UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
DEPARTAMENTO COMPONENTES

Plugin para la extracción de Información Semántica de Objetos Móviles para GeneSIG

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

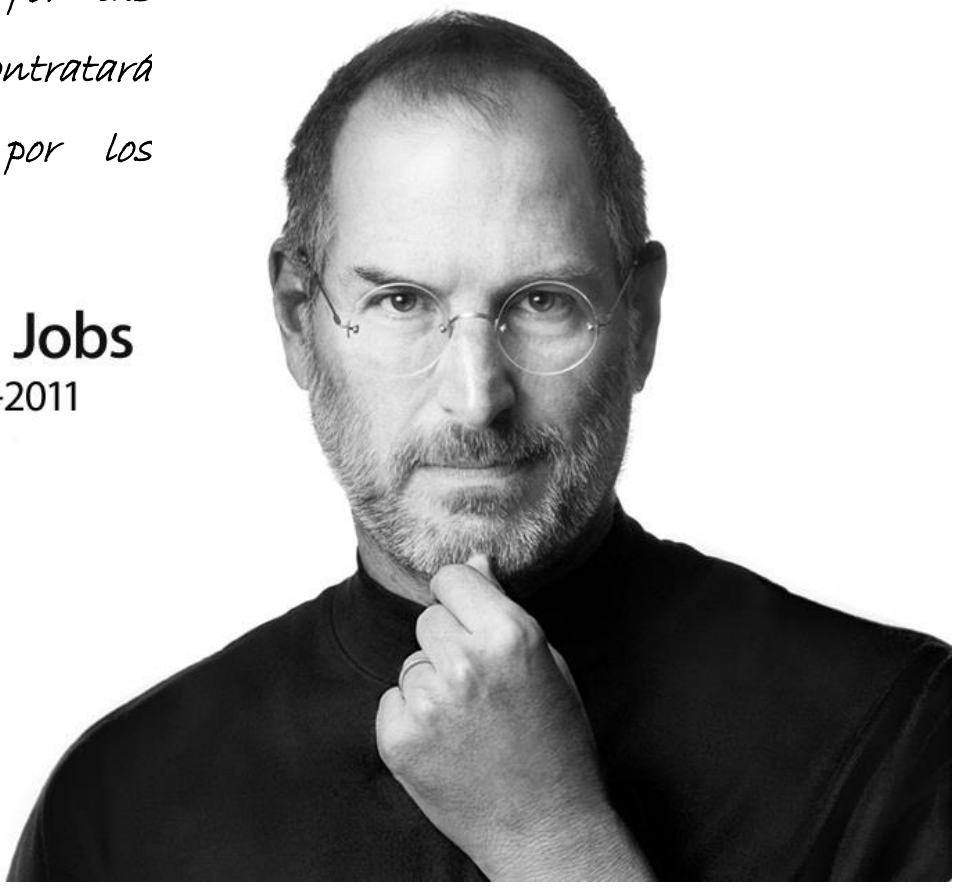
Autor: Elizabeth Rey García

Tutor: MsC. Yuniel Eliades Proenza Arias

La Habana, Junio del 2015

"Si tú no trabajas por tus sueños, alguien te contratará para que trabajes por los suyos...."

Steve Jobs
1955-2011



Declaración de autoría

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

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

_____

Elizabeth Rey García

Autor

_____

MsC. Yuniel Eliades Proenza Arias

Tutor

Datos de contacto

Autor: Elizabeth Rey García

Edad: 23.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniera Informática.

Síntesis del Autor: Graduado de la vocacional Vladimir Ilich Lenin en el año 2010 y graduado de Ingeniero Informático en el año 2015. Se ha desempeñado como estudiante destacado durante su carrera.

E-mail: erey@estudiantes.uci.cu.

Tutor: MsC. Yuniel Eliades Proenza Arias.

Edad: 33.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero Informático, Máster en Ingeniería de Software e Inteligencia Artificial.

Síntesis del Tutor: Graduado de Ingeniería Informática en el año 2006 de la Universidad de Holguín y CUJAE. Máster en Ingeniería de Software e Inteligencia Artificial por la Universidad de Málaga en el 2011. Se ha desempeñado como analista y desarrollador de aplicaciones Web y Desktop. Tiene experiencia en el trabajo con Sistemas de Información Geográfica y el desarrollo de Ontologías.

E-mail: yproenza@uci.cu.

Agradecimientos

Agradecimiento especial a mi madre que me ha dado todo.

Agradecimiento a mi novio por todo su apoyo, amor y comprensión.

Agradecimiento especial a Gretel por su apoyo.

Agradecimiento especial a Ariel por toda su paciencia y su sabiduría.

Agradecimiento a Leiny por todos sus consejos.

Agradecimiento a mi tutor por la confianza depositada.

Resumen

GeneSIG es una plataforma diseñada para el montaje de Sistemas de Información Geográficos que, a pesar de sus múltiples funcionalidades, no es capaz de realizar un análisis semántico de la información que permita incrementar la eficiencia en la búsqueda de contenido geo-referenciado. Para lograr este propósito es necesario hacer uso de una ontología como una base de conocimientos que logre modelar los conceptos y relaciones del dominio y posteriormente poblarla de acuerdo a la información almacenada en el sistema. Esta acción de insertar instancias actualmente solo puede realizarse de forma manual haciendo de este un proceso largo y complejo y dando lugar a la entrada de datos erróneos y la pérdida de información. El objetivo del presente trabajo es desarrollar un Plugin para la extracción de información semántica de objetos móviles para GeneSIG que permita mejorar dicho proceso haciendo uso de OntoMov, primera ontología dedicada a la representación de objetos en movimiento, en un Sistemas de Información Geográfico especializado en el análisis de la trayectoria y nivel de afectaciones de los huracanes en Cuba. Para ello se emplean Prodesoft, como proceso de desarrollo, los lenguajes del lado del servidor Java y PHP, el sistema gestor de base de datos PostgreSQL y OWL-API, para el manejo directo con la ontología. Como resultado principal se obtiene un Plugin capaz de poblar la ontología con los datos más significativos de un huracán dado y sus principales relaciones espaciales con la Isla que permitiría mejorar la búsqueda y recuperación de información dentro del sistema.

Palabras Clave: ONTOMOV; Ontología; Sistema de Información Geográfica.

Abstract

GeneSIG is a platform designed for mounting Geographic Information Systems that, despite its many features, it is currently unable to perform a semantic analysis of information that allows to increase efficiency in the search for geo-referenced content. For this purpose it is necessary to use an ontology as a knowledge base that achieves to modeling domain concepts and relationships and then populate it according to the information stored in the system. This action of inserting instances currently can only be performed manually making of this a long and complex process resulting in erroneous data entry and data loss. The goal of this work is to develop a Plugin for the extraction of semantic information of moving objects for GeneSIG that improves this process using OntoMov, first ontology dedicated to the representation of moving objects in a Geographic Information System specialized in the analysis of the trajectory and level of damages of hurricanes in Cuba. It is used Prodesoft as a software development process, side to Java and PHP server languages, PostgreSQL as a System Manager Database and OWL-API, for the direct operations with the ontology. As a main result it is obtained a Plugin able to populate the ontology with the most significant data of a given hurricane and its main relations with the island that would improve information retrieval within the system.

Keywords: ONTOMOV; ontology; Geographic Information System.

Índice General

Índice de Tablas

Índice de Figuras

Introducción

Los Sistema de Información Geográfica (SIG) son sistemas de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados, para la solución de los problemas complejos del manejo y planeamiento territorial (Candeaux Duffatt, y otros, 1994). Surgieron como resultado de la necesidad de disponer rápidamente de información para resolver problemas en un contexto real; y en un corto tiempo han alcanzado límites inesperados en los campos de aplicación e investigación convirtiéndose en una potente herramienta para la toma de decisiones dentro del entorno empresarial.

Debido al avance de las tecnologías y herramientas de la Web Semántica, los SIG han sido beneficiados con la inclusión cada vez más frecuente de geo-ontologías convirtiéndolos en Sistemas de Información Geográficos Gobernados por Ontologías (SIGGO). Una ontología puede considerarse una especificación formal de una conceptualización común descrita en un lenguaje que contiene los conceptos, propiedades y relaciones más relevantes (Gruber, 1993). Una vez poblada la ontología con instancias de conceptos y relaciones como abstracciones de entidades de la vida real, se puede considerar que se ha creado una base de conocimientos a partir de la cual se realiza el procesamiento de la información semántica. Esto facilita su legibilidad por parte de las máquinas logrando centrarse en el significado de los datos, en lugar de su sintaxis o estructura, agilizando la búsqueda de información.

En el Centro de Desarrollo de Geo-informática y Señales Digitales (GEYSED) ubicado en la Facultad 6 de la Universidad de las Ciencias Informáticas (UCI) se encuentra el proyecto GeneSIG. Este a su vez es una plataforma encaminada a realizar la representación y análisis geoespacial de información geográfica donde su estructura arquitectónica permite personalizar sus funcionalidades a cualquier negocio que lo requiera a través de la reutilización de sus componentes (Pantoja Zaldívar, y otros, 2010). Esto posibilita la creación de SIG que integrados a la sociedad constituyen un activo de gran relevancia dentro de las empresas e instituciones ya que gran parte de la información tratada está relacionada con localizaciones geográficas o coordenadas espaciales. Sin embargo, hasta el momento no cuenta con mecanismos que le permita realizar un análisis semántico de ningún tipo con los datos que maneja haciendo esquemático el proceso de búsqueda de información geográfica.

El trabajo con los SIGGO en el proyecto como fase superior a los SIG permitiría crear una base de conocimientos a través de la información extraída del sistema que le otorgaría un nivel de razonamiento e

inferencia capaz de arribar a conclusiones sobre los distintos campos de aplicación. Uno de estos campos perteneciente al trabajo con SIG lo constituye el análisis de cuerpos espaciales en movimiento enfocados en la trayectoria y grado de afectación de los huracanes sobre la isla. La incorporación de un procesamiento semántico a través de geo-ontologías al sistema otorgaría un alto valor añadido, por cuanto sería capaz de buscar elementos que no están directamente asociados a un parámetro espacial puramente, sino a una semántica espacial y permitiría asimilar la información almacenada y definir la magnitud de las afectaciones en las distintas provincias, las relaciones espaciales que se aplican, entre otras.

Sin embargo, a pesar de contar con los recursos necesarios para el desarrollo del proyecto, la limitación principal se encuentra en las herramientas de creación de ontologías, ya que no están preparadas para su poblado automático y debido a la enorme cantidad de datos geográficos almacenados en la base de datos espacial del proyecto, se hace sumamente complejo el proceso de creación de la base de conocimientos. El desarrollador se vería obligado a añadir de forma manual cada una de las instancias del dominio dentro de la ontología resultando en un proceso lento donde el factor humano presente da pie a la entrada de datos erróneos.

Además, la extracción del conocimiento a insertar dentro de la ontología surge después de un análisis de los datos geográficos almacenados y su posterior transformación en información sensible dentro del dominio. Este proceso usualmente se realiza dependiendo de la interpretación del especialista extrapolada en un algoritmo que defina las instancias de los conceptos y las relaciones dentro de la ontología que actualmente el sistema no posee, lo cual disminuye considerablemente la credibilidad de los resultados obtenidos. Teniendo en cuenta que una medida inadecuada que se tome ante la incidencia de huracanes sobre la isla afecta directamente a la sociedad, y esta podría estar determinada en gran parte por la calidad, exactitud y actualidad del conocimiento brindado por dicho SIGGO resulta inconcebible la existencia de un mal manejo de la misma.

A partir de la situación descrita anteriormente se ha identificado el siguiente **problema de investigación**: ¿Cómo mejorar el proceso de poblado de la geo-ontología asociada a un SIGGO especializado en el análisis de la trayectoria y nivel de afectaciones de los huracanes en Cuba?

Para darle solución al problema planteado se tiene como **objetivo general:** Desarrollar un Plugin para la extracción de información semántica de objetos móviles para GeneSIG a partir de datos geográficos de huracanes. Con este fin se determinó como **objeto de estudio de la investigación:** El proceso de

manipulación de geo-ontologías como forma de representación de la información semántica asociada al dominio espacio-temporal dentro de los SIG enfocado en el siguiente **campo de acción**: El poblado con individuos de geo-ontologías en el dominio de objetos móviles.

La **hipótesis** a defender plantea que: El desarrollo de un Plugin para la extracción de información semántica de objetos móviles para GeneSIG a partir de datos geográficos de huracanes permite la disminución de errores y del tiempo empleado en el proceso de poblado de su geo-ontología asociada.

Para dar cumplimiento al objetivo general del presente trabajo las **tareas de la investigación** que se proponen son las siguientes:

1. Analizar los conceptos fundamentales y las principales soluciones existentes referentes al poblado de las ontologías para validar la relevancia y la pertinencia de la investigación.
2. Seleccionar las tecnologías, herramientas y la metodología de desarrollo de software que se adapten mejor a las condiciones específicas del proyecto.
3. Definir los requisitos funcionales y no funcionales que debe poseer el Plugin a desarrollar para satisfacer las necesidades existentes.
4. Modelar el Plugin haciendo uso de las normas establecidas en el proyecto para dejar sentadas las bases para su posterior desarrollo.
5. Validar el funcionamiento correcto del Plugin desarrollado para garantizar el total cumplimiento de los requisitos del proyecto y corroborar la hipótesis planteada.

A lo largo de la investigación fueron empleados los siguientes **métodos teóricos**:

Analítico – Sintético: Se aplicó al realizar un estudio de las principales tecnologías existentes que permiten el manejo de ontologías con el fin de sintetizar sus principales funcionalidades y ventajas que puedan ser aplicadas al desarrollo del Plugin.

Histórico – Lógico: Se utilizó para realizar un estado del arte de la problemática analizada con el fin de conocer si existe alguna tecnología o sistema que permita automatizar el poblado con individuos de una geo-ontología especializada en la representación de objetos móviles.

Hipotético-Deductivo: Se emplea para la confirmación o no de la hipótesis planteada contrastando los resultados alcanzados a través de las pruebas a realizar en la evaluación del Plugin para la extracción de información semántica de objetos móviles para GeneSIG.

Se emplea además la tormenta de ideas como **técnica** para el levantamiento de requisitos funcionales y no funcionales.

La presente investigación científica cuenta con una estructura de 4 capítulos que pueden ser descritos de la siguiente forma:

Capítulo #1: Se realiza una descripción más abarcadora de la situación problemática y se analizan los principales conceptos que permitan una mejor comprensión de la investigación a realizar, así como las soluciones existentes asociadas al poblado con individuos de geo-ontologías en el dominio de objetos móviles.

Capítulo #2: Se analizan las tecnologías y las herramientas que mejor se ajustan a la propuesta de solución describiendo brevemente sus funcionalidades y ventajas en su utilización, así como también se elige la metodología adecuada para dirigir el proceso de desarrollo de software.

Capítulo #3: Se realiza el modelado y el levantamiento de los requisitos funcionales y no funcionales del Plugin y a partir de los mismos se procede al diseño de la arquitectura y al diseño detallado de las clases generando el modelo de dominio, el diagrama de componentes, el diagrama de despliegue y los diagramas de clases del diseño que sientan las bases para las futuras etapas de implementación y prueba.

Capítulo #4: Se explica el Plugin resultante haciendo énfasis en las funcionalidades implementadas, se realizan las pruebas de software que permitan garantizar el cumplimiento de los requisitos de la investigación y se corrobora la hipótesis planteada.

Posteriormente se presentan las conclusiones generales, recomendaciones y referencias bibliográficas utilizadas en el trabajo.

