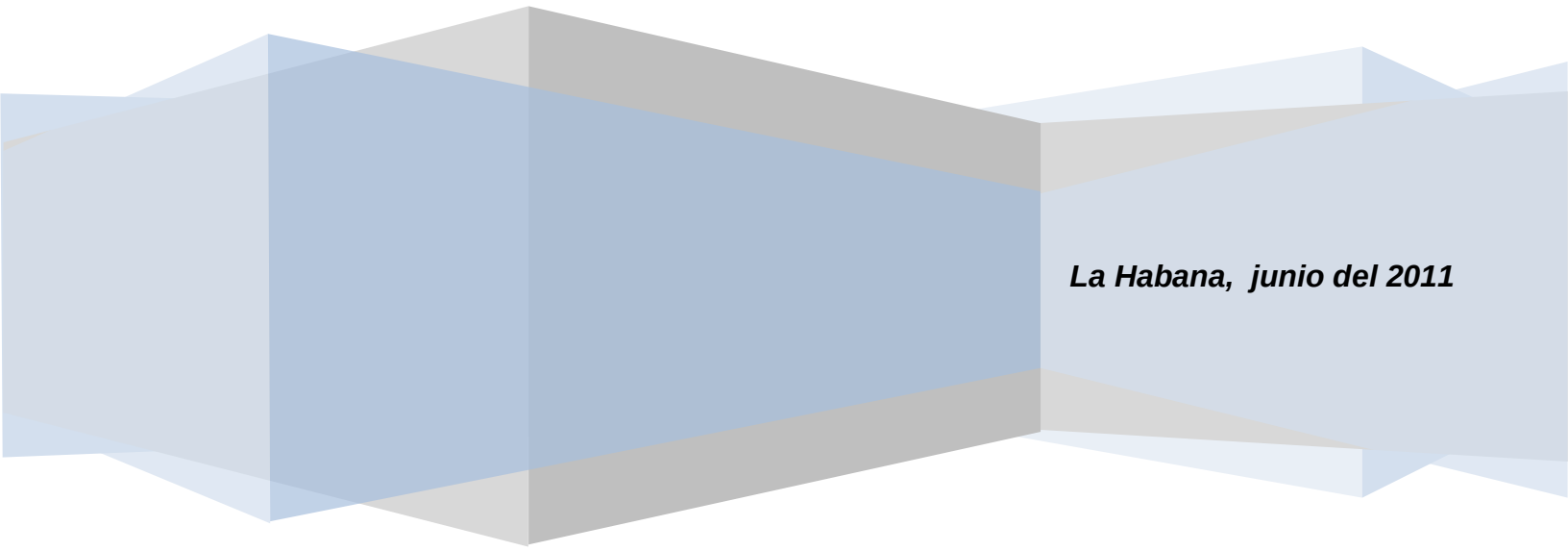
Universidad de las Ciencias Informáticas
Facultad 6

Título: Desarrollo de un visor de ontologías en OWL para la plataforma GeneSIG

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autor: Aurelio Díaz Nuñez
Tutor (es): ***Ing.*** *Adrián Gracia Águila*
Msc. *Manuel Enrique Puebla Martínez*
Co-Tutor: ***Msc.*** *Yordany Llovera López*



La Habana, junio del 2011

Si he conseguido ver más lejos, es porque me aupado en hombros de gigantes.

Isaac Newton

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Desarrollo de Geo-informática y Señales Digitales (GEySED) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2011.

Autor:

Aurelio Díaz Núñez

_____

Tutores:

Ing. Adrián Gracia Águila

_____

MCs. Manuel Enrique Puebla Martínez

_____

Dedicatoria

A mis padres por guiarme con empeño por el camino correcto.

A mi abuela Teresa por ser tan dedicada.

A mi hermana y mis sobrinas por recordarme siempre su presencia.

Y a todas las personas que me apoyaron continuamente.

A ustedes, mi amor incondicional.

Agradecimientos

A mi pareja, Yordany Llovera López (La Negra), por ser la fuerza, la voluntad, el arrojo, por ser mi más cercana colaboradora, a ti mis más sinceros agradecimientos.

A mis tutores Adrián Gracia y Manuel E. Puebla por comprender la importancia de esta etapa en mi vida y colaborarme atentamente.

A Alain León Companioni por su vital colaboración.

A mi compañero de trabajo Adrian Fuentenegra por su voluntad.

A mi hermano, Roberto Sánchez por ayudarme a vencer esos grandes molinos de vientos de la juventud.

A mis compañeros de distracciones, estudios y malas experiencias competitivas Alain López, Adrián Martínez, Leiber Fornaris y Adrián Viñas por estar siempre que los necesito.

"A todos, eternos agradecimientos."

Resumen

En la contemporaneidad las ontologías han alcanzado gran relevancia en el campo de la ciencia y la tecnología, pues sus funcionalidades contribuyen a eliminar confusiones conceptuales y terminológicas. El concepto de ontología en si mismo, ha avanzado a pasos agigantados desde el hecho que permite representar una base de conocimientos a partir de la realidad de los conceptos. Por la importancia que amerita, su uso se ha extendido al desarrollo de los Sistemas de Información Geográfica, permitiendo almacenar información semántica de los objetos geográficos representados.

La plataforma GeneSIG, perteneciente al Centro de Desarrollo de Geo-informática y Señales Digitales (GEySED) de la facultad 6, cuenta con un gran volumen de información, ocasionando que existan dificultades en el proceso de análisis y evaluación de la misma, lo que implica que la información consultada por los usuarios no satisfaga del todo sus necesidades.

El presente trabajo investigativo ha tenido como principal objetivo el desarrollo de un componente que se integre a la plataforma GeneSIG posibilitando visualizar una jerarquía de conceptos. Para ello se realizó un estudio del estado del arte de las ontologías, de las herramientas que se usan para su desarrollo y de los diferentes plug-ins que permiten su visualización.

Como resultado final se obtuvo un visor de ontologías en OWL, este constituye un aporte importante y significativo debido a que permite aumentar las funcionalidades de la plataforma GeneSIG, tiene el poder de visualizar una jerarquía de conceptos como un modelo genérico y demostrativo, y cuenta con una interfaz sencilla e intuitiva.

Palabras clave

Ontologías, Geo-ontologías, Sistemas de Información Geográfica (SIG), OWL.

Índice de contenidos

Introducción

El desarrollo ascendente de las Tecnologías de la Información y las Comunicaciones (TIC) es un hecho incuestionable, forma parte de la cultura tecnológica que caracteriza al mundo contemporáneo y que se ha expandido a todos los ámbitos y estratos de la sociedad ampliando las posibilidades de desarrollo social. El E-mail, la fibra óptica, los satélites y la World Wide Web (WWW) forman parte de las TIC y juntas han revolucionado la manera en la cual las sociedades obran recíprocamente.

El origen de la World Wide Web (WWW) y su desarrollo supusieron el uso generalizado de la misma, siendo un medio flexible que propicia el intercambio de la información. Esta está a punto de sufrir un nuevo cambio, donde los ordenadores podrán interpretar la información sin la intervención humana. Permitirá encontrar respuestas de una forma rápida y sencilla gracias a la mejor estructuración de la información.

Con el desarrollo de la Web y los deseos de compartir y rehusar el conocimiento, la investigación sobre las ontologías como posibles sistemas para representar el conocimiento almacenado en las páginas Web, comenzó a cobrar importancia en la medida en que el problema de la búsqueda y recuperación de información se agudizó, y el WWW Consorcio comenzó a desarrollar el proyecto de la Web Semántica.

La palabra “ontología”, en el ámbito de las tecnologías de la información y la comunicación, ha suscitado un gran interés, estas forman parte del soporte de la Web Semántica, se basan en la descripción del mundo real y son usadas generalmente para definir el conocimiento de un dominio. Las ontologías definen de forma compartida y consensuada los conceptos y relaciones de los diferentes dominios. El uso de ontologías proporciona un salto cualitativo en funcionalidad y posibilidades, permitiendo la comunicación entre personas, organizaciones y aplicaciones.

En la actualidad mundial, las ontologías han cobrado un gran auge en el campo de la ciencia y la tecnología, pues sus funcionalidades contribuyen a eliminar confusiones conceptuales y terminológicas. En el ámbito internacional, varios investigadores han dedicado algunas de sus obras al estudio de este tema por la importancia que recobra. Entre los autores se destacan: Manuel Noguera García con la tesis doctoral ¨ Modelado y análisis de sistemas CSCW siguiendo un enfoque de ingeniería dirigida por

ontologías¨, Miguel Feliz Mata Rivera con la tesis doctoral ¨Recuperación y ponderación de información geográfica desde repositorios no estructurados conducidas por ontologías¨, Francisco José Álvarez Montero con la tesis doctoral ¨ Construcción de recursos lingüísticos basados en ontologías con control de relaciones semánticas¨ y Rolando Quintero Téllez con la tesis doctoral ¨ Representación Semántica de Datos Espaciales Raster¨.

El investigador Miguel Ángel Avián, en su artículo ¨ Ontologías, qué son y para qué son¨, expresó… ¨las ontologías serán imprescindibles en la Web Semántica y en los futuros sistemas de gestión empresarial porque permitirán que las aplicaciones estén de acuerdo en los términos que usan cuando se comunican. Mediante ellas, será mucho más fácil recuperar información relacionada temáticamente, aún cuando no existan enlaces directos entre las páginas web¨. (Avián 2005)

Este fenómeno se ha convertido en un tema de investigación y aplicación en varios países. En este sentido, Cuba se encuentra inmersa en el estudio de las ontologías para su aplicación en diversos organismos y proyectos en desarrollo. Se destaca el Proyecto CYTED, centro creado en el año 2004 y orientado a las investigaciones teóricas y aplicadas en el área del Reconocimiento de Patrones y la Minería de Datos.

Para ello se han convocado eventos y talleres, entre los cuales pueden señalarse: II Taller de ontologías ¿semántica para qué?, desarrollado en la Habana en junio del 2007, dedicado a la evaluación y potenciación del papel de las Infraestructuras de Datos Espaciales en el desarrollo sostenible en América Latina y el Caribe.

Existen varios profesionales cubanos que se encuentran investigando temas vinculados con la Semántica Espacial, las Ontologías y las Geo-ontologías, entre los que se destacan: el Dr. Eduardo Garea Llano, jefe del Departamento de Reconocimiento de Patrones del CENATAV, la Dra. Mercedes Delgado Fernández, Decana de la Facultad de Ingeniería Industrial de la CUJAE, el Dr. Roberto Pérez, director del Sistema Nacional de Monitoreo Ambiental, Dr. Carlos Balmaseda, profesor de la Universidad Agraria de La Habana y el Dr. Rafael Espín, profesor de la CUJAE.

El uso de las ontologías, profundizado por las diversas investigaciones realizadas sobre el tema, ha contribuido al desarrollo de los Sistemas de Información Geográfica (SIG), permitiendo almacenar información semántica de los objetos geográficos representados. Cada vez son más las organizaciones

que en el mundo están incorporando la tecnología SIG, pues su funcionamiento es significativo para la representación y el estudio de la información geográfica. Estos son software que permiten ingresar datos mediante operaciones sencillas, manejarlos, analizarlos, combinarlos en modelos o por superposición, producir nueva información y mostrar el resultado de estas operaciones en mapas o tablas.

Los SIG constituyen hoy uno de los campos más dinámicos y novedosos de aplicación de la informática, con un indudable efecto en la sociedad. En correspondencia con ello, su uso se ha hecho extensivo al estudio del impacto ambiental, a la planificación urbanística, a los estudios de viabilidad, a la utilización de recursos naturales y a las compañías de servicios públicos (Electricidad, Teléfonos, Abastecimiento de aguas, Saneamiento, etc.).

Las expectativas creadas sobre SIG están también presentes en Cuba. Sus principales usuarios en el país no son los cibernéticos o especialistas informáticos, la mayoría son geólogos, cartógrafos, geógrafos, desarrolladores, arquitectos e ingenieros, quienes conocen y operan Sistemas de Información Geográfica en sus investigaciones y proyectos. Pueden señalarse los proyectos de protección de la biodiversidad y desarrollo sustentable de los diferentes ecosistemas y la gestión de la estadística de salud de Cuba a través de la representación cartográfica, por solo citar alguno de ellos. Entre las empresas e instituciones que se destacan en el desarrollo y aplicación de los SIG como parte del uso extensivo de las tecnologías se encuentran GEOCUBA, Ministerio de Ciencia, Tecnología y Medio Ambiente, Instituto de Planificación Física, Ministerio de las Fuerzas Armadas y diferentes Universidades.

