Universidad de las Ciencias Informáticas.

Facultad 6

Herramienta para mapeo semi-automático de geo-ontologías al estándar de la WGS84.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:
Ángel Osmel Rodríguez Téllez.

Tutores:
Msc. Yuniel E. Proenza Árias.

Ing. Adrián Gracia Águila.

La Habana, 2013
¨Año del 55 Aniversario del Triunfo de la Revolución¨

Declaración de Autoría.

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los___ días del mes de_____ del año_____.

Autor:

Ángel Osmel Rodríguez Téllez

Tutor:

Yuniel E. Proenza Arias

Tutor:

Adrián Gracia Águila

Nada es comparable a la satisfacción de

Tutor: Yuniel Eliades Proenza Arias

Graduado de Ingeniería Informática en la Universidad de Holguín en el año 2006. Máster en Ciencias en Ingeniería de Software e Inteligencia Artificial por la Universidad de Málaga en el año 2011. Se ha desempeñado como profesor en la Universidad de las Ciencias Informáticas, además de otras tareas asociadas a la producción. Tiene experiencia laboral en la línea del desarrollo de sistemas, sistemas de información geográfica, la arquitectura de software y el desarrollo de ontologías.

E-mail: yproenza@uci.cu

Tutor: Adrián Gracia Águila

Ingeniero en Ciencias Informáticas graduado en la Universidad de las Ciencias Informáticas (UCI) en el 2009. Actualmente se desempeña como profesor de la Universidad de las Ciencias Informáticas y Líder de la LPS Aplicativos SIG.

E-mail: agracia@uci.cu

Dedico este trabajo:

Especialmente a la persona que aunque no se encuentra entre nosotros guió mi vida hasta el momento en que no pudo continuar con la suya. A ti: MIMA, haz estado y estarás siempre conmigo.

A mi mamá y a mi papá, personas que siempre han estado a mi lado y me han mostrado su cariño verdadero. Los quiero con la vida.

A mis dos hermanas Diana y Adrie, espero que les sirva de ejemplo y que sepan que cuando se pone empeño no hay nada que no se pueda hacer.

Agradecimientos:

Agradezco a mi mamá y a mi papá por ser tan especiales para mí, por impulsarme a escoger el camino correcto y estar a mi lado y apoyarme en lo bueno y lo malo que ha pasado por mi vida.

A mis hermanitas queridas Diana y Adrie por darme su cariño siempre que están a mi lado.

A mis abuelos Geroncio, María y Monino, por apoyarme y ayudarme tanto desde que vine al mundo.

A mi primo Tony, por ser mi hermano, mi amigo y por ayudarme incondicionalmente cada vez que lo he necesitado.

A mi tía Vivian, mi tío Omar y todas esas personas que hacen a mi familia grande (no solo por numerosa) por ayudarme y preocuparse por mí.

A mi tutor Proenza que supo guiarme por el camino correcto, dando su apoyo ilimitado para que yo saliera adelante.

A mi tutor Adrián Gracia por aconsejarme en los momentos más oportunos.

A mi tribunal de tesis por sus críticas constructivas y enseñarme que cuando se quiere se puede.

A mis amigos que han convivido conmigo y saben que no es muy fácil soportar mis boberías: Diosbel, Ortiz, Chuchy, El Merco, El Manu y Ale.

A mis amigos de la vieja escuela Rodolfo, Papito, Franier, Oslanier, El Wilfre, Liu, Kare, Ari, David, Fariñas y Tristá. Que tiempos aquellos.

En fin a todos mis amigos y compañeros de aula que de una forma u otra ayudaron a que este sueño se realizara.

El término ontología se ha empleado desde hace muchos siglos en el campo de la filosofía y del conocimiento y hace ya varias décadas cobró especial relevancia en el campo de la biblioteconomía y la documentación. Hoy ha sufrido un nuevo impulso debido al desarrollo de la representación semántica de la información donde prima la idea de transformar la red no sólo en un espacio de información, sino también en un espacio de conocimiento.

El desarrollo de los llamados Sistemas de Información Geográfica Gobernados por Ontologías (SIGGO) ha captado la atención de la comunidad científica en años recientes. La representación semántica de información geográfica permite consultar la información a más alto nivel de abstracción y de manera homogénea sea cual sea el origen o el tipo de información representada.

Debido a la existencia de deficiencias en el esfuerzo de lograr tales representaciones se hace necesario crear una herramienta que las transforme a un estándar determinado para lograr el objetivo principal de esta investigación que es garantizar la visualización de geo-ontologías.

Palabras Clave: Ontología, Geo-ontología, Sistemas de Información Geográfica Gobernados por Ontologías.

Índice:

Índice de figuras:

Índice de Tablas:

Introducción

El ser humano como especie principal y más avanzada del planeta ha venido evolucionando con el transcurrir de los años con el fin de mejorar su vida social. Para lograr que estas transformaciones sean positivas ha tenido que desarrollar esa capacidad que lo hace tan especial: el poder de pensar, de esta forma ha podido mejorar todas las esferas de la sociedad. En la búsqueda de tales resultados ha sido necesario mezclar los conocimientos obtenidos en cada una de las ciencias que ha desarrollado durante este proceso evolutivo, los cuales han dado origen a un conjunto de elementos u objetos que permiten a los seres humanos poder visualizar estos avances científicos, y a su vez facilitar el proceso de investigación para la obtención de nuevos resultados que permitan mejorar la vida del hombre.

Dentro de estos elementos aparecieron las Tecnologías de la Información y las Comunicaciones (TIC), que son el ***"conjunto de medios (radio, televisión y la telefonía convencional) de comunicación y las aplicaciones de información que permiten la captura, producción, almacenamiento, tratamiento, y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética"*** (1). Las TIC incluyen la electrónica como tecnología base que soporta el desarrollo de las telecomunicaciones, la informática y el mundo audiovisual.

Estas tecnologías se han introducido en todos los campos posibles en los cuales ha marcado un incremento en cuanto a la eficiencia y calidad de la producción, específicamente en la Geoinformática, donde se han obtenido resultados increíbles que han facilitado a las personas realizar disímiles tareas con mapas a un nivel mucho más exacto. Estas mejoras se deben a la introducción de nuevas técnicas de representación de la información que con el transcurrir de los años han ganado gran aceptación en la comunidad científica. La utilización de estas tecnologías trae consigo la aparición de los Sistemas de Información Geográfica (SIG) los cuales son una acumulación de datos geográficos espacialmente georreferenciados y procedimientos que ayudan a las personas a tomar decisiones para realizar una tarea determinada, mediante la captura, manejo, manipulación, análisis y modelado de los mismos (2). Dentro de los SIG se encuentran los Sistemas de Información Geográfica Gobernados por Ontologías (SIGGO), que no son más que SIG en los que se incorporan el uso de ontologías como un componente activo. Una ontología según Gruber es ***"una especificación explícita y formal sobre una conceptualización compartida"*** (3), este término ha ganado espacio en muchos de los campos de investigación de la

informática como una alternativa eficaz de representación de la información y base para el razonamiento sobre el conocimiento almacenado.

El campo de investigación en torno a las ontologías en estos momentos es uno de los más aplicables en las tecnologías de varios países. En Cuba por ejemplo, se destaca la labor del Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV), surgido en el 2004 y está orientado a investigaciones teóricas y aplicadas al área de Reconocimiento de Patrones (RP) y Minería de Datos (MD).

El desarrollo de los llamados Sistemas de Información Geográfica Gobernados por Ontologías ha captado la atención de la comunidad científica en años recientes. La representación semántica de información geográfica permite consultar la información a más alto nivel de abstracción y de manera homogénea sea cual sea el origen o el tipo de información representada. Existen esfuerzos recientes de incluir además la capacidad de mostrar en mapas la información semántica incluida en una geo-ontología (ontología que contiene datos espaciales). Las propuestas principales presentan dos deficiencias esenciales: 1) se adaptan solamente a estándares específicos, dificultando visualizar la información de cualquiera de las geo-ontologías existentes o las nuevas que se puedan desarrollar fuera de los estándares establecidos y 2) no permiten realizar un mapeo de los modelos de representación a los estándares necesarios para la visualización. La principal herramienta utilizada para la visualización es Map4rdf[1], la que permite visualizar solamente información representada en una geo-ontología siguiendo el estándar de la WGS84[2].

Por lo que el **problema científico** de la presente investigación se centra en: ¿cómo garantizar la visualización sobre mapas de geo-ontologías que no estén representadas en el estándar WGS84 utilizando la herramienta Map4rdf?

El **objeto de estudio** de la investigación en curso es la representación de información geo-espacial mediante ontologías. Por tanto, el **campo de acción** estaría enmarcado en la representación de información geo-espacial mediante ontologías utilizando el estándar de la WGS84.

Para dar solución al problema identificado anteriormente se propone el siguiente **objetivo general**: elaborar una herramienta para el mapeo semi-automático de geo-ontologías al estándar WGS84.

[1] Herramienta para la visualización de geo-ontologías.
[2] Sistema de coordenadas geográficas mundial que permite localizar cualquier punto de la Tierra.

La **idea a defender** de este trabajo es: si se realiza una herramienta para el mapeo semi-automático de geo-ontologías al estándar de la WGS84 se garantizará su visualización utilizando Map4rdf.

Para darle cumplimiento al objetivo trazado se determinó que las **tareas de la investigación** estarán encaminadas a:

- Identificar propuestas para la representación de información geo-espacial mediante ontologías utilizando el estándar WGS84.
- Seleccionar de entre las propuestas identificadas la que se utilizará como base para la representación de la información usando WGS84.
- Identificar propuestas de herramientas para el mapeo de ontologías del dominio geo-espacial al estándar WGS84.
- Caracterizar las herramientas identificadas, previendo la posibilidad de adaptación de alguna de ellas para la solución del problema planteado y destacando en cada caso las insuficiencias presentes.
- Seleccionar las tecnologías a utilizar en el desarrollo de la herramienta, así como la metodología de desarrollo de software a seguir teniendo en cuenta los requerimientos de la aplicación y las características del entorno de trabajo.
- Elaborar la documentación establecida por la metodología de desarrollo de software seleccionada como resultado del proceso de desarrollo.
- Elaborar la herramienta para el mapeo de geo-ontologías a WGS84.
- Realizar pruebas de software a la herramienta como verificación de su correcto funcionamiento.
- Probar la visualización de las ontologías generadas en la herramienta Map4rdf.
- Elaborar un artículo asociado al desarrollo y utilidad de la herramienta.

A lo largo de esta investigación se han tenido en cuenta **métodos científicos** que son de suma importancia para la realización satisfactoria de la misma que a continuación se presentan:

Teóricos:

- **Histórico – lógico:** Se utiliza con el objetivo de realizar un estudio sobre la evolución y el desarrollo que han tenido los Sistemas de Información Geográfica Gobernados por Ontologías, lo que permite desarrollar una valoración sobre en qué etapa se encuentran dichos sistemas y cuál es la tecnología más factible a utilizar para el desarrollo de los mismos y realizar una valoración sobre el estado del arte del objeto de estudio para realizar la proyección futura en el campo de acción escogido.

- **Analítico - Sintético:** Se utiliza para estudiar y analizar la documentación y bibliografías de diferentes autores, para sustentar desde el punto de vista teórico y práctico los elementos que se relacionan con el proceso de mapeo semi-automático de geo-ontologías al estándar WGS84.

- **Modelación:** La modelación es justamente el proceso mediante el cual se crean modelos con vista a investigar la realidad, por lo que es el más importante en el proceso de construcción del software. Se utiliza para realizar los modelos ingenieriles de la solución que se plantea.

El presente trabajo de diploma consta de 3 capítulos:

En el **capítulo 1** se tratan aquellos temas que constituyen la fundamentación teórica de la investigación a realizar, entre ellos el estudio de las soluciones existentes y cómo funciona el proceso de visualización de una geo-ontología al estándar de la WGS84 con la herramienta Map4rdf. En el **capítulo 2**, denominado Herramientas y Tecnologías, el cual describe las herramientas y tecnologías a emplear. El **capítulo 3** está dedicado al diseño del sistema, a través de los diferentes artefactos propuestos por la metodología de desarrollo del software seguida, se realiza la selección de la arquitectura a utilizar y se describe el proceso de implementación y pruebas, el cual abarca todo lo relacionado con la implementación del sistema, además del diseño y aplicación de las pruebas para comprobar la obtención de los resultados esperados.

Capítulo # 1 Fundamentación Teórica

En el presente capítulo se describen los principales conceptos asociados al problema planteado el cual se centra en cómo garantizar la visualización sobre mapas de geo-ontologías que no estén representadas en el estándar WGS84 utilizando la herramienta Map4rdf.

1.1- Conceptos asociados al dominio del problema.

Los conceptos asociados al dominio del problema se toman a partir de las definiciones más actualizadas; estos conceptos son útiles para el entendimiento del problema en cuestión y ayudan a entender las relaciones que existen entre ellos.

1.1.1 Sistemas de Información Geográfica.

Dada la necesidad de almacenar y manipular información georreferenciada surgen los sistemas de información geográfica posibilitando realizar varias manipulaciones con mapas tales como superposiciones de forma rápida y precisa, transformaciones de escala, representaciones gráficas y gestión de bases de datos. De modo que toda actividad humana puede e incluso debe tener asociada la presencia de un SIG.

Un SIG puede ser concebido como una especialización de un sistema de bases de datos, caracterizado por su capacidad de manejar datos geográficos, que están georreferenciados y los cuales pueden ser visualizados como mapas. (2)

El autor de la presente investigación afirma que un SIG no es más que la integración de programas informáticos, periféricos y medios implementados, capaces de capturar, analizar y representar información de los objetos georreferenciados a través de mapas temáticos[3] en un ordenador. Dicha información puede ser consultada, compartida o modificada según las necesidades del usuario del SIG. Son sistemas que facilitan la toma de decisiones respecto a la información georreferenciada que estos brindan.

[3] Mapas que representan cualquier fenómeno geográfico de la superficie terrestre.

1.1.2. Ontología.

El término ontología en la contemporaneidad es empleado por científicos de la información en un sentido diferente al que usan los filósofos, contribuyendo a que su estudio y aplicación en los sistemas de información[4] constituyen un campo de exploración reconocido. Cuando se trata de hacer explícito y clarificar el conocimiento de un dominio dado, el uso de ontologías puede ser de gran ayuda permitiendo formular un exhaustivo y riguroso esquema conceptual, con la finalidad de facilitar la comunicación y compartir la información entre diferentes sistemas. Orientados hacia esta dirección, diferentes autores han abordado el tema de las ontologías.

