el momento una metodología que sea utilizada de forma universal. Existen diversas y cada una con sus características propias, pero en todas, el propósito es el mismo y es que el proceso sea configurable.

Las tradicionales establecen durante todo el proceso de desarrollo un mayor énfasis en la planificación y control del proyecto, y en la especificación precisa de los requisitos y el modelado. Por otro lado, las ágiles están más orientadas a la generación de código con ciclos muy cortos de desarrollo, enfocándose a equipos de desarrollo pequeños, haciendo especial hincapié en aspectos humanos asociados al trabajo en equipo e involucrando activamente al cliente en el proceso.

Metodología de desarrollo a utilizar

AUP-UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. Cuenta con tres fases las cuales son: Inicio, Ejecución y Cierre (RODRIGUES, 2014).

- Inicio: se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

- Ejecución: fase se ejecutan las actividades requeridas para desarrollar el software. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

- Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.8 Conclusiones del capítulo

Con la investigación antes expuesta se arribó a las siguientes conclusiones:

- El estudio de varios conceptos como: el de Breuker, (1999) y Gruber, (1993) fueron claves a la hora de explicar los conceptos asociados al dominio del problema. El entendimiento de estos conceptos permite comprender el objetivo de la herramienta propuesta.
- Las deficiencias encontradas la plataforma de Contenido Unificado para la Búsqueda Avanzada hacen necesario la creación de un componente que permita la generación automática de ontologías de dominio, lo que permite desambiguar la información.
- El estudio realizado de las principales herramientas, tecnologías y lenguajes de programación a utilizar, permitió realizar la selección de los más apropiados para dar solución al problema plan.

Capítulo 2. “Características de la propuesta de solución”.

2.1 Introducción

En este capítulo se abordan las características que el sistema a implementar debe poseer. Se identifican las clases del dominio y la relación que existe entre ellas es mostrada a través del diagrama de clases del modelo del dominio Se identifican los requisitos funcionales (RF) y no funcionales (RNF) que debe cumplir el servicio web. Se explica, además, la arquitectura y el diseño del sistema a desarrollar.

2.1 Descripción de la propuesta de solución

El proceso que será automatizado es la generación de ontologías de dominio. Para dar solución al problema existente se decide desarrollar un servicio web, El sistema debe ser capaz de construir ontologías a partir de documentos categorizados previamente.

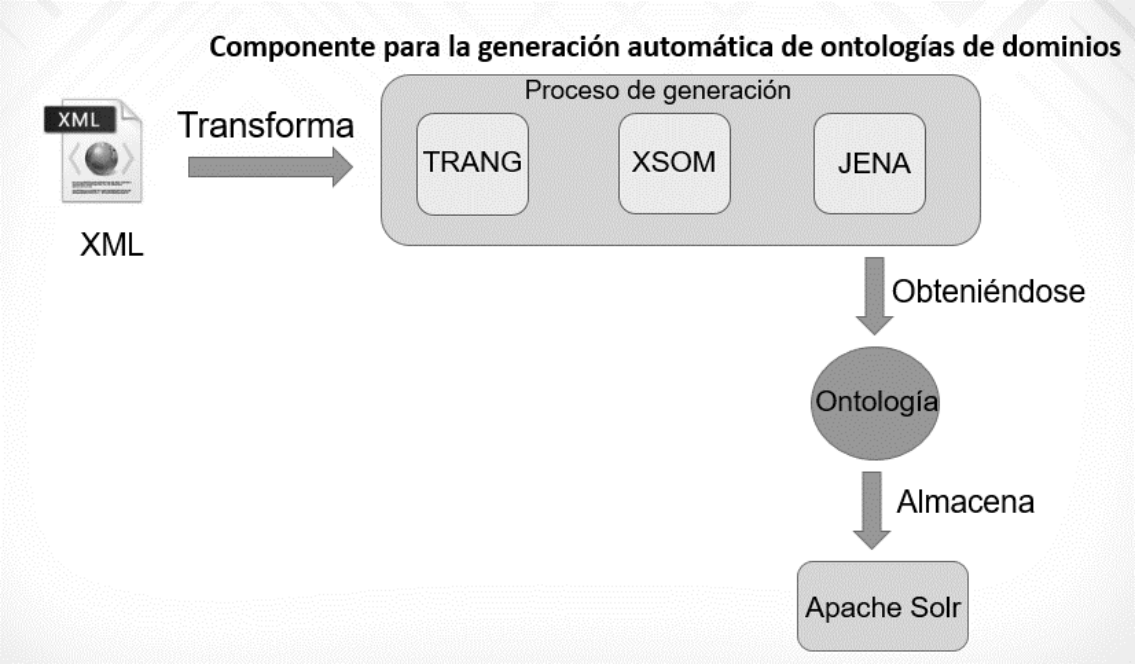

ILUSTRACIÓN 1: PROCESO PARA LA GENERACIÓN DE ONTOLOGÍAS. FUENTE: ELABORACIÓN PROPIA.

2.2 Modelo de Dominio

Según IBM[11] un modelo de dominio describe los tipos de dominio que admite una organización y sus restricciones. El modelo de dominio se crea con el fin de representar el vocabulario y los conceptos clave del dominio del problema. El modelo de dominio también identifica las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema, y comúnmente identifica sus atributos. El modelo de dominio proporciona una visión estructural del dominio que puede ser complementado con otros puntos de vista dinámicos, como el modelo de casos de uso.

El modelo de dominio se crea con el fin de representar el vocabulario y los conceptos clave del dominio del problema. El modelo de dominio también identifica las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema, y comúnmente identifica sus atributos. Un modelo de dominio que encapsula los métodos dentro de las entidades se asocia más bien con modelos orientados a objetos.

Descripción de clases del modelo del dominio

Usuario: Persona que realiza las búsquedas**.**

Interfaz: Constituye la vista mediante la cual se le muestran los resultados al usuario.

Rastreador: Mecanismo que se encarga de recopilar los documentos de la Web.

Solr (Indexador): Mecanismo que se encarga de indexar los documentos.

Base de Datos: conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Servicios: es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

[11]https://www.ibm.com/support/knowledgecenter/es/SS9UM9_9.1.2/com.ibm.datatools.logical.ui.doc/topics/cdommod.html