En consonancia con este desarrollo progresivo de las Tecnologías de la Información y las Comunicaciones en Cuba, es imprescindible destacar el Proyecto Futuro: La Universidad de las Ciencias Informáticas (UCI). Centro creado para la formación y superación de profesionales de la informática y la producción de software y servicios asociados para la industria nacional y la exportación. Esta institución cuenta con varios proyectos de desarrollo distribuidos por facultades. En la Facultad 6 se encuentra el Centro de Desarrollo de Geo-informática y Señales Digitales (GEySED), y dentro de esta la Línea de Desarrollo GeneSIG, que es una plataforma soberana con más de 50 funcionalidades utilizada para el montaje de SIG con diferentes propósitos.

A partir del estudio sobre el funcionamiento de la plataforma GeneSIG, se han identificado la existencia de datos dispersos, duplicidad de datos y un ineficiente flujo de los datos por falta de una tecnología de comunicación y de un sistema que garantice el suministro eficiente de la información.

Analizando la situación descrita anteriormente se ha identificado el siguiente **problema científico:**

La plataforma GeneSIG no cuenta con un servicio de visualización de conocimientos y relaciones que presten asistencia a los usuarios para que logren interpretar mejor la información que se maneja.

Realizando un análisis del problema planteado, se determinó como **objeto de la investigación**: las Geo-ontologías como forma de representación de la información y el conocimiento en los Sistemas de Información Geográfica y el **campo de acción**: tecnología para la visualización de Geo-ontologías.

La **idea a defender** plantea que: el desarrollo de un visor de Geo-ontologías permitirá representar y visualizar de forma detallada la información y el conocimiento de los Sistemas de Información Geográfica.

Partiendo de esta temática se plantea el siguiente **objetivo**: desarrollar un visor de Geo-ontologías sobre la plataforma GeneSIG.

Se trazan como **tareas investigativas**:

1. Analizar el estado del arte relacionado con las Geo-ontologías como un sub-conjunto de las ontologías y las diferentes formas de representar las mismas.
2. Analizar el estado actual de la realización de visores de ontologías para la Web Semántica.
3. Fundamentar el lenguaje que se aceptará para las ontologías a visualizar.
4. Fundamentar las herramientas a utilizar en la construcción de la solución.
5. Argumentar la Metodología de Desarrollo de Software a usar en el proceso.
6. Identificar los requisitos funcionales y no funcionales del módulo de visualización.
7. Diseñar el módulo de visualización.
8. Implementar el módulo de visualización.

9. Evaluar la solución propuesta.

Para el desarrollo de la investigación se emplearon los siguientes métodos teóricos.

Métodos teóricos

- **Análisis de documentos:** Posibilitó realizar el análisis de documentos de autores nacionales y extranjeros referentes al tema, para llevar a cabo la fundamentación y el desarrollo de la investigación.
- **Histórico – lógico:** Se utilizó para hacer un análisis retrospectivo sobre el desarrollo, la utilización y el alcance de las ontologías y de los visores de geo-ontologías a nivel nacional e internacional.
- **Análisis – síntesis:** Se aplicó al análisis bibliográfico, de definiciones y enfoques de diferentes autores sobre el desarrollo de la Web Semántica, las Ontologías, las Geo-ontologías y los Sistemas de Información Geográfica.
- **Inducción – deducción:** Se empleó para conocer la realidad sobre el funcionamiento de la plataforma GeneSIG y sus posibilidades para desarrollar el visor de Geo-ontologías.

La tesis está compuesta por la presente introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía y anexos que complementan la información comprendida en ella.

Capítulo 1: Está dedicado al análisis del estado del arte de las Geo-ontologías como un sub-conjunto de las Ontologías y las diferentes formas de representar las mismas, más específicamente conocer el estado actual de la realización de visores de ontologías para la Web Semántica. Como elemento importante se realizará el análisis y fundamentación del lenguaje que se aceptará para las ontologías a visualizar.

Capitulo 2: Se centra en la valoración de las herramientas a utilizar en la construcción de la solución, así como en la selección y argumentación de la Metodología de Desarrollo de Software a emplear en el proceso.

Capitulo 3: Está dedicado a identificar y modelar los requisitos funcionales y no funcionales de la aplicación. Se realizará el Modelo de Casos de Uso del Sistema.

Capitulo 4: Comprende el diseño, implementación y aplicación de casos de pruebas al módulo de visualización.

Capítulo 1: Tendencias actuales de las ontologías

1.1. Ontologías

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 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." (Gutiérrez 1997)

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. (Gangemi 1998)

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 como los conceptos están interrelacionados. Las ontologías son usadas para mejorar la comunicación entre personas o computadoras. (Jasper 1999)

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. (Van Heijst 1997)

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 cuando plantea ***"Una ontología es una***

***especificación formal y explícita de una conceptualización compartida*"**. (Gruber 1993) Conceptualización es una abstracción, un enfoque simplificado del mundo que se desea representar, construido a partir de los conceptos y 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: (Gruber 1993)

- **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. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.

- **Funciones:** son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como 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 describen los conceptos de algún dominio, sus propiedades y la relación de estos conceptos entre si. A partir del desarrollo de las ontologías devienen diferentes clasificaciones. Gangemi y otros autores distinguen tres tipos de ontologías: (Gangemi 1998)

- **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/o objeto de la ontología. Identificado con este criterio, Guarino clasifica las ontologías en función de su nivel de dependencia de una tarea definida: (Guarino 1998)

- **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, 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. (Gruber 1993)

- **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.

1.2. Ontologías geográficas

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 geo-espacial¨ plantean que las ontologías que tratan la temática geo-espacial presentan características particulares de este campo de estudio. Debido a su nivel de especialización, a estas ontologías geográficas se les han denominado geo-ontologías. (Garea 2009) 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.(Oliva 2009)

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 geo-espaciales. 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.

1.3. Estado actual de los visores de ontologías para la Web Semántica

Las ontologías se han convertido en un elemento significativo para lograr expresar el conocimiento, ya sea entre agentes de software o personas, pero su construcción es una tarea compleja. Desde mediados de los noventas se han desarrollado herramientas que facilitan esta labor. A continuación se distinguen los siguientes grupos de herramientas: (Maroto 2007)

- **Herramientas para la evaluación de ontologías:** permiten evaluar el contenido de las ontologías construidas.
- **Herramientas para la fusión y alineación de ontologías:** permiten fusionar y alinear las distintas ontologías existentes para un mismo dominio.
- **Herramientas de aprendizaje de las ontologías:** permiten construir ontologías de manera semiautomática a partir de textos en lenguaje natural mediante técnicas de aprendizaje automático y análisis del lenguaje natural.
- **Herramientas para el desarrollo de ontologías:** permiten construir ontologías desde cero. Estas herramientas además de ser usadas para la edición y navegación de ontologías, ayudan en las fases de documentación, exportación e importación en diferentes formatos y lenguajes ontológicos. Algunos de los ejemplos de este tipo de herramientas son KAON, WebODE, WebOnto, Protégé, OntoEdit entre otras.

Protégé y OntoEdit son algunas de las herramientas que más se utilizan, éstas ayudan a implementar un amplio entorno para el diseño, la administración y la visualización de ontologías.

A continuación se especifican las características de los editores de ontologías Protégé y OntoEdit. Se explican además, los *plug-ins*[1] que emplean para la visualización de ontologías.

[1] Es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Protégé

Protégé es un software para el soporte de desarrollo de ontologías creado en la Universidad de Standford por el grupo SMI (Standford Medical Informatics), sus capacidades gráficas facilitan la edición de ontologías. Es una herramienta basada en Java, de código abierto, que por su arquitectura expandible, puede ampliarse por medio de plug-ins que mejoran las posibilidades de exportación e importación. La interfaz de usuario se configura fácilmente, contiene una serie de pantallas llamadas *Tabs*, cada pantalla muestra un aspecto diferente de la ontología en una vista especializada (Knublauch 2006). Este editor utiliza dos tipos de razonadores: Pellet y FACT. Protégé permite el diseño, implementación y manipulación de ontologías.

La plataforma Protégé proporciona dos modos diferentes de modelar ontologías:

- **Protégé-Frames[2]:** permite crear y almacenar ontologías basadas en marcos utilizando un modelo de conocimiento compatible con OKBC.
- **Protégé-OWL:** es una extensión que permite desarrollar ontologías utilizando el lenguaje ontológico estándar OWL. Está concebido como un plug-ins de Protégé que puede cargar y guardar ontologías en OWL y en RDF, así como editar y visualizar las clases, propiedades y reglas.

Protégé hace uso de los siguientes plug-ins gráficos para visualizar las ontologías:

- **Jambalaya:** plug-in que permite ver todos los componentes de la ontología gráficamente, facilita la navegación interactiva de las mismas. Jambalaya soporta una amplia gama de algoritmos de representación.

En la figura 1 se muestra una pantalla de la ontología desarrollada en Protégé. Del lado izquierdo está la jerarquía de clases y del lado derecho aparece una representación en Jambalaya.

[2] Marcos

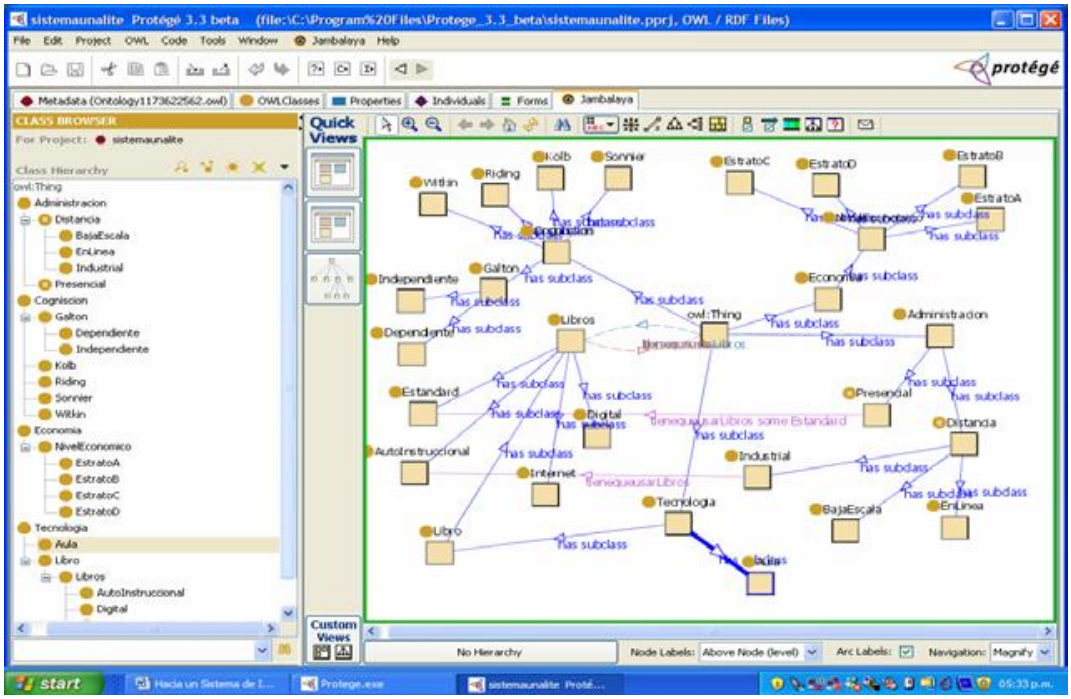


Fig. 1 Pestaña de Jambalaya de Protégé.

Jambalaya presenta requisitos funcionales con el fin de ayudar a comprender las bases de conocimientos almacenadas:

- **Proporcionar información:** permitir una visión general de la ontología y la visualización de las relaciones no estructurales.
- **Facilitar la navegación:** permitir la exploración y visualización de los resultados de la búsqueda.
- **Apoyo de Edición Gráfica:** proporcionar técnicas para la adquisición y edición del conocimiento visual.
- **Facilitar la colaboración:** proporcionar un mecanismo para compartir y ahorrar aspectos específicos de una ontología.
- **Vista principal:** la vista principal representa la pantalla donde se maneja la mayoría de las operaciones del usuario.

- **Vista en jerarquía:** la vista jerárquica presenta una visualización de alto nivel utilizando un diseño de representación en forma de árbol para mostrar una visión general de la base de conocimiento.
- **Vista de mapas de árboles:** esta herramienta utiliza un algoritmo de mapa de árbol. Por lo general puede mostrar más nodos que cualquier otra herramienta, siendo útil para obtener un sentido de la estructura general de una base de conocimientos.
- **Buscador:** Jambalaya actualmente soporta la funcionalidad de búsqueda limitada.
- **Zoom:** permite observar con mayor nivel de detalle o ver con mayor profundidad las representaciones de las bases de conocimiento.