Claudio Gutiérrez señala que **"Ontología es teoría de los objetos. Una ontología es la colección de entes (objetos) a que se refieren los enunciados de cada disciplina particular. El tema de la ontología es el tema de lo que hay."** (4)

Gangemi y otros autores se refieren a una ontología como un entendimiento común y compartido de un dominio, que puede comunicarse entre científicos y sistemas computacionales. (5)

Plantea el investigador Robert Jasper que una ontología puede tomar una variedad de formas, pero necesariamente deberá incluir un vocabulario de términos y alguna especificación de su significado. Afirma que esto incluye definiciones y una indicación de cómo los conceptos están interrelacionados. Las ontologías son usadas para mejorar la comunicación entre personas o computadoras. (6)

Para Van Heijst y otros autores una ontología es una especificación explícita a nivel de conocimiento de una conceptualización. Es un conjunto de distinciones que son significativas para un agente. (7)

Existen diferentes puntos de vistas que explican en qué consisten las ontologías o para qué se utilizan. Atendiendo a esta diversidad y partiendo del análisis de las definiciones expuestas anteriormente, el autor de la presente investigación asume la que ofrece Tom Gruber por ser la que ofrece una información más detallada y la que mejor se adentra en este tema, la cual expresa: **"*Una ontología es una especificación formal y explícita de una conceptualización compartida*"**. (3) Conceptualización es una abstracción, un enfoque simplificado del mundo que se desea representar, construido a partir de los conceptos y

[4] Conjunto de elementos orientados al tratamiento y administración de datos e información.

relaciones identificados. Cuando se emplea el término formal, se hace referencia a la necesidad de que el conocimiento deba ser obvio y comprensible por los ordenadores.

Los estudios realizados permiten percibir que una ontología necesariamente incluirá definiciones e interrelaciones entre conceptos. A criterio de Tom Gruber las ontologías tienen los siguientes componentes que permitirán representar claramente el conocimiento de algún dominio: (3)

- **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 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 categorizar-clase, asignar-fecha, etc.
- **Instancias:** se utilizan para representar objetos determinados de un concepto.
- **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 C1, A es B”.

Una ontología es una base de datos que describe los conceptos de algún dominio, sus propiedades y la relación de estos conceptos entre sí. A partir del desarrollo de las ontologías devienen diferentes clasificaciones. Gangemi y otros autores distinguen tres tipos de ontologías: (5)

- **Ontologías de un dominio:** en las que se representa el conocimiento especializado de un dominio o sub-dominio, como la medicina, la cardiología, etc.
- **Ontologías genéricas:** en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos.
- **Ontologías representacionales:** en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan meta-ontologías.

Teniendo en cuenta el punto de vista de la gestión del conocimiento que se adopte, se pueden diferenciar varios tipos de ontologías. Entre las caracterizaciones más frecuentes en la literatura estudiada se destaca una de ellas, el nivel de generalidad y/u objeto de la ontología. Identificado con este criterio, Guarino clasifica las ontologías en función de su nivel de dependencia de una tarea definida: (8)

- **Ontologías de alto nivel:** describen conceptos generales tales como tiempo, materia, objeto, evento, acción, etc., que son independientes de un dominio particular.
- **Ontologías de dominio:** describen el vocabulario relacionado al dominio genérico mediante la especialización de los términos introducidos en las ontologías de alto nivel.
- **Ontologías de tareas:** describen el vocabulario relacionado a una tarea mediante la especialización de los términos introducidos en las ontologías de alto nivel.
- **Ontologías de aplicación:** describen conceptos tanto de un dominio como de una tarea particular, que son muchas veces especializaciones de ambas ontologías relacionadas.

Las ontologías apoyan, en mayor o menor medida, a facilitar los procesos de intercambio entre personas y agentes en un sistema colaborativo. Entre los motivos que justifican la necesidad del uso de ontologías se destacan:

- Compartir un entendimiento común acerca de la estructura y semántica de la información entre personas y agentes de software.
- Habilitar la reutilización del conocimiento del dominio.
- Hacer explícitas las premisas acerca de un dominio.
- Separar el conocimiento de un dominio de conocimiento operacional.
- Analizar formalmente el conocimiento de un dominio.

A criterio de Tom Gruber las ontologías se diseñan cuando se selecciona el cómo será representado algo en una ontología. Propone un conjunto de pautas y criterios de diseño para ontologías cuyo fin es compartir conocimiento e inter-operar con otros sistemas basados en la conceptualización compartida. (3)

- **Claridad:** las definiciones deben ser objetivas e independientes del contexto social o computacional. Estas deben ser documentadas en lenguaje natural, a su vez una definición completa es la designada antes que una definición parcial.

- **Coherencia:** una ontología debe ser coherente, lo que significa que debe sancionar inferencias que son consistentes con las definiciones.
- **Extensibilidad:** una ontología debe estar planteada para explicar los usos del vocabulario compartido. Se debe poder precisar nuevos términos para usos específicos basándose en el vocabulario.
- **Dependencia mínima de la codificación:** la conceptualización debe ser detallada a nivel del conocimiento, sin depender de una codificación particular a nivel de símbolos.
- **Compromiso ontológico mínimo:** una ontología debe tener pocas excepciones acerca del mundo permitiendo a quienes las utilizan la libertad de especializar e instanciar la ontología como se necesite.

Los elementos planteados anteriormente permiten constatar el papel significativo que ocupan las ontologías en la resolución de interoperabilidad semántica entre sistemas de información y su utilización en un contexto determinado. Para que su funcionalidad sea correcta, las ontologías deben desarrollarse teniendo en cuenta sus diferentes clasificaciones, componentes, necesidades, funciones, pautas y criterios para su implementación.

1.1.3 Sistemas de Información Geográfica Gobernados por Ontologías.

"Un SIG que utiliza ontologías para generar nueva información a partir de un conjunto previo de datos es lo que se denomina Sistema de Información Geográfica Gobernado por Ontologías (SIGGO)". En los SIGGO las ontologías son una componente más, como lo es la base de datos temáticos o espaciales que interviene y coopera de la misma manera para alcanzar los objetivos para los cuales fue creado. (9)

Los SIGGO no son más que SIG en los que se incorporan el uso de ontologías como un componente activo. Los SIGGO son construidos utilizando clases derivadas de las ontologías y como consecuencias de esto se extrae el conocimiento embebido en estas para aportar mayor eficiencia y robustez al sistema (9). Los SIGGO surgen a partir de que los SIG convencionales no soportan sus negocios a través de la expresividad semántica de los objetos espaciales, pero una vez logrando que un SIG se integre con un sistema proveedor de geo-ontologías pueda hacer uso del conocimiento embebido en estas y a su vez

logre generar nuevo conocimiento que consiga almacenarse y ser utilizado posteriormente, se convierte entonces en un SIGGO.

Estructura de un SIGGO.

Los SIGGO tienen dos fases que son fundamentales en su estructura (10).

- **Fase de generación del conocimiento:** comprende la especificación de las ontologías utilizando un editor de ontologías, la generación de nuevas ontologías a partir de las ya existentes y la traducción de ontologías a componentes de software.

- **Fase de Uso del conocimiento:** se apoya en los productos obtenidos en la fase anterior: una serie de ontologías especificadas en un lenguaje formal y en una serie de clases. Las ontologías están disponibles para ser navegadas por el usuario final y para su utilización en la generación de aplicaciones, análisis y finalmente las posibles alternativas de decisión.

1.1.4 Geo-ontología.

Las ontologías se crean conforme a las necesidades que existan en un dominio, pueden cambiarse, adaptarse o personalizarse para propósitos específicos. En el caso de los Sistemas de Información Geográfica (SIG) que utilizan información semántica, éstas constituyen una herramienta fundamental. Orientados hacia esta dirección, los investigadores Francisco Vera Voronisky y Eduardo Garea Llano en su artículo ¨Alineamiento de ontologías en el dominio geoespacial¨ plantean que las ontologías que tratan la temática geoespacial presentan características particulares de este campo de estudio. Debido a su nivel de especialización, a estas ontologías geográficas se les han denominado geo-ontologías. (11) Más adelante aseveran que las geo-ontologías cumplen con todas la características de una ontología convencional, pero tienen además propiedades propias de este dominio.

Según Eduardo Garea Llano una geo-ontología es una ontología que ofrece una descripción de entidades geográficas y difiere de otras ontologías por la presencia predominante de relaciones topológicas. Las geo-ontologías pueden utilizarse para hacer explícita la semántica del contenido de la información geográfica de servicios Web, con el fin de mejorar el descubrimiento y recuperación en la Web. (12)

Las geo-ontologías cumplen con todas la características de una ontología convencional, pero presentan ciertas particularidades propias del dominio geográfico. Las ontologías geográficas contienen una clase de relaciones espaciales que describen como están ubicados algunos objetos en el espacio en correspondencia a algún objeto de referencia. Las relaciones espaciales contienen las relaciones topológicas entre entidades, y éstas a su vez describen propiedades como la adyacencia, la conectividad y la intersección entre clases geoespaciales. Otras de las propiedades particulares de las ontologías geográficas son la posición espacial o la geometría de un objeto. Una ontología geográfica describirá objetos a los que se les asigne una localización en la superficie terrestre y a las relaciones entre éstos.

Las investigaciones y trabajos estudiados referentes al tema permiten constatar que una geo-ontología permite organizar y dar sentido al conocimiento del domino geográfico. En la actualidad diferentes ramas como la Topografía, Geología, Hidrología, entre otras, han sido beneficiadas por la utilización de ontologías en la representación de información.

1.1.5 WGS84

El WGS84 es un sistema de coordenadas geográficas mundial que permite localizar cualquier punto de la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas. WGS84 son las siglas en inglés de World Geodetic System 84 (Sistema Geodésico Mundial 1984).

El Sistema Geodésico Mundial es un estándar para su uso en la cartografía, geodesia y navegación. Cuenta con un estándar de coordenadas de la Tierra, un estándar de referencia de la superficie esférica (el dato o elipsoide de referencia) para los datos de altitud primas, y una superficie equipotencial gravitacional (el geoide) que define el nivel del mar nominal. El origen de coordenadas de WGS 84 está destinado a ser ubicado en el centro de la masa de la Tierra, se cree que el error es menos de 2 cm.

Se trata de un estándar en geodesia, cartografía, y navegación, que data de 1984. Tuvo varias revisiones (la última en 2004), y se considera válido hasta una próxima reunión (aún no definida en la página web oficial de la Agencia de Inteligencia Geoespacial). Se estima un error de cálculo menor a 2 cm. por lo que es en la que se basa el Sistema de Posicionamiento Global (GPS).

Consiste en un patrón matemático de tres dimensiones que representa la tierra por medio de un elipsoide, un cuerpo geométrico más regular que la Tierra, que se denomina WGS 84. El estudio de este y otros modelos que buscan representar la Tierra se llama Geodesia. (13)

1.1.6 Map4rdf.

Es una herramienta de visualización facetada para explorar y visualizar conjuntos de datos RDF[5] (Resource Description Framework por sus siglas en inglés) enriquecidos con información geométrica. Permite la visualización geoespacial, mediante Google Maps, de los recursos contenidos en un SPARQL[6] endpoint (servicio que interpreta el lenguaje de consultas). Además, no sólo permite la representación de puntos, sino también de recursos de tipo LineString (línea), Polygon (polígono) y recursos con información geoespacial disponibles en el SPARQL endpoint. (14)

Para utilizar esta herramienta, se debe desplegar en un servidor de aplicaciones, y modificar el fichero de configuración. En el caso de este prototipo, se desplegará, en un servidor Tomcat[7].

En el fichero de configuración, sólo hace falta cambiar dos parámetros para que funcione la aplicación, dichos parámetros son:

- endpoint.url: dirección del SPARQL endpoint con el que vamos a trabajar.

- geometry.model: para indicar el tipo de modelo de geometría que tienen los recursos.

1.2 Objeto de Estudio.

El objeto de estudio de esta investigación es la representación de información geo-espacial mediante geo-ontologías, centrando la atención en la representación de información geo-espacial mediante geo-ontologías utilizando el estándar de la WGS84 y para esto es necesario enfocarse en el estado en que se encuentra el desarrollo de soluciones para el mapeo semi-automático de geo-ontologías.

La herramienta Map4rdf solo representa geo-ontologías que estén en el estándar de la WGS84 pero existen varias que no están en dicho estándar, imposibilitando representarlas con Map4rdf, de ahí la necesidad de desarrollar un software que modifique dichas geo-ontologías llevándolas al estándar de la WGS84 para su representación. Es importante tener conocimiento de lo que significa desarrollar una herramienta de este tipo, la cual, aunque dista de las especificidades más robustas en lo referente al

[5] Lenguaje de objetivo general para representar la información en la web.
[6] Acrónimo recursivo del inglés Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF.
[7] Servidor HTTP y contenedor de servlets (objetos que corren dentro y fuera del contexto de un contenedor).

mapeo de geo-ontologías, sí cumple con un objetivo: obtener siempre una representación de una geo-ontología con la herramienta Map4rdf.

Actualmente no existe un software que modifique una geo-ontología llevándola al estándar de la WGS84, lo que ubica al desarrollo de esta investigación como novedoso y muy útil para la comunidad científica interesada en el trabajo con geo-ontologías.

Conclusiones Parciales.

Durante el desarrollo de este capítulo se trataron los principales conceptos asociados al dominio del problema, tales como ontología, geo-ontología como subconjunto de las ontologías, Sistemas de Información Geográfica, Sistemas de Información Geográfica Gobernado por Ontologías, el estándar de la WGS84 y la herramienta Map4rdf para la representación de las geo-ontologías. Este esbozo permitió entender la necesidad latente de realizar una herramienta para el mapeo semi-automático de geo-ontologías al estándar de la WGS84 que garantice su visualización utilizando Map4rdf.

Capítulo # 2 Herramientas y Tecnologías

En el proceso de desarrollo de un software es necesario definir una metodología de desarrollo como guía para alcanzar una solución, dicha metodología se encarga de estructurar, planear y controlar una serie de procedimientos esenciales para la construcción de una aplicación. Para darle un formato a la solución del problema en cuestión se definirá un lenguaje de modelado así como una herramienta CASE para representarlo y de esa forma dar paso al eslabón más importante que es la elaboración de la herramienta que se requiere, al llegar a este punto es esencial definir un framework de desarrollo adecuado y el lenguaje de programación que cubran las necesidades del software a construir. En el presente capítulo quedarán definidas las herramientas y tecnologías antes mencionadas con el objetivo de hacer menos engorroso el trabajo en aras de encontrar la solución esperada.

2.1. Metodología de desarrollo de software.

Su necesidad viene dada por la engorrosa tarea de la creación de aplicaciones de software que respondan a las habilidades y restricciones de los estándares del software en el mundo. Ello dificulta gradualmente el trabajo de los desarrolladores, lo que da pie al surgimiento a las metodologías de desarrollo de software. Estas definen quién debe hacer qué, cuándo y cómo debe hacerlo para obtener los distintos productos en un determinado proceso de software. Su objetivo principal es guiar a los desarrolladores en el diseño e implementación de nuevas aplicaciones de probada calidad. Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo, las cuales se pueden clasificar en dos grupos:

- **Las metodologías pesadas:** Están orientadas al control de los procesos, estableciendo rigurosamente la definición de roles, herramientas, actividades, artefactos y notaciones para el modelado y documentación detallada. Prestan mayor énfasis a la planificación y control del proyecto, en especificar de forma precisa los requisitos y el modelado.

- **Las metodologías ágiles:** Proporcionan una serie de pautas, principios y técnicas pragmáticas que aseguran la entrega del proyecto con menos complicación y satisfactoriamente. Esto lo cumple a través de muestra de versiones parcialmente funcionales del software al consumidor en cortos

períodos de tiempo para que se pueda evaluar y sugerir cambios en el producto según se va desarrollando. Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software.