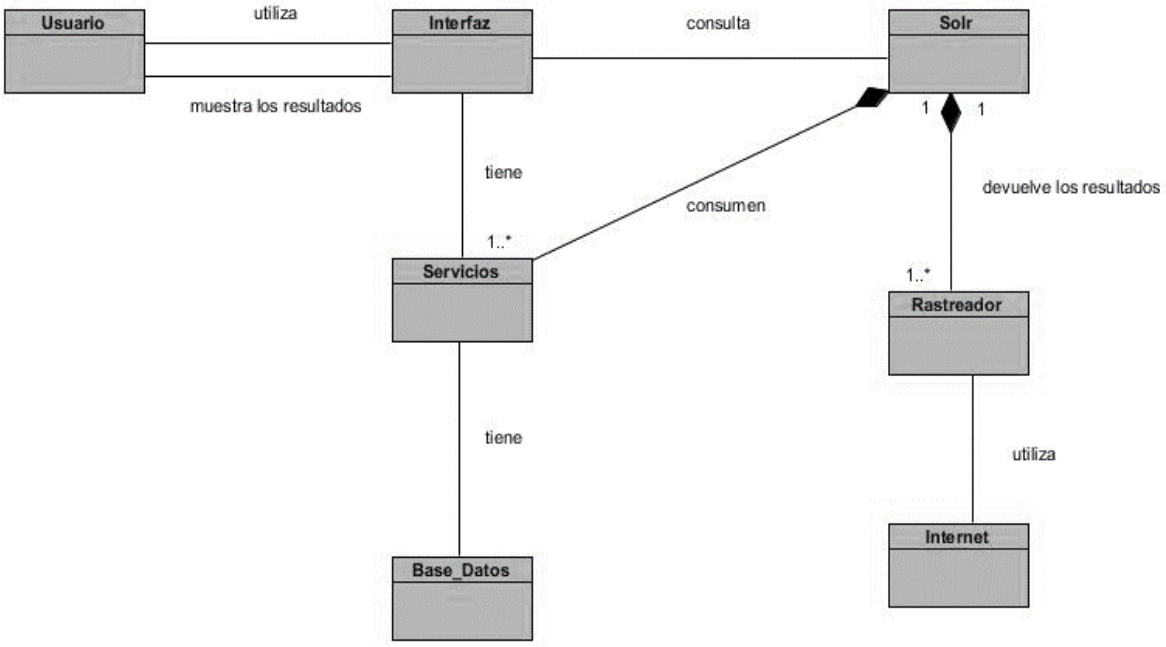

ILUSTRACIÓN 2: MODELO DE DOMINIO. FUENTE: ELABORACIÓN PROPIA.

2.3 Especificación de requisitos de software

Según Pressman, (2009), la ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software.

El análisis de requisitos es una tarea de ingeniería del software que cubre el hueco entre la definición del software a nivel sistema y el diseño de software. El análisis de requerimientos permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software.

Existen dos tipos de requisitos los Requisitos funcionales (RF) y los Requisitos no funcionales (RNF).

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema.

- RF1. Cargar formalmente el archivo XML.
- RF2. Generar formalmente el XML-SCHEMA.
- RF3. Parsear el XML-SCHEMA.
- RF4. Identificar las claves de una ontología.
- RF5. Identificar las propiedades de una ontología.
- RF6. Traducir los identificadores de las clases y propiedades de una ontología.
- RF7.Construir la ontología.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales describen atributos sólo del sistema o del ambiente del sistema que no están relacionados directamente con los requisitos funcionales. Los requisitos no funcionales incluyen restricciones cuantitativas, como el tiempo de respuesta o precisión, tipo de plataforma (lenguajes de programación y/o sistemas operativos, etc.) (Rojo, 2012).

Requisitos de software

- Se requiere la instalación del servidor web Nginx en su versión 1.9.11 y PHP 5.5.9 o superior para poder visualizar la interfaz web.

Requisitos de hardware

- Para la interfaz web: 2 GB RAM, CPU de 2 núcleos y al menos 40 GB de Disco Duro.
- Para el servidor de base datos Solr: 2 GB RAM, CPU de 2 núcleos y al menos 70 GB de Disco Duro.

Requisitos de usabilidad

- Los dispositivos clientes que utilizarán la herramienta deben poseer navegadores web que soporten HTML5, CSS3 y Java Script.

Requisitos de seguridad

- La información manejada por el sistema estará protegida de acceso no autorizado y divulgación.
- La comunicación entre el cliente y el servidor web será realizada a través del protocolo HTTPS.

Requisitos de Licencia

- Uso de la licencia PHP *License*.
- Uso de la licencia *Apache Software*.
- Uso de la licencia BSD de *PosgreSQL*.

2.4 Historias de usuario

Unos de los artefactos más importantes generados por la metodología de desarrollo de software son las historias de usuario (HU), las cuales son utilizadas para la especificación de requisitos funcionales del sistema. La Historia de Usuarios sirve para registrar los requerimientos de los clientes según el negocio y son utilizadas para poder realizar la estimación de cada una de las iteraciones durante la fase de planificación (Balarezo, 2013). Durante el análisis en la etapa de planificación – definición fueron definidas 6 historias de usuarios, en correspondencia con las necesidades del cliente. A continuación, se muestra la descripción de las historias de usuarios.

TABLA 3: HISTORIA DE USUARIO #1. FUENTE: ELABORACIÓN PROPIA.

Número: HU_1	
Programador: Lázaro Martínez Odio	**Nombre:** Representar formalmente el archivo XML.
Prioridad en negocio: Baja	**Iteración Asignada:** 2
Descripción: Se selecciona el archivo XML.	

TABLA 4: HISTORIA DE USUARIO #2 FUENTE: ELABORACIÓN PROPIA

Número: HU_2	
Programador: Lázaro Martínez Odio	**Nombre:** Generar formalmente el XML-SCHEMA.
Prioridad en negocio: Alta	**Iteración Asignada:** 1

Descripción: El documento XML se transforma a XML-Schema usando la API de Trang para Java. El Trang toma como entrada un esquema escrito en sintaxis XML y produce como resultado un esquema escrito en XML-Schema.

TABLA 5: HISTORIA DE USUARIO #3 FUENTE: ELABORACIÓN PROPIA.

Número: HU_3	
Programador: Lázaro Martínez Odio	**Nombre:** Parsear el XML-SCHEMA
Prioridad en negocio: Alta	**Iteración Asignada:** 1
Descripción: El XML-Schema se analiza utilizando XML-Schema Object Model (XSOM). XSOM es una biblioteca de Java que permite a las aplicaciones analizar fácilmente los documentos de esquema XML e inspeccionar la información en ellos.	

TABLA 6: HISTORIA DE USUARIO #4 FUENTE: ELABORACIÓN PROPIA.

Número: HU_4	
Programador: Lázaro Martínez Odio	**Nombre:** Identificar las claves de una ontología.
Prioridad en negocio: Alta	**Iteración Asignada:** 2
Descripción: Los tipos de atributos contenidos en el complexType definen las clases equivalentes.	

TABLA 7: HISTORIA DE USUARIO #5 FUENTE: ELABORACIÓN PROPIA.

Número: HU_5	
Programador: Lázaro Martínez Odio	**Nombre:** Identificar las propiedades de una ontología.
Prioridad en negocio: Alta	**Iteración Asignada:** 2
Descripción: A partir del análisis del archivo XML-schema se definen las propiedades de las	

ontologías.

TABLA 8: HISTORIA DE USUARIO #6 FUENTE: ELABORACIÓN PROPIA

Número: HU_6	
Programador: Lázaro Martínez Odio	**Nombre** Traducir los identificadores de las clases y propiedades de una ontología.
Prioridad en negocio: Media	**Iteración Asignada:** 3
Descripción: Se hace coincidir las extenciones o base de la restricción para generar la relación del subClassOf para las definiciones del complexType, y luego generar el owl.	

TABLA 9: HISTORIA DE USUARIO #7 FUENTE: ELABORACIÓN PROPIA.

Número: HU_7	
Programador: Lázaro Martínez Odio	**Nombre** Construir la ontología.
Prioridad en negocio: Alta	**Iteración Asignada:** 1
Descripción: Las claves y propiedades identificadas respectivamente se incluyen en la ontología.	

2.5 Arquitectura

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma[12]. En el modelo cliente servidor, el cliente, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador Web), envía un mensaje solicitando un determinado servicio al servidor de aplicaciones (hace una petición) (también denominado software intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo. El servidor de datos, que proporciona al servidor de aplicaciones los datos que requiere.

[12] Carnegie Mellon. Software Engineering Institute. Client/Server Software Architectures an Overview http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm

En este caso el sistema puede ser visto como un conjunto de servicios que se proporcionan a los clientes que hacen uso de dichos servicios. Los servidores y los clientes se tratan de forma diferente en estos sistemas.