Jambalaya no se centra en una técnica de visualización en particular, este ofrece una variedad de puntos de vistas que el usuario puede seleccionar. La vista que por defecto brinda Jambalaya es la anidada, otras técnicas de visualización que incluye son: diseño de árbol (tanto horizontal como vertical), diseño de la fuerza-dirigido, el diseño mapa de árbol, y un diseño radial. Todas estas técnicas persiguen un mismo objetivo ayudar a los usuarios a comprender grandes bases de conocimientos.

- **OWLViz:** es un plug-in que permite la visualización gráfica y la navegación sobre las jerarquías de clases de las ontologías OWL. Necesita de la aplicación GraphViz[3] para poder funcionar. Una vez instalados OWLViz y GraphViz, se genera una pestaña en el entorno de Protégé denominada OWLViz. En ella existe una barra de herramientas que provee diferentes funcionalidades al plug-in.

Entre las principales funciones de esta pantalla se destacan:

- Mostrar clases, mostrar sub-clases, mostrar súper-clases, ocultar clases, ocultar sub-clases, ocultar súper-clases, ocultar clases especificando el radio, ocultar todas las clases, mostrar solo las clases primitivas.

En la figura 2 se puede observar una pantalla de la ontología desarrollada en Protégé. Del lado izquierdo está la jerarquía de clases y del lado derecho aparece una representación en OWLViz.

[3] Librería de gráficos

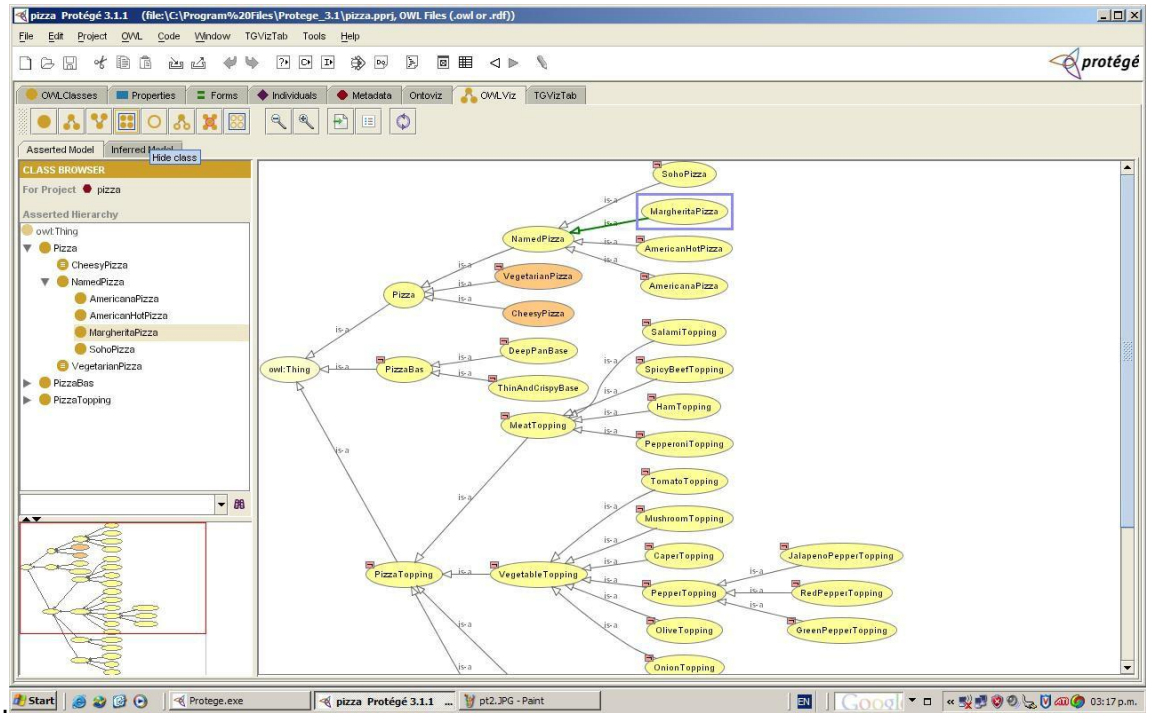


Fig. 2 Pestaña de OWLViz de Protégé.

Los plug-ins Jambalaya y OWLViz fueron creados para visualizar bases de conocimiento permitiendo explorar espacios de información complejos, a pesar de poseer rasgos favorables estas herramientas no poseen las características apropiadas que le permitan ser una propuesta de solución. Jambalaya y OWLViz no poseen una interfaz que sea compatible con la plataforma GeneSIG, ya que ambos componentes son parte de una aplicación de escritorio desarrollada en Java, mientras que GeneSIG es una aplicación web que posee una interfaz desarrollada en ExtJS.

OntoEdit

OntoEdit es otra de las herramientas de edición de ontologías que apoya el desarrollo y mantenimiento de las mismas, creado por el Instituto AIFB de la Universidad de Karlsruhe. OntoEdit permite almacenar y posteriormente manipular ontologías en una base de datos relacional. OntoEdit es una aplicación Java que posee una interfaz de visualización mediante la que es posible navegar y editar la ontología de manera gráfica. Una ventaja añadida en OntoEdit es que los términos de la ontología pueden ser definidos en varias lenguas.

Este editor de ontologías cuenta con otras funcionalidades importantes, entre las que se destacan:

- Importar estructuras del directorio y tablas de Excel.
- Visualizar y editar ontologías en un gráfico.
- Dispone de reglas que eliminan fallos en la visualización del gráfico.

A continuación se hace referencia a los elementos que componen una ontología en OntoEdit y que se consideran significativos para su correcto desarrollo: términos, propiedades y relaciones.

- **Términos:** representan los conceptos extraídos de un dominio del conocimiento. Cada término posee atributos, nombre y descripción que identifican al término.
- **Propiedades:** representan otros atributos definidos por el usuario. Cada propiedad posee los atributos nombre y tipo de valor. Al definir una propiedad deberá estar asociada a un determinado término o a varios de ellos de forma independiente.
- **Relaciones entre términos:** Un término puede estar relacionado con uno o más términos. Cada relación posee un atributo (nombre). Subclases: es un tipo de relación en el que las propiedades de la superclase son heredadas por la subclase.

OntoEdit cuenta con el **Plug-in Visualizador,** plug-in que posee la opción de editar todos los componentes de una ontología, además de tener la posibilidad de hacer búsquedas o identificar partes, así como de visualizar una ontología en forma de grafo dirigido.

En la figura 3 se puede observar una pantalla de la ontología desarrollada en OntoEdit.

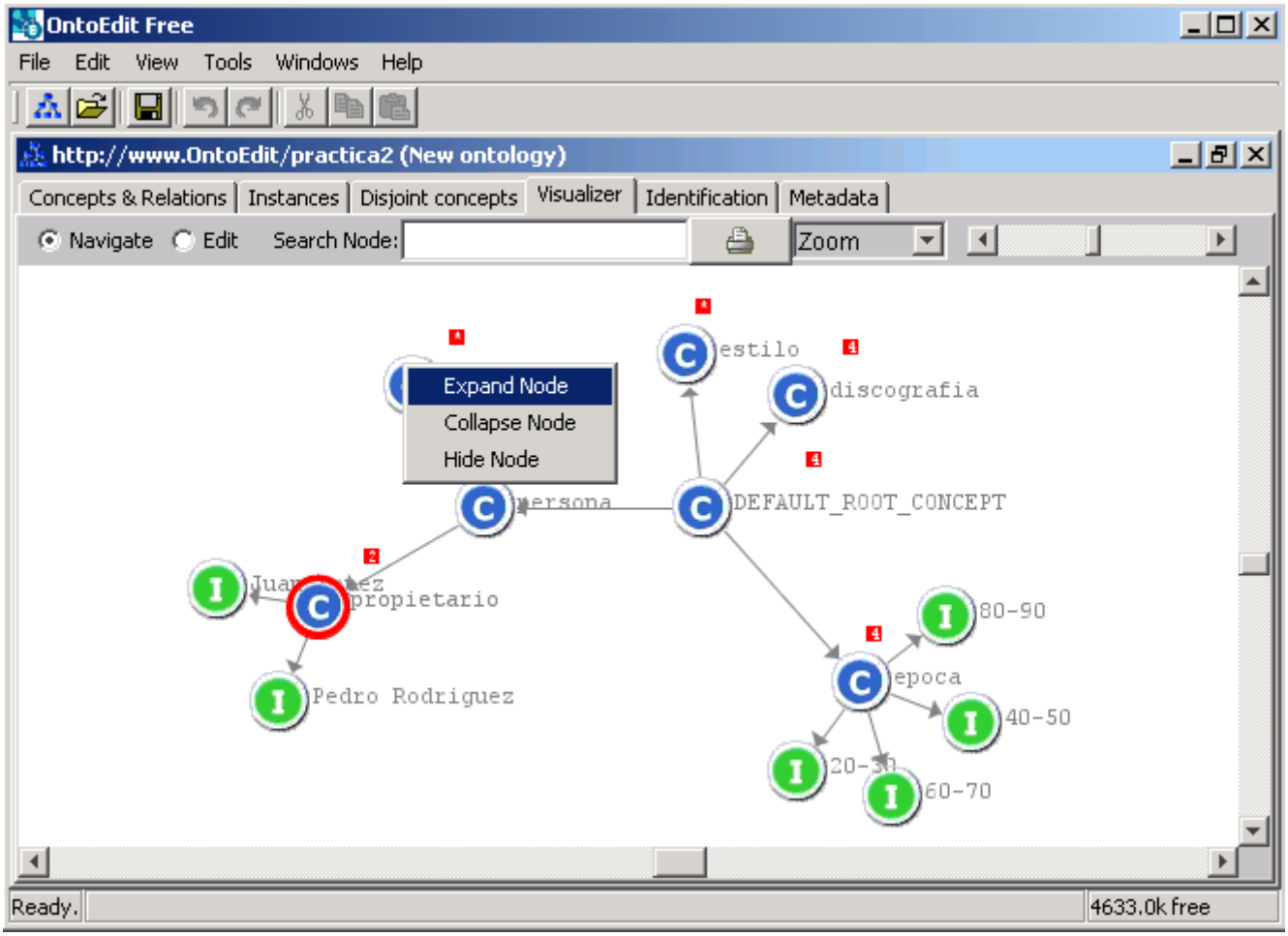


Fig. 3 Plug-in Visualizador de OntoEdit.

En la Fig. 3 pueden observarse varias funciones del plug-in visualizador, de ellas se considera como función principal el Buscador. Este posee un cuadro de texto en el que se ingresa el término a buscar, si se encuentran diversos nodos que coincidan con el término buscado, OntoEdit procede a crear una lista de nodos hallados, después de seleccionar el nodo deseado se procede a resaltar el nodo dentro del grafo dirigido.

El plug-in Visualizador al igual que los plug-ins Jambalaya y OWLViz posee rasgos favorables, pero no presenta las características necesarias que le permita ser una propuesta de solución, debido a que su interfaz de usuario no es compatible con la que presenta la plataforma GeneSIG, pues esta última es una

aplicación web que posee una interfaz gráfica desarrollada en ExtJS y el plug-in Visualizador es un componente que forma parte de una aplicación de escritorio desarrollada en Java.

A raíz de las potencialidades que ofrece el uso de visores ontológicos han surgido proyectos importantes que diseñan mapas conceptuales a partir de ontologías. En la actualidad existen iniciativas en este sentido, tales como, el proyecto Cmap Ontology Development Environment (CODE) del Institute for Human and Machine Cognition (IHMC) para la aplicación de editores de mapas conceptuales en la visualización de ontologías (CODE 2004).

Como se ha podido comprobar estas iniciativas son fruto del deseo de potenciar el empleo de los visualizadores de ontologías, permitiendo a los usuarios ver el alcance de las mismas. La visualización del conocimiento ayuda a realizar una lectura asociativa donde el usuario pasa a ser un lector activo que participa de forma directa en la construcción de su conocimiento.

1.4. Lenguaje ontológico OWL

El conocimiento se puede codificar en diferentes lenguajes ontológicos como XML SCHEMA, RDF, RDF SCHEMA y OWL. Este grupo de lenguajes está bajo el auspicio del W3C y han sido estandarizados, pues permiten la especificación de los contenidos en los documentos diseñados para la Web (OWL 2004). A continuación se muestran estos lenguajes ordenados de menor a mayor capacidad expresiva:

- **XML Schema:** es un lenguaje que permite definir la estructura de la información en documentos XML para que sea conforme a un esquema determinado. La semántica acerca del contenido de la información que se puede expresar con este lenguaje es limitada. XML Schema no se suele considerar como un lenguaje de ontologías.