2.1.1 Proceso Unificado de Desarrollo de Software (RUP).

RUP (Rational Unified Process según sus siglas en inglés) es una metodología de desarrollo de software pesada, la cual se centra en la definición detallada de los procesos y tareas a realizar y las herramientas a utilizar. Pretende prever todo el desarrollo del producto de antemano por lo que requiere una extensa documentación. Jacobson, Booch y Rumbaugh describen esta metodología de la siguiente manera: ***"Se puede definir como un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos"*** (15)

La metodología RUP cuenta con 3 características esenciales: (16)

- **Dirigido por casos de uso:** Los casos de uso constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo del software.
- **Centrado en la arquitectura:** Se establece una arquitectura candidata al inicio del desarrollo del sistema que funcionará como guía y se irá perfeccionando en las restantes fases de desarrollo.
- **Iterativo e incremental:** El desarrollo iterativo garantiza la corrección de los errores en cada iteración, brindando la posibilidad de que los elementos sean integrados continuamente, lo que garantiza un producto más robusto y de mayor calidad.

RUP se repite a lo largo de una serie de ciclos de desarrollo que constituyen la vida de un sistema, donde cada ciclo concluye con una versión del producto. Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Todas las fases se subdividen en N iteraciones y terminan con un hito.

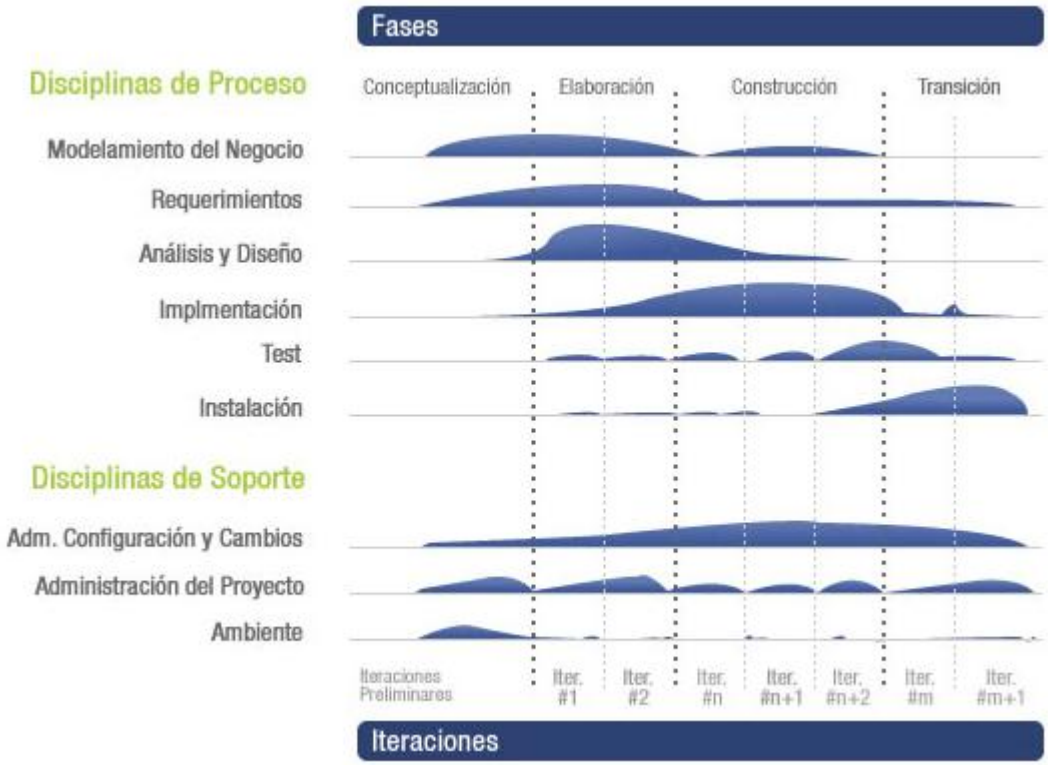

Figura 1: Fases y flujos que utiliza RUP.

Las nociones de casos de uso y de escenarios utilizadas en RUP han demostrado ser una manera excelente de capturar los requerimientos funcionales y asegurarse que direccionan el diseño, la implementación y la prueba del sistema, logrando así que el sistema satisfaga las necesidades del usuario. Una característica a destacar de RUP es que describe como controlar, rastrear y monitorear los cambios, en ambientes en los cuales el cambio es inevitable, para permitir un desarrollo iterativo exitoso.

RUP se propone como metodología para la realización de la herramienta para el mapeo semi-automático de geo-ontologías al estándar WGS84 que garantice su visualización utilizando Map4rdf. Se garantiza que con su aplicación se genere la documentación y artefactos imprescindibles para que los futuros clientes tengan como base una guía para su entendimiento.

2.2. Herramienta CASE.

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) se definen como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Investigación Preliminar, Análisis, Diseño, Implementación e Instalación).

El Instituto Nacional de Estadística e Informática en el artículo ¨Herramientas CASE¨, pone a disposición de sus lectores otras definiciones que se considera importante mencionar para entender mejor la terminología CASE. (23)

Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas.

Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Las herramientas CASE alcanzaron su valor en el año 1985, convirtiéndose en una familia de métodos favorablemente estructurados para el planeamiento, análisis y diseño del proceso de desarrollo del software. Su importancia en la actualidad ha propiciado que muchas instituciones y universidades realicen trabajo de investigación consciente de las ventajas que ofrecen. Entre ellas se encuentran:

Progreso en la calidad, fiabilidad, utilidad y rendimiento.

El entorno de producción de documentación para software mejora la comunicación, mantenimiento y actualización.

Reducción del costo de producción de software.

2.1.1 Visual Paradigm.

Existen varias herramientas CASE orientadas a OWL[8], entre las que pueden mencionarse: ArgoUML, Poseidon, MagicDraw UML, Visual Paradigm, BorlandTogether. Para dar soporte al modelado visual de la propuesta se asume Visual Paradigm, lo cual se argumenta a continuación:

Visual Paradigm para UML (Lenguaje Unificado de Modelado), es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software orientado al desarrollo y la construcción de objetos, a las mejoras, la realización de comprobaciones y el despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código inverso, generar código desde diagramas y generar documentación. Además, el código de ingeniería inversa es compatible con Java, C++, .Net DLL o EXE, CorbaIDL, XML, XML Schema, Hibernate y JDBC.

Visual Paradigm soporta estándares de la industria clave, tales como Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI entre otros, ofrece un completo conjunto de herramientas para la captura de requisitos, la planificación de programas, la planificación de controles, la clase de modelado y modelado de datos. Es compatible con las notaciones de UML más recientes para el modelado y proporciona una interfaz intuitiva centrada que se integra perfectamente con aplicaciones como Eclipse/WebSphere, JBuilder, NetBeans, IntelliJ IDEA, JDeveloper y Web Logic Workshop ofreciendo una sincronización sofisticada y en tiempo real de los códigos y modelos.

A partir del estudio realizado se identificaron varias ventajas que ofrece Visual Paradigm:

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.

[8] Acrónimo del inglés Web Ontology Language, un lenguaje de marcado para publicar y compartir datos usando ontologías en la web.

- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

2.3. Lenguaje de Modelación.

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. (19)

El lenguaje de modelado predefinido por Visual Paradigm es el Lenguaje Unificado de Modelado (UML), el cual se describe a continuación:

2.3.1. Lenguaje Unificado de Modelado (UML).

El Lenguaje Unificado de Modelado (UML) es un lenguaje para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas. Se convirtió en estándar del Object Management Group (OMG) en 1997 y representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas grandes y complejos. (20)

Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML permite modelar de manera conceptual procesos de negocio y funciones de sistema, también actividades específicas como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. UML facilita un material de apoyo que le permita al lector definir diagramas propios, y a su vez, entender el modelamiento de diagramas ya existentes.

A continuación se definen las principales ventajas del Lenguaje Unificado de Modelado (UML):

UML brinda la posibilidad de modelar variados tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. Establece una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos, facilitando que diferentes diseñadores modelando sistemas desiguales, puedan entender los diseños de los otros con claridad. Este lenguaje ofrece además nueve diagramas en los cuales modelar sistemas:

- Diagramas de Casos de Uso para modelar los procesos 'business'.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y de los conceptos más usados orientados a objetos. El ingeniero e investigador José Enrique González Cornejo plantea en su artículo ¨ El Lenguaje de Modelado Unificado (UML) ¨ que esta herramienta debe apoyar todos los diagramas de los nueve que componen UML. Debe soportar la diagramación de casos de uso, permitir definir la visión estática con diagramas de clases y diagramas de objeto, permitir la definición de la visión dinámica, tales como los diagramas de secuencia, la actividad de los estados, de colaboración y el despliegue de componentes que forman el sistema. (21)

Resulta conveniente tener en cuenta las consideraciones que sobre las ventajas de UML ofrece Santiago Meliá en su Tesis Doctoral “WebSA: Un Método de Desarrollo Dirigido por Modelos de Arquitectura para Aplicaciones Web”. (22)

1. Dispone de una semántica y sintaxis precisa, permitiendo conocer de forma no ambigua lo que el diagrama indica.
2. Tiene una alta capacidad expresiva lo que posibilita representar todos los aspectos de la arquitectura web.
3. Tiene capacidad para representar el sistema a diferentes niveles de abstracción, así cada uno de los miembros de desarrollo, arquitectos, diseñadores y analistas pueden enfocarse en los problemas que les ocupan olvidándose de los demás aspectos.
4. Los modelos pueden ser intercambiables entre diferentes herramientas que tengan soporte de UML, así los modelos pueden ser reutilizados.

UML ha tenido un gran éxito a nivel mundial, pero presenta un conjunto de desventajas que a continuación se relacionan:

- UML no es un método de desarrollo. No expresa cómo pasar del análisis al diseño y de este al código. No forma una serie de pasos que llevan a producir código a partir de unas especificaciones.

- UML al no ser un método de desarrollo es independiente del ciclo de desarrollo que se vaya a seguir, puede ajustar en un tradicional ciclo en cascada, en un evolutivo ciclo en espiral o incluso, en los métodos ágiles de desarrollo.

- Diversos desarrolladores consideran que UML es algo impreciso dentro de su notación, por ejemplo: al hacer referencias a un diagrama con servidores, no se sabe si los servidores simbolizados se encuentran operativos, restringidos, pasivos, etc. Por este motivo se le califica como un poco “inexacto”.

- UML no se presta con facilidad al diseño de sistemas distribuidos. En tales sistemas cobran importancia factores como transmisión, serialización, persistencia, etc. UML no cuenta con

maneras de describir tales factores. No se puede, por ejemplo, usar UML para señalar que un objeto es persistente o remoto.

El autor de la investigación considera que UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo, y que permite establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

El estudio realizado sobre las ventajas y desventajas que ofrece UML, ha permitido concluir que estas no influyen directamente en el desarrollo de la solución propuesta en la investigación.

2.4. Framework de Desarrollo.

Se puede definir como una estructura tecnológica de soporte, definido con artefactos o módulos de software concretos, con el objetivo de crear las bases para la mejor organización y desarrollo de los proyectos de software. Los frameworks de desarrollo en la informática han marcado un hito significativo, pues poseen significativas ventajas que simplifican el proceso de implementación y estandarización de los procesos en alguna medida. Refiere ser toda una tecnología o modelo de programación que contiene máquinas virtuales, compiladores, bibliotecas de administración de recursos en tiempo de ejecución y especificaciones de lenguajes, además de incluir una biblioteca de componentes reutilizables. Fueron diseñados con el fin de facilitar el desarrollo de software posibilitando a los proyectos identificar los requerimientos y prestar menos atención a los detalles de programación, quizás de bajo nivel, para proveer un sistema funcional.

Luego del estudio de las aplicaciones prácticas que desarrolla y las diferentes conceptualizaciones del término, se pueden precisar como las principales ventajas de la utilización de un framework:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes de software. Los framework son los paradigmas de la reutilización.
- El uso y la programación de componentes que siguen una política de diseño uniforme.

Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que genera como resultado bibliotecas más fáciles de aprender a usar.

2.4.1. Jena.

Es un framework de código abierto para desarrollar aplicaciones en Java con tecnologías de la Web Semántica. Incluye un motor de inferencia[9] basado en reglas, entornos para generar modelos RDF, RDFSchema, OWL y un intérprete y servidor de SPARQL, además de diversos servicios de almacenamiento como adaptadores a bases de datos relacionales.

Jena proporciona a los desarrolladores un API para el tratamiento de los grafos RDF. Estos grafos son representados internamente como un modelo abstracto que se puede consultar mediante el lenguaje de consultas para RDF, SPARQL. Además Jena permite el tratamiento del lenguaje OWL, al utilizar el modelo semántico de RDF, y permite que los razonadores DL, como Pellet o FaCT++ se puedan conectar de forma externa a través del interfaz DIG.

Independientemente de la capacidad de Jena para conectarse con motores de inferencia externos, Jena tiene su propio subsistema de inferencia que consiste en un motor híbrido con encadenamiento hacia-delante y hacia-atrás. Proporciona varios razonadores:

- Razonador transitivo.
- Razonador para RDF(S).
- Razonador para OWL.
- Motor de reglas multipropósito, que puede ser utilizado para el procesamiento de RDF, deducción de nuevo conocimiento y la transformación de grafos RDF.

Como características principales de **Jena** se destacan:

- API para trabajar con RDF programáticamente.
- API para trabajar con ontologías en distintos lenguajes:
  - ✓ RDFSchema
  - ✓ OWL

[9] Programa de control cuya función es seleccionar las reglas posibles a satisfacer el problema.

- Capacidad de procesamiento de consultas SPARQL.
- Motores de inferencia y conectores a motores externos.
- Mecanismos de almacenamiento para RDF.
- Servidor HTTP de RDF.

Jena permite gestionar todo tipo de ontologías (añadir hechos, borrarlos y editarlos), almacenarlas y realizar consultas contra ellas. Soporta RDF, DAML y OWL y es independiente del lenguaje. Los recursos no están ligados estáticamente a una clase java particular.

2.5. Lenguaje de Programación.

Un lenguaje de programación es una técnica de comunicación estandarizada para expresar las instrucciones a una computadora. Se trata de un conjunto de reglas sintácticas y semánticas utilizadas para definir los programas de ordenador. (17)

Los lenguajes de programación se encargan de relacionar a los seres humanos con los ordenadores, y así manipularlos para automatizar tareas y llevar a cabo instrucciones donde se controla su comportamiento y se realizan diferentes operaciones. Estos lenguajes le permiten al programador especificar con precisión los datos que una computadora tomará sobre una decisión determinada, dichos datos pueden ser almacenados o transmitidos.

2.5.1. Java.

Java fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. El lenguaje fue diseñado con las siguientes características en mente:

- **Simple:** elimina la complejidad de los lenguajes como "C#" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional.

- **Familiar:** como la mayoría de los programadores están acostumbrados a programar en C# o en C++, la sintaxis de Java es muy similar al de estos.
- **Robusto:** el sistema de Java maneja la memoria de la computadora por ti. No te tienes que preocupar por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que uno se lo indique.
- **Seguro:** el sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- **Portable:** como el código compilado de Java es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- **Independiente a la arquitectura:** al compilar un programa en Java, el código resultante un tipo de código binario conocido como byte code. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- **Multithreaded:** un lenguaje que soporta múltiples threads[10], es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.
- **Interpretado:** Java corre en máquina virtual, por lo tanto es interpretado.
- **Dinámico:** Java no requiere que se compilen todas las clases de un programa para que este funcione. Si se realiza una modificación a una clase, Java se encarga de realizar un Dynamic Bynding[11] o un Dynamic Loading[12] para encontrar las clases.