Capítulo 1: Fundamentación Teórica del Plugin para la extracción de Información Semántica de Objetos Móviles para GeneSIG

1.1. Introducción

En este capítulo se realiza un estudio de las principales definiciones de ontología, geo-ontología, SIG y SIGGO como conceptos necesarios para comprender el objeto de estudio de la investigación y se realiza un análisis del estado actual del proceso de poblado de ontologías enfocado en los sistemas existentes que manejan ontologías como bases de conocimiento.

1.2. Conceptos asociados al dominio del problema

Resulta necesario conocer los conceptos relacionados al poblado con individuos de geo-ontologías como son: SIG, SIGGO, Ontologías y Geo-ontología ya que facilitan la comprensión del objeto de estudio de la investigación.

Un **SIG** ha sido definido por diversos autores como:

"Sistemas de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados, para la solución de los problemas complejos del manejo y planeamiento territorial" (Candeaux Duffatt, y otros, 1994).

"Un SIG constituye un conjunto de herramientas para seleccionar, almacenar, recuperar, transformar y exhibir datos espaciales del mundo real para un sistema particular con propósito definido" (Garrete, y otros, 2009).

"Un SIG es un sistema que integra tecnología informática, personas e información geográfica, y cuya principal función es capturar, analizar, almacenar, editar y representar datos geo-referenciados" (Olaya, 2010).

Dicho concepto, según el autor, podría generalizarse como: Un conjunto de métodos, software y hardware que almacena, analiza, manipula y representa datos geográficos. Estas funcionalidades los han posicionado en un lugar significativo dentro de los marcos de la protección civil, la demografía, los recursos mineros, el análisis de mercados, la gestión territorial, entre otros.

Dichas utilidades podrían expandirse con la inclusión del nivel de razonamiento e inferencia que proporcionan las **Ontologías**, unas de las herramientas más versátiles de la Web Semántica. Este concepto ha evolucionado grandemente desde su primera aparición en el siglo XVll en el campo filosófico hasta alcanzar una posición dentro de las ciencias de la computación y la información hoy en día. Su concepto más citado dentro de la literatura refiere que: *"Una ontología es una especificación explícita de una conceptualización"* (Gruber, 1993). A partir de este el término ha evolucionado dentro de la comunidad adquiriendo más connotación como se puede observar a continuación:

"Una ontología es una jerarquía estructurada de términos para describir un dominio que puede ser usado como el esqueleto de una base de conocimientos" (Swartout , y otros, 1997).

"Una ontología es un set de axiomas lógicos designados a representar el significado intencional de un vocabulario" (Guarino , 1998).

"Una ontología puede tomar una variedad de formas, pero necesariamente incluirá un vocabulario de términos y alguna especificación de su significado. Esto incluye conceptos y una indicación de cómo estos se relacionan unos con otros, lo cual, colectivamente impone una estructura en el dominio y restringe la posible interpretación de los términos" (Uschold , y otros, 1999).

"Una ontología define un conjunto de primitivas con las cuales se modela un dominio de conocimiento o discurso, en el que, dichas primitivas son típicamente clases, atributos y relaciones entre las clases" (Navarro, 2012).

A partir de estas definiciones se puede englobar el termino Ontología como una representación de manera explícita y a través del lenguaje natural de un modelo abstracto de algún fenómeno en el mundo, basado en conceptos y las relaciones entre ellos permitiendo la comprensión de estructuras de información entre personas o agentes de software. En sentido general, una ontología se considera la base del procesamiento semántico capaz de formar una red de conceptos, relaciones y axiomas para representar, organizar y entender un dominio de conocimiento. Esta capacidad resulta especialmente útil a la hora de reutilizar el conocimiento representado, especialmente en procesos de búsqueda y recuperación de información.

Los componentes de una ontología varían de acuerdo al dominio de interés y a las necesidades de los desarrolladores, sin embargo entre los básicos se pueden mencionar a (Navarro, 2012):

Conceptos: Son las ideas básicas que se intentan formalizar. La forma más común de definir los conceptos es mediante clases que definen el estado interno que tendrán los objetos.

Propiedades (slots): Las propiedades son las características intrínsecas a cada concepto, que permiten discernir claramente entre uno y otro.

Relaciones: Representan la interacción y enlace entre los conceptos del dominio y especifican las dependencias entre unos y otros.

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.

Instancias: Representan objetos determinados de una clase, son las implementaciones específicas de un concepto, con valores concretos a sus propiedades.

Axiomas: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

Estas ontologías pueden ser clasificadas de acuerdo a su dependencia y relación con una tarea específica como se describe a continuación (Guarino , 1998):

Ontologías de Alto Nivel o Genéricas: Describen conceptos más generales.

Ontologías de Dominio: Describen un vocabulario relacionado con un dominio genérico.

Ontologías de Tareas o de Técnicas básicas: Describen una tarea, actividad o artefacto.

Ontologías de Aplicación: Describen conceptos que dependen tanto de un dominio específico como de una tarea específica y, generalmente son una especialización de ambas.

Precisamente se puede afirmar que los SIG podrían emplear en especial ontologías que posean características específicas del dominio geográfico y son llamadas comúnmente **Geo-ontologías**. Especialistas dentro de este campo la han definido como:

“Las geo-ontologías son estructuras en las que se puede llevar a cabo la integración entre datos, metadatos y conocimiento espacial” (Garea Llano, y otros, 2007).

"Las ontologías que tratan la temática geoespacial presentan características particulares de este campo de estudio. Debido a su nivel de especialización, a estas ontologías geográficas se les han denominado geo-ontologías" (Vera Voronisky, y otros, 2009).

"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" (Garea Llano, y otros, 2009).

El autor a partir de estos conceptos ha llegado a la conclusión de que las geo-ontologías son representaciones formales del dominio geográfico que cumplen con todas las características de una ontología convencional, pero cuentan además con la presencia predominante de relaciones topológicas. Ramas como la Topografía, Geología, Hidrología y Meteorología han sido beneficiadas con la utilización de geo-ontologías en la representación de información en sus distintos sistemas.

Esta nueva gama de posibilidades en cuanto a las capacidades de los SIG han dado lugar a los **SIGGO**. Este término también ha sido referido dentro de la literatura como:

"Un SIG que utiliza ontologías para generar nueva información a partir de un conjunto previo de datos es lo que se denomina SIGGO. En los SIGGO las ontologías son una componente más, como lo es la base de datos temáticos o espaciales que interviene y coopera de la misma manera para alcanzar los objetivos para los cuales fue creado" (Garea Llano, y otros, 2010).

"Son SIG que se caracterizan por un extensivo uso de ontologías en las fases de desarrollo y en el uso del sistema" (Fonseca, y otros, 1999).

"Los SIGGO no son más que SIG en los que se incorporan el uso de ontologías como un componente activo. Los SIGGO son construidos utilizando clases derivadas de las ontologías y como consecuencias de esto se extrae el conocimiento embebido en estas para aportar mayor eficiencia y robustez al sistema" (Garea Llano, y otros, 2010)

Se puede afirmar entonces, según el autor, que los SIGGO surgen a partir de la necesidad de realizar un manejo semántico de la información almacenada sobre los objetos espaciales y entre sus principales funciones se encuentra hacer uso del conocimiento embebido para generar nuevo conocimiento que consiga almacenarse y ser utilizado posteriormente.

Los SIGGO tienen dos fases que son fundamentales en su estructura (Garea Llano, y otros, 2009). Primeramente la fase de generación del conocimiento, que comprende la especificación de las ontologías utilizando un editor de ontologías, la generación de nuevas ontologías a partir de las ya existentes y la traducción de ontologías a componentes de software. Luego comienza la fase de uso del conocimiento. Esta pone a las ontologías obtenidas en la fase anterior a disposición para ser navegadas por el usuario final y para su utilización en la generación de aplicaciones, análisis y finalmente las posibles alternativas de decisión.

1.3. Necesidad del Plugin. OntoMov

ONTOMOV es considerada la primera geo-ontología capaz de representar objetos en movimiento y, hasta el momento, la única en su campo. Surge producto del proyecto conjunto entre la Universidad de las Ciencias Informáticas (UCI) y la Universidad de Málaga en España (UMA), pionero en el trabajo con geo-ontologías en la representación semántica de objetos móviles. La misma importa conceptos de ontologías existentes debido a la generalidad de la terminología considerada y la posibilidad de representar conceptos espacio-temporales usando la aproximación de 4D-fluents[1] (Welty, y otros, 2006). Dada su importancia en el análisis de movimiento considera conceptos asociados a la trayectoria de los objetos. Fue desarrollada usando la herramienta Protegé[2] con el lenguaje OWL[3]. Posee como funcionalidades fundamentales la captura y representación de la conceptualización del dominio de objetos en movimiento para su posterior reconocimiento, clasificación e intercambio usando las jerarquías de conceptos del dominio y reglas de inferencia SWRL[4].

El presente trabajo surge como una necesidad dentro del Grupo Khaos[5], de la referida institución española, y la plataforma europea de ciencia y tecnología COTS-MOVE. Se busca poner en práctica ONTOMOV como una base de conocimiento sobre la plataforma GENESIG, aplicada en un SIG para el análisis de la trayectoria y grado de afectación de huracanes sobre Cuba. Está respaldada por los proyectos: "Hacia una Plataforma para la Explotación y Análisis de Datos Vinculados en Biología de Sistemas" y "Una Plataforma

[1] **4D-fluents:** Enfoque para la representación de la evolución de la información temporal en ontologías.
[2] **Protegé:** Herramienta para la creación de ontologías.
[3] **OWL:** Lenguaje de desarrollo usado para la construcción de ontologías.
[4] **SWRL:** Lenguaje de Reglas para la Web Semántica.
[5] **Khaos:** Grupo de investigación con amplia experiencia en tecnologías de Web Semántica, Integración de Datos y de Aplicaciones y Bases de Datos. Puede encontrar más información al respecto en la siguiente dirección: *http://khaos.uma.es/*

Colaborativa para el Análisis del Big Data" de la UMA. Para lograr poblar la ontología se requiere de automatizar el proceso de extracción de información semántica a partir de la información almacenada en el sistema.

1.4. Estado actual del proceso de poblado de las Ontologías

Con el fin de explotar al máximo las posibilidades del uso de Geo-ontologías dentro de este contexto se hace necesario poblar la ontología. El **poblado de ontologías** es una actividad de adquisición de conocimiento que descansa en métodos automáticos o semi-automáticos para transformar datos desestructurados, semi-estructurados y estructurados en instancias de datos. Esta evolución de la ontología está definida como un proceso compuesto por las siguientes tareas (Castano, y otros, 2008):

1. Extracción de la Información.
2. Interpretación Semántica.
3. Selección de Patrones y Evolución.

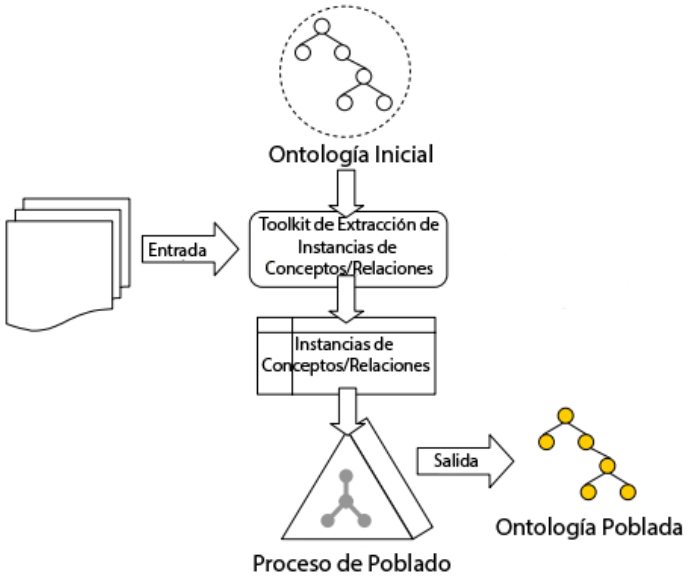


Fig. #1: El proceso de poblado de las ontologías (Petasis, y otros, 2011).

Hasta hace poco tiempo solo se buscaba manejar los conceptos y las relaciones de las ontologías, no los mismos objetos, limitando su uso al mero análisis de la taxonomía sin explotar las posibilidades de las ontologías como bases de conocimiento. Por esto los programas para la creación de ontologías como

Protegé, carecen de las funcionalidades que permitirían el poblado automático de sus elementos. Se deseaba únicamente compartir un entendimiento común acerca de la estructura y semántica de la información entre personas y agentes de software, no del contenido como tal.

Actualmente se han ido desarrollando ontologías como bases de conocimiento capaces de aprovechar estas potencialidades para el análisis semántico de la información de un dominio. Sin embargo, esta incapacidad de los programas de creación de ontologías para poblarlas automáticamente continúa siendo una problemática vigente. Por esta razón han ido surgiendo paulatinamente una serie de alternativas que serán descritas a continuación:

Herramientas para el poblado de Ontologías:

OntoClassify

OntoClassify es una herramienta que usa métodos de descubrimiento de conocimiento para el poblado eficiente de distintas ontologías. Genera un modelo para cada concepto habilitando una estimación eficiente para la asociación de nuevas instancias extraídas de la web. Ha sido reconocido como un bloque de creación útil para aplicaciones de web semántica avanzada e incorporado en varios sistemas. Incluye la construcción de ontologías de flujo de datos, ontologías semiautomáticas y ontologías de contextualización (Davies, y otros, 2009). A pesar de estas características, dicha herramienta no es capaz de realizar el proceso de extracción de información desde una base de datos espacial, ya que solo considera el texto en línea como fuente de datos.

OwlExporter

OwlExporter es una herramienta capaz de poblar ontologías ya existentes de dominio específicos en cualquier formato OWL, así como ontologías PNL (Programación Neurolingüística). Esta utiliza dos ontologías para poblar desde un texto: La primera ontología es una ontología de dominio específico que modela los conceptos y relaciones que son relevantes para el dominio dado (por ejemplo, la biología, la arquitectura, la ingeniería de software); y el segundo ontología es una ontología PNL independiente dominio que contiene los conceptos de uso común en la ingeniería lingüística. Rellena automáticamente la ontología PNL con relaciones como “hasEntity” o “containsSentence” que asocian los individuos de la ontología de dominio a los individuos de la ontología PNL. Permite su inclusión dentro de sistemas de poblado de ontologías (Witte, y otros, 2010). Sin embargo, dicha herramienta considera un análisis exclusivo a partir de

texto plano en la web y resultaría sumamente trabajoso adaptar los procedimientos propios de la PNL que emplea al dominio de objetos móviles.

Sistemas de Poblado de Ontologías:

Además, dentro de este contexto fueron creados un grupo de sistemas para el poblado de ontologías donde la gran mayoría comparten la arquitectura antes descrita en la Fig. #1: es usado un paquete de herramientas para la identificación de términos con el objetivo de crear instancias de conceptos y relaciones que posteriormente son asimilados en las ontologías. A continuación se presenta un análisis comparativo de los principales enfoques y sistemas prácticos que han sido presentados en la literatura para poblado de ontologías (Petasis, y otros, 2011):

Artequakt: Proyecto que se propone generar automáticamente biografías de artistas y sus pinturas utilizando fragmentos de información extraídos de la web y mantenerlas en una base de conocimiento. Su objetivo es acumular información de una amplia variedad de fuentes y enfocarla a los intereses de un lector en particular. Usa heurísticas para remover instancias redundantes de la ontología. Emplea una selección de herramientas y técnicas de extracción de información seguida de herramientas de construcción narrativa desarrolladas como parte del proyecto. El nivel de personalización de la biografía radica en su capacidad de adaptarse a los requerimientos de un determinado lector como son: “biografías adecuadas para niños”, “uso del color en sus pinturas”, etc. Su proceso está compuesto por 3 partes fundamentales: Extracción de Conocimiento, Representación y Almacenamiento del conocimiento y Generación Narrativa.