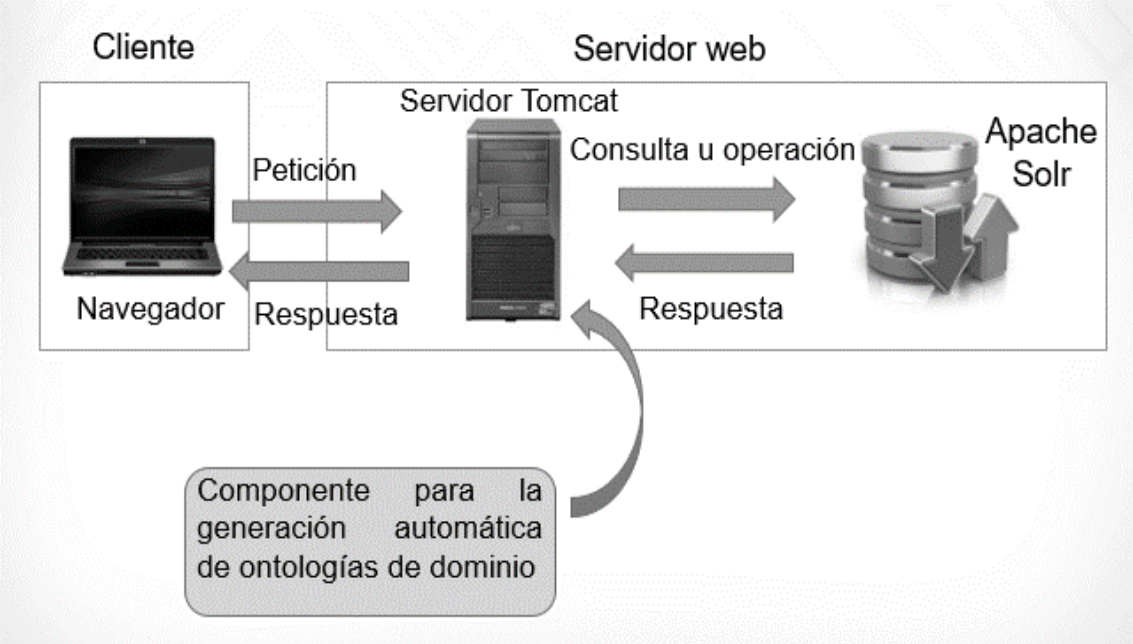

ILUSTRACIÓN 3: ARQUITECTURA DEL SISTEMA. FUENTE: ELABORACIÓN PROPIA.

2.6 Patrones de diseño

Los patrones de diseño de *software* son soluciones reutilizables de problemas recurrentes que aparecen durante el proceso de diseño de *software* orientado a objetos" (Larman, 2002).

Durante el desarrollo del sistema se usaron los siguientes patrones:

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, del inglés *General Responsibility Assignment Software Patterns*) permiten, como su nombre lo indica, la asignación de responsabilidades a objetos. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. (LARMAN, 2002).

Controlador

Para este caso se hace necesario conocer que un evento del sistema es una operación que se realiza en este, generada por un usuario externo. Un controlador, es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Su uso se evidencia en la clase UploadController.

Alta Cohesión

Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Se puede afirmar que cada una de las clases del sistema tiene alta cohesión, de manera que estas poseen la característica de tener las responsabilidades estrechamente relacionadas.

Creador

Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. En este caso el patrón se refleja en la clase *GenerarXSD*, encargada de crear los nuevos archivos XSD.

Bajo Acoplamiento

Debe haber pocas dependencias entre las clases. El objetivo de este patrón consiste en mantener un bajo nivel de dependencia de otros elementos, por lo que constituye un principio que debe estar presente en todas las decisiones de diseño con lo que se reduce el impacto de los cambios. Este patrón se evidencia en todas las clases.

Experto

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Este patrón se utilizó en la clase *UploadController*

2.7 Modelo de Diseño

El modelo de diseño se utiliza como medio de abstracción del modelo de implementación y el código fuente del *software*. Su objetivo fundamental es transmitir, a través de la representación mediante

diagramas, una comprensión en profundidad de los aspectos relacionados con los requerimientos no funcionales y restricciones concernientes a los lenguajes de programación (Larman, 2002).

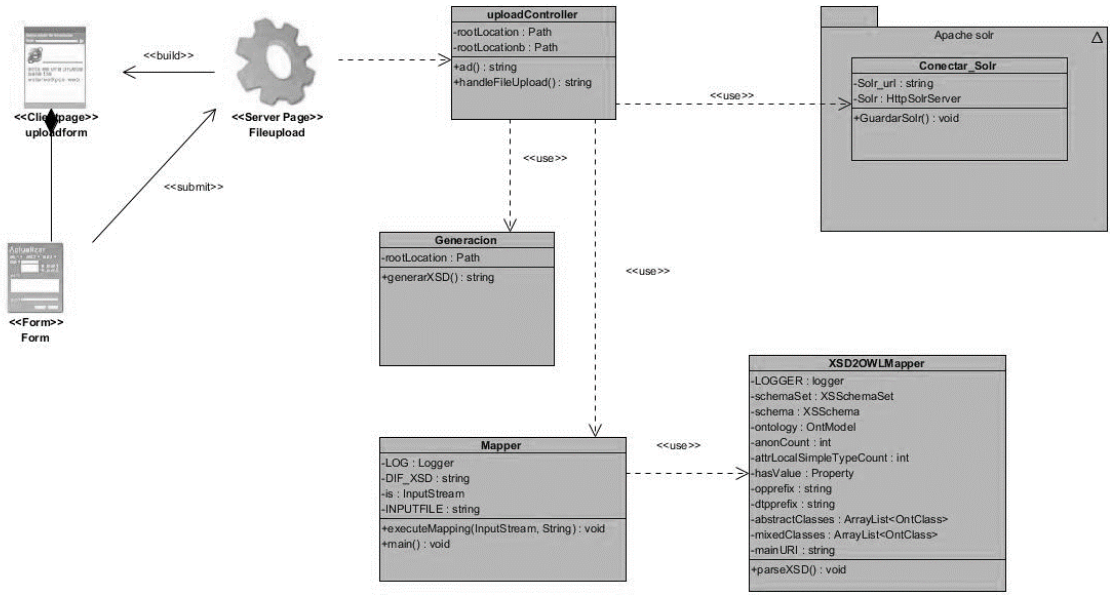

ILUSTRACIÓN 4: DIAGRAMA DE CLASES. FUENTE ELABORACIÓN PROPIA.

2.8 Diagrama de Secuencia

Ilustran las interacciones en un tipo de formato con aspecto de una valla, en el que cada objeto nuevo se añade a la derecha, mostrando claramente la secuencia y ordenación en el tiempo de los mensajes con una notación simple (Larman, 2003).