Java puede funcionar como una aplicación sola o como un applet[13], que es un pequeño programa hecho en Java. Los applets de Java se pueden "pegar" a una página de Web (HTML), y con esto puedes tener un programa que cualquier persona que tenga un browser compatible podrá usar.

[10] Hilos de ejecución.
[11] Mecanismo por el cual se escoge, en tiempo de ejecución, el método que responderá a un determinado mensaje.
[12] Mecanismo por el cual un programa puede, en tiempo de ejecución, cargar una biblioteca en la memoria.

Java proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Proporciona además una recopilación de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variados, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software.

2.6. Entorno de Desarrollo Integrado (IDE).

Un Entorno de Desarrollo Integrado o en inglés, Integrated Development Environment (IDE), es un ambiente para escribir la lógica de aplicación y el diseño de interfaces de aplicaciones. Se puede caracterizar como el programa informático compuesto de un conjunto de herramientas que utilizan los programadores para generar código. Estos pueden ser multiplataforma, además soportan diversos lenguajes de programación, tienen un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de forma amigable.

2.6.1. NetBeans.

Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. NetBeans IDE[14] dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco sus funcionalidades son ampliables mediante la instalación de packs.

Entre las características que presenta NetBeans se encuentran:

- Desarrollo de aplicaciones multiplataforma sobre: MacOS, Windows, Linux.
- Add-ons para desarrollo Móvil, desarrollo Web gráfico, integración con SOA, optimización de aplicaciones y desarrollo con C y C++.

[13] Componente de una aplicación que se ejecuta en el contexto de otro programa.
[14] Entorno de desarrollo integrado.

- Cliente CVS integrado.
- Crecimiento de plataforma por medio de plug-ins. Entre los plug-ins que existen se tienen los siguientes:
  - ✓ Herramientas de java que sirven para la mejora de desarrollo de aplicaciones.
  - ✓ Herramientas de modelado UML.
  - ✓ Herramientas XML.

NetBeans fue elegido como entorno de desarrollo ya que simplifica la tarea de programación enormemente, permitiendo centrarse en las particularidades de la aplicación a desarrollar y evitando la complejidad inherente a cualquier desarrollo sobre una plataforma gráfica. Presenta gran área de edición donde el desarrollador interactúa con el código fuente, panel que permite la gestión de los archivos del proyecto y panel de visualización de los mensajes de error resultado de una compilación.

Conclusiones parciales.

El desarrollo de este capítulo posibilitó realizar un análisis y caracterización de algunas de las tecnologías actuales que serán utilizadas para el desarrollo de la propuesta, determinando lo siguiente:

Como metodología estándar para el análisis, implementación y documentación del sistema se escogió RUP y UML se eligió como lenguaje de modelado. Como herramienta UML que soporta el ciclo de vida completo del desarrollo del software se eligió Visual Paradigm 6.4. Se determinó el lenguaje de programación Java para desarrollar el servidor de servicios que permitirá interpretar el lenguaje ontológico OWL, utilizando las potencialidades que ofrece el framework Jena 2.6.4.

Capítulo #3 Propuesta de la solución

El desarrollo de un software puede considerarse una acción fácil que no requiere un análisis profundo para realizar la programación de un sistema, pero esto es mucho más complicado a la vista de un especialista, pues para lograr los objetivos propuestos se necesita de un amplio razonamiento del problema y sus posibles soluciones. El objetivo del presente capítulo es llegar a un mayor entendimiento del ámbito en el que se desarrollará la aplicación, el cual quedará claro mediante el modelo de dominio, artefacto generado a partir de que no existe definido un negocio, se especificarán detalladamente los requerimientos funcionales y no-funcionales en materia de software, con la finalidad de que se usen como base para la construcción del Modelo de Casos de Uso del Sistema y el posterior planteamiento de una propuesta para el desarrollo de un sistema de software.

El presente capítulo está orientado también hacia el desarrollo del sistema que se propone. Para ello se construirá el modelo de diseño, que constituye un plano muy cercano a la implementación, y contribuirá a una arquitectura sólida. Como parte del modelo del diseño se hará una propuesta de los diagramas de clases del diseño, estos diagramas de clases serán la base para los diagramas de componentes y el diagrama de despliegue, que también serán presentados. Una vez modelado el sistema se realizará un proceso de pruebas con el objetivo de verificar que la herramienta cumpla con los requerimientos planteados.

3.1 Modelo de Dominio.

Es el artefacto que está representado por uno o más diagramas de clases que capturan los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los elementos que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio. El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema.

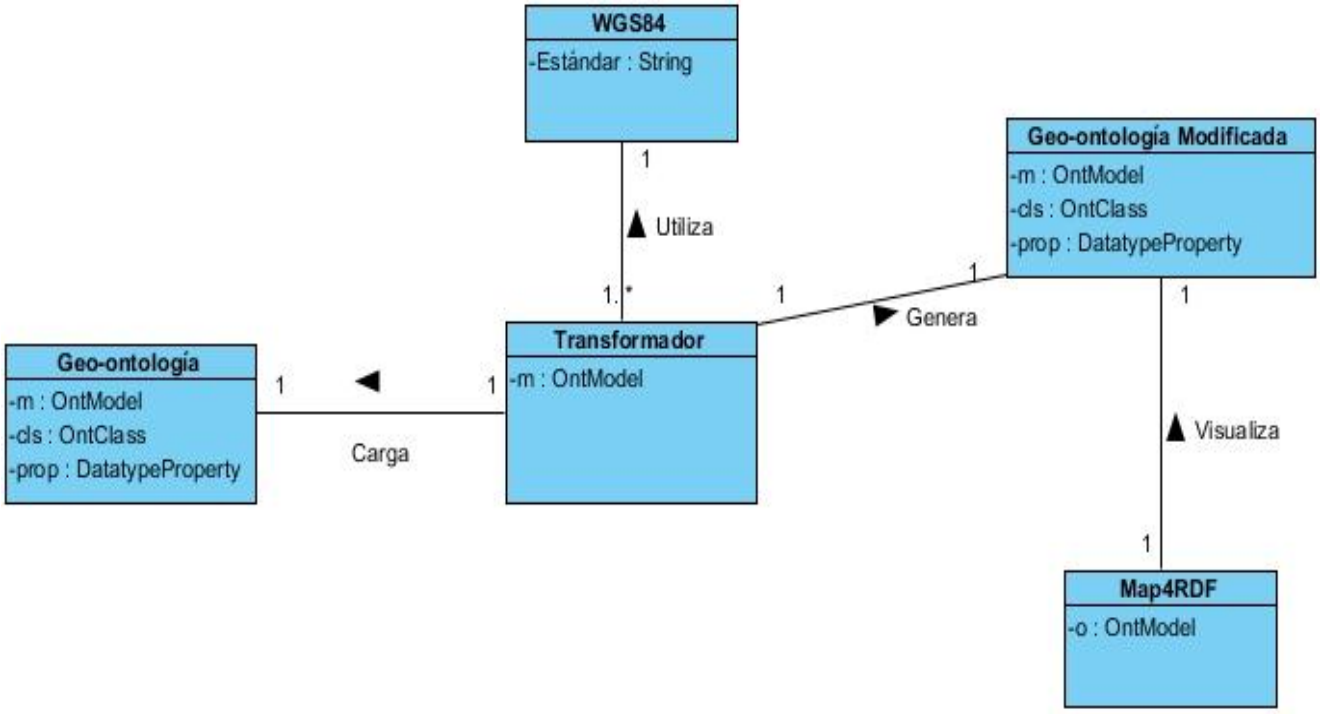

Figura 2: Modelo de Dominio

3.2 Descripción del Modelo de Dominio.

El sistema consiste en una herramienta (**Transformador**) que se encargará de cargar una **Geo-ontología** que no está en el estándar de la **WGS84** y haciendo uso de dicho estándar la transformará de forma semi-automática generando una **Geo-ontología Modificada** que podrá ser visualizada a través de la herramienta **Map4RDF**.

3.3 Requisitos del Sistema.

No son más que el conjunto de tareas que tienen que ver con la determinación de las necesidades y la satisfacción de las condiciones de un producto, teniendo en cuenta los requisitos impuestos por el cliente. Define de forma precisa el producto de software que se va a construir.

3.3.1 Requisitos Funcionales (RF).

Los requisitos funcionales son las capacidades o condiciones que el sistema debe cumplir.

- **RF1.** Cargar geo-ontología.
  - ✓ Crear un modelo de geo-ontología.
- **RF2.** Mapear geo-ontología al estándar WGS84.
  - ✓ Mostrar el contenido de las clases de una geo-ontología.
  - ✓ Modificar el contenido de una geo-ontología.
    - Agregar clases.
    - Agregar relaciones.
    - Almacenar el modelo en un fichero.
- **RF3.** Visualizar geo-ontología.
- **RF4.** Brindar servicio para mapeo de geo-ontologías.

3.3.2 Requisitos No Funcionales (RNF).

No son más que las cualidades o propiedades que el producto debe tener. Son un límite en el sistema o en el proceso de desarrollo. Según Dorfman y Thayer un requisito no funcional es un requisito de software que describe no lo que el software hará, sino como lo hará, como por ejemplo, requisitos de rendimiento. (18) Los requisitos no funcionales son difíciles de verificar/testear, y por ello son evaluados subjetivamente.

Requisitos de usabilidad: estos requerimientos describen los niveles convenientes de usabilidad, dados los usuarios finales del producto y para ello debe estudiarse las especificaciones de los perfiles de usuarios y las clasificaciones de sus niveles de experiencia. Depende de la definición de los usuarios, cuantificable por los criterios de evaluación.

- **RNF1.** El sistema deberá ser utilizado por especialistas que posean al menos conocimientos básicos del manejo de ontologías.

Requisitos de interfaz externa: estos requerimientos son los que describen la apariencia del producto.

- **RNF2.** Debe contar con una interfaz amigable, navegable y sencilla.

Restricciones en el diseño e implementación: especifica o restringe la codificación o construcción de un sistema. Son restricciones que han sido ordenadas y deben ser cumplidas estrictamente.

- **RNF3.** Para el desarrollo del análisis y el diseño del sistema deberá ser utilizada la metodología RUP, usando el lenguaje de modelado UML y como herramienta CASE para llevarlo a cabo el Visual Paradigm en su versión 6.4.

- **RNF4.** Para la implementación se utilizará IDE NetBeans y la librería Jena.

Requisitos de operatividad y portabilidad: requerimientos que definen las particularidades que debe ostentar el software para facilitar su traslado a otras plataformas o entornos de desarrollo.

- **RNF5.** La aplicación debe ser compatible con los Sistemas Operativos: Windows 2000 NT, Windows XP, Windows 7 y GNU/Linux.

Requerimientos de Software: se especifican las condiciones o capacidades que el sistema debe cumplir.

- **RNF6.** Las estaciones de trabajo que harán uso del sistema deberán tener instalado:
  - ✓ Windows 2000 NT, Windows XP Profesional, Windows 7 ó GNU/Linux en cualquier distribución.

Requerimientos de Hardware: son los que especifican las características lógicas para cada interfaz entre el producto y los componentes de hardware del sistema, incluyendo la estructura lógica, direcciones físicas, el comportamiento esperado, entre otros aspectos.

- **RNF7.** Las computadoras clientes que utilizarán el software a desarrollar deberán tener como mínimo 128 MB de memoria de tipo RAM, al menos 10 GB de disco duro y un procesador 1.90 MHz como mínimo.

- **RNF8.** El servidor de servicios debe tener como mínimo 2GB de RAM, 20GB de disco duro y un procesador 3 GHz como mínimo para cada uno.

Requerimientos de Fiabilidad: definen la probabilidad de que el software funcione adecuadamente durante un período determinado bajo condiciones operativas específicas.

- **RNF9.** El tiempo medio de reparación, en caso de un fallo es de 7 días.
- **RNF10.** La información y las funcionalidades del sistema estarán disponibles y el usuario podrá acceder a ellas las 24 horas de los 7 días de la semana.

Requerimientos de Eficiencia: ofrecen tiempos de respuesta aceptables para el usuario.

- **RNF11.** El tiempo de respuesta estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento.
- **RNF12.** Al igual que el tiempo de respuesta, la velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar la aplicación.

3.4 Actores del Sistema.

Actor es toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, además de entidades abstractas. En el caso de los seres humanos se pueden ver a los actores como definiciones de rol, por lo que un mismo individuo puede corresponder a uno o más Actores. En la siguiente tabla se enuncia la descripción delos actores de este sistema:

Actor	Descripción
Usuario	Ente que interactúa con el software, es el encargado de realizar todas las acciones para posteriormente adquirir el conocimiento.
Aplicación Externa	Aplicación que necesite utilizar el software para su funcionamiento.

Tabla # 1 : Descripción de los actores del sistema.

3.5 Diagrama de Casos de Uso del Sistema (DCUS).

El modelo de casos de uso permite que los desarrolladores del software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas.

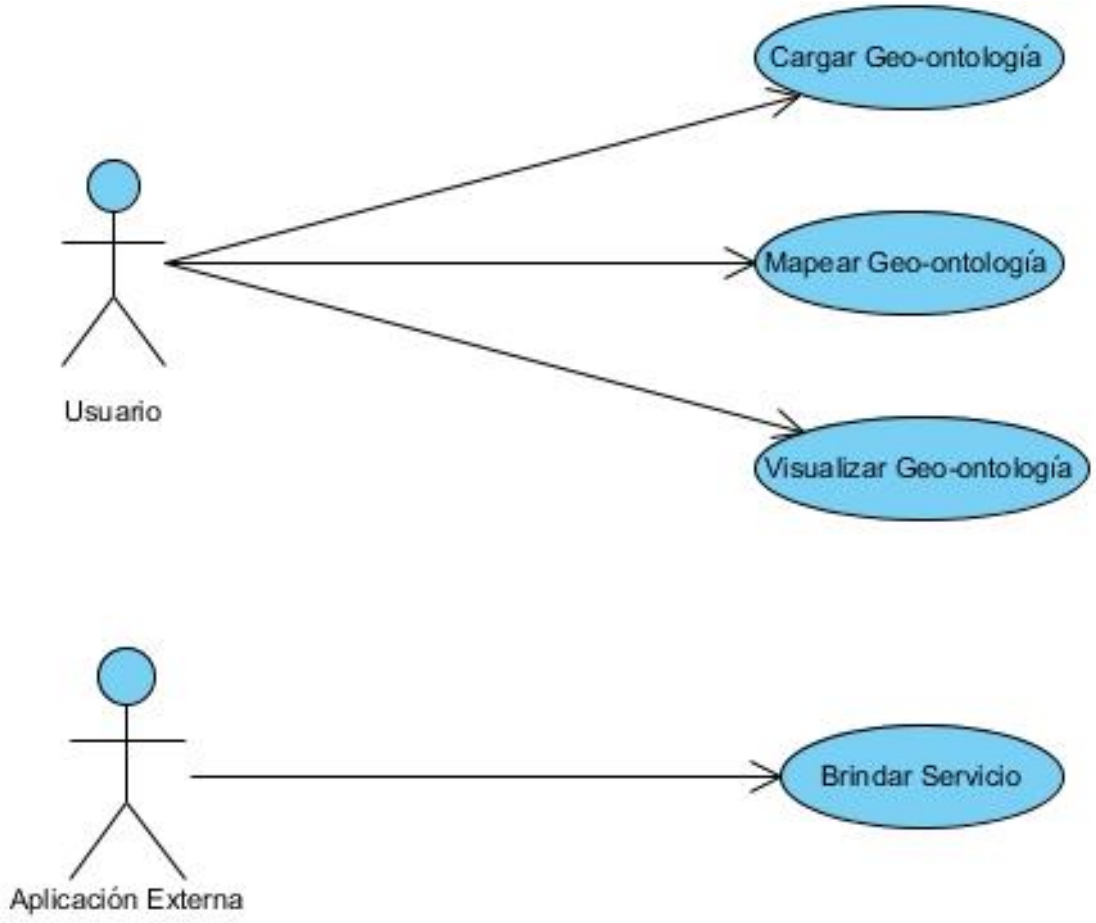

Figura 3 Diagrama de Casos de Uso del Sistema.