- **RDF:** es un modelo de datos para describir objetos y relaciones entre ellos, con una semántica simple. RDF fue concebido como un marco de trabajo para representar metadatos acerca de recursos Web.

- **RDF Schema:** es un vocabulario para describir propiedades y clases de recursos RDF. Cuenta con vocabulario y semántica para describir jerarquías de generalización de dichas propiedades y clases.

- **OWL:** amplía el vocabulario proporcionado por RDF Schema para describir propiedades y clases. OWL es un lenguaje basado en una representación XML, por lo tanto es un lenguaje legible que permite compartir ontologías y reutilizar conocimiento orientado a la Web para definir nuevas ontologías. Es un lenguaje basado en marcos, lo cual hace que sea fácil representar una ontología en términos de Orientación a Objetos.

OWL es el lenguaje ontológico más extendido a partir del surgimiento de la Web Semántica y recomendado como estándar por el World Wide Web Consortium (W3C). OWL tiene una mayor capacidad para representar contenido interpretable por un ordenador en la web al proveer un vocabulario rico y estándar, con una semántica formal, para expresar conceptos y relaciones entre ellos.

El Lenguaje de Ontologías Web (OWL) es un lenguaje de marcado que ha alcanzado un gran auge, compatible con la World Wide Web en general y con la Web Semántica en particular. OWL tiene la capacidad de expresar una semántica más rica que la proporcionada por cualquier otro lenguaje ontológico. (Evermann 2008)

Los documentos OWL en comparación con los documentos estructurados en XML, RDF y RDF Schema, están provistos de más vocabulario para la descripción de propiedades y clases:

- Relaciones entre clases (disyunción)
- Cardinalidad (“exactamente uno”)
- Igualdad
- Tipificación más rica de propiedades
- Equivalencia
- Características de las propiedades(simetría, transitividad)
- Clases enumeradas

En comparación con otros lenguajes, OWL como lenguaje ontológico aporta varias mejoras a la Web Semántica, entre las que pueden señalarse:

- Capacidad de ser distribuidas a través de varios sistemas.

- Escalable a las necesidades de la Web.
- Compatible con los estándares Web de accesibilidad e internacionalización.
- Abierto y extensible.

Una ontología OWL está compuesta por varios elementos:

- Individuos: representan objetos dentro del dominio de interés.
- Propiedades: relaciones binarias entre individuos.
- Clases: elementos que contienen individuos.

1.4.1. Sub-lenguajes de OWL

OWL se presenta con tres niveles del lenguaje, en función de lo expresivo, lo que permite para sistemas más sencillo, limitar la expresividad buscando una mayor efectividad de cálculo. De menos a más expresivo se encuentran:

- **OWL Lite:** está diseñado para ser utilizado en situaciones donde solamente es necesaria una jerarquía de clasificación y restricciones simples de las clases. Este es el sub-lenguaje más sencillo que posee OWL. Se trata de un lenguaje muy simple, y posee un razonamiento muy eficaz, a cambio se obtiene un nivel de expresión muy limitado. Posee las funcionalidades básicas para la definición de clases y algunos recursos muy limitados para especificar restricciones. En cuanto a la cardinalidad, solo permite restringir la cardinalidad a 0 ó 1.

- **OWL DL:** incluye todas las construcciones de la lengua del OWL, pero pueden ser utilizadas solamente bajo ciertas restricciones. Este tipo de OWL está basado en lógica descriptiva, es más completa que la anterior y su aplicación es más compleja. Este sub-lenguaje ofrece el máximo poder expresivo sin perder computabilidad ni la capacidad de ser decidible.

- **OWL Full:** es la expresividad máxima y la libertad sintáctica de RDF a la vez que permite que una ontología aumente el significado (RDF u OWL) del vocabulario predefinido. Este tipo de OWL ofrece el máximo poder expresivo, pero debe pagar a cambio el alto precio de perder la capacidad

de ser decidible, es decir, no siempre puede dar una respuesta ante cualquier consulta que se realice al modelo.

Cada uno de estos sub-lenguajes es una extensión de su predecesor más simple, cumpliendo con el siguiente conjunto de relaciones:

- Cada ontología OWL Lite es una ontología OWL DL.
- Cada ontología OWL DL es una ontología OWL Full.

Los desarrolladores de ontologías que usen OWL deben decidir qué sub-lenguaje se ajusta mejor a sus necesidades.

OWL es desde febrero de 2004 una recomendación de W3C. En la actualidad los desarrolladores y estudiosos de la Web Semántica están concentrando sus esfuerzos en desarrollar herramientas y sistemas gobernados por este lenguaje. Se asume para la implementación de las geo-ontologías el modelo de conocimiento basado en el lenguaje OWL Full, ya que se puede ver como una extensión de RDF que posee características de modelado suficientes para poder expresar todo un modelo de dominio. Cada documento OWL (Lite, DL, Full) es un documento RDF y todos los documentos RDF son documentos OWL Full.

Conclusiones parciales

El desarrollo de este capítulo permitió conocer las definiciones, componentes, clasificaciones y funciones de las ontologías. Posibilitó conocer el estado actual de los visores de ontologías, las herramientas que pueden emplearse para facilitar la construcción de ontologías y los diferentes lenguajes ontológicos, determinando lo siguiente:

- La ontología se define como un enfoque simplificado del mundo que se desea representar, construido a partir de los conceptos y relaciones identificados.
- Los componentes de una ontología que permiten representar el conocimiento son: conceptos, relaciones, funciones, instancias y axiomas.

- En el campo de los Sistemas de Información Geográfica las ontologías son utilizadas para representar el conocimiento del dominio geográfico.
- Las ontologías se clasifican en ontologías de dominio, ontologías genéricas, ontologías representacionales, ontologías de alto nivel, ontologías de tareas y ontologías de aplicación. Para este trabajo se utilizarán las ontologías de aplicación, puesto que las mismas describen conceptos que dependen tanto de un dominio específico como de una tarea específica, y generalmente son una especialización de ambas.
- Existen otras herramientas empleadas en el diseño de ontologías, entre las que se destacan Protégé y OntoEdit. Ambas utilizan plug-ins para la visualización de ontologías, Protégé utiliza los plug-ins Jambalaya y OWLViz, y OntoEdit el plug-in Visualizador. Para el desarrollo de este trabajo no se seleccionó ninguno de los plug-ins como propuesta para la solución.
- Las ontologías se representan en varios lenguajes tales como XML Schema, RDF, RDF Schema y OWL. Se identificó el lenguaje OWL como el más propicio para la implementación de la geo-ontología teniendo en cuenta sus particularidades.

Capítulo 2: Tendencias y tecnologías actuales

Con el objetivo de crear un software de máxima calidad, desarrollado en el tiempo planificado y que satisfaga las necesidades de la plataforma GeneSIG, así como la de sus usuarios, es preciso hacer un análisis general de las metodologías existentes para el desarrollo del software. En consecuencia con ello, en el presente capítulo se realiza un estudio que posibilite la documentación del proyecto en cuestión, excluyendo todos los riesgos que puedan afectar el desarrollo del mismo. Se referencian las herramientas necesarias para la obtención del producto final deseado y se hace una selección de la metodología más ajustada a la propuesta de solución, teniendo en cuenta las facilidades que puede aportar al trabajo.

2.1. 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 (Lovelle 2005).

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 modelado 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. (González 2001)

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¨ (Meliá 2008):

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 o pasivos. 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 requisitos 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 éstas no influyen directamente en el desarrollo de la solución propuesta en la investigación.

2.2. Proceso Unificado de Desarrollo de Software (RUP)

Plantean los ingenieros Juan Pablo González y Jorge Galves que RUP es un proceso para el desarrollo de un software que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Tiene tres características esenciales, **está dirigido por los Casos de Uso**: que orientan el proyecto a la importancia para el usuario y lo que este quiere, **está centrado en la arquitectura**: que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y **es iterativo e incremental**: donde divide el proyecto en mini-proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada (González y Galves 2007).

Seguidamente hacen referencia a los **6 principios clave que maneja RUP:**

- **Adaptar el proceso:** el proceso deberá adaptarse a las necesidades del cliente puesto que es muy importante interactuar con él. Las características propias del proyecto u organización, el tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto en un área sub-formal.

- **Equilibrar prioridades:** los requisitos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio se podrán corregir desacuerdos que surjan en el futuro.

- **Demostrar valor iterativamente:** los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

- **Colaboración entre equipos:** el desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requisitos, desarrollo, evaluaciones, planes, resultados, etc.

- **Elevar el nivel de abstracción:** este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (framework) por nombrar algunos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.

- **Enfocarse en la calidad:** el control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

RUP como metodología que brinda al equipo de desarrollo una idea de cómo realizar el sistema, presenta cuatro fases en las que se divide el proyecto en curso:

1. Inicio o Conceptualización. (Puesta en marcha)
2. Elaboración. (Definición, análisis y diseño)
3. Construcción. (Implementación)
4. Transición. (Fin del proyecto y puesta en producción)

A continuación se muestra una imagen que representa las cuatro fases por las que transita el software:

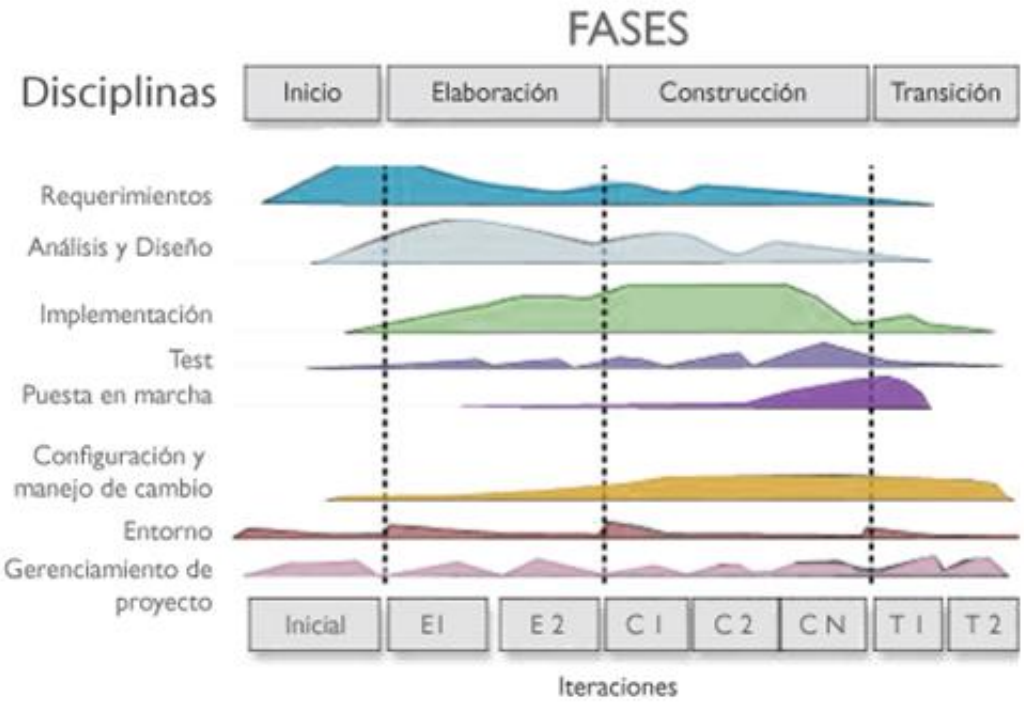


Fig. 4 Esfuerzo en actividades según fase del proyecto.

La presente investigación considera importante mencionar algunas de las ventajas que ofrece RUP:

- Progreso visible en las primeras etapas.
- Temprana retroalimentación que se ajuste a las necesidades reales.
- Mitigación temprana de posibles riesgos altos.
- Conocimiento adquirido en una iteración puede aplicarse de iteración a iteración.
- Gestión de la complejidad.

2.3. 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 (Instituto Nacional de Estadística e Informática., 1999):

- 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.

Visual Paradigm para UML 6.4

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

Visual Paradigm para UML 6.4 es una herramienta UML 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, Corba IDL, 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 WebLogic 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.
- 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.

Visual Paradigm para UML en su versión 6.4, incluye siete nuevas mejoras:

- Importa diagramas de Microsoft Office Visio a Visual Paradigm.
- Soporta patrones de diseño.

- Soporta las tres formas del Diagrama Entidad Relación, conceptual, lógica y física.
- Mejora la trazabilidad de elementos utilizando el historial de revisiones.
- Soporta líneas anidadas.
- Muestra como elemento estereotipado del modelo un ícono de imagen.
- Muestra y oculta elementos del diagrama según se desee.