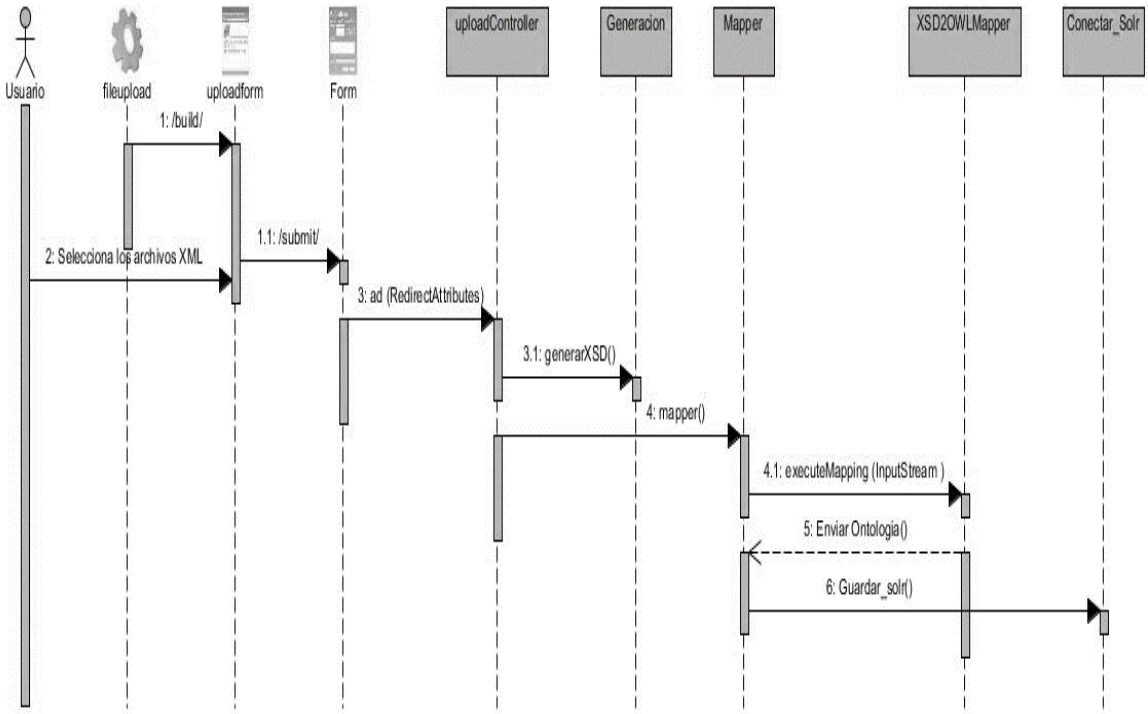

ILUSTRACIÓN 5: DIAGRAMA DE SECUENCIA. FUENTE ELABORACIÓN PROPIA

2.9 Base de Datos

Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos o podríamos decir que es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos (Redondo, 2017).

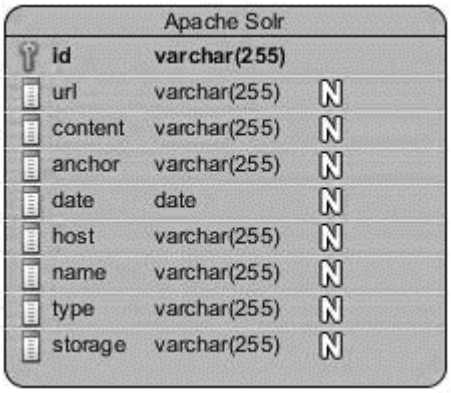

ILUSTRACIÓN 6: DIAGRAMA MODELO DE DOMINIO DE APACHE SOLR. FUENTE ELABORACIÓN PROPIA

2.10 Conclusiones parciales

En el presente capítulo se ha podido evidenciar las principales características y elementos significativos de la arquitectura de software y de información del repositorio, así como los diferentes requisitos funcionales y no funcionales. Por lo que se puede constatar que:

- La obtención de los requisitos funcionales y no funcionales obtenidos a partir del proceso de identificación de los requisitos, garantizará que la solución responda a las necesidades del cliente y de los usuarios finales, sirviendo de guía para el desarrollo de las distintas funcionalidades del sistema.
- La utilización de los patrones de diseño permitió identificar aspectos importantes de la estructura del componente para la generación de ontologías de dominio, que garantiza una mayor organización.
- Los artefactos generados constituyeron una guía fundamental para el desarrollo del componente.

Capítulo 3: “Implementación y prueba del componente de generación automática de ontologías de dominios”

En el presente capítulo se presentan los diferentes mecanismos utilizados para llevar a cabo el desarrollo y validación del componente para la generación automática de ontologías de dominios, quedando definido el diagrama de componente que brinda detalles principales de la distribución del sistema y los estándares de codificación empleados durante la implementación.

3.1 Diagrama de componentes

El diagrama de componentes muestra los componentes de un sistema de software conectados por las relaciones de dependencias lógicas entre cada uno de ellos. Provee una vista arquitectónica de alto nivel del sistema, ayudando a los desarrolladores a visualizar el camino de la implementación. Cada componente representa una unidad del código (fuente, binario o ejecutable), que permite mostrar las dependencias en tiempo de compilación y ejecución. La realización del diagrama posibilita tomar decisiones respecto a las tareas de implementación y los requisitos (Rivera, 2008).

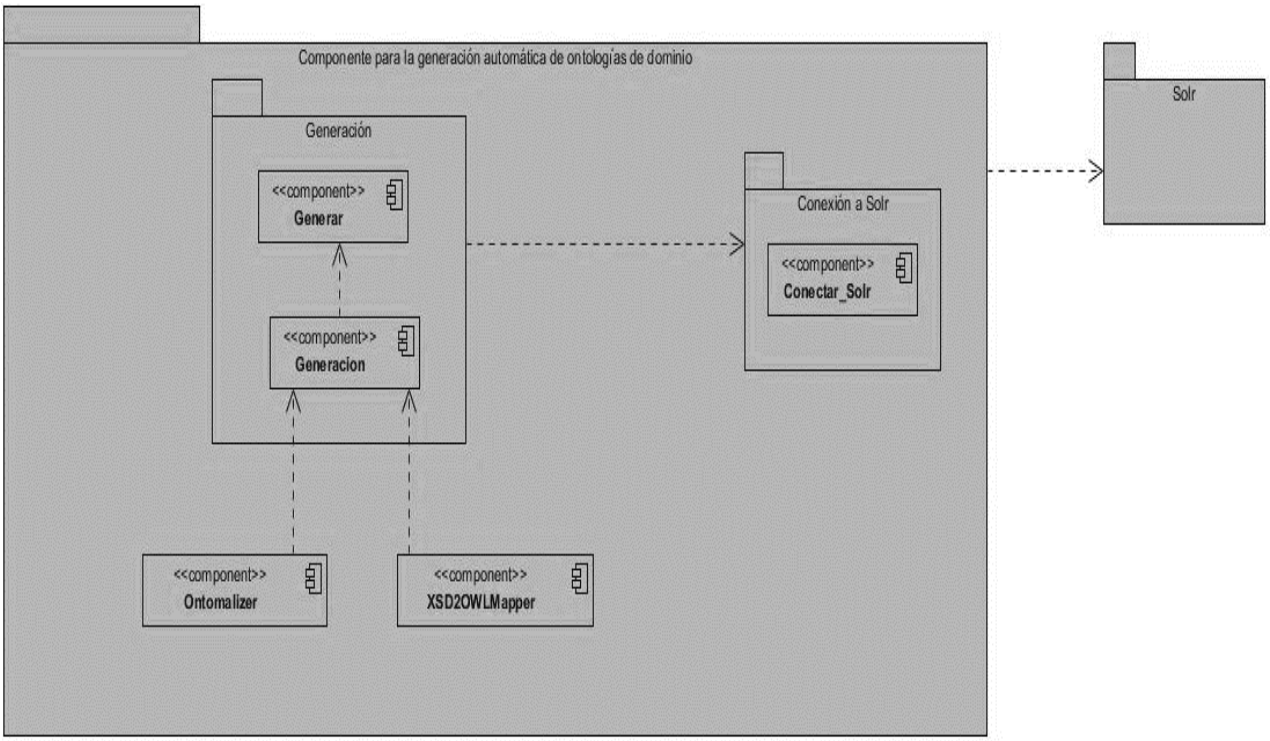

ILUSTRACIÓN 7: DIAGRAMA DE COMPONENTE DEL COMPONENTE PARA LA GENERACIÓN DE ONTOLOGÍAS DE DOMINIO FUENTE: ELABORACIÓN PROPIA.