3.6 Descripción de Casos de Uso del Sistema.

Caso de Uso "Cargar Geo-ontología".	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario selecciona del menú principal la opción "Cargar geo-ontología" y aparece la interfaz común de usuario "Cargar archivo", donde el actor realiza una búsqueda del fichero, al cargarse el mismo quedarán representadas jerárquicamente en un jTree las propiedades de la geo-ontología.
Precondición	Verificar que exista al menos una ontología.
Referencias	Requisito Funcional #1: Cargar Geo-ontología.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario elige la opción "Cargar geo-ontología".	1.1. El sistema muestra la interfaz común de usuario "Cargar archivo".

2. El usuario realiza una búsqueda y carga el fichero.	2.1. El sistema verifica que los campos se hayan llenado con la información correcta. 2.2. El sistema verifica la completitud del fichero. 2.3. El sistema visualiza de forma jerárquica las clases o propiedades de la geo-ontología cargada.
Flujo Alterno de Eventos	
	2.1.1. Si el sistema detecta que el fichero es corrupto muestra un mensaje de Error.
Prototipo de interfaz	

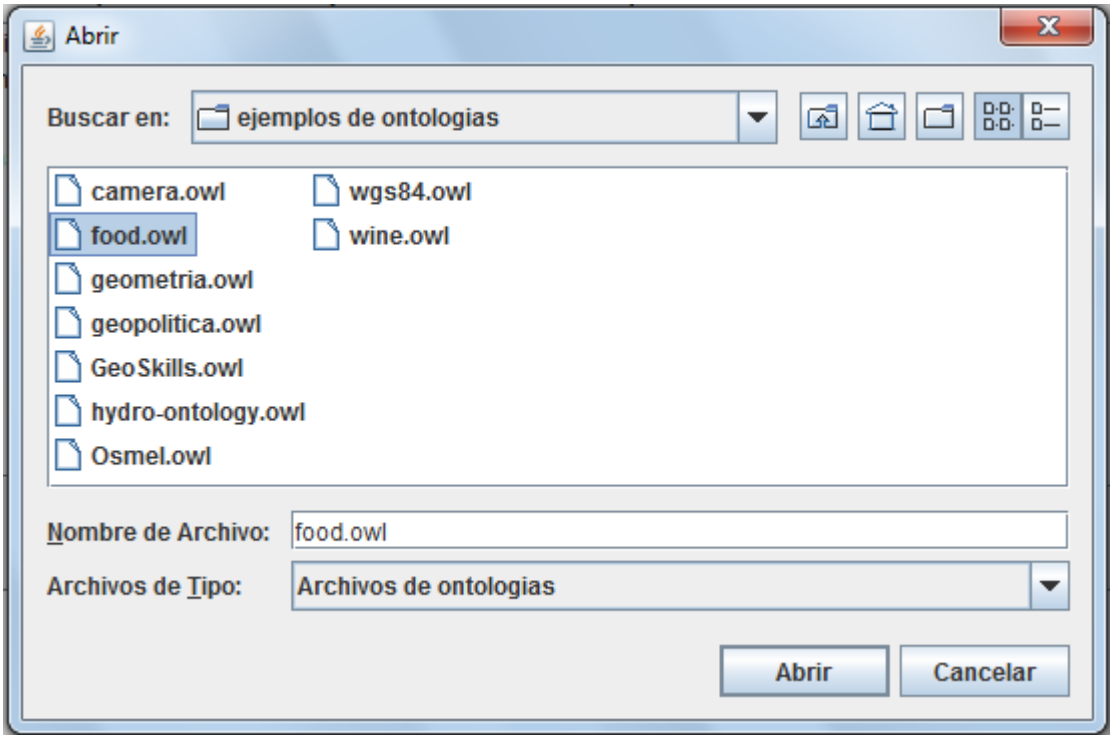

Tabla # 2: Descripción textual del caso de uso "Cargar Geo-ontología".

Caso de Uso "Mapear Geo-ontología".	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario agrega a un listado las clases de la geo-ontología que desea mapear.
Precondición	Verificar que exista una ontología cargada.
Referencias	Requisito Funcional #2: Mapear Geo-ontología.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona las clases.	1.1. El sistema agrega las clases a un listado.
2. El usuario selecciona la opción “Equivalentes”.	2.1. El sistema modifica el contenido de las clases de la geo-ontología llevándolo al estándar de la WGS84. 2.2. El sistema muestra un mensaje de confirmación.
Flujo Alterno de Eventos	
	2.1.1. Si el sistema detecta un error muestra un mensaje.
Prototipo de interfaz	

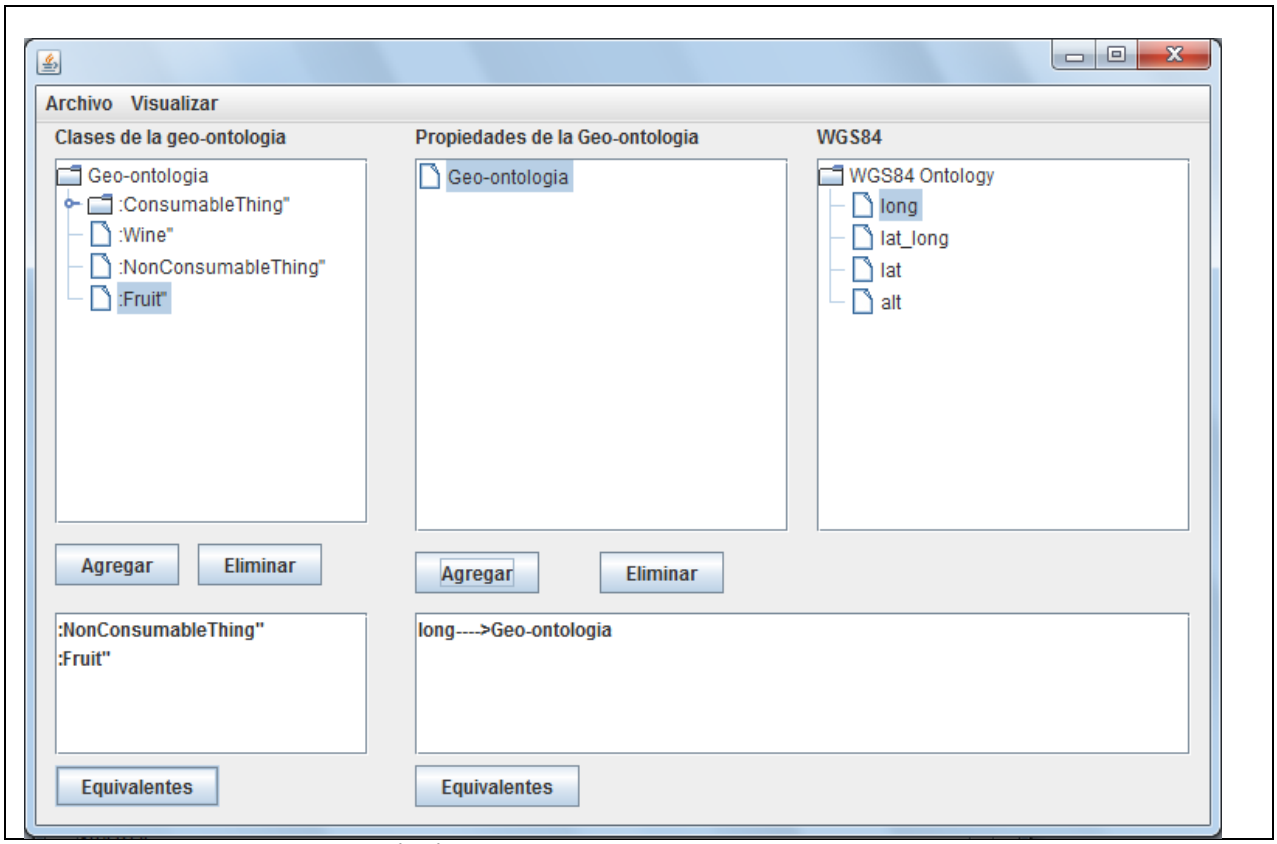

Tabla # 3: Descripción textual del caso de uso "Mapear Geo-ontología".

Caso de Uso "Visualizar Geo-ontología".	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario elige del menú principal la opción "Geo-ontología Modificada"
Precondición	Verificar que exista una ontología cargada.
Referencias	Requisito Funcional #3: Visualizar Geo-ontología.

Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario elige la opción “Geo-ontología Modificada”.	1.1. El sistema muestra en una ventana las clases y propiedades (de forma jerárquica) de la geo-ontología modificada.
Flujo Alterno de Eventos	
	1.1.1. Si el sistema detecta un error muestra un mensaje.
Prototipo de Interfaz	

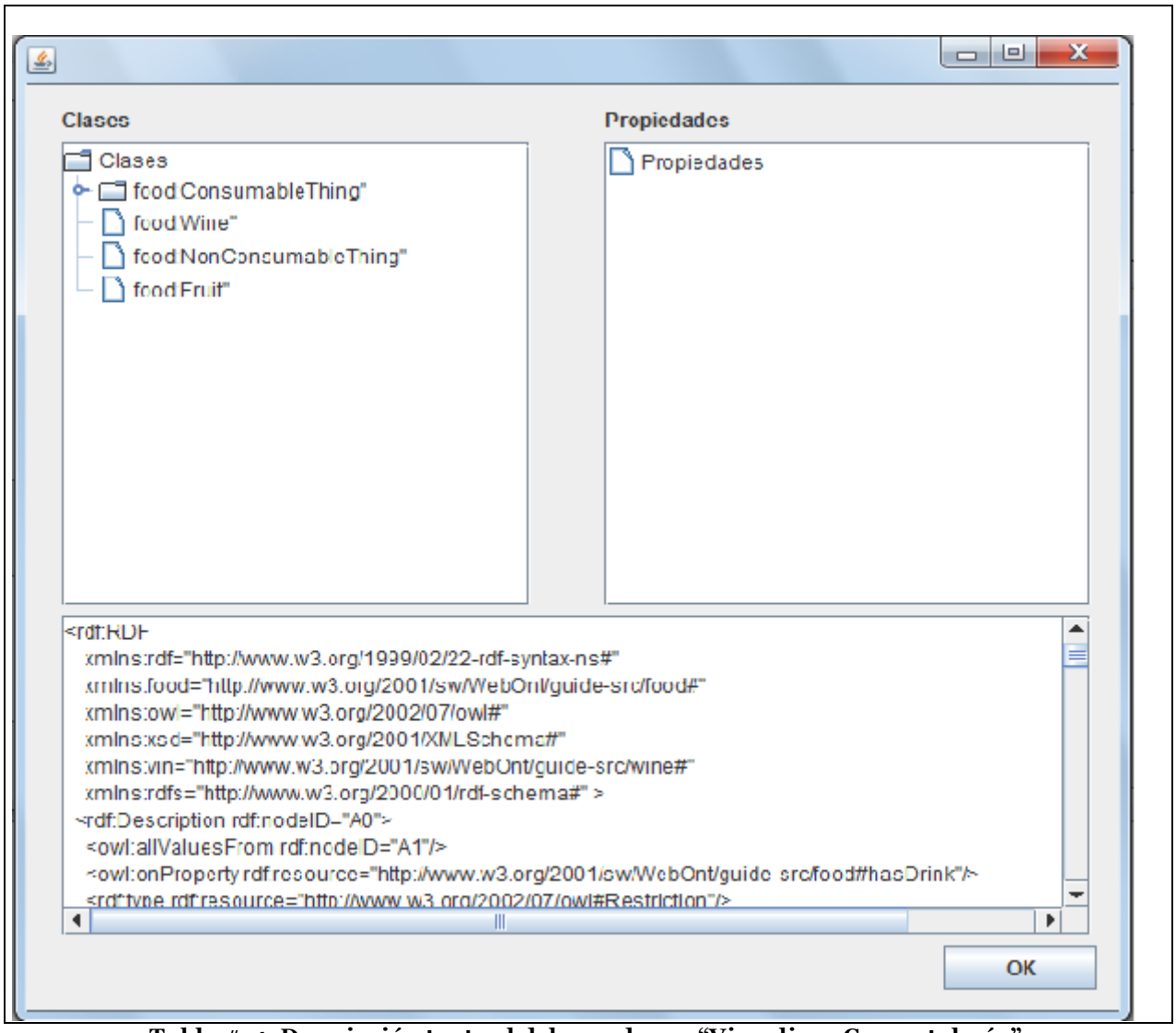

Tabla # 4: Descripción textual del caso de uso "Visualizar Geo-ontología".

Caso de Uso "Brindar Servicio".	
Actor	Aplicación Externa

Resumen	Este caso de uso se inicia cuando una aplicación externa consuma los servicios brindados por la herramienta.
Precondición	Verificar que la aplicación externa se pueda integrar a la herramienta.
Referencias	Requisito Funcional #4: Brindar Servicio.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. La aplicación externa consume los servicios brindados.	1.1. El sistema brinda los servicios necesarios a la aplicación externa.
Flujo Alterno de Eventos	

Tabla # 5: Descripción textual del caso de uso "Brindar Servicio".

3.7 Arquitectura de la solución.

Una Arquitectura de Software, también denominada Arquitectura Lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software de un sistema de información.