World Wide Knowledge Base (Web->KB): Su principal propósito es desarrollar una base de conocimiento probabilística y simbólica que funcione de espejo sobre el contenido de la red de redes. De ser exitoso este proyecto daría acceso a la información textual en la web en un formato comprensible por las computadoras habilitando una recuperación de información mucho más sofisticada y la resolución de problemas. Se propone que el sistema pueda ser entrenado para extraer conocimiento simbólico de hipertexto usando una variedad de métodos de aprendizaje de máquina. Las instancias de relaciones son recuperadas examinando los hipervínculos que conectan las páginas web.

SmartWeb Ontology-based Annotation (SOBA): Es un componente para la extracción de información de páginas web de fútbol para poblar automáticamente una base de conocimientos que puede ser usada para responder preguntas específicas del dominio. Puede extraer información de fuentes heterogéneas tales como: estructuras tabulares, textos y capturas de imágenes de una forma semánticamente integrada con la

ayuda de reglas de asignación. Se puede afirmar que almacena la información extraída en una base de conocimiento y luego la utiliza para interpretar y vincular la nueva información extraída con aquella ya existente.

Bootstrapping Ontology Evolution with Multimedia Information Extraction (BOEMIE): El proyecto BOEMIE aboga por un análisis de contenido de multimedia impulsado por ontologías (semántica de extracción a partir de imágenes, vídeo, texto, audio / voz) a través de un novedoso método sinérgico que combina la extracción de multimedia y la evolución de ontologías de manera bootstrapping. Este último consiste en la extracción continua de información de documentos multimedia con el fin de poblar y enriquecer las ontologías y el despliegue de estas para mejorar la robustez y precisión de la extracción.

Como se pudo observar básicamente 3 enfoques abordan el poblado de ontologías: el sistema Artequakt aplica heurísticas escritas manualmente con el fin de fundir instancias que refieren al mismo objeto o evento real. Estas heurísticas se evalúan después de que un lote de instancias ha poblado la ontología. El sistema SOBA, por otra parte, realiza comprobaciones utilizando reglas de asignación especial, durante la creación de la instancia, con el fin de reutilizar los casos que se refieren al mismo objeto o evento real en lugar de crear otros nuevos. El enfoque seguido por BOEMIE realza el de Artequakt, mediante el uso de aprendizaje de máquina de en lugar de heurísticas manualmente desarrollados. En general, estos sistemas analizados son bastante completos en el sentido de que pueblan la ontología con instancias tanto de conceptos como de relaciones. Se muestran diferencias a la hora de llevar a cabo el reconocimiento de entidades a partir de las ontologías para posteriormente encontrar candidatos a la clase y, a pesar de existir similitudes entre estos, todos utilizan enfoques, algoritmos y herramientas distintas y específicas del dominio.

Sin embargo, se hace imposible la utilización de alguno de estos sistemas debido a que ninguno analiza la extracción de información semántica desde una base de datos espacial y no son capaces de representar relaciones n-arias[6], incluyendo el factor tiempo, imprescindible dentro del campo de objetos móviles. Además, el proceso de transformación de los datos geográficos con el objetivo de crear instancias en ontología depende grandemente de las características puntuales del dominio de los datos en todos los

[6] **Relaciones N-arias:** Generalización de las relaciones binarias. La relación está conformada por una tupla de n términos.

casos, haciendo sumamente trabajoso adaptarlos al análisis de la trayectoria y grado de afectaciones de los huracanes sobre la isla.

Herramientas para el poblado de Ontologías a partir de una Base de datos

La meta de esta aproximación es establecer una correlación entre la base de datos y la ontología para realizar el proceso de poblado con el contenido almacenado en la misma. Este enfoque incluye 2 procesos principales: La transformación del esquema de la base de datos a la estructura de la ontología para posteriormente migrar el contenido de la base de datos a instancias de la ontología. Esto puede realizarse de 2 formas (Ghawi, y otros, 2007):

- Como un proceso por lotes descargando todas las instancias de base de datos al repositorio de la ontología.
- Como un proceso impulsado por consultas mediante la transformación sólo de las instancias de bases de datos que son la respuesta a una petición dada.

DB2OWL

El objetivo de este módulo es crear una nueva ontología de una base de datos relacional. La conciliación de diferentes fuentes de datos se realiza mediante la asignación de las distintas tablas y columnas a la ontología de referencia utilizando servicios web. La ontología generada contendrá sólo las clases y propiedades, pero no las instancias, que permanecerán en la base de datos y podrán ser recuperadas y traducidas, según sea necesario en respuesta a las consultas de los usuarios. El prototipo de este herramienta ha sido desarrollada en Java usando Jena[7] y trata con BD desplegadas en Oracle y MySQL (Ghawi, y otros, 2007). O sea, es una herramienta que logra establecer una correlación entre la base de datos y la ontología que esta construye. Sin embargo, no concibe el poblado de una ontología ya existente y no analiza la elaboración de una base de conocimientos, sino que plantea analizar individualmente cada instancia con la participación directa de un usuario. Además, actualmente no es capaz de analizar la información geográfica almacenada en PostgreSQL.

[7] Jena: Marco de trabajo de la Web Semántica de código abierto para Java.

1.5. Propuesta de Solución

En el análisis de soluciones y alternativas existentes que automatizan el poblado con instancias de ontologías no se encontró ningún sistema capaz de adaptarse al dominio geográfico de cuerpos móviles. Sin embargo, las características comunes presentes en cada uno de estos demuestran la viabilidad que supone el desarrollo de un nuevo sistema con características específicas y distintivas acorde a la problemática de la investigación. Se propone que dicho sistema sea capaz de acoplarse en forma de Plugin a la plataforma GeneSIG y pueda incluirse en un SIG especializado en el análisis de la trayectoria y nivel de afectaciones de los huracanes. Se concibe que este SIG funcione utilizando una base de datos externa que contiene datos referentes a cada uno de los puntos de la trayectoria de los huracanes de los cuales resulta importante almacenar en la base de conocimientos sus coordenadas geográficas, su valor en el tiempo y su posición dentro de la base de datos espacial que permita acceder al resto de la información. Además es importante extraer las relaciones topológicas que presentan que doten al sistema de un entendimiento semántico de consultas como: “¿El huracán Ana pasó al sur de La Habana?” o “Huracanes que cruzaron Pinar del Río.”, solo por poner unos ejemplos. Esto mejoraría drásticamente la búsqueda de contenido geográfico en el SIG ya que le permitiría razonar y hacer inferencias según los elementos insertados y mostrar un resultado acorde a las necesidades del usuario.

1.6. Conclusiones Parciales

A través del análisis de los principales conceptos se logró elaborar una panorámica en cuanto al objeto de estudio de la investigación donde se observa principalmente la estructura de una ontología y los elementos que la componen que pueda propiciar luego la creación de las instancias de los distintos conceptos con sus relaciones correspondientes. Además, luego de analizar las herramientas y sistemas existentes diseñados para el poblado de ontologías queda ratificada la viabilidad del proyecto a realizar en el presente trabajo mejorando el proceso de poblado de OntoMov. Sin embargo, resulta imposible la reutilización de alguno de estos sistemas ya que resultaría sumamente trabajoso adaptarlos al dominio de objetos móviles y a la infraestructura de GeneSIG, en particular.

Capítulo 2: Herramientas, Tecnologías y Metodología a emplear en la construcción de la solución

2.1. Introducción

En este capítulo se fundamentan las tecnologías y las herramientas que serán utilizadas en el desarrollo del Plugin para GeneSIG de forma tal que sus funcionalidades puedan acoplarse totalmente a la plataforma. Además, se selecciona la metodología que guiará el proceso de desarrollo y permitirá generar la documentación y artefactos correspondientes que faciliten el proceso de capacitación del personal en el uso del Plugin.

2.2. Herramientas y Tecnologías adoptadas de GeneSIG

Antes de adentrarse en el análisis de las ventajas del uso de las herramientas y tecnologías que soportan la plataforma GeneSIG se hace necesario entender totalmente su funcionamiento y características principales que permitan esbozar el panorama sobre el cual está ambientado el desarrollo del Plugin para la extracción de información semántica de objetos móviles. GeneSIG podría definirse como:

"La Plataforma GeneSIG es un producto encaminado a realizar la representación y análisis geoespacial de información geográfica y su estructura arquitectónica permite personalizar sus funcionalidades a cualquier negocio que lo requiera a través de la reutilización de sus componentes. Puede ser considerado como un Sistema de Información Geográfica Web único y extensible, basado en estándares OpenGIS que incluye funcionalidades operativas de las aplicaciones de esta tecnología" (Pantoja Zaldívar, y otros, 2010).

A partir de lo antes analizado se puede concluir que GeneSIG es una plataforma que tiene como características principales la modularidad, la reusabilidad y la compatibilidad. Además, permite la representación geoespacial de la información asociada a cualquier dominio, proporcionando a su vez servicios de acceso a la información geográfica, para su consulta, análisis y visualización, mediante una interfaz de usuario sencilla y de fácil manejo. Dicho producto se encuentra en su versión 2.0 y para su desarrollo fueron utilizadas varias tecnologías y herramientas, de las cuáles resulta necesario para el desarrollo del proyecto analizar las siguientes:

Entorno de Desarrollo (IDE): NetBeans IDE 8.0

Lenguajes de desarrollo: PHP 5.3 y JavaScript.

Sistema Gestor de BD: PostgreSQL 8.4 (PostGIS 1.5.1)

Herramienta de Modelado: Visual Paradigm for UML 8.0

Lenguaje de Modelado: Lenguaje Unificado de Modelado 2.0

Frameworks: CartoWeb 3.0, ExtJS 3.0

Servidor Web: Apache 2.5.6

2.2.1. NetBeans IDE 8.0

NetBeans IDE es un software diseñado para el desarrollo de aplicaciones Java de escritorio, móviles y web. Además, provee un set de herramientas para desarrolladores en PHP y C/C++. Tiene la potencialidad de ser un software libre de código abierto bajo la licencia GPL y tiene una gran comunidad de usuarios y desarrolladores a lo largo del mundo. Incluye un potente editor capaz de analizar el código fuente sintáctica y semánticamente mientras que proporciona plantillas, consejos y generadores de código. Cabe destacar también funcionalidades como el auto-completamiento de código y la integración de frameworks de desarrollo. Proporciona distintas vistas de los datos: desde múltiples ventanas de proyectos hasta herramientas útiles para la creación y gestión eficiente de aplicaciones. Es un software extensible y multiplataforma, ya que puede ser instalado en todos los sistemas operativos que soporten Java como Windows, Linux o Mac. Admite la creación de aplicaciones Web en PHP 5.x, tiene soporte para AJAX y reconoce las librerías ExtJS desarrolladas con el lenguaje JavaScript, lo cual devino beneficioso en el desarrollo de GeneSIG y se adopta al desarrollar el presente trabajo (NetBeans Community, 2015).

2.2.2. PHP 5.3

Pre-procesador de Hipertexto PHP (del inglés: Hypertext Preprocessor), es un lenguaje de programación script rápido y flexible de propósito general del lado del servidor diseñado para el desarrollo web y puede ser embebido en páginas HTML. Su sintaxis se basa en C, Java y Perl y es fácil de aprender. Es multiplataforma y de código abierto, caracterizado por su estabilidad, seguridad y simplicidad y puede ser desplegado en la mayoría de los servidores web. PHP funciona ya sea como un módulo o como un procesador CGI (del inglés: Common Gateway Interface) y da soporte a una amplia gama de bases de datos. Permite usar programación procedural, programación orientada a objetos o una mezcla de las dos (Achour , y otros, 1997-2015). GeneSIG está programado en PHP 5 y está enfocado al desarrollo de

aplicaciones web en el mismo lenguaje de programación. Aunque la versión mínima de PHP requerida para ejecutar GeneSIG es PHP 5.2 para la realización del presente módulo se elige una versión más actual y por lo tanto superior.

2.2.3. JavaScript

JavaScript (también conocido por ECMAScript) es un lenguaje de programación ligero que se utiliza principalmente para crear páginas Web dinámicas del lado del cliente, exactas en cuanto a posición y presentación de su contenido. Puede funcionar tanto como un lenguaje procedural o un lenguaje orientado a objetos. Es un lenguaje multi-paradigma y basado en prototipos que soporta estilos de programación funcionales, orientados a objetos e imperativos. La sintaxis básica es intencionalmente similar a Java y C++ para reducir el número de nuevos conceptos necesarios para aprender el lenguaje. Capacidades dinámicas de JavaScript incluyen el tiempo de ejecución de la construcción de objetos, creación de scripts dinámicos y la recuperación de código fuente. Además, permite una máxima interactividad entre el usuario y la página y la verificación de los datos introducidos por el usuario antes de enviar el formulario al servidor (Mozilla Developer Network, 2005-2015). Cualquier navegador puede interpretar el código JavaScript dentro de las páginas web y debido a sus grandes beneficios en el funcionamiento de la interfaz de cara al usuario en la plataforma GeneSIG, se emplea en el desarrollo del presente Plugin.

2.2.4. PostgreSQL 8.4

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos, libre y gratuito de código abierto publicado bajo la licencia BSD (del inglés: Berkeley Software Distribution). Es compatible con una gran parte del estándar SQL y ofrece muchas características modernas como: consultas complejas, claves foráneas, triggers, vistas, integridad transaccional, soporte multi-usuario, optimización de consultas y control de concurrencia multi-versión. Es altamente extensible ya que permite la adición de nuevos tipos de datos, funciones, operadores, funciones de agregado, métodos de índice y leguajes procedurales. Se considera un software multiplataforma (Linux, Un ix, BSDs, Mac OS, Beos, Windows) capaz de manejar complejas rutinas y reglas (The PostgreSQL Global Development Group, 2014). Se definió como Sistema Gestor de Bases de Datos a utilizar en la plataforma GeneSIG, teniendo en cuenta que es un proyecto de software libre de gran estabilidad. Resulta importante destacar es que pese a que en los repositorios actuales se encuentra versiones superiores de PostgreSQL se recomienda instalar la 8.4.

2.2.5. PostGIS 1.5.1

PostGIS es una extensión de base de datos espacial agrega tipos de datos adicionales (geometry, geography, raster y otros) a la base de datos PostgreSQL liberada bajo la Licencia Pública General de GNU (GPLv2). También agrega funciones, operadores y mejoras de índice que se aplican a estos tipos espaciales y añade soporte para objetos geográficos que permiten ejecutar consultas de ubicación en SQL. Esto aumenta el poder del núcleo de PostgreSQL haciéndolo un sistema rápido y robusto ideal para su utilización en SIG (Ramsey, y otros, 2015). En el proyecto GeneSIG se emplea integrado en forma de módulo almacenando la información geográfica en una columna del tipo geometry en formato WKB[8]. Por esto se decide adoptar su utilización en el presente trabajo.

2.2.6. Visual Paradigm for UML 8.0

Visual Paradigm for UML (VP-UML) es una potente y multiplataforma herramienta CASE para el modelado UML. Posee interoperabilidad con otras herramientas CASE y la mayoría de los principales IDEs. Se utiliza libre de costo bajo la llave de licencia con fines académicos disponible para educación superior. Permite crear diversos los tipos de diagramas UML 2.x incluyendo el diagrama de clases, el diagrama de componentes, el diagrama de despliegue, entre otros, así como el modelado de base de datos necesarios en el desarrollo del presente trabajo (Visual Paradigm International, 2015). Además, esta es empleada en el proyecto GeneSIG para la generación de los distintos artefactos asociados por lo cual se decide adoptar como parte de las herramientas de la investigación.