3.2 Modelo de despliegue

El modelo de despliegue o diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos (2007). En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta:

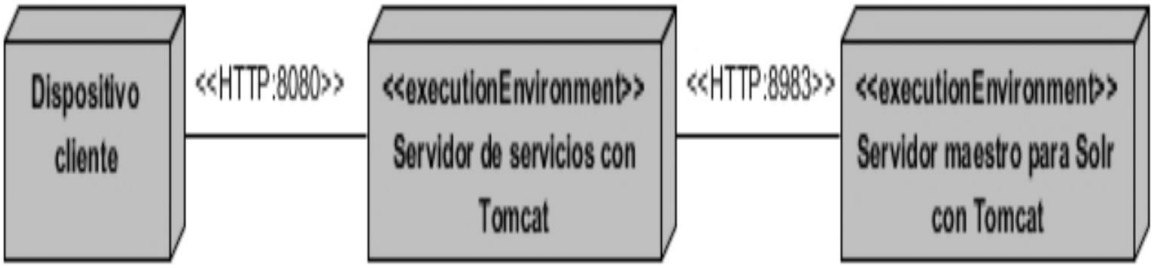

ILUSTRACIÓN 8: DIAGRAMA DE DESPLIEGUE. FUENTE: ELABORACIÓN PROPIA.

Descripción de los elementos de interface y comunicación

<<HTTPS>>: Protocolo para establecer la conexión segura entre la PC cliente y el servidor de aplicaciones a través del puerto.

<<TCP>>: Protocolo para establecer la conexión entre el servidor de aplicaciones y el servidor de base de datos a través del puerto definido para el gestor de base de datos. La conexión entre estos servidores permitirá ejecutar un conjunto de órdenes y obtener rápidamente respuesta a las mismas.

3.3 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. El estándar de codificación debería establecer cómo operar con la base de código existente (Microsoft, 2017).

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia se le pueda realizar el correcto mantenimiento (Calleja, 2009).A continuación se especifican los estándares de codificación empleados en la construcción de la solución.

Indentación

La unidad de indentación de bloques de sentencias son 4 espacios.

```
public String ad(RedirectAttributes redirectAttributes) throws Exception{
    try {
        Path file = rootLocation.resolve(storage.getFile(rootLocation.toFile()));
```

Comentarios

Los comentarios deben añadir claridad al código. Deben contar el por qué y no el cómo. Deben ser concisos, declarados en el mismo idioma y gramaticalmente correctos

```
String typeName = ct.getName();
// these are extensions so look at the base type to see what it is
String baseTypeName = ct.getBaseType().getName();
```

Declaraciones

Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.

```
public void generarXSD() {
    String a = "C:\\Users\\h3h\\Desktop\\Lachi\\tesisLazaro\\trang";
    String nombre = storage.getNombre(storage.getFile(rootLocation.toFile()));
```

Inicialización

Inicializar cada variable en su declaración a menos que su valor inicial dependa de algún cálculo.

```
List<String> b = Arrays.asList("java", "-jar", "trang.jar", "upload\\" + storage.getFile(rootLocation.toFile()),
ProcessBuilder pb = new ProcessBuilder().command(b).directory(new File(a));
try {
    Process p = pb.start();
```

Cada tipo de elemento debe nombrarse con una serie de reglas determinadas

Paquetes: Todo en minúsculas. Cada palabra es más específica que la anterior

```
package com.tesis.tesis.repo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

Clases e interfaces: Nombres. La inicial en mayúscula.

```
@Service
public class Storage {
    private static final Path rootLocation = Paths.get("trang/upload");
```

Métodos: Deben ser verbos. La primera letra de la primera palabra en minúsculas, el resto de las palabras empiezan por mayúsculas

```
public void deleteAll() {
    FileSystemUtils.deleteRecursively(rootLocation.toFile());
}
```

3.3 Validación de la propuesta de solución

En la ingeniería de software la validación es el proceso de revisión que verifica que el sistema de software producido cumple con las especificaciones y que logra su cometido. Las pruebas de software son procedimientos realizados para verificar la calidad de un software y pueden ser aplicadas periódicamente. Estas tienen como objetivo identificar posibles errores en la aplicación.

Existen varias estrategias de pruebas que suelen ser utilizadas, dentro de las que se pueden mencionar:

1. Pruebas Unitarias.
2. Pruebas Funcionales.

Pruebas Unitarias

Se llaman pruebas unitarias porque descomponen las funciones del programa en comportamientos discretos que se pueden probar como unidades individuales. El uso de pruebas unitarias aumenta la calidad del código debido a que comprueban el comportamiento del código en respuesta a casos estándar, límite e incorrectos de datos de entrada, así como cualquier suposición explícita o implícita creada por el código. Las pruebas unitarias deben poder cubrir casi la totalidad del código de nuestra aplicación. Una prueba unitaria será tan buena como su cobertura de código. La cobertura de código marca la cantidad de código de la aplicación que está sometido a una prueba.

A continuación, se realiza las pruebas unitarias mediante el uso de la técnica de camino mínimo al método generarXSD de la clase Generación, el cual a partir de un archivo XML común se convierte en un archivo XML-schema, utilizando la API Trang.