Uno de los estilos de arquitectura de software más conocidos es la arquitectura en capas, la cual tiene como objetivo principal separar la lógica del negocio de la lógica de diseño, la implementación de la herramienta para mapeo semi-automático de geo-ontologías al estándar de la WGS84 se basará en esta

arquitectura pero con la particularidad de que esta herramienta no presenta una base de datos para el almacenamiento de información, por tanto la arquitectura quedará distribuida en 3 capas fundamentales:

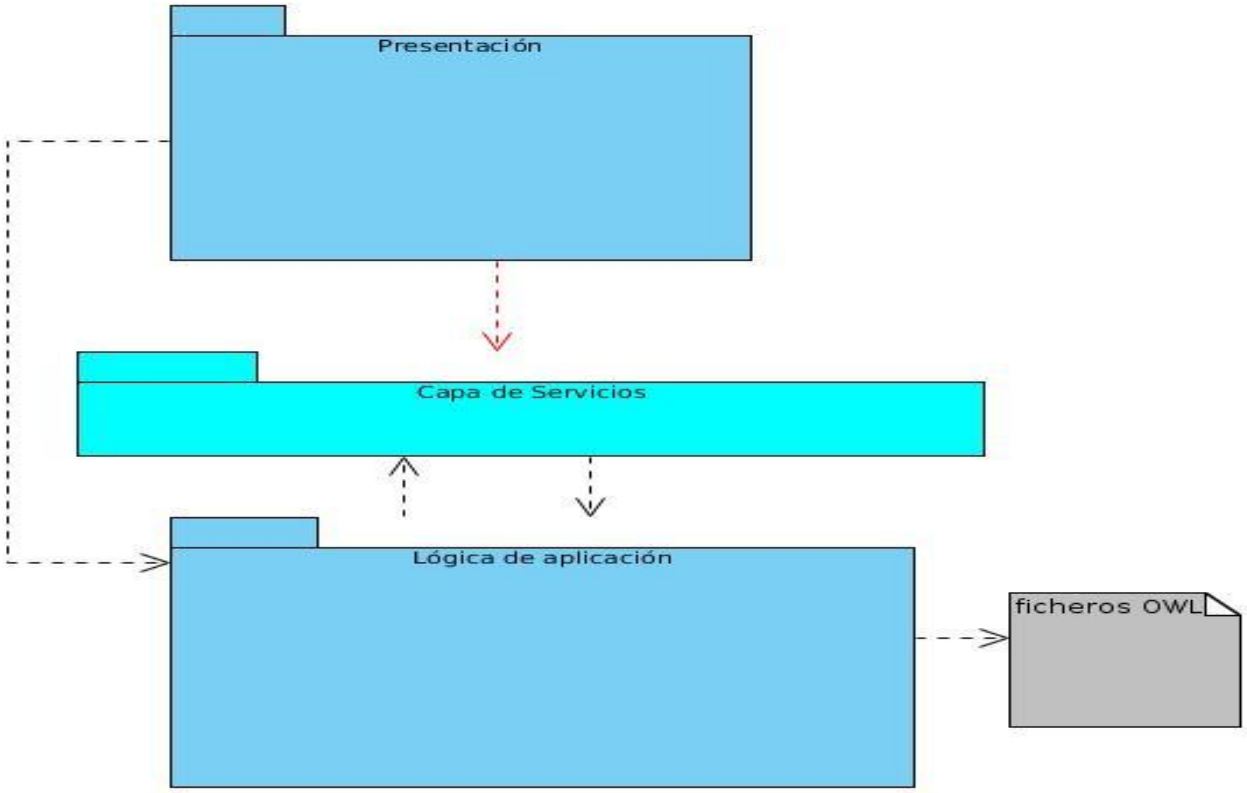

Figura 4 Arquitectura.

- **Presentación:** es la capa que ve el usuario, es decir, toda interfaz que presente el software, la cual comunica y captura la información del usuario en un mínimo de procesos. También es conocida como interfaz gráfica y debe tener la característica de ser entendible y fácil de usar. Esta capa se comunica únicamente con la capa de negocio.

- **Lógica del Negocio:** es donde radican las principales funcionalidades que se ejecutan del software, se reciben las peticiones del usuario y se envían las respuestas. Se denomina también capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados.

- **Servicios:** Es la capa de cooperación que presenta el sistema, el cual interactúa con otras aplicaciones que requieren los servicios brindados que no son más que una forma estándar de publicar y utilizar dichos servicios, conocidos comúnmente como servicios web (web services).

3.8 Modelo de diseño.

Comprende el refinamiento y formalización adicional del modelo de análisis tomando en cuenta los detalles de implementación. El resultado del modelo de diseño consiste en especificaciones mucho más detalladas en cuanto a que se incluyen operaciones y atributos de los objetos. Se requiere un modelo de diseño puesto que el modelo de análisis no es lo suficientemente formal para alcanzar el código fuente.

3.8.1 Patrones de diseño.

Los Patrones de diseño, en su forma más sencilla, son buenas soluciones que pueden ser aplicadas a problemas recurrentes que surgen durante el diseño de un sistema o aplicación. Estos no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables. A continuación se describen los patrones utilizados en el desarrollo de la aplicación:

Patrones para Asignar Responsabilidades (GRASP): describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (19).

- **Experto:** se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad.
- **Creador:** este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.
- **Alta Cohesión:** El objetivo de este patrón es asignar responsabilidades de tal forma que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.
- **Bajo Acoplamiento:** especifica cómo dar soporte a una dependencia escasa y a un aumento de la reutilización, enfocándose en asignar una responsabilidad para mantener bajo acoplamiento. El bajo

acoplamiento estimula la asignación de responsabilidades de forma tal que la inclusión de éstas no incremente el acoplamiento, creando clases más independientes y con mayor resistencia al impacto de los campos, que aumentan la productividad y la posibilidad de reutilización.

Patrones GoF (*Gang of Four*):

- **Singleton:** Un programa que instancia múltiples copias, probablemente tenga un error, pero la utilización del patrón singleton hace que tales errores sean inofensivos puesto que dicho patrón garantiza el acceso único a una clase mediante una única instancia. De esta forma se controla el acceso a las clases.
- **Fachada:** Permitirá simplificar la interfaz del sistema que se propone.

3.9 Diagrama de clases del diseño.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Usualmente contiene la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de atributos.
- Navegabilidad.
- Dependencia.

Un diagrama de este tipo contiene las definiciones de las entidades del software en lugar de conceptos del mundo real. El UML no define concretamente un elemento denominado “diagrama de clases del diseño”, sino que se sirve de un término más genérico: “diagrama de clases”. Craig Larman en su libro UML y Patrones, Introducción al Análisis y Diseño Orientado a Objetos, opta por incluir el término “diseño” en “diagrama de clases” para acentuar que se trata de una perspectiva desde el punto de vista del diseño de las entidades de software y no de una concepción analítica sobre los conceptos del dominio. (20)

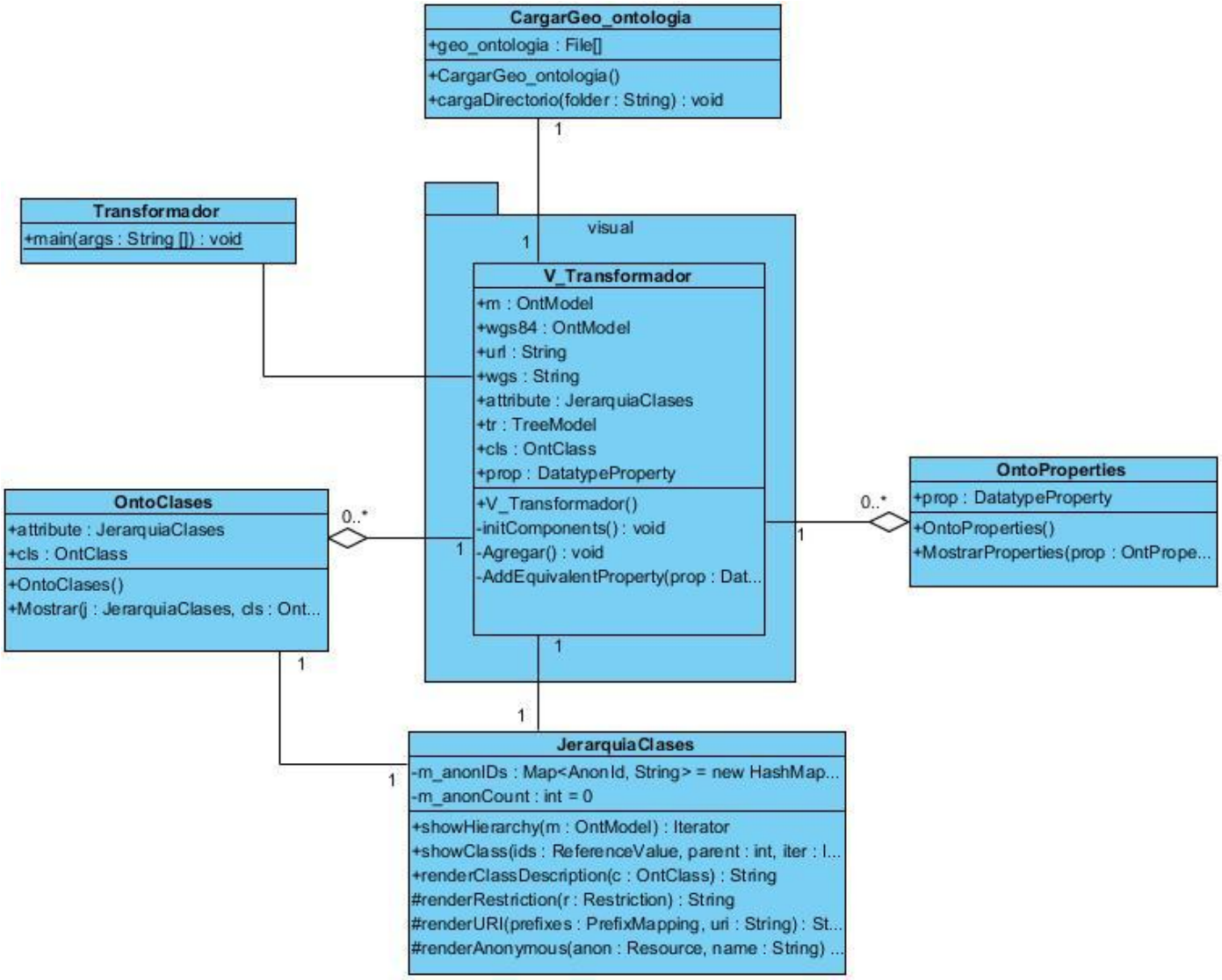

Figura 5: Diagrama de Clases del Diseño.

3.10 Diagramas de secuencia del sistema

El diagrama de la secuencia de un sistema es una representación que muestra, en determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del

sistema. A todos los sistemas se les trata como una caja negra; los diagramas se centran en los eventos que trascienden las fronteras del sistema y que fluyen de los actores a los sistemas. (20)

RUP plantea la realización de un diagrama de secuencia por cada caso de uso, los cuales se presentan a continuación.

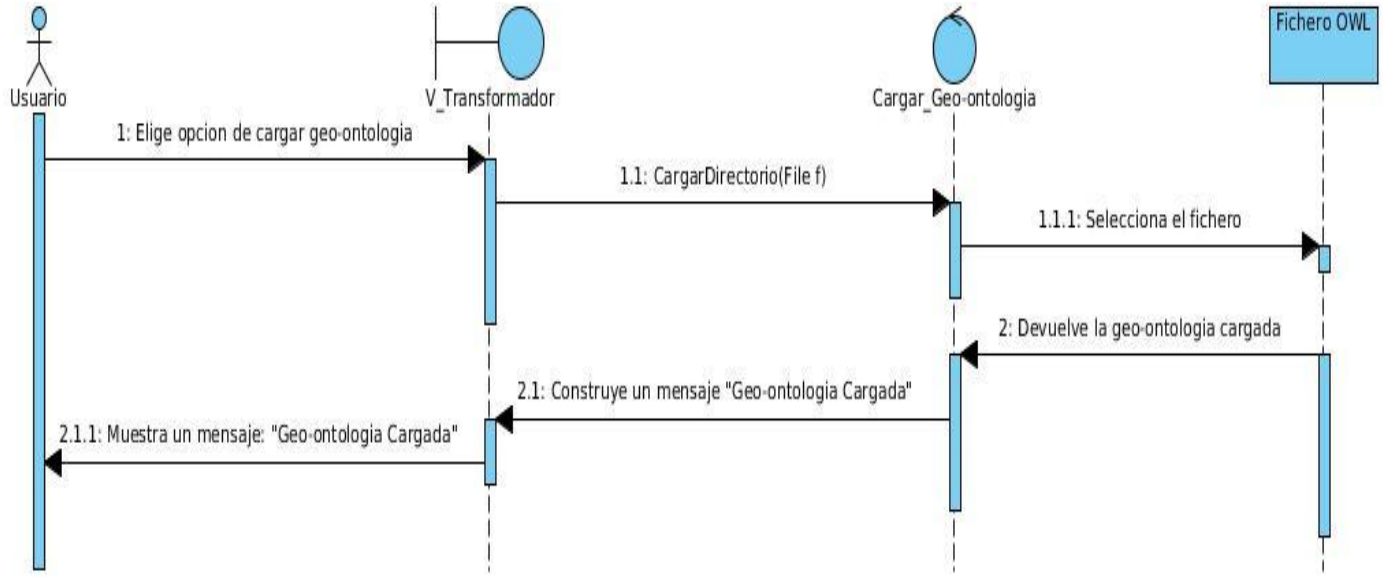

Figura 6: Diagrama de Secuencia del CU "Cargar Geo-ontología".

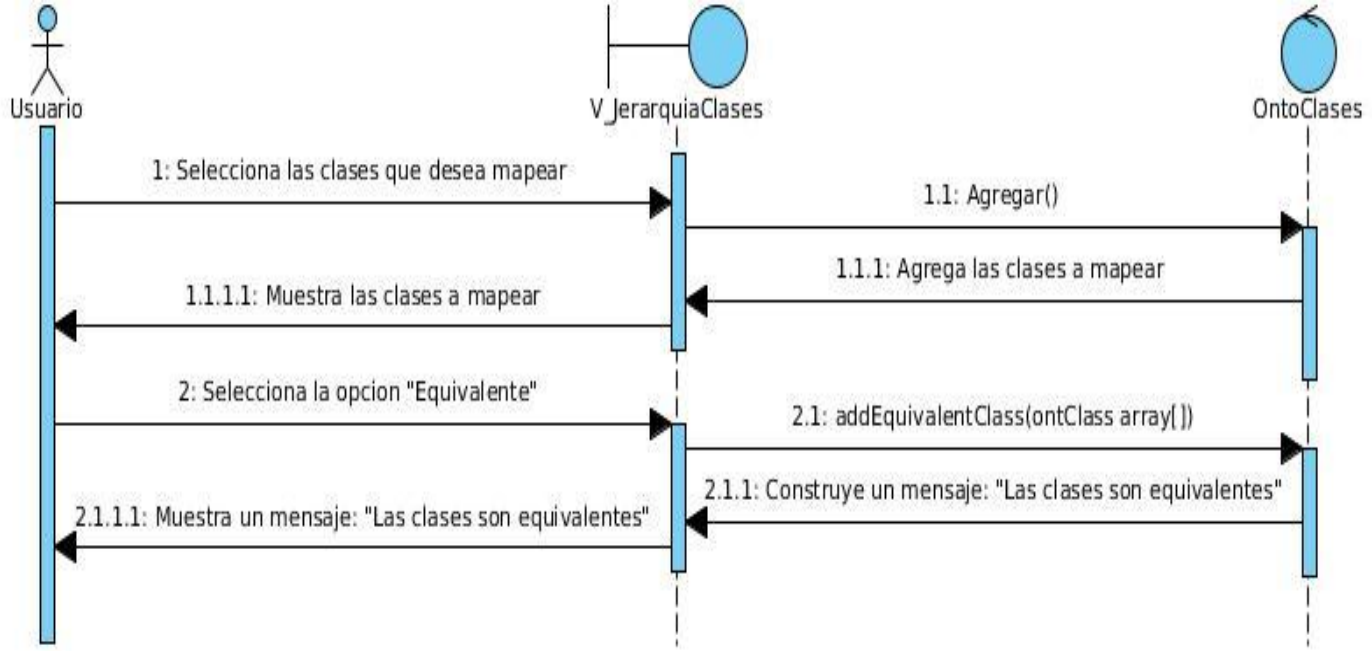

Figura 7: Diagrama de Secuencia del CU "Mapear Geo-ontología".