2.2.7. Lenguaje Unificado de Modelado 2.0 (UML)

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. El mismo permite realizar una descripción altamente detallada del sistema, de procesos del negocio, de componentes de software reutilizables, entre otros (Alhir, 2003). Particularmente en el presente trabajo resulta especialmente útil por cuanto permite la creación del diagrama de componentes, el diagrama de despliegue del sistema y los diagramas de clases del diseño.

[8] WKB: Codificación o sintaxis en formato ASCII estandarizada diseñada para describir objetos espaciales expresados de forma vectorial.

2.2.8. CartoWeb 3.0

Es un software de publicación WebGIS construido en PHP y basado en el motor de UMN MapServer que explota AJAX. El mismo ofrece un framework con una arquitectura modular y extensible conveniente para la construcción de aplicaciones avanzadas y personalizadas (Camptocamp SA, 2005). Es liberado bajo la licencia GNU GPL y sirve de columna vertebral para la plataforma soberana GeneSIG, permitiendo separar la lógica encargada del diálogo con el servidor de mapas y la provisión de los servicios al cliente. Además, cuenta con numerosas características propias de un geoportal, que incluyen la posibilidad de ir añadiendo o desarrollando nuevos Plugins. Esta viene siendo una de las mayores fortalezas del sistema que a su vez respalda la realización del presente trabajo por cuanto indica la posibilidad de crear nuevas funcionalidades que se adapten a la arquitectura de GeneSIG.

2.2.9. ExtJS 3.0

Es una biblioteca JavaScript ligera y de alto rendimiento, compatible con la mayoría de los navegadores, como Internet Explorer, Firefox, Safari y Opera, que permite crear páginas e interfaces Web dinámicas. Posibilita controles tales como panel de pestañas, ventanas, árboles, entre otros y está dirigido al desarrollo de aplicaciones Web interactivas usando tecnologías como AJAX, HTML y DOM. Posee componentes UI personalizables y extensibles, su API es fácil de usar y tiene licencias de códigos abiertos y comerciales. Además, permite crear aplicaciones complejas utilizando componentes predefinidos y hace posible realizar cambios sobre las páginas sin necesidad de recargarlas, aumentando la interactividad, velocidad y usabilidad en las aplicaciones (Ramon, 2009). El análisis de estas funcionalidades de ExtJS ha permitido comprobar las potencialidades del mismo para crear una interfaz de usuario funcional y compatible con la plataforma GeneSIG por lo que se adopta en el presente trabajo.

2.2.10. Servidor Web Apache 2.5.6

El servidor HTTP Apache es un servidor web de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. Soporta host virtuales y se compone de una sección core y diversos Módulos de Multiprocesamiento que implementan funcionalidades en el manejo de peticiones de forma dinámica. Esto permite que el servidor pueda ser personalizado de forma muy precisa instalando o desinstalando servicios a demanda de los usuarios del mismo y adaptado a diferentes entornos y necesidades. Además posee una configuración simple y potente

a través de ficheros y soporta scripts PHP, elementos que lo hacen ideal para cargar y desplegar la aplicación a desarrollar en el presente trabajo (The Apache Software Foundation, 2015).

2.3. Herramientas y Tecnologías utilizadas para el manejo de la ontología

Para la realización del presente trabajo se emplean además herramientas y tecnologías que de forma original no mantienen relación con la plataforma GeneSIG, sino más bien con la propuesta de solución en sí. Entre estas se incluyen aquellas que permiten el consumo de servicios web, desplegados desde java, relacionados con la gestión de instancias de conceptos y relaciones dentro de la ontología. A continuación se realiza una breve descripción de los mismos.

2.3.1. Java 7

Java es un lenguaje de programación de alto nivel orientado a objetos de propósito general y basado en clases. Su sintaxis se asemeja a la de C y C++, pero elimina sus características menos usadas y más confusas proporcionando de esta forma simplicidad y facilidad de trabajo. Incluye la gestión de almacenamiento automático y es compilado a un conjunto de instrucciones de código de bytes y a formato binario. Cuenta además con potentes entornos de desarrollo como el Netbeans IDE y una abundante documentación tanto en formato duro como digital. Proporciona portabilidad al software implementado ya que el mismo podrá ser ejecutado sobre cualquier plataforma utilizando el JDK (del inglés: Java Development Kit) (Gosling, y otros, 2015). Además, Java constituye una tecnología que permite el desarrollo de implementaciones SOAP y aplicaciones que usen SOAP para su comunicación así como la construcción de servicios web, por lo cual se adopta en el presente trabajo.

2.3.2. SOAP (Simple Object Access Protocol)

SOAP (del inglés: Simple Object Access Protocol) es un protocolo de comunicación ligero basado en XML para el intercambio de información estructurada en un ambiente descentralizado y distribuido. No define la aplicación, ni la semántica de implementación, solamente, proporciona un modelo de empaquetamiento modular y los mecanismos para la codificación de los datos dentro de los módulos. Entre sus principales características se puede mencionar (XML Protocol Working Group, 2007):

- Es inherente a cualquier lenguaje de programación.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- Permite la interoperabilidad entre múltiples plataformas.

La utilización de los servicios web resulta de gran utilidad en el presente proyecto debido a la independencia que otorgan a las aplicaciones. Esta permite implementar cada parte por separado garantizando un diseño flexible y tolerante a cambios.

2.3.3. OWL-API

OWL-API es una API de Java para la creación, manipulación y serialización de ontologías OWL. Su última versión da soporte a OWL2 y está disponible bajo las licencias de Open Source LGPL y Apache. Su punto central de acceso es el OWLOntologyManager, que es usado para cargar, crear y acceder a las distintas ontologías. Para su correcto funcionamiento requiere de la versión 6 de Java o una superior. Incluye los siguientes componentes (Palmisano , 2015):

- Analizador sintáctico y escritor RDF/XML
- Analizador sintáctico y escritor OWL/XML
- Analizador sintáctico y escritor de Sintaxis Funcional OWL.

En el presente trabajo se selecciona dicha herramienta debido a que permite adherirse a una aplicación desarrollada en JAVA con el fin de insertar instancias de conceptos y relaciones en una ontología escrita en el lenguaje OWL2 desarrollada utilizando la herramienta Protegé.

2.4. Metodología de Desarrollo

La metodología de desarrollo se selecciona acorde a los requerimientos del grupo de trabajo de GeneSIG que solicita obtener como resultado del presente trabajo un conjunto de artefactos y documentación asociada acorde con el proyecto. Es por esta razón que se adopta PRODESOFT, que a pesar de no ser una metodología para el desarrollo como tal, se presenta ideal como proceso de desarrollo del Plugin de extracción de información semántica de objetos móviles para GeneSIG.

2.4.1. PRODESOFT 1.5

Proceso de Desarrollo de Software diseñado por la UCID (Unidad de Compatibilización e Integración de Software para la Defensa) actual XETID (Empresa de Tecnologías de la Información para la Defensa). Concibe el ciclo de vida del proyecto como una composición de 5 fases: inicio, modelación, construcción, explotación experimental y despliegue, donde cada fase terminará en un hito con el objetivo fundamental de evaluar y decidir el paso a la siguiente fase de desarrollo. Su modelo de desarrollo de software describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones con una

combinación entre los modelos basado en Componentes, el Iterativo y el Incremental. En las primeras fases las iteraciones son realizadas con mayor énfasis en la determinación del alcance del proyecto, la planificación, identificación y descripción de requisitos y la creación de la línea base de la arquitectura, para luego centrarse en el análisis, diseño, implementación y pruebas de los requisitos funcionales. Las iteraciones sucesivas se construyen sobre los artefactos de desarrollo tal como quedaron al final de la última iteración, lo cual permite reducir los riesgos más significativos para el éxito del proyecto. Su premisa de desarrollo basado en componentes permite alcanzar un mayor nivel de reutilización de software y ejecutar las pruebas de forma individual antes de probar los componentes ya ensamblados (UCID: SOFTWARE-DEFENSA, 2012). Se adopta en el presente trabajo debido al éxito de su implementación en el proyecto GeneSIG como guía para su desarrollo y la generación de los artefactos necesarios.

2.5. Conclusiones Parciales

A través del análisis de las principales características de la metodología, herramientas y tecnologías adoptadas, enfocadas en las ventajas y facilidades de cada una de estas, se logra crear un marco de trabajo adecuado a las necesidades y características propias del Plugin como complemento de la plataforma GeneSIG. De forma general, son herramientas y tecnologías en su mayoría libres de código abierto en correspondencia con los estándares del país y la universidad. Además, se encuentran ubicadas entre las mejores dentro de su área, donde su éxito en el desarrollo de numerosos proyectos ha validado su selección en el presente trabajo. PRODESOFT, por otra parte, garantiza la completitud y calidad del software guiando eficientemente no solo el proceso de desarrollo de software, sino también la organización, la gestión, la configuración y cambios y el soporte del Plugin. Estos factores en su conjunto han permitido al equipo de desarrollo producir un software más robusto, predecible, reutilizable y de fácil mantenimiento.

Capítulo #3 Descripción de la Propuesta de Solución

3.1. Introducción

Durante el desarrollo de este capítulo se realiza el modelado y el levantamiento de los requisitos funcionales y no funcionales del Plugin de extracción de información semántica de objetos móviles para GeneSIG. A partir de estos se procede al diseño de la arquitectura y al diseño detallado de las clases del diseño correspondientes a las primeras fases del modelo de desarrollo de software definido por PRODESOFT. Esto permite la generación de artefactos como son el modelo de dominio, las especificaciones de requisitos, el diagrama de componentes, el diagrama de despliegue, los diagramas de diseño, entre otros, que dejan sentadas las bases para las futuras etapas de implementación y prueba.

3.2. Modelado del Negocio

El propósito principal de esta fase en el proceso de desarrollo de software es realizar una exploración del dominio del problema especificando los conceptos y las relaciones que existen entre estos, que permitan determinar las necesidades operacionales que logren un entendimiento final del negocio y den paso a la fase inicial del sistema (UCID: SOFTWARE-DEFENSA, 2012).

3.2.1. Modelo Conceptual

El propósito de esta actividad es identificar y representar conceptos relacionados con el dominio del problema obteniéndose como resultado el Modelo Conceptual. Éste explica los conceptos más significativos en el dominio del problema, teniendo como cualidad esencial representar elementos del mundo real (UCID: SOFTWARE-DEFENSA, 2012).

En el presente trabajo se analiza un SIGGO que maneja datos geográficos. Sin embargo es propósito de la investigación incorporar dentro de dicho sistema un módulo semántico que utilice dichos datos geográficos convirtiéndolos en información sensible del dominio con la cual enriquecer la ontología que utiliza el sistema con instancias de los elementos que contiene, ya sean conceptos o relaciones.

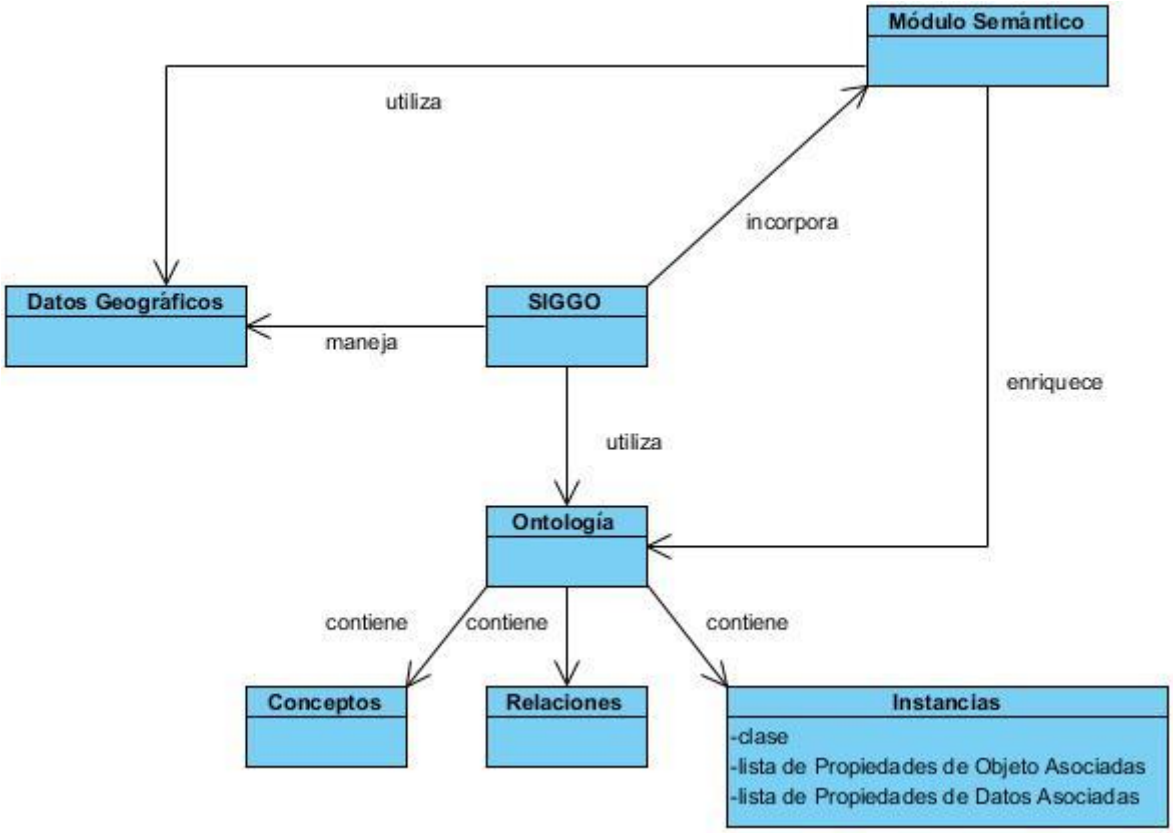


Fig. #2: Diagrama de Modelo Conceptual.

Definiciones de clases del modelo conceptual:

Conceptos: Clase que define el estado interno de los objetos.

Dato Geográfico: Datos espaciales geo-referenciados requeridos como parte del sistema tanto de forma implícita como explícita.

Instancia: Realización específica de un concepto o relación determinados.

Módulo Semántico: Contiene los algoritmos necesarios para transformar los datos geográficos extraídos en información semántica.

Ontología: Forma de representación del dominio de objetos móviles que al ser enriquecida con la información semántica se trasforma en una base de conocimientos.

Relaciones: Representan la interacción y dependencias entre los conceptos del dominio.

SIGGO: Entidad que maneja la información geográfica y utiliza ontologías para generar nueva información a partir de un conjunto previo de datos.

3.3. Definición de Requisitos

El propósito de esta fase es la definición de requisitos que especifican las condiciones o capacidades que el sistema debe cumplir y las restricciones bajo las cuales debe operar, logrando un entendimiento entre el equipo de desarrollo y el especialista funcional, y especificando las necesidades reales de forma que satisfaga sus expectativas (UCID: SOFTWARE-DEFENSA, 2012).

3.3.1. Requisitos Funcionales

Los requisitos funcionales definen las condiciones o capacidades que el sistema será capaz de realizar. Estos describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Estos requisitos, al tiempo que avanza el proyecto de software, se convierten en los algoritmos, la lógica y gran parte del código del sistema (UCID: SOFTWARE-DEFENSA, 2012).

El Plugin debe ser capaz de:

RF1. Insertar instancias dado un umbral.

RF2. Insertar instancias dada una provincia.

RF3. Actualizar las relaciones espaciales.

A continuación se muestra las especificaciones de dichos RF que permitan un mejor entendimiento del funcionamiento del Plugin:

El requisito **Insertar instancias dado un umbral** permite insertar instancias de cada punto en la trayectoria del huracán y las relaciones que presenta con las provincias y municipios afectados dentro de un umbral de distancia dado.

Tabla #1: Especificación del requisito funcional Insertar instancias dado un umbral.