Visual Paradigm para UML 6.4 es el elegido como software especializado en el manejo del UML. Esta herramienta CASE es una potente plataforma de desarrollo visual compatible con el ciclo completo de una aplicación, combina el modelado con excelentes herramientas para generación de códigos, ingeniería inversa y la interoperabilidad con otras aplicaciones.

2.4. Entornos de Desarrollo Integrado

Un Entorno de Desarrollo Integrado es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. En la actualidad las herramientas que usualmente componen un Entorno de Desarrollo Integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

NetBeans IDE 6.9.1

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[4] 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 proporciona NetBeans se encuentran:

[4] Ambiente de desarrollo integrado

Desarrollo de aplicaciones multiplataforma sobre: MacOS, Windows, Linux.
Add-ons para desarrollo Móvil, desarrollo Web grafico, integración con SOA, optimización de aplicaciones y desarrollo con C y C++.
Cliente CVS integrado.
Crecimiento de plataforma por medio de plug-ins. Entre los plug-ins que existen se tienen los siguientes:

- Herramientas java que sirven para la mejora de desarrollo de aplicaciones.
- Herramientas de modelado UML.
- Herramientas XML.

NetBeans fue elegido como entorno de desarrollo debido a que simplifica las tareas de programación, 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.

2.5. Lenguajes de programación

Es un conjunto de símbolos, caracteres y reglas (programas) que les permiten a las personas comunicarse con la computadora. Los lenguajes de programación tienen un conjunto de instrucciones que permiten realizar operaciones de entrada/salida, cálculo, manipulación de textos, lógica/comparación y almacenamiento/recuperación. Entre ellos pueden señalarse Delphi, Visual Basic, Pascal y Java.

2.5.1. Lenguajes del lado del cliente

Los lenguajes del lado del cliente basan su procesamiento en el cliente web, es decir, que se ejecutan en el navegador del usuario. En este sentido existen varios lenguajes, pero la presente investigación asumirá Javascript para la solución que se propone, lo cual se justifica seguidamente:

Javascript

Javascript es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado

del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único con que cuenta este lenguaje, es el propio navegador. Debido a su compatibilidad con la mayoría de los navegadores modernos, es actualmente el lenguaje más utilizado.

Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, en ocasiones con ligereza. Una de sus ventajas radica en que las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia.

Entre las acciones típicas que se pueden realizar en Javascript se encuentran dos variantes. Por un lado, los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, Javascript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

- ***ExtJS 3.0***

  Actualmente existen múltiples librerías de Javascript que permiten realizar todo tipo de maravillas en el navegador web. ExtJS es una de estas librerías Javascript de alto rendimiento que permite construir aplicaciones complejas haciendo uso de tecnologías como AJAX, DHTML y DOM. Esta librería incluye:

  - Componentes UI del alto performance y personalizables.
  - Modelo de componentes extensibles.
  - Un API fácil de usar.
  - Licencias de códigos abiertos y comerciales.

  ExtJS permite crear Aplicaciones Ricas en Internet (RIA) mediante Javascript, lo que significa que éstas se ejecutan en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las

páginas sin necesidad de recargarlas, aumentando la interactividad, velocidad y usabilidad en las aplicaciones. (García, Fernández y Rodríguez 2009)

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).

El análisis de las funcionalidades de ExtJS como framework ha permitido comprobar las potencialidades de la misma para crear una interfaz de usuario funcional y compatible con la plataforma GeneSIG. Se incorporan a este criterio los beneficios que permite el uso de ExtJS y que a continuación se relacionan:

- **Relación entre Cliente-Servidor balanceado:** se distribuye la carga de procesamiento, permitiendo que el servidor pueda atender más clientes al mismo tiempo.

- **Eficiencia de la red:** disminuye el tráfico en la red pues las aplicaciones cuentan con la posibilidad de elegir que datos desea trasmitir al servidor y viceversa (Criterio este que puede variar con el uso de aplicaciones de pre-carga).

- **Comunicación asíncrona:** en este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

- ***ECOTree***

ECOTree es un componente de Javascript para la confección de árboles jerárquicos que implementa las posiciones de los nodos, permite proporcionarle diferentes tamaños y colores. Los nodos pueden incluir un título, un hipervínculo y metadatos ocultos. Los sub-árboles pueden ser contraídos y expandidos a voluntad. ECOTree permite realizar búsquedas de nodos por el título y los metadatos. Este componente permitirá la elaboración y representación de árboles jerárquicos

que ayudarán a entender un esquema general, así como el grado de diferenciación e integración funcional de los conceptos que componen una ontología.

2.5.2. Lenguajes del lado del servidor

Los lenguajes de lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Existen varios lenguajes, pero solamente se asumen Java y PHP para el desarrollo de la solución que se propone, los cuales se justifican a continuación:

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 interprete 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, 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 o un Dynamic Loading para encontrar las clases.

Java puede funcionar como una aplicación sola o como un applet[5], 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.

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 byte-codes: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software.

- ***Jena 2.6.4***

[5] Componente de una aplicación que se ejecuta en el contexto de otro programa

Es un framework de código abierto para desarrollar aplicaciones en Java con tecnologías de la Web Semántica. Incluye un motor de inferencia basado en reglas, entornos para generar modelos RDF, RDF Schema, 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:
  - RDF Schema
  - OWL
- 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.

PHP

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, y con extensa documentación. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

Está dotado de una extensa librería de funciones, lo que le permite realizar numerosos tipos de aplicaciones web. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por citar algunos ejemplos.

Algunas de las más importantes capacidades de PHP son:

- Compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, ODBC y PostgresSQL.

- Incluye funciones para el envío de correo electrónico, cargar archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

En la solución que se propone PHP forma parte de la arquitectura de GeneSIG, permitiendo la comunicación entre Java y Javascript.

2.6. Servidores Web

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web es el encargado de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo en correspondencia con los comandos solicitados (textos complejos con

enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.). En este punto se considera necesario realizar la siguiente aclaración: mientras que comúnmente se utiliza la palabra servidor para referirse a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones descritas.

Existen varios servidores web importantes, entre los que se destacan Internet Information Services (IIS), Servidor HTTP Apache, Cherokee y Tomcat (también conocido como Jakarta Tomcat o Apache Tomcat).

Servidor HTTP Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. La arquitectura del servidor Apache es muy modular. El servidor consta de una sección core y diversos módulos que aportan muchas de las funcionalidades que podrían considerarse básicas para un servidor web. Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

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:

- UML en su versión 6.4 se eligió como lenguaje de modelado y como metodología estándar para el análisis, implementación y documentación del sistema se escogió RUP. Como herramienta UML que soporta el ciclo de vida completo del desarrollo del software se eligió Visual Paradigm 3.4.

- 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.

- Se seleccionó el lenguaje de programación PHP para consumir desde GeneSIG los servicios del servidor desarrollado en Java.

- Javascript es el lenguaje elegido junto con la librería de componentes ExtJS 3.0 para desarrollar la interfaz web del visor ontológico. También se hará uso del componente Javascript ECOTree que permitirá representar la base de conocimiento de una ontología en forma de árboles jerárquicos.

Capítulo 3: Características del sistema

El desarrollo de un software puede considerarse una acción fácil que solo requiere de un efímero análisis para realizar la programación de un sistema, esto a la vista de un especialista es mucho más complicado, para lograr los objetivos propuestos se necesita de un profundo razonamiento del problema y sus posibles soluciones. El objetivo del presente capítulo es especificar detalladamente los requisitos 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.

3.2. Requisitos funcionales

Los requisitos funcionales describen servicios o funciones:

- ✓ **RF1.** Cargar una ontología.
- ✓ **RF2.** Seleccionar una ontología.
- ✓ **RF3.** Representar una ontología.
- ✓ **RF4.** Operar con la representación.

3.3. Requisitos no-funcionales

Los requisitos No-funcionales son un límite en el sistema o en el proceso de desarrollo. Según Dorfman y Thayer un requisito no funcional es un requisito software que describe no lo que el software hará, sino cómo lo hará, como por ejemplo, requisitos de rendimiento (Dorfman y Thayer 1990). Los requisitos no funcionales son difíciles de verificar/testear, y por ello son evaluados subjetivamente.

Requisitos de usabilidad: estos requisitos 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 lenguaje ontológico OWL.
- ✓ **RNF2.** Debe contar con una interfaz amigable, navegable y sencilla.

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

- ✓ **RNF3.** El sistema debe tener un diseño visual sugestivo, el mismo hará uso de diferentes tonos del color azul permitiendo que el usuario cree un sentimiento de confort y relajación con el sistema, lo que posibilitará una mejor comprensión de la información.

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

- ✓ **RNF4.** La aplicación debe ser compatible con los Sistemas Operativos: Windows 2000 NT, Windows XP y GNU/Linux.

Requisitos de software: se especifican las condiciones o capacidades que el sistema debe cumplir.

- ✓ **RNF5.** Las estaciones de trabajo que harán uso del sistema deberán tener instalado:
  - Windows 2000 NT, Windows XP Profesional ó GNU/Linux en cualquier distribución.
  - Navegador Web Mozilla Firefox.
  - Servidor Web Apache 2.0 o superior.

Requisitos 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.

- ✓ **RNF6.** 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.

- ✓ **RNF7.** 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.

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

- ✓ **RNF8.** 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.

- ✓ **RNF9.** Para la implementación se utilizará IDE NetBeans y la librería de Javascript ExtJS.

3.4. Descripción del sistema propuesto

La propuesta consiste en un software que va a poseer del lado izquierdo una jerarquía de clases y del lado derecho la vista principal. La misma cuenta con un grupo de funcionalidades que ayudarán a la exploración y edición del conocimiento: vista horizontal, vista vertical, expandir nodos y colapsar nodos. Este sistema permitirá al usuario cargar ontologías para poder tener una visión general de las mismas. El visor de ontologías en su primera versión será un sistema autónomo desarrollado con el propósito de aumentar las funcionalidades de la plataforma GeneSIG como un módulo.

3.4.1. Descripción del actor

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 del actor de este sistema:

Actor	Descripción
Usuario	Agente externo que interactúa con el software, es el encargado de realizar todas las acciones para posteriormente adquirir el conocimiento. El mismo debe poseer un conocimiento básico del lenguaje ontológico OWL.

Tabla #1: Descripción del actor del Sistema

3.4.2. Casos de uso del sistema

El caso de uso es un documento narrativo que describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso. Los casos de uso son historias o casos de utilización de un sistema; no son exactamente los requisitos ni las especificaciones funcionales, sino que ejemplifican e incluyen tácticamente los requisitos en las historias que narran (Larman 2004).

3.4.3. Diagrama de casos de uso del sistema

Los diagramas de casos de uso muestran la funcionalidad del sistema desde la perspectiva que tienen los usuarios y lo que el sistema debe hacer para satisfacer los requisitos propuestos. Pueden mostrar el comportamiento de un sistema completo o de una parte. A continuación se muestra el diagrama de casos de uso del sistema y las descripciones textuales de cada uno de los casos de uso.

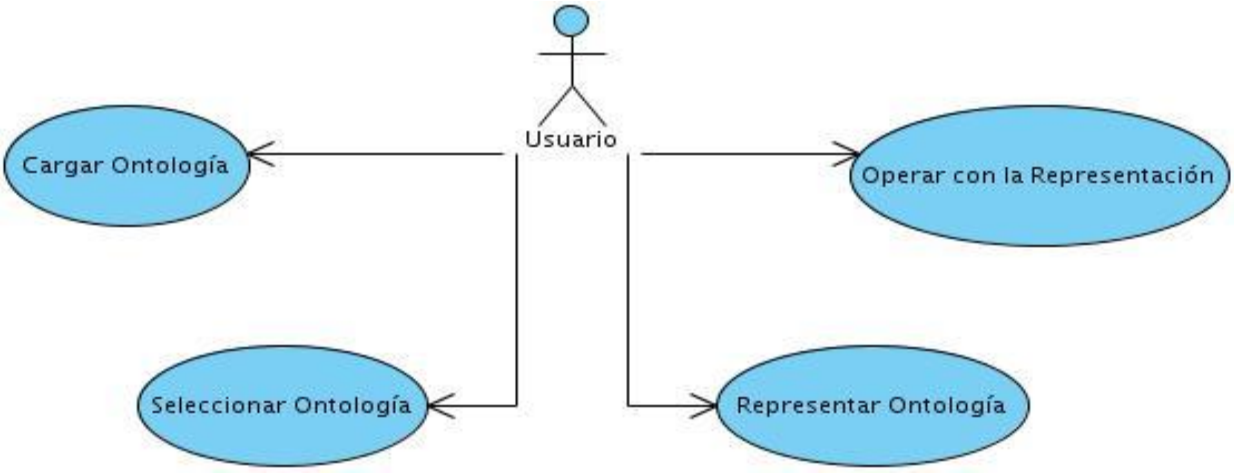


Fig. 5 Diagrama de Casos de Uso del Sistema

3.4.4. Descripción textual de los casos de uso del sistema

Caso de Uso "Cargar una ontología".	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario selecciona del menú principal la opción "Cargar ontología" y aparece la interfaz común de usuario "Subir archivo", donde el actor realiza una búsqueda del fichero, al cargarse, el mismo quedará salvado en la lista desplegable que aparece en la barra de herramientas.
Precondición	Verificar que exista al menos una ontología.
Referencias	Requisito Funcional #1: Cargar una ontología.
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El usuario elige la opción “Cargar 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 la ontología cargada en la lista desplegable de la barra de herramientas.
Flujo Alterno de Eventos	
	2.1.1. Si el sistema detecta que el fichero es corrupto muestra un mensaje de Error.