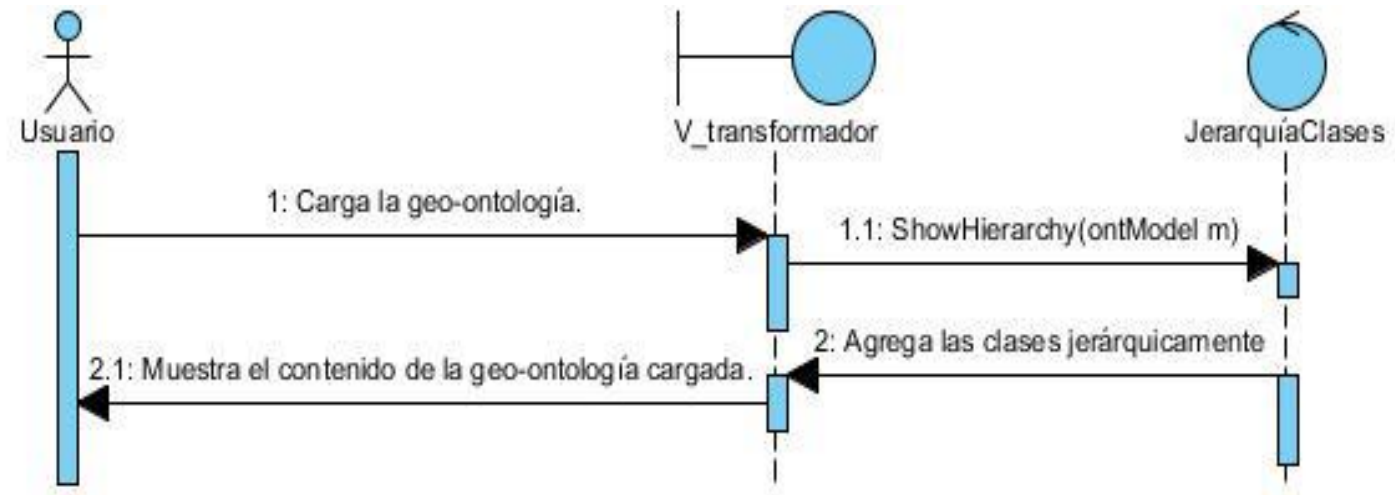

Figura 8: Diagrama de Secuencia del CU "Visualizar Geo-ontología".

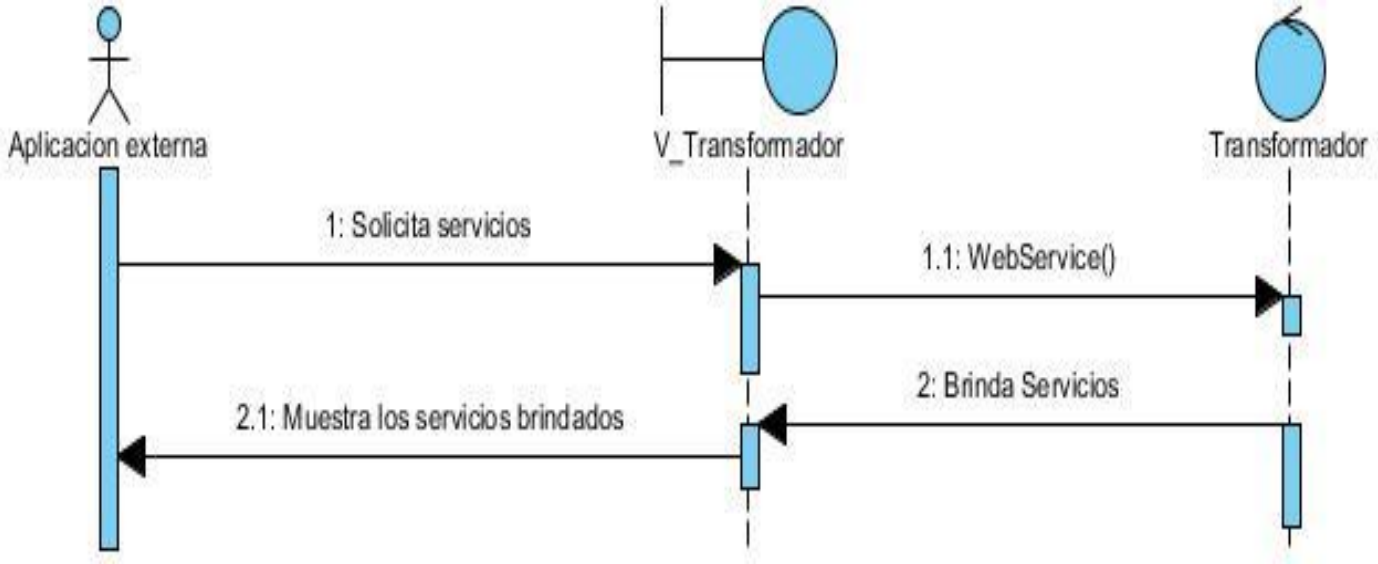

Figura 9: Diagrama de Secuencia del CU "Brindar Servicio".

3.11 Modelo de despliegue

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o más nodos, dispositivos, y conectores, entre estos. El modelo de despliegue también mapea procesos dentro de estos elementos de procesamiento, permitiendo la distribución del comportamiento a través de los nodos que son representados. A continuación se muestra el diagrama de despliegue modelado para la aplicación a desarrollar:

Figura 10: Diagrama de Despliegue.

3.12 Modelo de implementación

Describe como los elementos del modelo de diseño (clases) se implementan en términos de componentes, como ficheros de código fuente, ejecutables, scripts y ficheros de código binario. Un modelo de implementación define cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación, en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

Los diagramas de componentes permiten modelar los aspectos físicos de un sistema, la vista de implementación estática de un sistema y los elementos físicos que residen en un nodo, tales como: ejecutables, tablas, librerías, archivos y documentos. Un diagrama de componentes muestra un conjunto de componentes y sus relaciones.

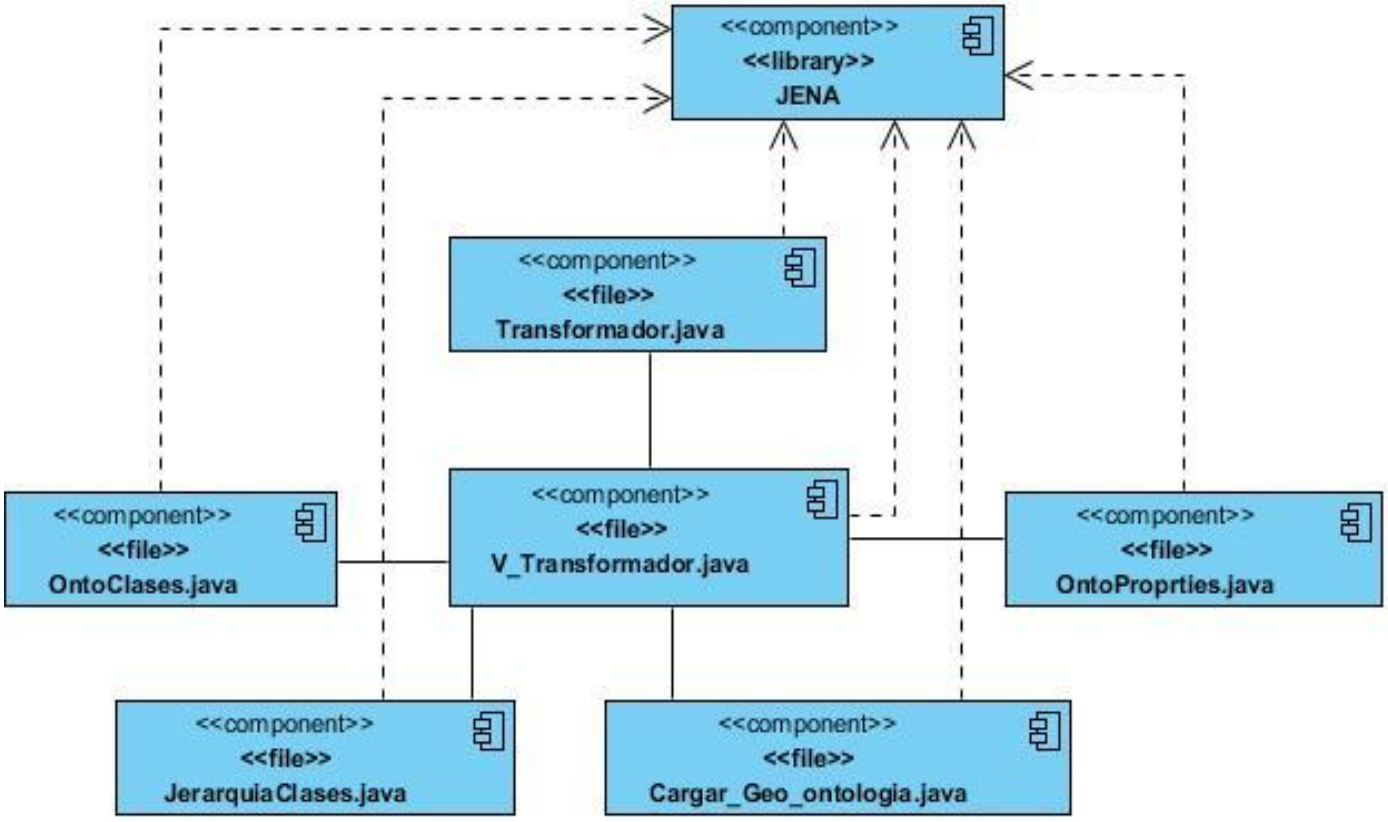

Figura 11: Diagrama de Componentes.

3.13 Pruebas de software

Un sistema de pruebas implica la operación o aplicación del mismo a través de condiciones controladas y la consiguiente evaluación de la información. Las condiciones controladas deben incluir tanto situaciones normales como anormales. El objetivo del sistema de pruebas es encontrar un error para determinar situaciones en donde algo pasa cuando no debe de pasar y viceversa. En una palabra, un sistema de pruebas está orientado a detectar.

Para la planeación de prueba que se va a aplicar al sistema evaluador, se determinó:

Pruebas de Caja Negra: el sistema de pruebas de caja negra no considera la codificación dentro de sus parámetros a evaluar, es decir, que no están basadas en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requerimientos establecidos, en la funcionalidad del sistema y su forma de interactuar con el medio que le rodea entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. Por tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

Para la realización de dichas pruebas llevadas a cabo sobre la interfaz del software, se escogen los casos de uso: “Cargar Geo-ontología” y “Mapear Geo-ontología” proporcionando unas entradas y estudiando las salidas para ver si son o no las esperadas:

3.13.1 Caso de prueba # 1

1. **Descripción General.**

El caso de uso se inicia cuando el usuario selecciona del menú **Archivo** que aparece en la barra de herramientas la opción **Cargar Geo-ontología**, al mismo tiempo aparece una ventana de cargar archivos en la cual el usuario podrá seleccionar el fichero que desee cargar y oprimir el botón **Abrir** para llevar a cabo esta acción, el caso de uso termina cuando el sistema muestra un mensaje de confirmación y aparecen representadas jerárquicamente las clases de la geo-ontología.

2. **Condiciones de Ejecución.**

- El archivo debe tener la extensión *.owl.
- El archivo no puede estar corrupto.

3. Secciones a probar en el Caso de Uso "Cargar Geo-ontología".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Cargar Geo-ontología.	EC 1.1: Cargar fichero.	El sistema muestra la interfaz común de usuario "Cargar archivo".
	EC 1.2: Abrir fichero.	El sistema muestra un mensaje de confirmación y representa jerárquicamente las clases.
	EC 1.3: Abrir fichero erróneo.	El sistema muestra mensaje de error.

4. Descripción de variable.

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Geo-ontología	Campo de subida de archivos	Sí	Se tiene que subir un archivo.

5. Matriz de Datos

SC 1 Cargar Geo-ontología

Escenario	Variable 1	Respuesta del Sistema	Resultado de la	Flujo Central

	Geo-ontología		Prueba	
EC 1.1: Cargar fichero.	NA	El sistema muestra la interfaz común de usuario "Cargar archivo".	Satisfactorio.	• Seleccionar la opción "Cargar Geo-ontología".
EC 1.2: Abrir fichero.	V (Food.owl)	El sistema muestra mensaje de confirmación.	Satisfactorio.	• Seleccionar la opción "Cargar Geo-ontología". • Seleccionar **Abrir** archivo. • Mostrar mensaje de confirmación y representa jerárquicamente las clases.
EC 1.3: Salvar fichero erróneo.	V (Food.txt)	El sistema muestra mensaje de error.	Satisfactorio.	• Seleccionar la opción "Cargar Geo-ontología". • Seleccionar **Abrir** archivo. • Mostrar mensaje de error.

3.13.2 Caso de prueba # 2

1. **Descripción General.**

El caso de uso se inicia cuando el usuario selecciona las clases de la geo-ontología que desea mapear y las va agregando a un listado a través del botón **Agregar**, luego presiona el botón **Equivalentes** modificando el contenido de la geo-ontología cargada, el caso de uso termina cuando el sistema muestra un mensaje de confirmación.

2. **Condiciones de Ejecución.**

- Debe existir una geo-ontología cargada.
- El listado debe tener al menos una clase seleccionada.

3. **Secciones a probar en el Caso de Uso "Mapear Geo-ontología".**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Mapear geo-ontología.	EC 1.1: Agregar las clases a mapear de la geo-ontología cargada.	El sistema muestra un listado con las clases seleccionadas.
	EC 1.2: Hacer equivalentes.	El sistema muestra un mensaje de confirmación.

4. **Descripción de variable.**

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Agregar	Listado de textos	Sí	Se debe seleccionar al

				menos una clase.
2	Equivalentes	Botón	Sí	Debe existir en el listado al menos una clase.

5. Matriz de Datos

SC 1 Mapear Geo-ontología

Escenario	Variable 1 Agregar	Variable 2 Equivalentes	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC1.1: Agregar las clases a mapear de la geo-ontología cargada.	V	NA	El sistema muestra un listado con las clases seleccionadas.	Satisfactorio.	• Seleccionar las clases que se desean mapear.
EC 1.2: Hacer equivalentes.	V	V	El sistema muestra mensaje de confirmación.	Satisfactorio.	• Seleccionar la opción "Equivalentes".

Para el proceso de pruebas de caja negra fue necesario realizar dos iteraciones debido a que en la primera se detectaron algunas no conformidades como la ausencia de mensajes de confirmación o error después de realizadas algunas acciones, las cuales fueron solucionadas y al término de la fase de

pruebas el sistema cumple con los requerimientos planteados en el proceso de análisis y diseño de la solución.