Conceptos tratados	Conceptos	Atributos
	Instancia	• Clase de Instancia

		• Listado de Propiedades de Objetos asociados • Listado de Propiedades de Datos asociados
Precondiciones	Precondiciones	Pre-requisito
	No procede.	No procede.
Descripción	1. Se activa el Plugin haciendo clic sobre el botón representado por la siguiente iconografía (). 2. Se selecciona el huracán a analizar. 3. Se selecciona la opción de inserción de instancias por Umbral. 4. Se introduce en el campo de texto correspondiente al umbral el número de kilómetros que componen el radio de alcance a contemplar en el proceso. 5. Se hace clic en el botón que contiene la opción Extraer en la parte inferior del panel. 6. El sistema extrae identificador, longitud, latitud, fecha y geometría asociada de cada punto de la trayectoria del huracán seleccionado de la base de datos. 7. El sistema inserta las instancias de huracán y de cada punto de su trayectoria. 8. Son insertadas las propiedades de datos asociadas a cada instancia. 9. Se determina el área de acción de cada punto de la trayectoria empleando el umbral dado. 10. Se identifican las provincias ubicadas dentro del área de acción por cada punto de acción. 11. Se define la afectación o no de las regiones por el huracán. 12. Se identifican las relaciones espaciales entre cada punto de la trayectoria del huracán y cada región afectada. (Al Norte de, Al Sur de de, Cruza, etc) 13. Se insertan las propiedades de objetos asociadas entre las instancias de acuerdo a las relaciones espaciales identificadas. 14. Se muestra el siguiente mensaje al usuario: “Las Instancias han sido adicionadas satisfactoriamente.”	
Validaciones	2. El botón Extraer solo es activado al seleccionar un huracán de la lista mostrada.	

	4. Se comprueba que se ha seleccionado la opción de insertar instancias por Umbral para activar el campo de texto que permite la inserción del umbral. 5. Se comprueba que se ha seleccionado una opción, de lo contrario muestra el siguiente mensaje de error: "Debe seleccionar la opción por la cual desea adicionar las instancias." 5. Se comprueba que el dato entrado en el campo de texto sea tipo entero menor que 1000 km. De no serlo muestra el siguiente mensaje de error: "Solo admiten números enteros entre 1 y 1000."
Complejidad	Alta
Prioridad	Alta
Post-condiciones	Se ha insertado una instancia dado un umbral.
Post-requisito	No procede

Por otra parte el requisito **Insertar instancias dada una provincia** permite determinar si el área seleccionada fue afectada por el huracán y, de ser así, insertar instancias de los distintos puntos del huracán, de los municipios afectados en la provincia y de sus relaciones espaciales en la ontología.

Tabla #2: Especificación del requisito funcional Insertar instancias dada una provincia.

Conceptos tratados	Conceptos	Atributos
	Instancia	• Clase de Instancia • Listado de Propiedades de Objetos asociados • Listado de Propiedades de Datos asociados
Precondiciones	Precondiciones	Pre-requisito
	No procede.	No procede.
Descripción	1. Se activa el Plugin haciendo clic sobre el botón representado por la siguiente iconografía ().	

	2. Se selecciona el huracán a analizar. 3. Se selecciona la opción de inserción de instancias Punto a Punto. 4. Se selecciona de la lista desplegada la provincia a analizar. 5. Se hace clic en el botón que contiene la opción Extraer en la parte inferior del panel. 6. El sistema extrae identificador, longitud, latitud, fecha y geometría asociada de cada punto de la trayectoria del huracán seleccionado de la base de datos. 7. El sistema inserta las instancias de huracán y de cada punto de su trayectoria. 8. Son insertadas las propiedades de datos asociadas a cada instancia. 9. Se identifican las relaciones espaciales entre cada punto de la trayectoria del huracán y cada región afectada. (Al Norte de, Al Sur de de, Cruza, etc) 10. Se insertan las propiedades de objetos asociadas entre las instancias de acuerdo a las relaciones espaciales identificadas. 11. Se muestra el siguiente mensaje al usuario: “Las Instancias han sido adicionadas satisfactoriamente.”
Validaciones	2. El botón Extraer solo es activado al seleccionar un huracán de la lista mostrada. 4. Se comprueba que se ha seleccionado la opción de insertar instancias Punto a Punto para activar la lista que permite la selección de la provincia a analizar. 5. Se comprueba que se ha seleccionado una opción, de lo contrario muestra el siguiente mensaje de error: “Debe seleccionar la opción por la cual desea adicionar las instancias.” 5. Se comprueba que se haya seleccionado una provincia de la lista. De no hacerlo se muestra el siguiente mensaje de error: “Debe seleccionar la provincia en la cual se basa el proceso de extracción.”
Complejidad	Alta
Prioridad	Alta
Post-condiciones	Se ha insertado una instancia dada una provincia.
Post-requisito	No procede

Además, el requisito **Actualizar relaciones espaciales** le permite al usuario elegir las relaciones a tener en cuenta en el en el proceso de extracción de información semántica. Dicha funcionalidad permite de un listado de relaciones válidas para el proceso poder elegir aquellas de interés al usuario para realizar el análisis de cada punto en la trayectoria del huracán y las relaciones que presenta con las provincias y municipios afectados.

Tabla #3: Especificación del requisito funcional Actualizar relaciones espaciales.

Conceptos tratados	Conceptos	Atributos
	No procede.	No procede.
Precondiciones	Precondiciones	Pre-requisito
	No procede.	No procede.
Descripción	1. Se activa el Plugin haciendo clic sobre el botón representado por la siguiente iconografía (). 2. Se hace clic en el botón que contiene la opción Actualizar en la parte inferior del panel. 3. El sistema muestra una ventana con 2 listados de relaciones válidas: El listado de relaciones a utilizar y el listado con las relaciones restantes (Por defecto aparecen todas las relaciones en el listado de las relaciones a utilizar). 4. Se selecciona una relación del listado de las relaciones a utilizar. 5. Se hace clic en el botón Quitar ubicado entre los 2 listados representado por el siguiente símbolo (>>). 6. Se elimina la relación del listado de las relaciones a utilizar y pasa a componer el listado de relaciones restantes. 7. Se selecciona una relación del listado de las relaciones restantes. 8. Se hace clic en el botón Añadir ubicado entre los 2 listados representado por el siguiente símbolo (<<). 9. Se elimina la relación del listado de las relaciones restantes y pasa a componer el listado de a utilizar.	

	10. Se selecciona la opción **Aplicar**. 10.1. Si se selecciona la opción **Aceptar** se actualizan las relaciones a utilizar y colapsa la ventana. 10.2. Si se selecciona la opción **Cancelar** se cancelan los cambios realizados y se colapsa la ventana. 11. Se actualizan las relaciones a utilizar. 12. Se muestra el siguiente mensaje al usuario: "Las relaciones a utilizar han sido actualizadas satisfactoriamente."
Validaciones	10. El botón Aplicar solo se activará si la lista de relaciones a usar no está vacía. 10.1. El botón Aceptar solo se activará si la lista de relaciones a usar no está vacía.
Complejidad	Alta
Prioridad	Media
Post-condiciones	Se han actualizado las relaciones espaciales.
Post-requisito	No procede

3.3.2. Requisitos no Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Estos definen cuan usable, seguro, conveniente y agradable es el producto desarrollado, lo cual, puede marcar la diferencia entre un producto bien aceptado y uno con poca aceptación (UCID: SOFTWARE-DEFENSA, 2012).

El Plugin debe poseer las siguientes cualidades:

Usabilidad

RNF1. Debe emplearse el uso de tooltips que expliquen de forma clara la función de los elementos más significativos dentro del formulario.

Confiabilidad

RNF2. Debe ser estable y fiable de forma tal que la información que maneja sea correcta y completa y que no existan fallos o su ocurrencia sea mínima.

Rendimiento

RNF3. La velocidad de procesamiento de la información será relativamente rápida, no mayor de 20 segundos en cada inserción.

RNF4. Debe permitir la ejecución del Plugin de forma simultánea desde múltiples PC cliente.

Restricciones de diseño

RNF5. El producto final debe diseñarse sobre una arquitectura cliente-servidor.

RNF6. Debe emplear los estándares establecidos para el diseño de interfaces y codificación así como usar el lenguaje del lado del servidor PHP y JavaScript del lado del cliente.

Interfaz

RNF7. Debe tener una interfaz gráfica uniforme incluyendo pantallas, menús y opciones.

RNF8. Los mensajes de error deberán ser lo suficientemente informativos para dar a conocer la severidad del error.

RNF9. Se deberá facilitar la entrada de datos a los usuarios, presentando campos de selección que permitan escoger los valores.

Portabilidad

RNF10. El Plugin debe ser compatible con los sistemas operativos Windows y Linux.

Software

RNF11. Para el cliente: Sistema operativo Windows o GNU/Linux y navegador Mozilla Firefox 3.6 o superior.

RNF12. Para el servidor:

- Sistema operativo GNU/Linux Debian en su versión 6.

- Un servidor Apache 2.0 o superior con módulo PHP 5.3 disponible. Este debe estar configurado con la extensión “pgsql”.
- Un servidor de base de datos PostgreSQL 8.4 o superior con su extensión “postgis” para datos espaciales.

Hardware

RNF13. Para el servidor de aplicaciones y de base de datos:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 2Gb de memoria RAM.
- Al menos 10Gb de espacio libre en disco duro.
- Tarjeta de red.

RNF14. Para el servidor de servicios de manejo de la ontología:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 2Gb de memoria RAM.
- Al menos 30Gb de espacio libre en disco duro.
- Tarjeta de red.

RNF15. Para el cliente:

- Requerimientos mínimos: Procesador Pentium IV a 1.2 GHz con 512Mb de memoria RAM.
- Tarjeta de red.

Licencia

RNF16. El Plugin será distribuido bajo licencia libre de código abierto GPL (General Public License), permitiéndose la modificación, uso y distribución libre del producto.

3.4. Diseño de la Arquitectura

El diseño de la arquitectura propone una vista detallada obteniéndose elementos arquitectónicamente significativos como la identificación de componentes y especifica los elementos para el despliegue y soporte del sistema basados en los patrones arquitectónicos y de diseño seleccionados (UCID: SOFTWARE-DEFENSA, 2012).

3.4.1. Análisis preliminar de reutilización e identificación de componentes

La Arquitectura del Sistema tiene la responsabilidad de diseñar la integración de los elementos relacionados con el área de la arquitectura tecnológica y de presentación fundamentalmente mediante la identificación de sus componentes (UCID: SOFTWARE-DEFENSA, 2012). Se puede definir un componente como: *"Una parte modular, desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces"* (Pressman, 2005).

Se concibe al mismo Plugin como un componente en sí a insertarse en la plataforma GeneSIG, por lo cual asume su distribución en paquetes estándar que garantiza su correcta integración concentrando el núcleo del sistema en el componente **ServerExtractorSemantico.php** donde se implementa la lógica del negocio. Está basado en el principio de estandarización para el desarrollo de software estableciendo una política de desarrollo orientada a la reutilización de componentes y consumo de servicios.

Descripción de los principales componentes:

extractorSemantico.ajax.js, extractorSemantico.dynamic.js y extractorSemantico.action.js: Componentes que en conjunto conforman la interfaz visual del Plugin desde donde se interactúa de forma directa con el usuario.

ClientExtractorSemantico.php: Componente encargado de manejar las peticiones realizadas al servidor.

ServerExtractorSemantico.php: Componente que contiene la implementación de las funcionalidades del Plugin.

ExtractorSemantico.php: Componente encargado de estandarizar la comunicación entre el cliente y el servidor del Plugin.

CartoWeb: Componente que representa los componentes base de la plataforma utilizados en el Plugin.

BD Espacial: Componente que representa la base de datos asociada al sistema.

GeoOntoService.java, GeoOntoIndividualManager.java y GeoOntoManager.java: Componentes que en conjunto conforman el servicio que permite el manejo de la ontología.

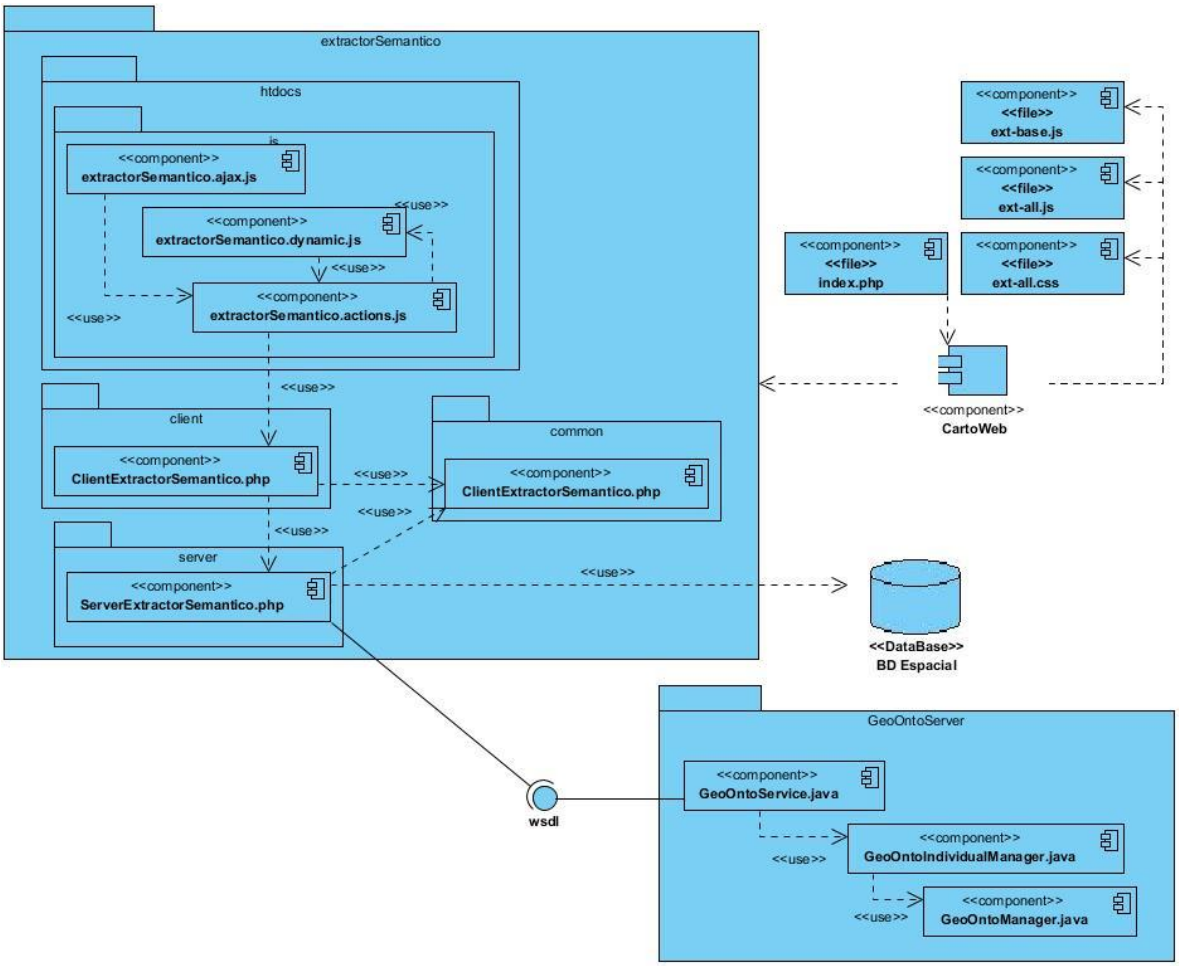


Fig. #3: Diagrama de Componentes

3.4.2. Descripción de los elementos de la Arquitectura del Sistema

La definición oficial de Arquitectura de Software propuesto por la IEEE 1471 resume que:

"La arquitectura de software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán y los principios que sustentan su diseño y evolución" (International Organization for Standardization, 2014).