```
public void generarXSD() {
    String a = "C:\\Users\\h3h\\Desktop\\Lachi\\tesisLazaro\\trang";
    String nombre = storage.getNombre(storage.getFile(rootLocation.toFile()));
    List<String> b = Arrays.asList("java", "-jar", "trang.jar", "upload\\" + storage.getFile(rootLocation.toFile()), "out\\" + nombre + ".xsd");
    ProcessBuilder pb = new ProcessBuilder().command(b).directory(new File(a));
    try {
        Process p = pb.start();
    } catch (IOException ioe) {
        System.out.println(ioe);
    }
```

FIGURE 1: IMPLEMENTACIÓN DEL MÉTODO GENERARXSD DE LA CLASE GENERACIÓN. FUENTE: ELABORACIÓN PROPIA.

Se enumeró el código y se representó en una gráfica para hacer más entendible el flujo de control lógico, mediante el empleo de nodos y aristas.

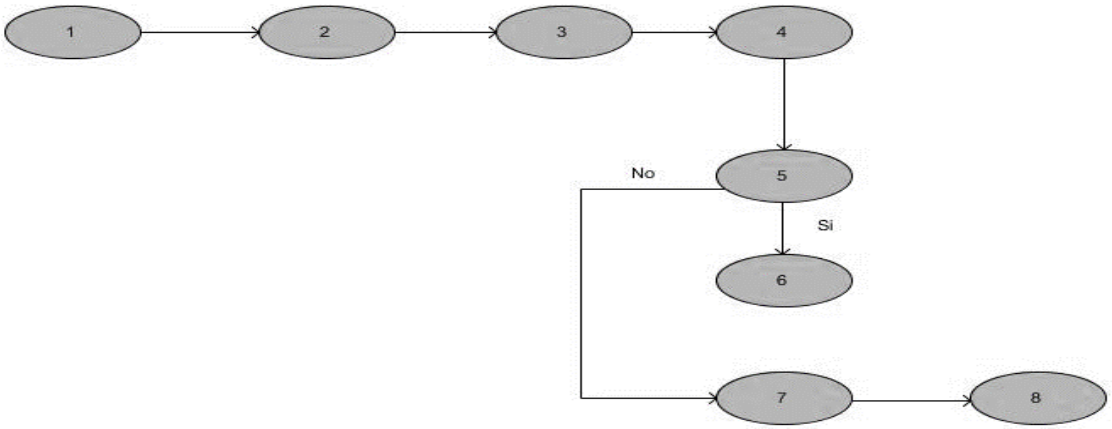

ILUSTRACIÓN 9: GRAFO DE FLUJO FUENTE: ELABORACIÓN PROPIA

Como se evidencia en la fig. 7se obtuvo como resultado 8 nodos y 7 aristas con lo que se calcula la complejidad condicional(C(c)), que no es más que, una simple métrica para determinar, la complejidad de un programa estructurado, la cual refleja directamente, el número de caminos independientes que un programa puede tomar durante su ejecución.

C(c)= cantidad aristas – cantidad nodos + 2

C(c)=7-8+2

C(c)=1

En general, el valor de complejidad ciclomática establece un límite superior en la cantidad de pruebas necesarias para cubrir todos los caminos decisorios del código en cuestión. Este es un límite superior, ya que no todos los caminos son necesariamente realizables. También el valor de la complejidad ciclomática define la cantidad de rutas independientes (Pressman, 2006).

TABLA 10: CAMINOS INDEPENDIENTES. FUENTE: ELABORACIÓN PROPIA.

Número de Ruta	Camino
1	1-2-3-4-5-6

TABLA 11: CANTIDAD DE ERRORES POR CADA ITERACIÓN LAS PRUEBAS DE INTEGRACIÓN FUENTE: ELABORACIÓN PROPIA.

Errores	**Primera iteración**	**Segunda iteración**	**Tercera iteración**
Detectados	5	3	0
Resueltos	5	3	0
Pendientes	0	0	0

.

Prueba funcionales

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. El objetivo de la prueba funcional es validar si el comportamiento observado del software probado cumple o no con sus especificaciones. La prueba funcional toma el punto de vista del usuario. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático. Dentro de los principales beneficios que tienen este tipo de pruebas está la mitigación del riesgo de aparición de fallos en producción, el cumplimiento de los objetivos de los proyectos en términos de calidad y resultados esperados principalmente. Además, la identificación temprana de riesgos y desviaciones asociadas a la calidad.

TABLA 12: CASO DE PRUEBA A LA HU "REPRESENTAR FORMALMENTE EL ARCHIVO XML ". FUENTE ELABORACIÓN PROPIA.

Escenario	Descripción	Valor	Respuesta del componente	Flujo Central
EC 1.1 Representar formalmente el archivo XML.	El usuario selecciona los archivos XML a introducir en el componente de generación automática de ontologías de domino.	Archivo XML	El componente carga los archivos para su posterior análisis.	El componente de generación automática de ontologías carga los archivos para su posterior análisis.

TABLA 13: CASO DE PRUEBA A LA HU "GENERAR FORMALMENTE EL XML-SCHEMA ". FUENTE: ELABORACIÓN PROPIA.

Escenario	Descripción	Valor	Respuesta del componente	Flujo Central
RF2. Representar formalmente el XML-SHEMA.	El componente de generación automática de ontologías de domino convierte el XML en XML-	Archivo XML	El componente de generación automática de ontologías de domino genera el archivo XML-Schema.	El componente utiliza la biblioteca Trang el cual utiliza como entrada un esquema escrito en sintaxis XML y produce como resultado un esquema escrito en XML-Schema.

	Schema.			

TABLA 14: CASO DE PRUEBA A LA HU " PARSEAR EL XML-SCHEMA ". FUENTE: ELABORACIÓN PROPIA.

Escenario	**Descripción**	**Valor**	**Respuesta del componente**	**Flujo Central**
RF3. Parsear el XML-SHEMA. .	El componente parsea el archivo XSD obteniendo los distintos complexTypes.	Archivo XSD	El componente de generación automática de ontologías de domino procesa el archivo XSD.	El componente utiliza la biblioteca XSOM el cual utiliza como entrada un XML esquema y obtiene los complexTypes y los namespace.

TABLA 15: CASO DE PRUEBA A LA HU " IDENTIFICAR LAS CLAVES DE UNA ONTOLOGÍA.". FUENTE: ELABORACIÓN PROPIA.

Escenario	**Descripción**	**Valor**	**Respuesta del componente**	**Flujo Central**
RF4. Identificar las claves de una ontología. .	El componente relaciona los elementos dentro de complexType para generar Restricciones y Clases definida por	owl:Class rdf:ID	El componente de generación automática de ontologías de domino identifica las propiedades de la ontología.	El componente de generación automática de ontologías de domino a partir de complexType genera las restricciones y las clases de la ontología.

	el complexType.			

TABLA 16: CASO DE PRUEBA A LA HU " IDENTIFICAR PROPIEDADES DE UNA ONTOLOGÍA.". FUENTE: ELABORACIÓN PROPIA.

Escenario	**Descripción**	**Valor**	**Respuesta del componente**	**Flujo Central**
RF5. Identificar las propiedades de una ontología.	El componente hace coincidir las definiciones de los elementos XML Schema para generar y OWL ObjectProperties y DatatypeProperties.		El componente de generación automática de ontologías de domino crea las clases de la ontología.	El componente a partir de las ObjectProperties y DatatypeProperties identificadas en el archivo XSD genera las claves de las ontologías.

TABLA 17: CASO DE PRUEBA A LA HU "TRADUCIR LOS IDENTIFICADORES DE LAS CLASES Y PROPIEDADES DE UNA ONTOLOGÍA". FUENTE: ELABORACIÓN PROPIA.

Escenario	**Descripción**	**Valor**	**Respuesta del componente**	**Flujo Central**
RF6. Traducir los identificadores de las clases y propiedades de una	El componente hace coincidir la extensión o base de restricción		El componente asocia las propiedades y las clases a la ontología.	El componente de generación automática de ontologías de domino hace coincidir la extensión o base de restricción, con los

ontología.	para generar la relación subClassOf para las definiciones de complexType.			identificadores de las clases.

TABLA 18: CASO DE PRUEBA A LA HU "GENERA LA ONTOLOGÍA". FUENTE: ELABORACIÓN PROPIA

Escenario	**Descripción**	**Valor**	**Respuesta del componente**	**Flujo Central**
RF7.Construir la ontología.	Las claves y propiedades identificadas respectivamente se incluyen en la ontología.	Archivo .n3	El componente de generación automática de ontologías de domino genera la ontología	El componente asocia las propiedades, las clases y los identificador es a la ontología.

Resultados de las pruebas funcionales

Para probar el correcto funcionamiento del sistema se realizaron dos iteraciones de pruebas. En la tabla 18 que a continuación se presenta, se muestran los resultados obtenidos en cada iteración de pruebas al componente de generación automática de ontologías de domino, así como la corrección de cada uno de los errores.

TABLA 19: CANTIDAD DE NO CONFORMIDADES POR CADA ITERACIÓN LAS PRUEBAS FUENTE: ELABORACIÓN PROPIA.

No	**Primera iteración**	**Segunda iteración**	**Tercera iteración**

conformidades			
Detectadas	5	3	0
Resueltas	0	0	0
Pendientes	0	0	0