Conclusiones parciales

En este capítulo se obtuvo un listado de los requerimientos funcionales entre los que se encuentran Cargar geo-ontología, Mapear geo-ontología, Visualizar geo-ontología y Brindar servicio. Se identificaron un conjunto de requisitos no-funcionales que definen las propiedades y restricciones del sistema a desarrollar: requisitos de usabilidad, requisitos de interfaz externa, requisitos de operatividad y portabilidad, requerimientos de software y hardware y las restricciones en el diseño y la implementación. Se describió la acción y capacidad delos actores que van a interactuar con el sistema. A partir de los requisitos funcionales identificados se elaboró el diagrama de casos de uso del sistema y se describieron textualmente cada uno de los casos de uso reconocidos. Todo lo anterior, ofrece una visión general del funcionamiento del sistema, mostrando con claridad que debe hacer y como lo debe realizar.

Además se obtiene como resultado un sistema completamente diseñado y construido en términos de clases del diseño. Se generaron los diagramas referentes al flujo de trabajo de implementación y se completó la modelación del transformador de geo-ontologías al estándar de la WGS84 en un grado más avanzado de la investigación. Para el desarrollo de la aplicación se utilizó una arquitectura distribuida en 3 capas fundamentales, al mismo tiempo que se emplearon los patrones de diseño GRASP y GOF. Se le realizaron pruebas al sistema desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno con el fin de determinar si el sistema es robusto y fácil de mantener, en caso de ocurrir un fallo.

Conclusiones Generales

- Los métodos investigativos aplicados permitieron ampliar el conocimiento acerca del estado actual del objeto de estudio.
- El proceso de tratamiento de la información concerniente al desarrollo y uso de las ontologías y sus subconjuntos en el ámbito nacional e internacional, proporcionó los elementos necesarios para fundamentar teóricamente el desarrollo de la propuesta.
- El estudio de las diferentes herramientas, metodologías y tecnologías actuales permitió identificar las más factibles para el modelado e implementación de la aplicación.
- El modelado del ciclo de desarrollo del software facilitó la visualización del sistema que se desarrolló, y conllevó a que las actividades planificadas fueran orientadas hacia la calidad.
- Se logró el desarrollo de un sistema capaz de transformar una geo-ontología al estándar de la WGS84.
- Se logró que el sistema cumpla con los requisitos funcionales y no-funcionales planteados para su desarrollo, por lo que se convierte en una herramienta capaz de brindar los resultados deseados.
- Se detallaron todos los artefactos generados a partir de la puesta en práctica del Proceso Unificado de Desarrollo como Metodología de Desarrollo de Software.

El producto de esta investigación es una herramienta cuyo aporte más significativo es la transformación de geo-ontologías al estándar de la WGS84. Por tanto se cumple el objetivo general que es obtener siempre una representación de una geo-ontología a través de la herramienta Map4rdf para facilitar el trabajo con las mismas.

Recomendaciones

Luego de haber finalizado la realización del presente trabajo de diploma se recomienda realizar un estudio aún más amplio con el objetivo de hacer el proceso de mapeo de una geo-ontología al estándar de la WGS84 de manera automática facilitando así el trabajo a los usuarios que interactúan con la representación de geo-ontologías a través de la herramienta Map4rdf.

Bibliografía

1. **Juan Carlos M. Coll, Contribuciones a las Ciencias Sociales.** eumed.net. [En línea] http://www.eumed.net/rev/cccss/02/vsp.htm..

2. **Webster, I. Braken & C.** Information technology in geography and planning. Londres & New York : Routledge, 1992. [En línea] 1992.

3. **Gruber, T.R.** "A translation approach to portable ontology specifications. Knowledge Acquisition 5.". [En línea] 1993.

4. **Gutiérrez, C.** "La lógica y el conocimiento.". [En línea] 1997. http://cariari.ucr.ac.cr/~claudiog/La_logica_y_el_conocimiento.html..

5. **A. Gangemi, D. Pisanelli, y G. Steve.** Ontology Alignment: Experiences With Medical Terminologies. Formal Ontology in Information System. N. Guarino. . [En línea] 1998.

6. **Jasper, M. U. Robert.** "A framework for understanding and classifying ontology applications. En: Proceedings of the IJCAI99 Workshop on Ontologies and Problem- Solving Methods (KRR5).". [En línea] 1999.

7. **G. Van Heijst, A.T. Schreiber, B.J. Wieligna,.** "Using Explicitit Ontologies in KBS Development¨: International Journal of Human Computer Studies 46.". [En línea] 1997.

8. **Guarino, N.** "¨Formal ontology in information systems¨. Proceedings of FOIS 98. IOS Press.". [En línea] 1998.

9. **Larin Fonseca, Rainer y Garea Llano, Eduardo.** *Relaciones topológicas para generación automática de ontologías en SIG.* La Habana : Ciencia Tierra y Esp, 2010.

10. **E. Garea-Llano, R. Oliva-Santos y C. Costales-Llerandi.** Integración Ontológica de Datos Metadatos y Conocimiento en Sistemas de Información Geográfica como Herramienta para la Interpretación Semántica de la Información Espacial. [En línea] 2009.

11. **Voronisky, E. G.Llano y F. V.** "Alineamiento de ontologías en el dominio geospacial.". [En línea] 2009.

12. **Oliva, L. M, Costales, C, Garea, E. y Maciá, F. .** Modelo de Anotación Semántica para Sistemas de Información Geográfica. “VI Congreso Internacional de Geomática 2009”. [En línea] 2009.

13. **Civil, Ingeniería.** http://ingecivilcusco.blogspot.com. [En línea] 2009. http://ingecivilcusco.blogspot.com/2009/09/sistema-geodesico-mundial-1984-wgs84.html.

14. **Aemet.** http://aemet.linkeddata.es. [En línea] 2011. http://aemet.linkeddata.es/technology.html.

15. **Jacobson.** www.knpue.com/TIC/materias/4/4sis/antologia.pdf. [En línea] 2000.

16. **Rofio, Oscar.** oscarrofio.netne.net/resumen.html. [En línea] 2003.

17. **Informática.** www.ecured.cu/index.php?title=Teoría_de_compiladores&oldid.... [En línea] 2006.

18. **Thayer, Drofman y.** www.getcited.org/pub/102863707. [En línea] 1990.

19. **informáticos, Departamento de lenguajer y sistemas.** http://www.lsi.us.es.http://www.lsi.us.es.[. [En línea] http://www.lsi.us.es/docencia/get.php?id=906..

20. **Larman, Craig.** http://www.taringa.net/posts/ciencia-educacion/12690915/UML-y-Patrones---Craig-Larman.html. [En línea] 2004.

21. **Fuentenegra Perez, Adrián, Gracia Águila, Adrián y Escalona Labrada, Kissy Yinet.** Revista Cubana de Ciencias Informáticas. *Revista Cubana de Ciencias Informáticas.* [En línea] [Citado el: 15 de octubre de 2012.] http://rcci.uci.cu/index.php/rcci/article/download/152/119. 1994-1536.

22. **Bosque-Sendra, J.** Sistemas de Información Geográfica. Madrid : Rialp., 1992. [En línea] 1992.

23. **Cebrian, J. A.** GIS Concepts. Cáceres : Departamento de Geografía y Ordenación del Territorio de la Universidad de Extremadura. [En línea] 1994.

24. **Sarabia-Lopez, Gerardo.** BÚSQUEDA Y PONDERACIÓN DE INFORMACIÓN CONTENIDA EN BASES DE DATOS ESPACIALES, UTILIZANDO JERARQUÍAS. México D.F. : s.n. [En línea] 2008.

25. **Pockin.** es.tldp.org/.../doc-modelado-sistemas.../doc-modelado-sistemas-uml.pdf. [En línea] 2008.

26. **Lovelle.** www.magma.com.ni/~jorge/upoli_uml/refs/Que_es_UML.doc. [En línea] 2005.

27. **Coenejo, José Enrique Gonzalez.** www.docirs.cl/uml.htm. [En línea] 2001.

28. **Meliá.** http://nlp.cic.ipn.mx/Publications/2004/Avances%20en%20la%20cienca%20de%20la%20computacion.pdf. [En línea] 2008.

29. **Informática, Instituto Nacional de Estadística e.** www.eclac.org/publicaciones/xml/3/22713/LCL2097.pdf. [En línea] 1999.

30. ECURED. [En línea] [Citado el: 26 de noviembre de 2012.] http://www.ecured.cu/index.php/TIC.

31. **Bordils i Rovaira, Francisco y Chavarría Díaz, Miguel.** Almacenamiento y transmisión de imágenes. PACS. [En línea] 2004. [Citado el: 27 de noviembre de 2012.] http://www.seis.es/seis/is/is45/IS45_54.pdf.

32. **Ferreira Moreno, Victor, Cabrera Fernández, Dalmys Amelia y Cifuentes de la Paz, Jorge Iván.** Infomed. [En línea] [Citado el: 27 de noviembre de 2012.] http://www.rcim.sld.cu/revista_15/articulos_pdf/siradiologico.pdf.

33. telecomsalud. [En línea] [Citado el: 9 de diciembre de 2012.] http://www.telecomsalud.com/dicom.htm.

34. Informática Médica. [En línea] [Citado el: 15 de Enero de 2013.] http://www.informaticamedica.cl/2012/07/que-es-dicom.html.

35. ECURED. [En línea] [Citado el: 18 de Enero de 2013.] http://www.ecured.cu/index.php/Microsoft_Visual_Studio.

36. ECURED. [En línea] [Citado el: 18 de Enero de 2013.] http://www.ecured.cu/index.php/PostgreSQL.

37. PostgreSQL-es. [En línea] [Citado el: 19 de Enero de 2013.] http://www.postgresql.org.es/sobre_postgresql.

38. ECURED. [En línea] [Citado el: 20 de Enero de 2013.] http://www.ecured.cu/index.php/UML.

39. Scribd. [En línea] [Citado el: 20 de Enero de 2013.] http://es.scribd.com/doc/7411856/Caracteristicas-de-C.

40. **Rodríguez López, Martha y Rodríguez García, Raymundo.** *Propuesta de aplicación de los perfiles de integración de IHE entre los sistemas alas PACS–alas RIS–alas HIS.* Ciudad de La Habana : s.n., 2010.

41. slideshare. [En línea] [Citado el: 26 de Enero de 2013.] http://www.slideshare.net/bernardolimachi/metodologia-rup-14288208.

42. VATES ingenieria de software. [En línea] [Citado el: 29 de Enero de 2013.] http://www.vates.com/cmmi/que-es-cmmi.html.

43. Tecnología y Synergix. [En línea] 10 de julio de 2008. [Citado el: 17 de abril de 2013.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

44. **Bauta Camejo, René R y Torres Galvez, Ariel.** *HERRAMIENTA PARA LA GESTIÓN Y EL ANÁLISIS DE TRAZAS EN EL SISTEMA DE GESTIÓN INTEGRAL CEDRUX.*

45. PDSIB. [En línea] [Citado el: 18 de Enero de 2013.]
http://itzamna.uam.mx/pdsib/PROYECTOS/PACS/PACS.HTML.

46. PediatricRad. [En línea] 1994. [Citado el: 15 de Enero de 2013.]
http://www.pediatricrad.info/wo/textes/pacsteor.htm.

47. Softonic. [En línea] [Citado el: 5 de Febrero de 2013.] http://sawmill.softonic.com/.

48. Imagine. [En línea] [Citado el: 7 de Febrero de 2013.]
http://www.imagine.com.co/conectividad/maildetective_mdaemon.php.

49. Free Download Manager. [En línea] [Citado el: 2 de 5 de 2013.]
http://www.freedownloadmanager.org/es/downloads/Centro_Syslog_65396_p/.

50. SparxSystems. [En línea] [Citado el: 12 de Febrero de 2013.]
http://www.sparxsystems.com.ar/products/ea/index.html.

51. ManageEngine. [En línea] [Citado el: 18 de marzo de 2013.]
http://www.manageengine.es/pages/productos/eventlog-analyzer/resumen/.

52. Ingeniería DRIC. [En línea] [Citado el: 18 de marzo de 2013.] http://www.dric.com.mx/eventlog-analyzer/que-es-eventlog-analyzer.

53. definiciones.de. [En línea] [Citado el: 2 de abril de 2013.] http://definicion.de/modelo-de-datos/.

54. definiciones.de. [En línea] [Citado el: 15 de enero de 2013.] http://definicion.de/seguridad-informatica.

55. **Zayas-Basán Mola, Yaquelin.** *La seguridad de la información clínica. Perfiles de seguridad IHE.* La Habana : s.n., 2012.

56. **de la Torre Llorente, César, y otros, y otros.** *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* s.l. : Krasis Press, 2010.

57. Zystrax. [En línea] [Citado el: 8 de febrero de 2013.] http://zystrax.blogspot.com/2009/07/analisis-de-los-ficheros-de-logs.html.

58. Deep Software. [En línea] [Citado el: 11 de febrero de 2013.] http://www.deep-software.com/.

Referencias Bibliográficas

Aguilar López, Dulce María. 'Búsqueda Web Basada en Ontologías de Dominio'. Consultado en: http://www.tamps.cinvestav.mx/sep262008t.

Barrera Juárez, Orlando. Sistema de Información Geográfica MóvilBasadoen ComunicacionesInalámbricasy Visualización de Mapas en Internet. México, 25 de Marzo de 2011. Consultado en: http://biblioteca.cicese.mx/catalogo/tesis/ficha.php?id=18638#.

Blog de la Web Semántica "Apollo: un editor de ontologías". Julio 17, 2002. Consultado en: http://apollo.open.ac.uk/index.html.

Cervera García, Arturo. Expresividad Limitada en el Uso de Editores de Ontologías. Universidad de Valencia, España. Febrero 2006. Consultado en: http://www.revista.unam.mx/vol.7/num2/art07/int07.htm.

Dávila, Jacinto. La Lógica, Los Agentes y la Web Semántica. Universidad de Los Andes, Venezuela. Consultado en: http://webdelprofesor.ula.ve/ingenieria/jacinto/ws/web-semantica.html.

De J. Carmona, Alvaro. Sistemas de Información Geográfica. Consultado en: http://www.monografias.com/trabajos/gis/gis.shtml.

Gutiérrez Lázaro, Juan Carlos. UML"Diagramas de Clases y Casos de Uso". 2009. Consultado en: http://www.fdi.ucm.es.

Ierache, Jorge; Bruno, M. Marcela y García Martínez, Ramón. Ontología para el Aprendizaje y Compartición de Conocimientos entre Sistemas Autónomos.

Ignacio J. Blanco; Carmen Martínez Cruz y M. Amparo Vila. Arquitectura para la integración de esquemas relacionales difusos basada en ontologías: una aplicación para la web. Septiembre de 2008.

Lenguajes del lado del cliente o servidor. 2006. Consultado en: http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

Lozano Tello, Adolfo. TESIS DOCTORAL: Métrica de Idoneidad de Ontologías. Universidad de Extremadura. Dpto. de Informática. Abril, 2002. Consultado en: http://quercusseg.unex.es/adolfo/tesis.htm.