Teniendo en cuenta que se empleará GeneSIG como herramienta base para el desarrollo y que esta fue desarrollada utilizando el framework CartoWeb es conveniente mantener la misma arquitectura para evitar futuras incompatibilidades. El framework CartoWeb implementa el estilo **Llamada Retorno** que define una arquitectura modular que enfatiza la modificabilidad y la escalabilidad, lo que permite poder separar la lógica de un servidor (cartoserver), encargado de la provisión de servicios, de un cliente (cartoclient). A continuación se expone una breve descripción de las principales características de los patrones **Orientado a Objetos** y **Basado en Componentes** adoptados de dicho estilo:

Patrón de arquitectura orientada a objetos: Los componentes del estilo se basan en los siguientes principios Orientados a Objetos: encapsulamiento, herencia y polimorfismo. Por lo tanto, los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación. Estos interactúan a través de invocaciones de funciones y procedimientos y conciben las interfaces separadas de las implementaciones (Boyd, y otros, 2013). Estas características permiten que se pueda modificar la implementación de un objeto sin afectar a sus clientes, descomponer problemas en colecciones de agentes en interacción así como utilizar los objetos como entidades reutilizables en el entorno de desarrollo.

Patrón de arquitectura basado en componentes: Dicho estilo enfatiza la portabilidad y se caracteriza por su modularidad, reusabilidad y compatibilidad. Considera a los componentes como las unidades de modelado, diseño e implementación y a sus interfaces e interacciones como el centro de incumbencias en el diseño arquitectónico. Dichas interfaces están separadas de las implementaciones y cada componente soporta algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución (Boyd, y otros, 2013). Esto convierte a GeneSIG en una plataforma flexible y personalizable, por cuanto está compuesto por Plugins; algunos son de obligatoria presencia para que pueda funcionar correctamente la plataforma y otros, como el que ocupa el presente trabajo, son creados de acuerdo a las funcionalidades que se desean añadir al sistema.

Además, como parte de la descripción de los elementos de la arquitectura se seleccionan los **Patrones de Diseño** a utilizar que contribuyan a crear un sistema más flexible, modular y reutilizable.

Primeramente los **Patrones GRASP** (del inglés: General Responsibility Assignment Software Patterns) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades (Larman, 2001). De estos se proponen utilizar los descritos a continuación:

Alta Cohesión: Concibe clases con responsabilidades moderadas en un área funcional que colaboran con otras clases para llevar a cabo las tareas (Larman, 2001). Constituye una premisa de GeneSIG donde cada componente realiza una labor única dentro del sistema y auto-identificable creando un sistema relativamente fácil de mantener, entender y reutilizar.

Bajo acoplamiento: Soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio (Larman, 2001). Constituye otra de las premisas de GeneSIG garantizando que cada componente dentro de la estructura del Plugin no dependan unos de otros a no ser estrictamente necesario.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos (Larman, 2001). Se aplica al crear nuevos objetos que en conjunto logren formar la representación visual del módulo en los componentes extractorSemantico.ajax.js, extractorSemantico.dynamics.js y extractorSemantico.actions.js así como al crear instancias que permitan el paso de información entre los componentes ServerExtractorSemantico.php y ClientExtractorSemantico.php.

Experto: Indica que la responsabilidad de la implementación de un método debe ser en la clase que contiene la información necesaria para cumplir con esta responsabilidad (Larman, 2001). Se aplica en **ServerExtractorSemantico.php** que tiene la información necesaria para el manejo directo con la base de datos espacial y la ontología que le permite dar solución a los distintos requerimientos funcionales.

Controlador: Representa una especie de fachada en la capa del dominio para la capa de la interfaz (Larman, 2001). O sea, puede evidenciarse en la forma en que la plataforma maneja las peticiones del usuario enviándolas al cliente del Plugin que les pueda dar solución, que en este caso sería **ClientExtractorSeamantico.**

Por otra parte los **Patrones de Diseño GOF** (del inglés: Gang of Four), que fueron identificados son:

Instancia única (Singleton): Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella (Larman, 2001). Este patrón se pone de manifiesto en **ServerExtractorSemantico.php** con el fin de garantizar que exista una única conexión con los servicios que manejan la ontología.

Patrón Comando (Command): Se utiliza para encapsular un comando en un objeto para que pueda ser almacenado, pasado a métodos y devuelto igual que cualquier otro objeto (Larman, 2001). Se evidencia su uso en **extractorSemantico.actions.js**, encargada de encapsular las peticiones que se hacen al servidor. En esta se ensamblan distintas acciones, para formar una acción compuesta que resulta de utilidad para el

manejo de las transacciones dentro del Plugin. Mediante el uso de este patrón se separa la petición de la acción que la satisface.

3.4.3. Diseño del modelo de despliegue

Se modela la distribución física de los nodos de hardware donde pueden ejecutarse los paquetes y componentes identificados. En el modelo se representa como se distribuyen los sistemas en cada nodo físico (UCID: SOFTWARE-DEFENSA, 2012).

En la solución a implementar como se observa en la Fig. #4 se pretende incluir el Plugin **extractorSemantico** a GeneSIG desde donde podrá ser usado por cualquier usuario que acceda a la misma vía web. Dicho Plugin permite poblar la ontología a través del consumo de servicios web proporcionados por un servidor especialmente dedicado a dicho propósito con datos extraídos de la base de datos espacial asociada a la plataforma.

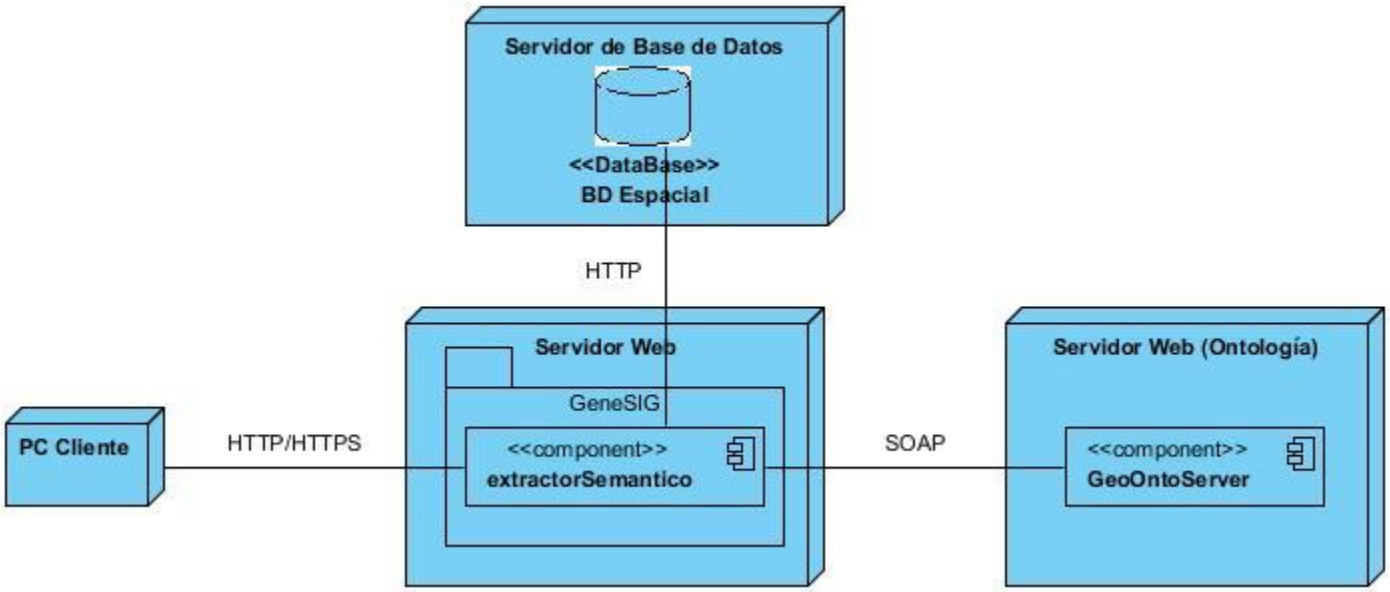


Fig. #4: Diagrama de Despliegue.

BD Espacial: Componente que contiene los datos espaciales a emplear en el proceso de extracción.

extractorSemantico: Componente para el poblado semi-automático de la ontología asociada al sistema.

GeoOntoServer: Componente especializado en el manejo de OntoMov, ontología encargada de representar de objetos móviles.

PC Cliente: Nodo desde el cual los usuarios pueden interactuar con el Plugin desarrollado por medio de las peticiones realizadas vía HTTP/HTTPS.

Servidor de Base de Datos: Nodo que contiene la base de datos asociada al Plugin.

Servidor Web: Nodo donde se encuentra ubicado GeneSIG.

Servidor Web (Ontología): Nodo encargado de soportar los servicios que permiten el poblado de la ontología.

3.5. Diseño de la Arquitectura de Datos

La Arquitectura de Datos precisa las clases de datos que apoyan las funciones del negocio definidas en el modelo de negocios (UCID: SOFTWARE-DEFENSA, 2012). Tiene como objetivo puntualizar los principales tipos y fuentes de datos de manera que sean: entendibles, consistentes y estables.

Para conformar dicha arquitectura se importaron distintas tablas al proyecto que conformaron una base de datos conjunta de donde se realiza el proceso de extracción de información. Su distribución se puede observar en el siguiente modelo de datos en la Fig. #5 donde la tabla “hurr” representa los huracanes almacenados, “province” hace referencia a las provincias y “municipality” a los municipios. Además se emplea la tabla “relation” para establecer las relaciones a tener en cuenta en el proceso de extracción.

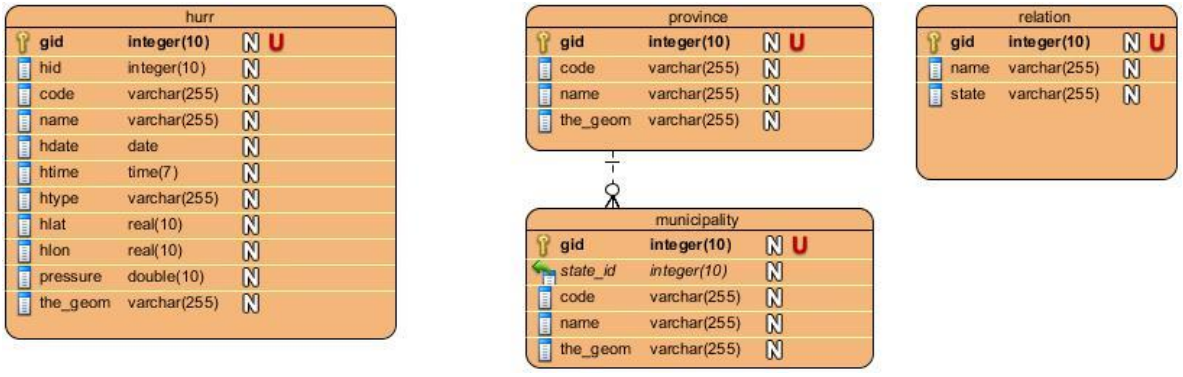


Fig. #5: Modelo de Datos

3.6. Diseño

El objetivo de esta sección es dejar el proyecto preparado para la implementación. Toma parte de los resultados de la fase de arquitectura, y termina con un diseño completo y listo para la implementación. Entre las actividades que tiene contenidas se encuentran (UCID: SOFTWARE-DEFENSA, 2012):

- Captura los requisitos o subsistemas individuales, interfaces y clases.
- Descompone los trabajos de implementación en partes más manejables.
- Crear un diseño utilizando una notación común.

3.6.1. Prototipos de Interfaz de Usuario

Prototipo de interfaz de usuario de los RF Insertar instancias dado un umbral e Insertar instancias dada una provincia: Está compuesto por un listado que contiene los huracanes almacenados en la base de datos del sistema en orden ascendente de acuerdo a su fecha de origen. Se muestra tanto el código como el nombre del huracán debido a la existencia de nombres repetidos. A continuación el usuario tiene la opción de seleccionar la opción por la cual desea realizar la extracción semántica pulsando en alguno de las opciones presentadas en el formulario, ya sea: “Por Umbral” o “Punto a punto” y posteriormente llenar sus campos correspondientes: ya sea el umbral de afectación o una provincia en específico. Consultar la Fig. #6.

Prototipo de interfaz de usuario del RF Actualizar relaciones espaciales: Está compuesto por un listado que contiene las relaciones espaciales a usar en el proceso de extracción semántica y otra que incluye aquellas relaciones que no se han de tener en cuenta. La dirección de las flechas que componen los botones centrales indica aquel que está dedicado a añadir o quitar relaciones de una lista para moverlos hacia la otra y viceversa. La interfaz culmina con una serie de botones en la parte inferior derecha que representan las acciones de Aplicar, Aceptar y Cancelar respectivamente. Consultar la Fig. #7.

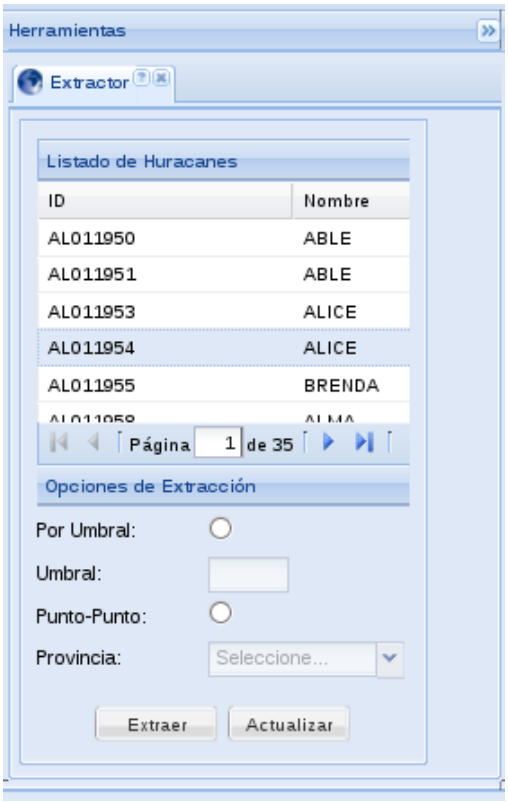


Fig. #6: Prototipo de interfaz de usuario de los RF Insertar instancias dado un umbral e Insertar instancias dada una provincia

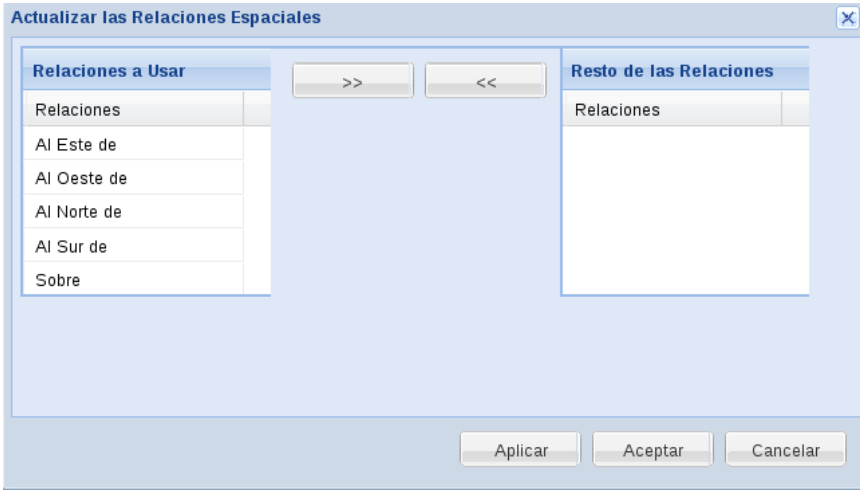


Fig. #7: Prototipo de interfaz de usuario del RF Actualizar relaciones espaciales

3.6.2. Diseño de Clases

En el diseño de clases se resume la definición de las clases que se pueden implementar en el software, se visualizan las relaciones entre estas y se muestra gráficamente la interacción de los objetos para comunicarse entre sí (UCID: SOFTWARE-DEFENSA, 2012). Basados en la especificación de requisitos y el modelo conceptual se elabora el diagrama de clases del diseño para cada requisito funcional del Plugin.