Tabla #2: Descripción textual del caso de uso “Cargar una ontología”

Caso de Uso “Seleccionar una ontología”.	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario selecciona de la barra de herramientas la opción “Seleccionar ontología”, despliega un combo-box y selecciona del mismo una de las ontologías cargadas, la misma quedarán representadas en el panel “Jerarquía de Estructuras”.
Precondición	Verificar que exista al menos una ontología cargada.
Referencias	Requisito Funcional #2: Seleccionar una ontología.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario elige la opción “Seleccionar ontología”.	1.1. El sistema despliega un combo-box con las ontologías cargadas.
2. El usuario selecciona la ontología deseada.	2.1. El sistema visualiza las clases de la ontología en el panel “Jerarquía de Estructuras”.

Tabla #3: Descripción textual del caso de uso “Seleccionar una ontología”

Caso de Uso “Representar una ontología”.	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario selecciona de la barra de funcionalidades la opción Vista horizontal o Vista vertical, las mismas muestran una visión jerárquica de la base de conocimiento seleccionada.
Precondición	Verificar que se haya seleccionado alguna ontología.
Referencias	Requisito Funcional #2: Representar una ontología.
Prioridad	Crítico
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El usuario puede elegir las opciones: ➢ Vista horizontal. ➢ Vista vertical.	1.1. El sistema realiza la operación según la opción seleccionada por el usuario. ➢ Si seleccionó “Vista horizontal”, ver sección “Vista horizontal”. ➢ Si seleccionó “Vista vertical”, ver sección “Vista vertical”. 1.2. El sistema procesa la información según la acción realizada por el usuario y actualiza la representación en la vista general.
Sección “Vista horizontal”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Vista horizontal”.	1.1. El sistema utiliza un diseño de representación en forma de árbol horizontal para mostrar una visión general de la ontología. 1.2. El sistema procesa la información a partir de la acción realizada por el usuario y actualiza la vista general.
Sección “Vista vertical”	
Acción del Actor	**Respuesta del Sistema**

1. El usuario selecciona la opción “Vista vertical”.	1.1. El sistema utiliza un diseño de representación en forma de árbol vertical para mostrar una visión general de la ontología. 1.2. El sistema procesa la información a partir de la acción realizada por el usuario y actualiza la vista general.

Tabla #4: Descripción textual del caso de uso “Representar una ontología”

Caso de Uso “Operar con la representación”.	
Actor	Usuario
Resumen	Este caso de uso se inicia cuando el usuario selecciona del menú principal las opciones: expandir nodos o colapsar nodos, las cuales permiten expandir o colapsar los nodos de la base de conocimiento visualizada.
Precondición	Verificar que haya sido representada alguna ontología.
Referencias	Requisito Funcional #4: Operar con la representación.
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El usuario puede elegir las opciones: ➢ Expandir nodos. ➢ Colapsar nodos.	1.1. El sistema realiza la operación según la opción seleccionada por el usuario. ➢ Si seleccionó “Expandir nodos”, ver sección “Expandir nodos”. ➢ Si seleccionó “Colapsar nodos”, ver sección “Colapsar nodos”. 1.2. El sistema procesa la información según la acción realizada por el usuario y actualiza la representación en la vista general.
Sección “Expandir nodos”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Expandir nodos”.	1.1. El sistema realiza la operación de “Expandir nodos” seleccionada por el usuario. 1.2. El sistema procesa la información a partir de la acción realizada y actualiza la vista general expandiendo todos los nodos desde la raíz.
Sección “Colapsar nodos”	
Acción del Actor	**Respuesta del Sistema**

1. El usuario selecciona la opción “Colapsar nodos”.	1.1. El sistema realiza la operación de “Colapsar nodos” seleccionada por el usuario. 1.2. El sistema procesa la información a partir de la acción realizada y actualiza la vista general colapsando todos los nodos hacia la raíz.

Tabla #5: Descripción textual del caso de uso “Operar con la representación”

Conclusiones parciales

En este capítulo se describen las características fundamentales del sistema que se propone, con el fin de ofrecer una visión general de su funcionamiento, y mostrando con claridad qué debe hacer y cómo lo debe realizar:

- Se obtuvo un listado de los requisitos funcionales entre los que se encuentran Cargar una ontología, Seleccionar una ontología, Representar una ontología y Operar con la representación.
- 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, requisitos de software y hardware y las restricciones en el diseño y la implementación.
- Se describió la acción y capacidad del actor que va 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.

Capítulo 4: Construcción de la solución propuesta

El presente capítulo está orientado hacia el desarrollo del sistema que se propone. Primeramente 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.

4.1. 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.

El sistema que se propone está soportado bajo una arquitectura distribuida basada en componentes que consume servicios, lo cual brinda altos niveles de escalabilidad y portabilidad, facilitando su mantenimiento, así como el desarrollo de nuevas funciones.

4.2. 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.

4.2.1. 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.

- **Solitario** (Singleton): un programa que instancias múltiples copias, probablemente tenga un error, pero la utilización del patrón **Solitario** 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. Este patrón se pone de manifiesto en la clase **ServerOntoCore** con el fin de garantizar por razón de una única instancia la conexión al **Servidor de Servicios**.

- **Comando** (Command): encapsular un comando en un objeto de tal forma que pueda ser almacenado, pasado a métodos y devuelto igual que cualquier otro objeto. Este patrón se hace notorio en la clase **AjaxHelper,** la cual se encarga de encapsular las peticiones que se hacen al servidor sin la necesidad de conocer el contenido de las mismas.

- **Fachada** (Facade): permitirá simplificar la interfaz del sub-sistema que se propone y que será parte de la plataforma GeneSIG.

4.3. 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 (Larman 2004).