Entre las no conformidades detectadas se encuentra la incorrecta ubicación del archivo xsd que se genera. Todas las no conformidades se solucionaron con la modificación del código fuente.

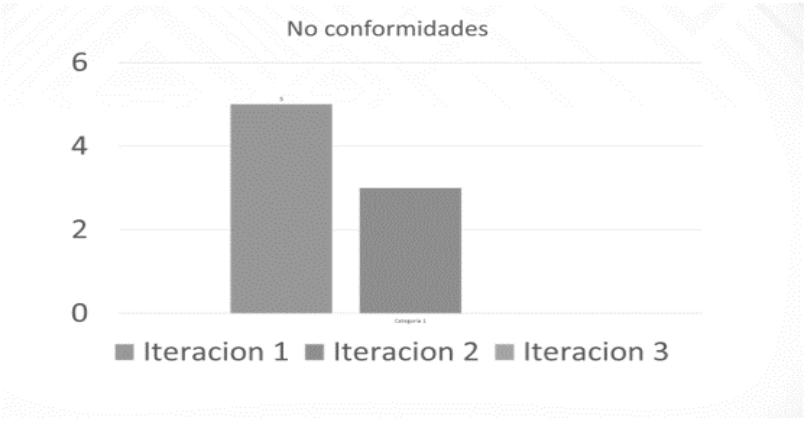

Ilustración 10: Resultado de las no conformidades. Fuente elaboración propia

3.4 Criterio de expertos

El juicio de expertos es un método de validación útil para verificar la fiabilidad de una investigación que se define como “una opinión informada de personas con trayectoria en el tema, que son reconocidas por otros como expertos cualificados en éste, y que pueden dar información, evidencia, juicios y valoraciones” (Escobar-Pérez y Cuervo-Martínez, 2008).

El juicio de expertos en su variante Delphi, es un método de pronóstico cualitativo muy popular, desarrollado por Olaf Helmer y otros en la RAND Corporantion a mediados de la década de 1960. Este procedimiento utiliza un grupo de expertos para el análisis que se mantienen aislados con objeto de minimizar el efecto de presión social y otros aspectos del comportamiento de pequeños grupos. Los expertos pueden ser especialistas internos o externos. Su uso en general requiere una considerable flexibilidad para satisfacer las necesidades de la situación, un análisis comparativo de la introducción y la expansión del nuevo producto, basando la comprobación en patrones de similitud (Fernández, 2012).

Para llevar a cabo este método se definieron los siguientes pasos:

1. Confeccionar un listado inicial de personas posibles de cumplir los requisitos para ser expertos en la materia a trabajar.
2. Selección de los expertos.
3. Se lleva a cabo las consultas a expertos.
4. Se analizan las respuestas.
5. Se realiza la valoración de la información obtenida.

En la siguiente tabla se muestran los expertos seleccionados:

TABLA 20: LISTADO DE EXPERTOS SELECCIONADOS. FUENTE: ELABORACIÓN PROPIA.

No.	Experto	Entidad	Años de experiencia
1	Eric Bárbaro Utrera Sust	CIDI	8
2	Miguel Ángel Chávez Alfonso	CIDI	7
3	Paúl Rodríguez Leyva	CIDI	7
4	Walter Daniel Camejo López	CIDI	4
5	Michel Lázaro Frómeta Burey	CIDI	7
6	Yoan Antonio López Rodríguez	FAC3-Programación	10
7	Hubert Viltres Sala	CIDI	8

Una vez seleccionado los expertos con los que se realizaría el trabajo se les presentan los aspectos a valorar previamente determinados por el investigador, a través de una tabla de Aspectos / Rangos de Valoración. Los rangos de valoración son 5, es decir, Muy Adecuado, Bastante Adecuado, Adecuado, Poco Adecuado e Inadecuado, a los que asignamos valor numérico del 1 al 5 en el mismo orden.

TABLA 21: SENTENCIAS A EVALUAR POR LOS EXPERTOS PARA VALIDAR LA HIPÓTESIS CIENTÍFICA. FUENTE: ELABORACIÓN PROPIA.

No	Afirmación planteada a los expertos
1	Considera que el componente extrae las clases y propiedades del archivo XML

	de manera correcta.
2	Considera que el componente genera las ontologías de forma correcta.
3	Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.
4	Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.
5	Considera que se logra esquematizar el conocimiento a través de las ontologías.

Una vez definido los expertos se prosigue al cálculo del coeficiente de Kendall, para medir el grado de correlación entre las variables de una muestra.

W = 12S/K2 (N3-N)

Dónde:

w = coeficiente de concordancia de Kendall. [0,1]. El valor 1 significa una concordancia de acuerdos total y el 0, un desacuerdo total.

S = suma de los cuadrados de las diferencias observadas con respecto a un promedio.

N = Tamaño de la muestra en función del número de tripletes, (número total de aspectos a evaluar).

K = número de expertos.

Li = sumatoria de las ligas o empates entre los rangos.

Planteamiento de la Hipótesis:

- **H0:** no existe concordancia entre los expertos.
- **H1**: existe concordancia entre los expertos.

X2 = K (N-1) W

X2=0.28

Para tener un 95% de confianza se utilizará α = 0.05. Si se cumple que X2 calculada <X2 (α, N-1) se obtiene que 0.28 < 9.4877 entonces se valida la hipótesis alternativa H1 de que existe concordancia entre los expertos.

Una vez plasmados los criterios de los expertos en cada rango de valoración para los diferentes aspectos en una tabla de Aspectos:

TABLA 22: DISTRIBUCIÓN DE FRECUENCIA PARA LOS DATOS PRIMARIOS OBTENIDOS. FUENTE ELABORACIÓN PROPIA.

Categorías evaluativas	Frecuencia Absoluta	Frecuencia Relativa
Muy adecuado	24	0.96
Bastante adecuado	1	0.04
Adecuado	0	0
Poco adecuado	0	0
Inadecuado	0	0

Por lo tanto, con un nivel de concordancia de un 100%, los expertos clasifican de nivel Muy adecuado las sentencias evaluadas, por lo que la hipótesis científica es apoyada por el juicio de los expertos. Demostrándose así el alto grado de calidad del componente para la generación de ontologías de dominio mediante el análisis de los indicadores analizados.

3.5 Conclusiones parciales

Con la investigación antes expuesta se concluye que:

- En el presente capítulo se especificó cómo está construido el componente a partir del diagrama de componentes, ofreciendo una vista arquitectónica de alto nivel que permitió identificar con claridad la estructura y relaciones que existen entre los diferentes elementos empleados en la implementación del componente.

- Al mismo tiempo la definición de los estándares de codificación, así como las pruebas tanto de caja blanca como funcionales que se utilizaron en la implementación del componente de generación automática de ontologías de domino, permitieron detectar los errores presentes, corregirlos en el menor tiempo posible y entregar al cliente una aplicación con mayor calidad.

- El proceso de validación de la hipótesis propuesta permitió demostrar la calidad del componente desarrollado.

CONCLUSIONES GENERALES

El desarrollo del presente trabajo ha posibilitado el cumplimiento de los objetivos y tareas propuestas, por lo que se arriban a las siguientes conclusiones:

- El estudio realizado de los sistemas homólogos, permitió identificar las principales funcionalidades para la implementación del componente de generación automática de ontologías de dominio.
- La selección de herramientas, lenguajes y tecnologías permitió la implementación del componente de generación automática de ontologías de dominio.
- La captura de requisitos facilitó el análisis y diseño de las funcionalidades desarrolladas para el componente.
- Como propone la metodología seleccionada se generaron los artefactos necesarios para la modelación del sistema.
- La utilización de la estrategia de pruebas garantizó la identificación temprana de las deficiencias en el componente que se desarrolló; corrigiéndose las mismas logrando aumentar así la calidad del componente desarrollado.

Recomendaciones

Una vez concluida la investigación y el desarrollo de la propuesta de solución se recomienda:

- Graficar la ontología que se genere, donde se representen las relaciones que existen entre los conceptos.

Referencias Bibliográficas

ARANO, S. "La ontología: una zona de interacción entre la Lingüística y la Documentación". Hipertext.net, núm. 2, 2003. http://www.hipertext.net/web/pag220.htm

ARANO, S. "Los tesauros y las ontologías en la Biblioteconomía y la Documentación". Hipertex.net, núm 3, 2005. http://www.hipertext.net/web/pag260.htm

BENEYTO, R. Documanía 2.0. 2013 [https://documania20.wordpress.com/2013/09/16/cuanta-informacion-se-genera-y-almacena-en-el-mundo/.