Como se ha abordado anteriormente el framework CartoWeb constituye la columna vertebral de la plataforma GeneSIG y por ello el proyecto adopta su estructura a la hora de diseñar sus componentes. En este caso se enfocará la atención en los Plugins dentro de GeneSIG, que como se analizó en capítulos anteriores, no son más que paquetes modulares de archivos (clases PHP, plantillas HTML, imágenes y otros recursos) que permiten añadir de forma convencional un conjunto de funcionalidades, que actúan como herramientas del propio sistema y le brindan la posibilidad de ser altamente modular y escalable.

Descripción de clases del diseño

El diseño propuesto se basa en la estructura física correspondiente a un Plugin de GeneSIG. Por ende se puede observar una separación entre el diseño de la interfaz mediante el uso de la librería ExtJS, el manejo de las peticiones del cliente y los métodos principales desarrollados en el servidor.

AxajPlugin.ExtractorSemantico: Clase encargada de inicializar y crear la interfaz visual del Plugin, crear las peticiones y procesar las respuestas del servidor.

ClientExtractorSemantico: Clase que maneja las peticiones del usuario y las respuestas del servidor sirviendo de intermediario.

ServerExtractorSemantico: Contiene la lógica del proceso incluyendo los métodos para comunicar al Plugin con la base de datos espacial y los servicios montados en Java para insertar instancias en la ontología.

GeoOntoService: Clase encargada de proveer los servicios de acceso a la ontología.

GeoOntoIndividualManager: Clase encargada de manejar las peticiones concernientes a los individuos.

GeoOntoManager: Clase encargada de gestionar directamente con la ontología.

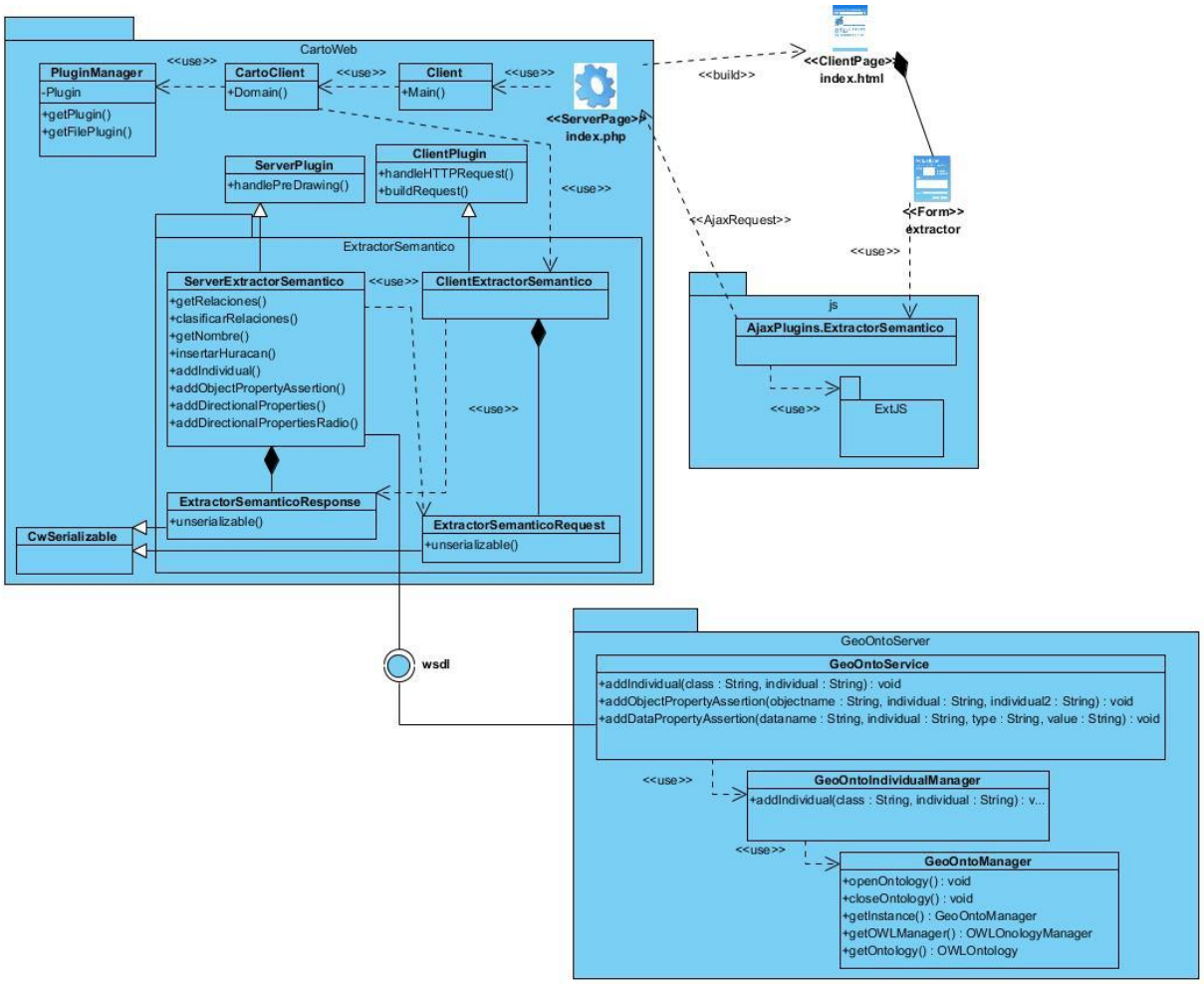


Fig. #8: Diagrama de Diseño. (RF1. Insertar Instancias dado un umbral y RF2. Insertar Instancias dada una provincia)

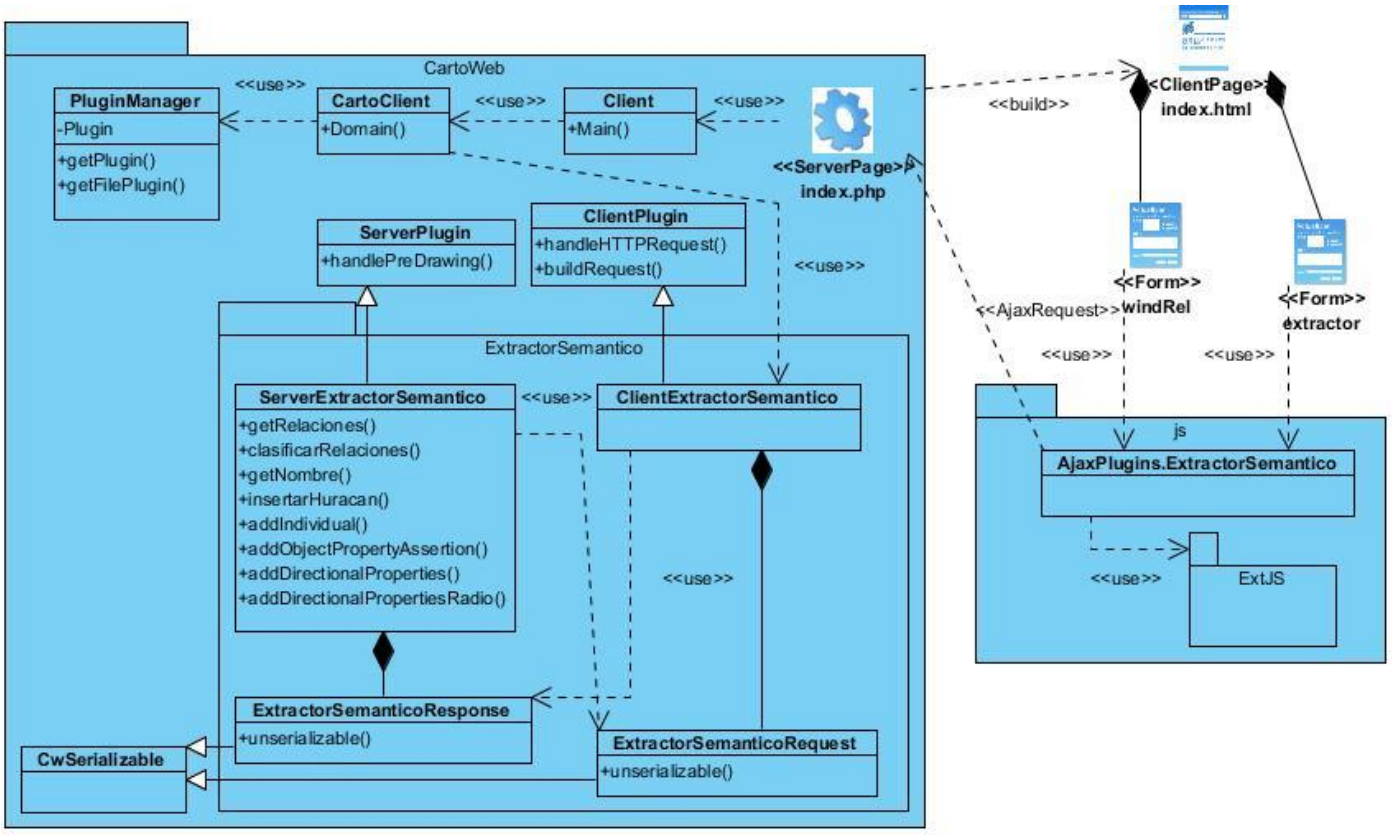


Fig. #9: Diagrama de Diseño. (RF3. Actualizar las relaciones espaciales)

3.7. Conclusiones Parciales

El diseño tanto lógico como físico de la arquitectura de la solución mediante el uso de los patrones seleccionados garantiza la compatibilidad del Plugin con la plataforma, el encapsulamiento y herencia de las clases, acorde a la arquitectura planteada por GeneSIG, y la correcta asignación de responsabilidades a cada componente. De esta forma, la modelación y el diseño crean la base para la implementación, como siguiente fase de desarrollo, que logra dar cumplimiento a los requisitos tanto funcionales como no funcionales descritos.

Capítulo 4: Implementación y Pruebas

4.1. Introducción

En el presente capítulo se analizan las fases de Implementación y Prueba del Plugin de Extracción Semántica de Objetos Móviles para GeneSIG. En la primera se tiene como propósito desarrollar el diseño propuesto para lo cual se tienen en cuenta los estándares de interfaces de usuario y de codificación adoptados en la implementación. Posteriormente se realizan las pruebas que permitan verificar la correcta implementación de los requisitos establecidos y corrobora los planteamientos de la hipótesis de la investigación. El desarrollo de la propuesta de solución concluye con una valoración de los resultados obtenidos durante esta fase.

4.2. Implementación

La Implementación comienza con el resultado obtenido del diseño detallado. El objetivo principal de esta fase es desarrollar el diseño de la arquitectura propuesto y el sistema como un todo. De forma más específica, los propósitos de la Implementación son (UCID: SOFTWARE-DEFENSA, 2012):

- Planificar las integraciones de sistema necesarias en cada iteración. Siguiendo para ello un enfoque incremental.
- Implementar clases, componentes y subsistemas encontrados durante el diseño.
- Integrar componentes y el sistema como un todo manteniendo como principio la estandarización y reutilización de códigos y componentes

4.2.1. Estándares de Interfaces de Usuario

Reglas a cumplir

1. La interfaz corresponderá con las características, metas y nivel de experiencia de los usuarios de la aplicación, siempre que no se atente contra las pautas para el diseño.
2. La información estará libre de errores gramaticales, ortográficos y tipográficos.
3. La taxonomía de los títulos encima de los controles estará según la estipulada, primera letra con mayúscula, el resto con minúscula.

4. En controles de tipo input o select cambiará el color de fondo indicando que el cursor está situado en él para modificar los datos. Una vez que el cursor deja de estar situado sobre estos, regresarán a su estado normal. (Garantizado por ExtJS).
5. En controles de tipo button cambiará el color de fondo indicando que el cursor está situado en él para presionar. Una vez que el cursor deja de estar situado sobre estos, regresarán a su estado normal. El cursor mantendrá la forma de puntero. (Garantizado por ExtJS).
6. Los iconos se corresponderán con las tareas a realizar.
7. Los nombres de los botones de comandos no serán abreviaturas y tendrán un significado claro para los usuarios del sistema.
8. Los botones de comando serán de tamaño y forma similares. (Garantizado por ExtJS).
9. Los nombres de los botones de opción (radiobutton) tendrán un significado claro para los usuarios del sistema.
10. Se define el español como idioma del sistema y se empleará en todos los textos.

Reglas a cumplir en componentes de tipo "grid"

1. Siempre utilizará una máscara (loadMask) Indicando siempre que se encuentre cargando.
2. La etiqueta mostrará un mensaje que dirá "Cargando…". (Garantizado por ExtJS).
3. Siempre que se cargue un grid, automáticamente se seleccionará la primera fila del conjunto de resultados.

Reglas a cumplir en componentes de tipo "formulario"

1. En un formulario siempre se utilizará una máscara (waitMsg) indicando cuando se encuentre ocupado intercambiando con el servidor.

Reglas a cumplir referente a "validaciones"

1. El mensaje de validación de un campo incorrecto se colocará en un "tooltips" correspondiente.
2. Si el usuario da clic en aplicar o aceptar y en el formulario hay campos no válidos, automáticamente los que falten por validar se marcaran como no válidos y se mostrará un mensaje de error que dirá: "Por favor verifique nuevamente que hay campos con valores incorrectos".

4.2.2. Estándares de Codificación

El uso de estándares de codificación permite que distintos desarrolladores puedan entender el código resultante en menos tiempo y que el código en consecuencia sea mantenible. A continuación son descritos algunos de los estándares de codificación que se tuvieron en cuenta en la implementación de la solución:

Se emplean **comentarios** para describir la codificación de la solución mejorando su comprensión por otros programadores. Teniendo en cuenta que los lenguajes que se utilizan son JavaScript y PHP, se definen dos tipos principales de comentarios:

De una sola línea:

//esto es un comentario

De varias líneas:

/ Esto es un comentario*

de varias líneas

**/*

Para nombrar las funciones y las clases se emplea el estándar **CamelCase** adoptado por la plataforma GeneSIG. En el caso particular de las funciones se emplea la forma **lowerCamelCase**, mientras las clases son nombradas utilizando la forma **UpperCamelCase** en inglés-español.

4.2.3. Ejemplos de instancias insertadas

Teniendo en cuenta que el Plugin desarrollado tiene como centro el poblado de la ontología asociada con instancias y las relaciones que se establecen entre ellas, se muestra a continuación un ejemplo en las Fig. #10 y la Fig. #11 de algunos de los resultados obtenidos observables mediante la herramienta Protegé.