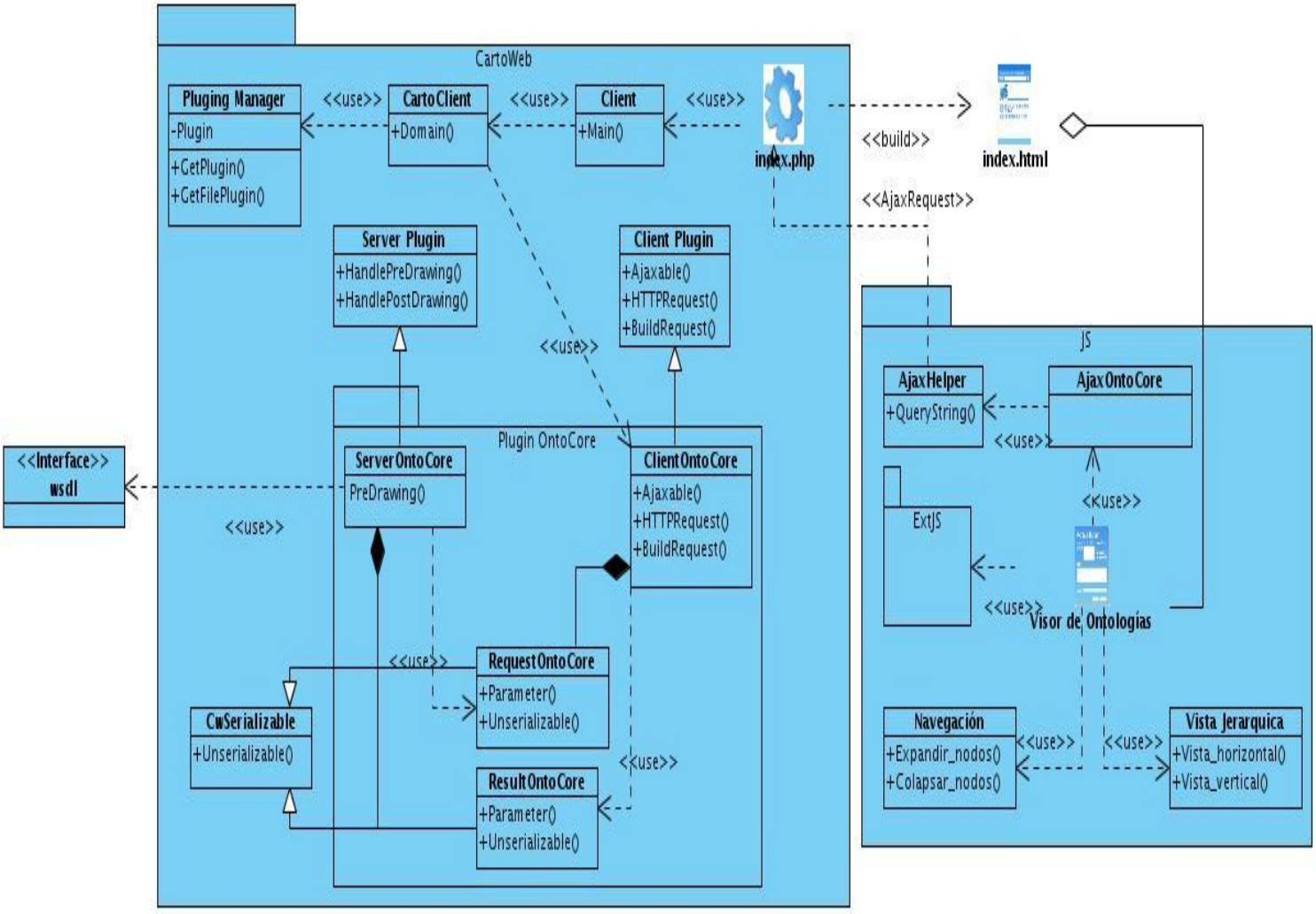


Fig. 5 Diagrama de Clases del OntoClient

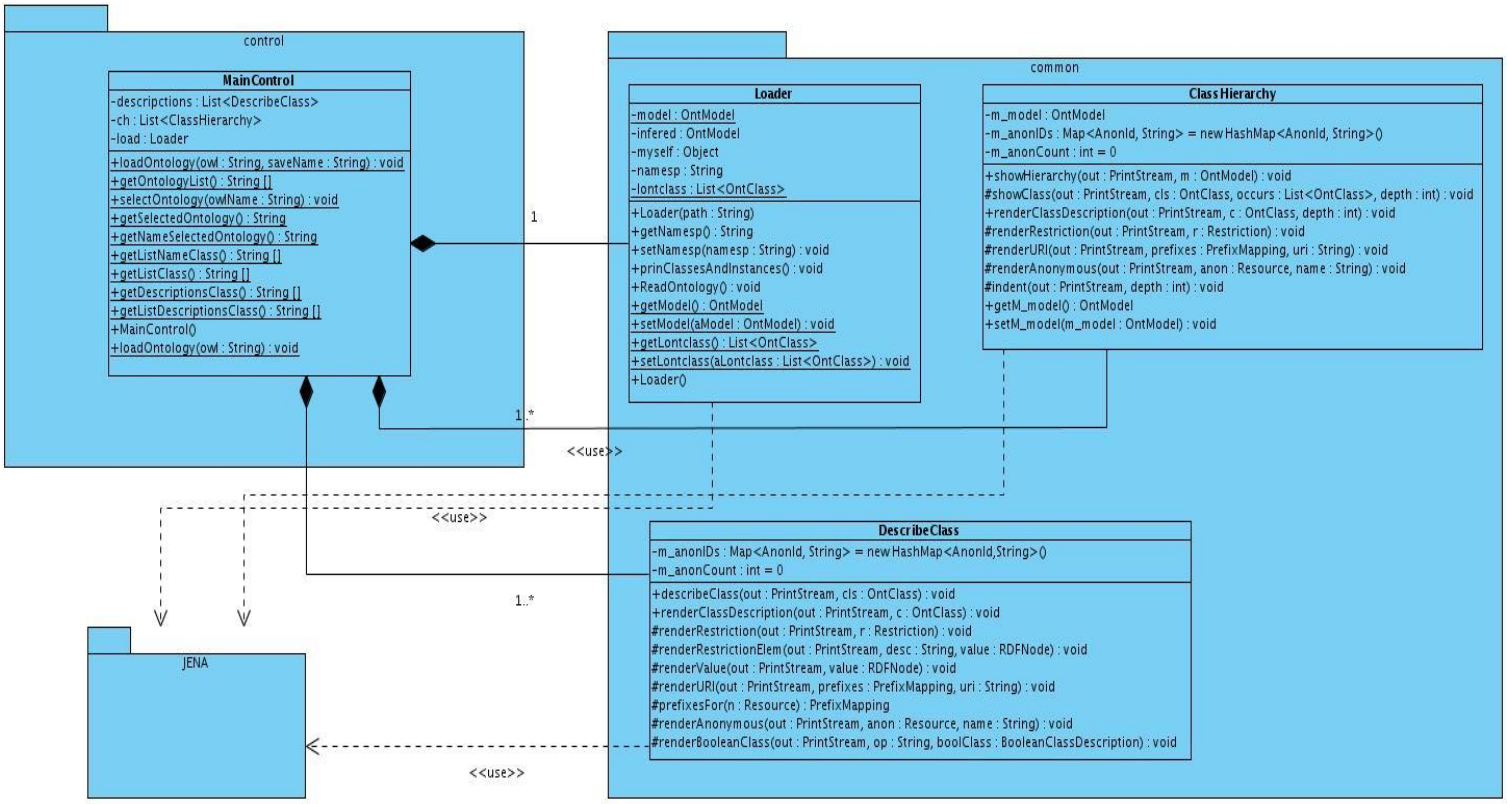


Fig. 6 Diagrama de Clases OntoCore Server

4.3. Principios de diseño

Especificar los principios en el momento de diseñar una aplicación permite que la misma se convierta en una herramienta amigable, atractiva y práctica. El diseño se debe realizar enfocado en lo que el cliente necesita, debido a que el éxito del sistema depende en gran medida de la aprobación de los clientes finales. Estos principios no sólo definen el aspecto estético de la interfaz, sino que facilitan que la misma brinde la información adecuada.

4.3.1. Estándares de la interfaz de la aplicación

Para el adecuado diseño del sistema se definen una serie de estándares orientados a garantizar su fortaleza:

- Brindar una interfaz sencilla y atractiva para todos los usuarios, de manera tal que cualquier persona con conocimientos básicos de informática pueda aprender a trabajar con la aplicación.
- Que el sistema sea capaz de proporcionar avisos eficaces y métodos de respuesta durante y después de la finalización de las tareas.
- Hacer fácil que el usuario utilice los objetos con los que debe interactuar (botones, enlaces, etc.).
- Los menús y etiquetas de botones deben comenzar con la palabra más importante.
- Los eventos importantes del sistema deben ser mostrados en una barra de estado.
- Eliminar la complejidad innecesaria del sistema.
- Minimizar las acciones repetitivas del sistema.
- Que el sistema use diferentes modos para visualizar de manera redundante la información (vista radial, vista jerárquica).

4.4. Diagrama 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 (Larman 2004).

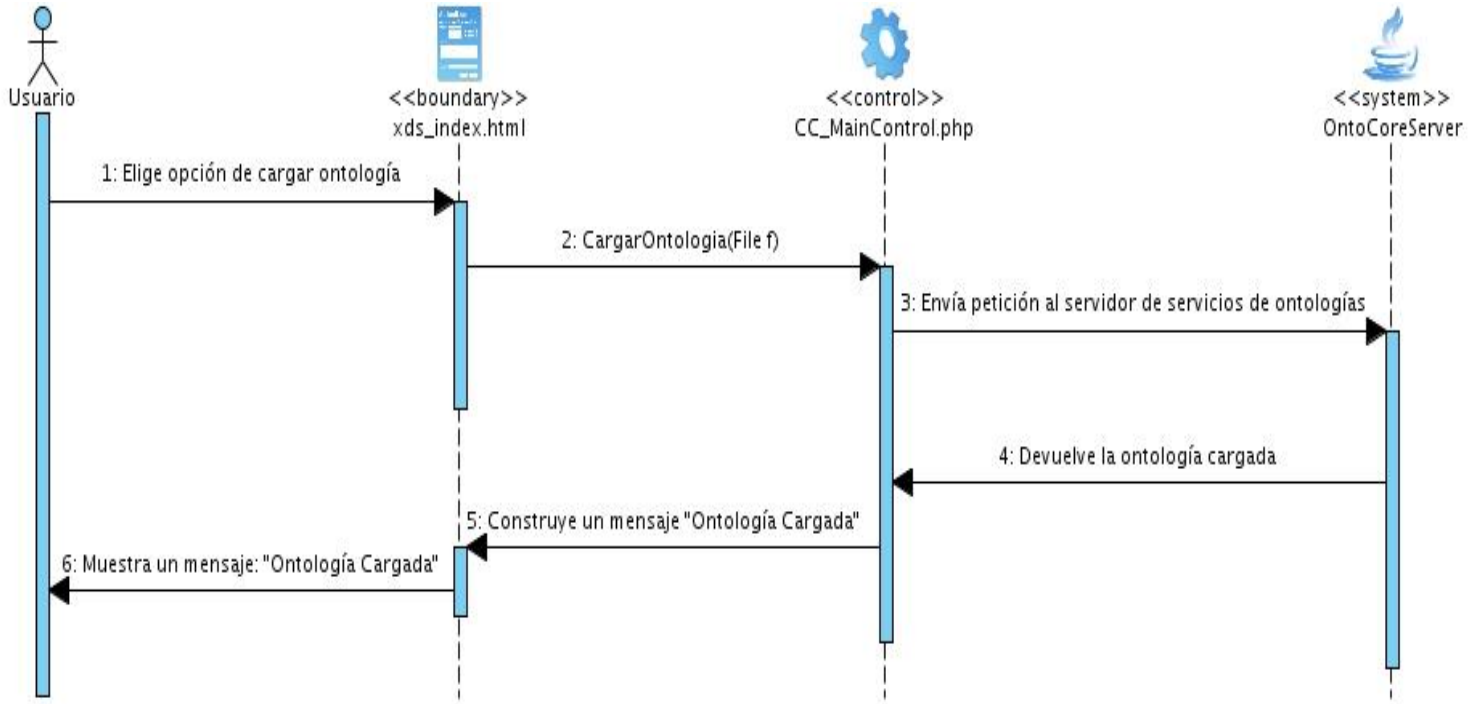


Fig. 7 Diagrama de Secuencia del CU: Cargar Ontología.

4.5. 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:

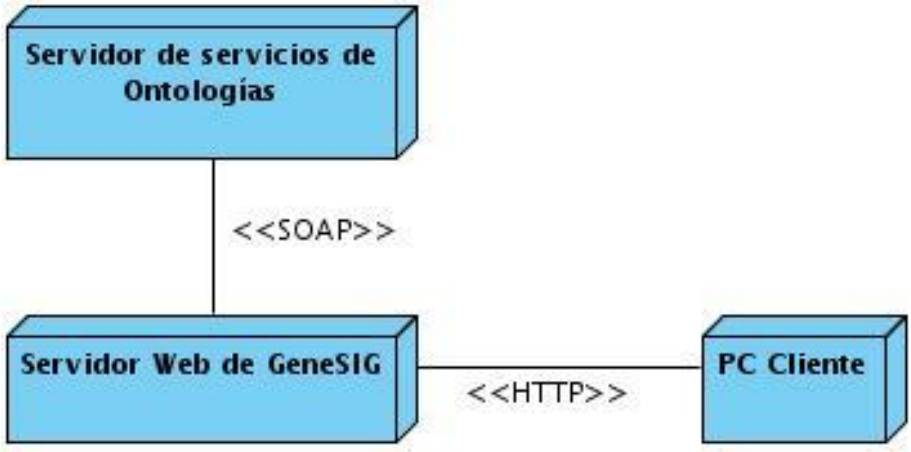


Fig. 8 Diagrama de Despliegue

4.6. 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.

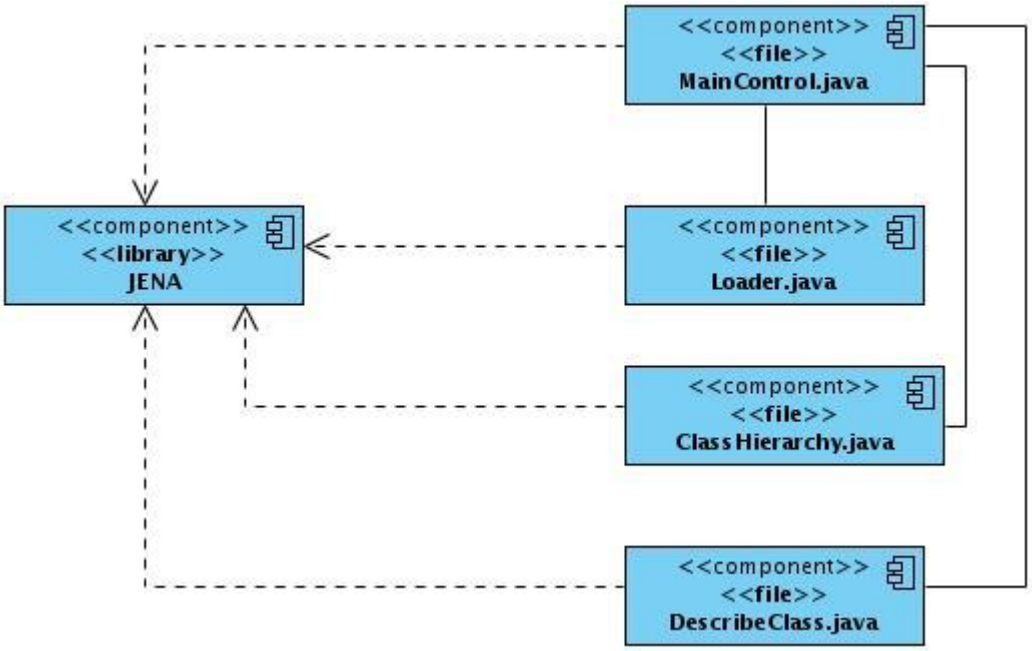


Fig. 9 Diagrama de Componentes del OntoCore Server

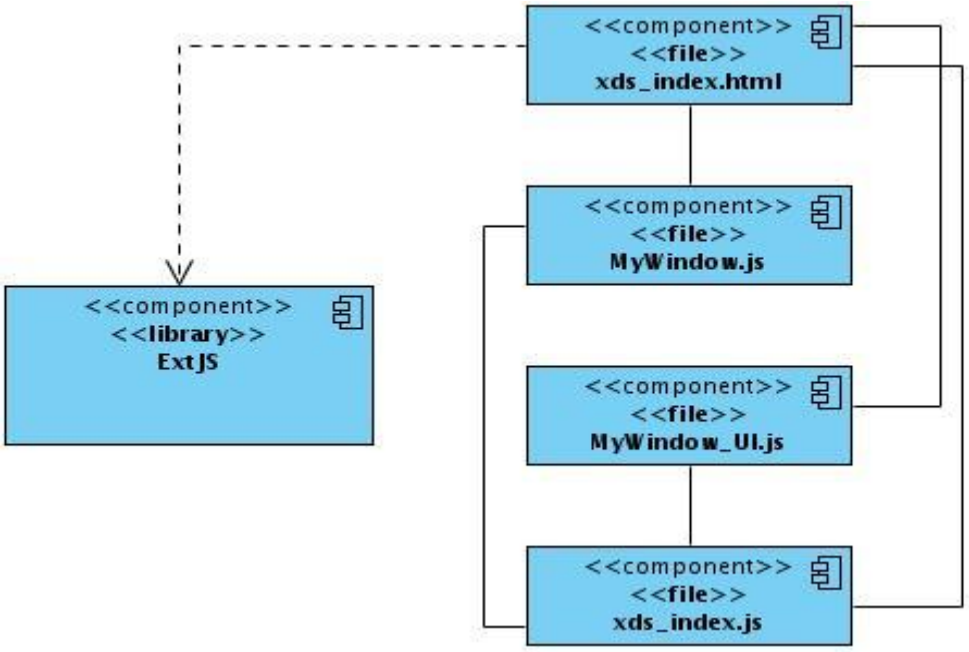


Fig. 10 Diagrama de Componentes del OntoClient

4.7. Sistema de pruebas

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 pasar y viceversa, en resumen, un sistema de pruebas está orientado a detectar.