BREUKER, A. *Indexing problem solving methods for reuse.* 1999.

BAKER, T. "Cores: A Forum on Shared Metadata Vocabularies". ERCIM News, No. 51, October 2002. Special Semantic Web. http://www.ercim.org/publication/Ercim_News/enw51 http://www.ercim.org/publication/Ercim_News/enw51/EN51.pdf

CALLEJA, M. *Carmen. Estándares de codificación.* 2009

CODINA, L.; CRISTÒFOL, R. *La Web Semántica.* 2006.

CURRÁS, E. Ontologías, taxonomías y tesauros: Manual de construcción y uso. Gijon, Trea, 2005.

CORCHO, O.; FERNÁNDEZ M. *Methodologies, tools and languages.* 2003.

DIENG-KUNTZ, R. "Corporate Semantic Webs". ERCIM News, No. 51, October 2002. Special Semantic Web. http://www.ercim.org/publication/Ercim_News/enw51 http://www.ercim.org/publication/Ercim_News/enw51/EN51.pdf

DING, Y.; FENSEL, D. "OntoWeb: The Thematic Network for the Semantic Web". ERCIM News, No. 51, October 2002. Special Semantic Web. http://www.ercim.org/publication/Ercim_News/enw51 http://www.ercim.org/publication/Ercim_News/enw51/EN51.pdf

DING, Y.; FOO, S. *Ontology Research and Development.* 2002.

FARAONI, F. *Desarrollo de una aplicacion web con Spring Framework para un gestor de un recetario.* 2015.

FERDINAND, M.; ZIRPINS, C.; et al. *Lifting XML Schema to OWL.* 2004.

FERNÁNDEZ, L. *PROCEDIMIENTO SEMI-AUTOMÁTICO PARA TRANSFORMAR.* 2009.

FERNÁNDEZ, S. Histodidactica. [En línea] 2012.

GHAWI, R.; CULLOT, N. *Building Ontologies from XML Data Sources.* 2009.

GOMEZ, A.; FERNÁNDEZ, M. *Towards a Method to Conceptualize Domain.* 1996.

GRUBER, T. *A translation approach to portable ontologies.* 1993. Vol. Knowledge Acquisition.

GARCÍA, A. "Instrumentos de representación del conocimiento: tesauros versus ontologías". Anales de Documentación, núm. 7, 2004.

GRUBER, T. "A translation approach to portable ontologies". Knowledge Acquisition, 5(2), 1993.

GRUBER, T. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University, CA, 1993.

IBM KNOWLEDGE CENTER. [En línea] 2018. https://www.ibm.com.

INTERNET LIVE STATS. [En línea] 2017. http://www.internetlivestats.com.

KAPOOR, B.; SAVITA S. A *comparative study ontology building tools for semantic web applications*. International Journal of Web & Semantic Technology (IJWesT), vol. 1(3), pp.1-13, 2010.

LENA, G. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* 1990.

LAPUENTE, M. [En línea] 8 de diciembre de 2013. http://www.hipertexto.info.

LARMAN, C. *UML y Patrones.* 2002.

LUTZ, M. *Learning Python, Fourth Edition.* 2010.

MORALES, E. "Ontologías". En Representación del conocimiento. http://dns1.mor.itesm.mx/%7Eemorales/Cursos/RdeC/node194.html

MIZOGUCHI, R. Tutorial on ontological engineering part 2: Ontology development, tools and languages. New Generation Computing Springer, vol.22 (1), pp. 61-96, 2004.

NETBEANS.ORG. [En línea] 2013.

***OMG UNIFIED MODELING LANGUAGE (OMG UML).* 2007.**

PRESSMAN, R. Ingeniería del Software. Un enfoque práctico. 5ta. Edición. 2002.

ROJO, A. y SILVANA V. *Requerimientos No funcionales para aplicaciones Web.* 2012.

ROVIRA, L. *La Web Semántica.* 2006.

SÁNCHEZ, R.; GIL, B. *Lenguajes documentales y ontologías.* 2007.

SWARTOUT, B.; SWARTOUT, B.; et al. *Toward Distributed Use of Large-Scale Ontologies.* 1997.

SINGH, A.; ANAND, P. *State of Art in Ontology Development Tools.* 2013.

SOMMERVILLE, I. Ingeniería del software. Pearson Educación, 2005.

TELLO, A. *Ontologías en la Web Semántica.* España.: s.n., 2012.

YAHIA, N.; MOKHTAR, S.; et al. *Automatic Generation of OWL Ontology from XML Data Source.* 2012.

ANEXOS

Anexo # 1: Encuestas

TABLA 23: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 1.FUENTE: ELABORACIÓN DEL PROPIO AUTOR.

Afirmación	Alternativas de respuestas				
	1	2	3	4	5
Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta				x	
Considera que el componente genera las ontologías de forma correcta.					x
Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.					x
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.				x	
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

TABLA 24: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 2.FUENTE: ELABORACIÓN DEL PROPIO AUTOR.

Afirmación	Alternativas de respuestas				
	1	**2**	**3**	**4**	**5**
Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta					x
Considera que el componente genera las ontologías de forma					x

correcta.					
Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.	x				x
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.				x	
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

TABLA 25: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 3.FUENTE: ELABORACIÓN PROPIA.

Afirmación	**Alternativas de respuestas**				
	1	**2**	**3**	**4**	**5**
Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta				x	
Considera que el componente genera las ontologías de forma correcta.				x	
Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.					x
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.				x	
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

TABLA 26: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 4.FUENTE: ELABORACIÓN PROPIA.

Afirmación	Alternativas de respuestas				
	1	2	3	4	5
Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta					x
Considera que el componente genera las ontologías de forma correcta.					x
Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.					x
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.					x
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

TABLA 27: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 5.FUENTE: ELABORACIÓN PROPIA.

Afirmación	Alternativas de respuestas				
	1	2	3	4	5
Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta				x	
Considera que el componente genera las ontologías de forma correcta.				x	
Considera que el componente genera las ontologías de dominio a					x

partir de archivos XML de forma correcta.					
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.					x
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

TABLA 28: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 6. FUENTE: ELABORACIÓN PROPIA.

Afirmación	**Alternativas de respuestas**				
	1	**2**	**3**	**4**	**5**
Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta				x	
Considera que el componente genera las ontologías de forma correcta.					x
Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.					x
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.				x	
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

TABLA 29: CUESTIONARIO DE ACTITUDES REALIZADO AL ENCUESTADO # 7. FUENTE: ELABORACIÓN PROPIA.

Afirmación	**Alternativas de respuestas**				
	1	**2**	**3**	**4**	**5**

Considera que el componente extrae las clases y propiedades del archivo XML de manera correcta				x	
Considera que el componente genera las ontologías de forma correcta.				x	
Considera que el componente genera las ontologías de dominio a partir de archivos XML de forma correcta.					x
Considera que agrupar el conocimiento de un dominio mejora el proceso de búsqueda de información.				x	
Considera que se logra esquematizar el conocimiento a través de las ontologías.					x

Anexo # 2: Respuestas dadas por los expertos para cada indicador

Expertos	**Indicadores**				
	1	**2**	**3**	**4**	**5**
1	4	5	5	4	5
2	5	5	5	4	5
3	4	4	5	4	5
4	5	5	5	5	5
5	4	4	5	5	5
6	4	5	5	4	5
7	4	4	5	4	5