Para la planeación de prueba que se va a aplicar al sistema, 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, no están basadas en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requisitos 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, o sea, 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 una ontología” y “Seleccionar una ontología” proporcionando unas entradas y estudiando las salidas para ver si son o no las esperadas:

4.7.1. Caso de prueba # 1

1. **Descripción General.**

El caso de uso se inicia cuando el usuario selecciona del menú **Acciones** que aparece en la barra de herramientas, la opción **Cargar ontología**, al mismo tiempo aparece una ventana con un campo de texto para el nombre, un campo para la carga de ficheros y un botón **Salvar** que permitirá salvar la ontología. El usuario debe elegir el nombre con el que salvará la ontología, subir el fichero que desee y oprimir el botón **Salvar** para llevar a cabo la acción, el caso de uso termina cuando el sistema muestra un mensaje de confirmación.

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 ontología".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Cargar ontología	EC 1.1: Cargar fichero.	El sistema muestra la interfaz común de usuario "Subir archivo".
	EC 1.2: Salvar fichero.	El sistema muestra mensaje de confirmación.
	EC 1.3: Salvar fichero erróneo.	El sistema muestra mensaje de error.
	EC 1.4: Salvar sin llenar los campos.	El sistema señala los campos vacíos.

4. Descripción de variable.

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	Si	Se tiene que escribir un nombre.
2	Ontología	Campo de subida de archivos	Si	Se tiene que subir un archivo.

5. Matriz de Datos

SC 1 Cargar Ontología

Escenario	Variable 1 Nombre	Variable 2 Ontología	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.1: Cargar fichero.	NA	NA	El sistema muestra la interfaz común de usuario “Subir archivo”.		• Seleccionar la opción “Subir archivo”.
EC 1.2: Salvar fichero.	V (Camera)	V (Cam.owl)	El sistema muestra mensaje de confirmación.		• Seleccionar la opción “Subir archivo”. • Seleccionar **salvar** archivo. • Mostrar mensaje de confirmación.
EC 1.3: Salvar fichero erróneo.	V (Camera)	V (Cam.txt)	El sistema muestra mensaje de error.		• Seleccionar la opción “Subir archivo”. • Seleccionar **salvar** archivo. • Mostrar mensaje de error.
EC 1.4: Salvar	NA	NA	El sistema señala		• Seleccionar **salvar**

sin llenar los campos.			los campos vacíos.		archivo. • Los campos se señalan indicando que se encuentran vacíos.

4.7.2. Caso de prueba # 2

1. **Descripción General.**

El caso de uso se inicia cuando el usuario selecciona de la lista desplegable que aparece en la barra de herramientas la ontología que desea representar, el caso de uso termina cuando el sistema representa en el panel “Jerarquía de Estructuras” la ontología seleccionada.

2. **Condiciones de Ejecución.**

   - Debe haberse cargado anteriormente alguna ontología.

3. **Secciones a probar en el Caso de Uso “Seleccionar ontología”.**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar ontología.	EC 1.1: Desplegar la lista de ontologías.	El sistema muestra la lista con las ontologías cargadas.
	EC 1.2: Seleccionar una ontología.	El sistema muestra en el panel “Jerarquía de Estructuras” la base de conocimientos de la ontología seleccionada.

4. **Descripción de variable.**

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Seleccionar ontología	Lista desplegable	Si	Se debe seleccionar una ontología de la lista.

5. Matriz de Datos

SC 1 Seleccionar Ontología

Escenario	Variable 1 Seleccionar ontología	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.1: Desplegar la lista de ontologías.	NA	El sistema muestra la lista con las ontologías cargadas.		• Seleccionar la ontología deseada.
EC 1.2: Seleccionar una ontología.	V	El sistema muestra en el panel “Jerarquía de Estructuras” la base de conocimientos de la ontología seleccionada.		• Seleccionar la ontología deseada.

Conclusiones parciales

En este capítulo se obtuvo 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 visor de geo-ontologías.
- Para el desarrollo de la aplicación se utilizó una arquitectura basada en componentes y que consume servicios, al mismo tiempo que se emplearon los patrones de diseño Solitario, Comando y Fachada.
- Entre los principios de diseño a tener en cuenta para el desarrollo de la interfaz de usuario del sistema se determinaron: garantizar la legibilidad de manera que la presentación visual sea clara para el usuario, lograr una interfaz consistente y dar al usuario un ambiente flexible para que pueda aprender rápidamente a usar la aplicación.
- 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, los resultados arrojados a partir de las pruebas efectuadas fueron satisfactorios.
- Se logró desarrollar un sistema amigable, eficaz y fácil de mantener.

Conclusiones generales

- Para la construcción del sistema fue necesario realizar un estudio sobre las diferentes herramientas que se utilizan para el diseño, la administración y la visualización de ontologías, lo que permitió identificar los requisitos funcionales que posee el sistema.

- La interfaz de usuario del sistema se desarrolló utilizando el lenguaje Javascript y la librería ExtJS 3.0, para desarrollar el servidor de servicios que permitió interpretar el lenguaje OWL se utilizó el lenguaje Java y la librería de Jena 2.6.4, y para consumir desde GeneSIG los servicios que ofrece el servidor se seleccionó el lenguaje de programación PHP.

- El modelado del ciclo de desarrollo del software facilitó la visualización del sistema que se construyó, y conllevó a que las actividades planificadas fueran orientadas hacia la calidad.

- Se logró el desarrollo de un sistema capaz de realizar representaciones conceptuales de una ontología.

- 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.

- Se le realizaron pruebas de caja negra al sistema con el fin de encontrar errores en las siguientes categorías:

  - ✓ Funciones incorrectas.
  - ✓ Errores de interfaz.
  - ✓ Errores de rendimiento.
  - ✓ Errores de inicialización y de terminación.

Los resultados obtenidos a partir de las pruebas realizadas fueron satisfactorios, se logró desarrollar un sistema eficaz y amigable donde el usuario sin tener que consultar un manual o ayuda puede interactuar con el mismo. La aplicación se logró en un 100% a partir de lo concebido.

- La herramienta desarrollada constituye un aporte importante y significativo porque permite aumentar las funcionalidades de la plataforma GeneSIG, tiene el poder de visualizar una jerarquía de conceptos como un modelo genérico y demostrativo con el fin de obtener una idea uniforme de las ontologías, cuenta también con una interfaz sencilla e intuitiva.

Recomendaciones

- Investigar sobre diferentes funcionalidades existentes en otras herramientas consultadas, con el objetivo de enriquecer y fortalecer el visor de ontologías que se propone en el presente trabajo investigativo.
- Perfeccionar la herramienta desarrollada, tomando en consideración las experiencias que se obtengan durante el transcurso de su puesta en práctica.

Bibliografía referenciada

Avián, M. Á. 2005. "Ontologías: qué son y para qué sirven. 2005."

CODE. (2004). "CODE: Cmap Ontology Development Environment." Consultado en: www.ihmc.us/users/phayes/CODE/index.html.

Dorfman, M. y Thayer, R. 1990. "Standards, Guidelines and Examples on System and Software Requirements Engineering", IEEE Computer Society Press.

Evermann, J. (2008). "A UML and OWL Description of Bunge's Upper-level Ontology Model. Software and System Modeling."

Gangemi, A., Pisanelli, D. y Steve, G. 1998. Ontology Alignment: Experiences With Medical Terminologies. Formal Ontology in Information System. N. Guarino.

García, R, Fernández, E y Rodríguez, D. 2009. Plataforma para Formación de Investigadores a Distancia. Universidad de Buenos Aires. Consultado en: http://laboratorios.fi.uba.ar/lsi/donofrio-uminsky-trabajoprofesional.pdf 21 de febrero de 2011

González, J. E. 2001. El Lenguaje de Modelado Unificado (UML) Consultado en: http://www.docirs.cl/uml.htm 12 de febrero de 2011

González, J. P y Galves, J. 2007. Fundamentos de la metodología RUP. Consultado en: http://www.monografias.com/trabajos-pdf4/ensayo-sobrte-rup/ensayo-sobrte-rup.pdf 18 de febrero de 2011

Gruber, T. R. (1993). "A translation approach to portable ontology specifications. Knowledge Acquisition 5."

Guarino, N. 1998. "¨Formal ontology in information systems¨. Proceedings of FOIS 98. IOS Press."

Gutiérrez, C. 1997. "La lógica y el conocimiento." Consultado en: http://cariari.ucr.ac.cr/~claudiog/La_logica_y_el_conocimiento.html.

Herramientas Case. Colección Cultura Informática. 1999. Consultado en: http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

Pérez, J. R. 2006. Clasificación de Usuarios Basada en la Detección de Errores Usando Técnicas de Procesadores de Lenguajes. Tesis Doctoral. Universidad de Oviedo. Consultado en: http://www.di.uniovi.es/~juanrp/investigacion/tesis/_Tesis_SICODE.pdf. 20 de febrero de 2011

Knublauch, H. M., M. 2006. Editing Description Logic Ontologies with the Protégé Owl Plugin. .

Larman C. 2004. UML y Patrones, Introducción al análisis y diseño orientado a objetos.

Lendinez S. Ingeniería Técnica en Informática de Gestión Proyecto Fin de Carrera APO: Aplicación para la Población de Ontologías.

Llano, F. V. V. y. E. G. 2009. "Alineamiento de ontologías en el dominio geospacial."

Lovelle, S. 2005. Perfil para representar una arquitectura de componentes en UML. CUJAE, CUBA: s.n.

Maroto, M. d. l. N. 2007. "Las relaciones conceptuales en la terminología de los productos cerámicos y su formalización mediante un editor de ontologías."

Meliá, S. 2008. WebSA: Un Método de Desarrollo Dirigido por Modelos de Arquitectura para Aplicaciones Web. Tesis Doctoral. Consultado en: http://www.dlsi.ua.es/~santi/papers/thesis_definitiva.PDF 19 de marzo

Muñoz, J. R. 2004. Metodología para la incorporación de medidas de seguridad en sistemas de información de gran implantación. Tesis doctoral. Consultado en: http://oa.upm.es/323/1/09200430.pdf 21 de febrero de 2011

OWLLite. Consultado en: http://www.w3.org/2007/09/OWL-Overview-es.html.

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

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

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

W3C (Editores) Mc Guinness, D. L., van Harmelen, F. (Editores). 2004. ""OWL Web Ontology Language: Overview"."

Bibliografía consultada

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 Móvil Basado en Comunicaciones Inalámbricas y 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.

ExtJS Framework. Consultado en: http://groupbit.com/curso-extjs-framework.

Fales Simón, Ricardo. Lenguajes ontológicos "Semántica de Servicios Web". Consultado en: http://www.slidefinder.net/l/lenguajes_ontol%C3%B3gicos_sem%C3%A1ntica_servicios_web/8020420.

Fernández Breis, Jesualdo Tomás. Título: Un entorno de integración de ontologías para el desarrollo de sistemas de gestión del conocimiento. 2003. Consultado en: http://www.tesisenred.net/handle/10803/10921.

Gil Peña, Jorge. Herramienta para construir bases de conocimiento a partir de información en la web. Madrid, Septiembre 2010.

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.

Macías Iglesias, José Antonio. TESIS DOCTORAL: "Autoría de Documentos Web Dinámicos Mediante Ontologías y Técnicas de Programación por Demostración". Junio de 2003.

Margaret-Anne Storey; Robert Lintern, Neil Ernst y David Perrin. Visualization and Protégé.

Modelando ontologías con subclases. Abril 21, 2009. Consultado en: http://tesis-e.blogspot.com/2009/04/modelando-ontologias-con-subclases.html.

Ontologías en la Web Semántica. Septiembre de 2010. Consultado en: http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf.

Peis Redondo, Eduardo; Hassan Montero, Yusef. Ontologías, metadatos y agentes: recuperación "semántica" de la información. Universidad de Granada – España.

Segura Maroto, Aurelio. Desarrollo de una aplicación Web 2.0 para la captura y análisis de la valoración del usuario. Septiembre, 2010.

Samper Zapater, José Javier. Ontologías para Servicios Web Semánticos de Información de Tráfico: Descripción y Herramientas de Explotación. Universidad de Valencia, 2005.

Sarabia López, Gerardo. "Búsqueda y Ponderación de Información Contenida en Bases de Datos Espaciales, Utilizando Jerarquías". Diciembre, 2008.

Silva, Dr. José Luis Batista. Aplicación de sistemas de información geográfica en Cuba. 11 de 2005. Consultado en: http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1051.