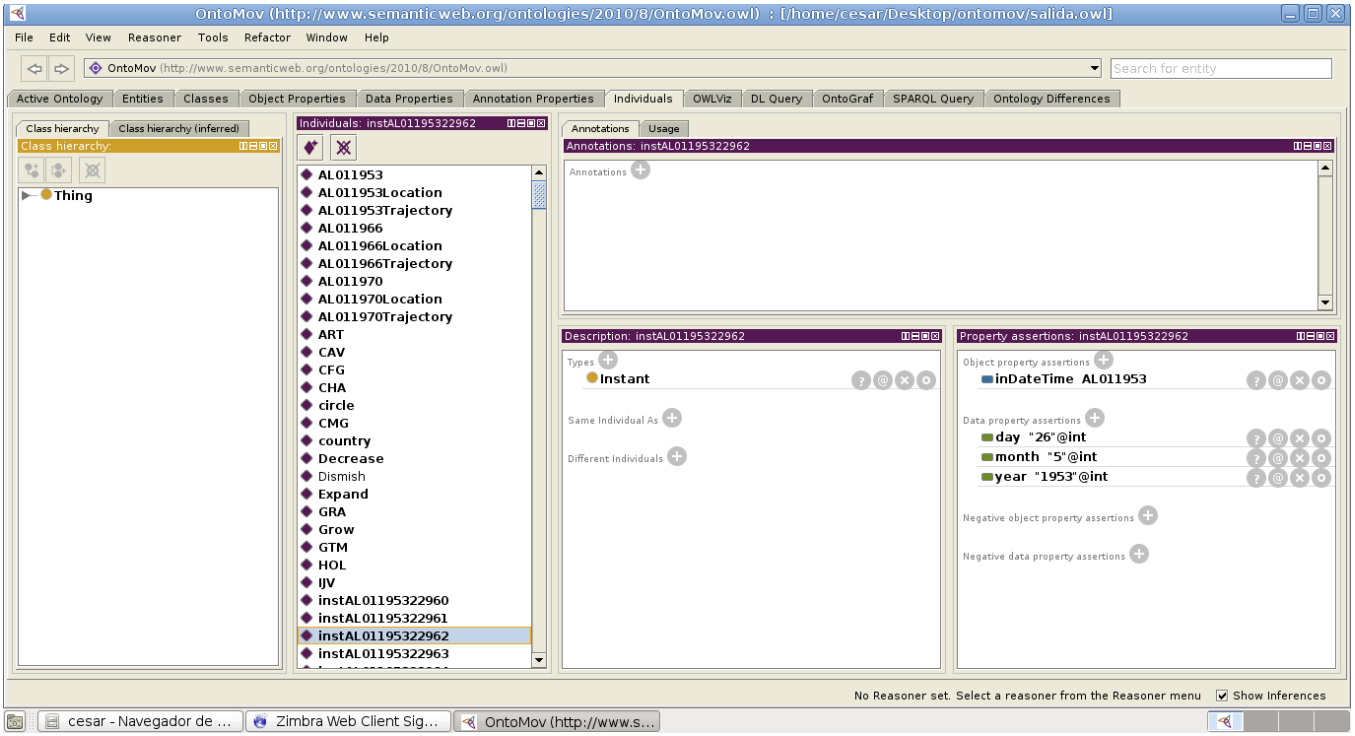


Fig. #10: Ejemplos de instancias y propiedades insertadas.

Son observables algunas instancias insertadas con el código del huracán como elemento clave y es notable también como una instancia del huracán AL011953 representada con la siguiente notación: inst + código del huracán + código del punto de la trayectoria, tiene asociada como propiedad el día, mes y año de ocurrencia.

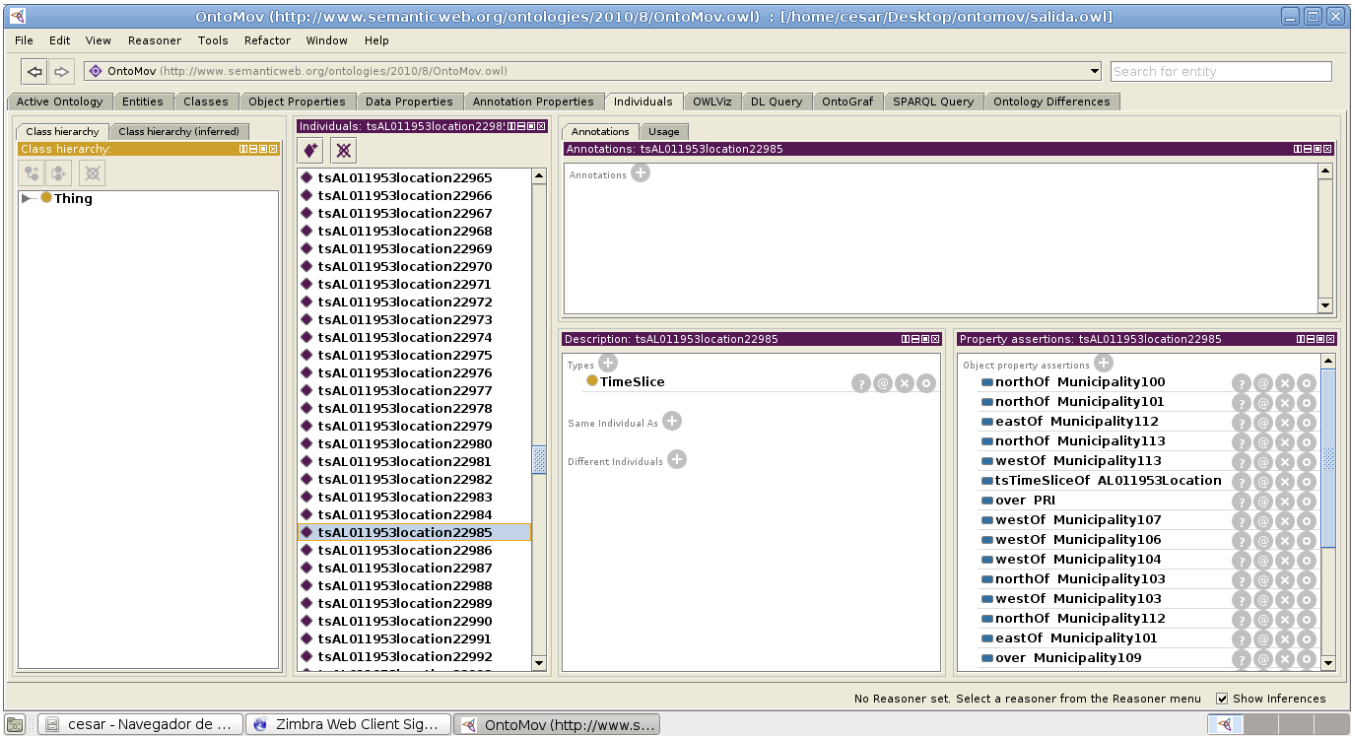


Fig. #11: Ejemplos de relaciones insertadas.

En este caso es observable como cada punto del huracán contiene una localización que se encarga de almacenar las relaciones del mismo con las distintas provincias y municipios. Por ejemplo la localización del punto de la trayectoria 22985 del huracán AL011953 se encuentra por encima de Pinar del Río, específicamente atravesando el municipio que tiene por código 109.

4.3. Pruebas

Las pruebas del software son una actividad de control de la calidad realizada en los proyectos para asegurar el correcto funcionamiento del software. Tienen como objetivos la verificación de la correcta implementación de los requisitos explícitamente establecidos, la adecuada integración de los componentes que conforman el sistema y la ejecución de casos de prueba que permitan detectar el mayor número de No conformidades y corregirlas antes de la entrega del software al cliente (UCID: SOFTWARE-DEFENSA, 2012).

Teniendo en cuenta las características particulares de las funcionalidades implementadas en el Plugin para la extracción de información semántica de objetos móviles para GeneSIG y la hipótesis a contrastar, que

plantea la disminución del tiempo empleado y la cantidad de errores en el proceso de poblado de la ontología, se realizaron las pruebas descritas a continuación:

Las pruebas de **Caja Negra** se aplican a la interfaz del software permitiendo obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa con el fin de encontrar la mayor cantidad de no conformidades existentes en el producto (Pressman, 2005).

Se empleó la técnica de **Partición Equivalente**, que se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2005). Los diseños de casos de prueba utilizados en esta etapa se pueden observar en las tablas: Tabla #4, Tabla #5 y Tabla #6 del presente capítulo.

4.3.1. Diseños de Casos de Prueba

Un diseño de caso de pruebas no es más que un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para determinar si el requisito de una aplicación está parcial o completamente satisfecho (UCID: SOFTWARE-DEFENSA, 2012). Esta fase resulta esencial en la validación del Plugin desarrollado por lo cual se ha diseñado un caso de prueba para cada requisito del sistema con el objetivo de asegurar que todas las funcionalidades sean comprobadas. A continuación se muestra un ejemplo de los diseños de casos de prueba realizados para la funcionalidad **Insertar Instancias dado un umbral**.

Tabla #4: DCP para el requisito Insertar Instancias dado un umbral.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Insertar instancias dado un umbral	Inserta instancias de los puntos de la trayectoria del huracán y sus relaciones espaciales con las provincias y municipios dentro de un umbral dado en la ontología.	EP 1.1: Se insertan las instancias correctamente.	1. Se activa el Plugin. 2. Se selecciona el huracán a analizar. 3. Se selecciona la opción de inserción por Umbral. 4. Se introduce el radio de alcance a

			contemplar en el proceso. 5. Se hace clic en el botón Extraer. 6. Se insertan las instancias.
		EP 1.2: No se insertan las instancias.	1. Se activa el Plugin. 2. Se selecciona el huracán a analizar. 3. Se selecciona la opción de inserción por Umbral. 4. Se introduce el radio de alcance a contemplar en el proceso. 5. Se hace clic en el botón Extraer. 6. El umbral no se encuentra entre los valores permitidos. Se muestra un mensaje de error. No se insertan las instancias.

Tabla#5: Descripción de variables para el requisito Insertar instancias dado un umbral.

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción del campo
1	umbral	campo de texto	No	Datos numéricos enteros.

Tabla #6: Juego de datos a probar para el requisito Insertar Instancias dado un umbral.

Id del escenario	Escenario	Variable 1 umbral	Respuesta del sistema
EP 1.1:	Se insertan las instancias correctamente.	V [0, 1000]	1. Se acepta el valor y continúa la funcionalidad.
EP 1.2:	No se insertan las instancias.	I (1000, +)	2. Se rechaza el valor mostrando un mensaje de error. La funcionalidad se detiene.

Tabla #7: DCP para el requisito Insertar Instancias dada una provincia.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Insertar instancias dada una provincia.	Inserta instancias de los puntos de la trayectoria del huracán y sus relaciones espaciales con la provincia seleccionada y sus municipios en la ontología.	EP 1.1: Se insertan las instancias correctamente.	1. Se activa el Plugin. 2. Se selecciona el huracán a analizar. 3. Se selecciona la opción de inserción Punto a Punto. 4. Se selecciona la provincia a analizar.

			5. Se hace clic en el botón Extraer. 6. Se insertan las instancias.
		EP 1.2: No se insertan las instancias.	1. Se activa el Plugin. 2. Se selecciona el huracán a analizar. 3. Se selecciona la opción de inserción Punto a Punto. 4. Se hace clic en el botón Extraer. 5. No se ha seleccionado la provincia. Se muestra un mensaje de error. No se insertan las instancias.

Tabla #8: Descripción de variables para el requisito Insertar instancias dada una provincia.

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción del campo
1	provincia	listado	No	Cadena de caracteres.

Tabla #9: Juego de datos a probar para el requisito Insertar Instancias dada una provincia.

Id del escenario	Escenario	Variable 1 provincia	Respuesta del sistema
EP 1.1:	Se insertan las instancias correctamente.	V "PRI"	1. Se acepta el valor y continúa la funcionalidad.
EP 1.2:	No se insertan las instancias.	I -	2. Se rechaza el valor mostrando un mensaje de error. La funcionalidad se detiene.

Tabla #10: DCP para el requisito Actualizar relaciones espaciales.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Actualizar relaciones espaciales.	Se actualizan las relaciones espaciales a tener en cuenta en el proceso de inserción a partir de aquellas seleccionadas por el usuario.	EP 1.1: Se actualizan las relaciones espaciales.	1. Se activa el Plugin. 2. Se hace clic en el botón Actualizar. 3. Se selecciona las relaciones a utilizar. 4. Se selecciona la opción **Aplicar.** 4.1. Se selecciona la opción **Aceptar** y colapsa la ventana.

			5. Se actualizan las relaciones a utilizar.
		EP 1.2: No se actualizan las relaciones espaciales.	1. Se activa el Plugin. 2. Se hace clic en el botón Actualizar. 3. Se selecciona las relaciones a utilizar. 4. Se selecciona la opción Cancelar. 5. Se cancelan los cambios realizados y se colapsa la ventana.

Tabla #11: Descripción de variables para el requisito Actualizar relaciones espaciales.

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción del campo
1	relaciones	listado	No	Arreglo de relaciones

Tabla #12: Juego de datos a probar para el requisito Actualizar relaciones espaciales.

Id del escenario	Escenario	Variable 1 relaciones	Respuesta del sistema
EP 1.1:	Se actualizan las relaciones correctamente.	V	1. Se actualizan las relaciones con el listado entrado.

		[“Este de”, “Sobre”, “Sur de”]	
EP 1.2:	No se actualizan las relaciones.	V -	2. Se rechaza el valor del listado y las relaciones se mantienen en su forma original.

4.3.2. Resultados de las pruebas realizadas

Los diseños de casos de prueba, una vez ejecutados, permitieron resaltar no conformidades relacionadas a las validaciones de los formularios asociados a los requisitos funcionales. Estas fueron corregidas en su totalidad luego de 4 iteraciones como se puede observar en la Fig. #12.

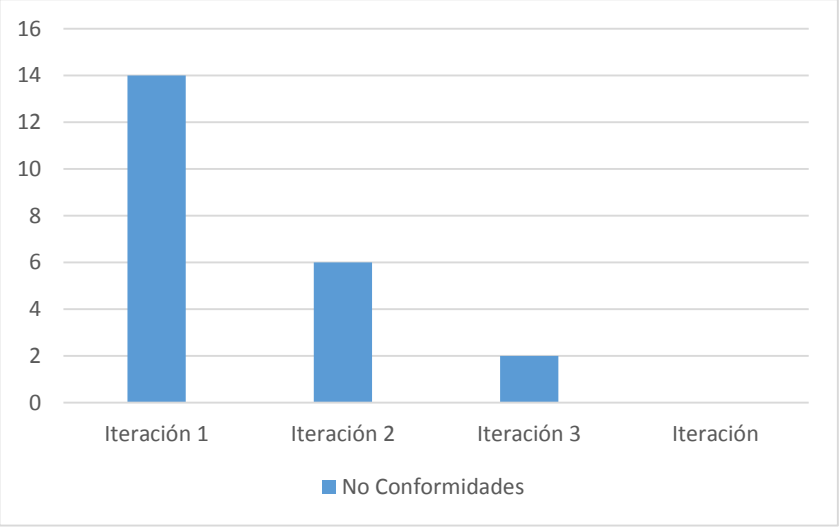


Fig. #12: Pruebas de Caja Negra. No Conformidades detectadas en cada iteración.

Cabe destacar que debido a la simplicidad de las interfaces y la abstracción del usuario referente a los procedimientos, el gran cúmulo de pruebas realizadas está concentrado en la verificación de las capacidades del sistema en cuanto a rendimiento y confiabilidad. Estos datos permitirían realizar una comparación entre las características del proceso realizado de forma manual y los resultados obtenidos con el uso del Plugin que permitan contrastar la hipótesis planteada:

"*El desarrollo de un Plugin para la extracción de información semántica de objetos móviles para GeneSIG a partir de los datos geográficos de huracanes almacenados permite la disminución de* ***errores*** *y del* ***tiempo*** *empleado en el proceso de poblado de su geo-ontología asociada.*"

Para lograr esto en un primer lugar era necesario partir del tiempo aproximado que le toma a un especialista realizar este proceso. Se toma una muestra de 4 especialistas en el manejo de la ontología de la facultad para insertar las instancias y relaciones de varios huracanes que cuenta con un aproximado de 100 instancias cada uno. A continuación se muestran los resultados de dicho experimento enfocado en las habilidades de los especialistas abstrayéndose de las capacidades de hardware:

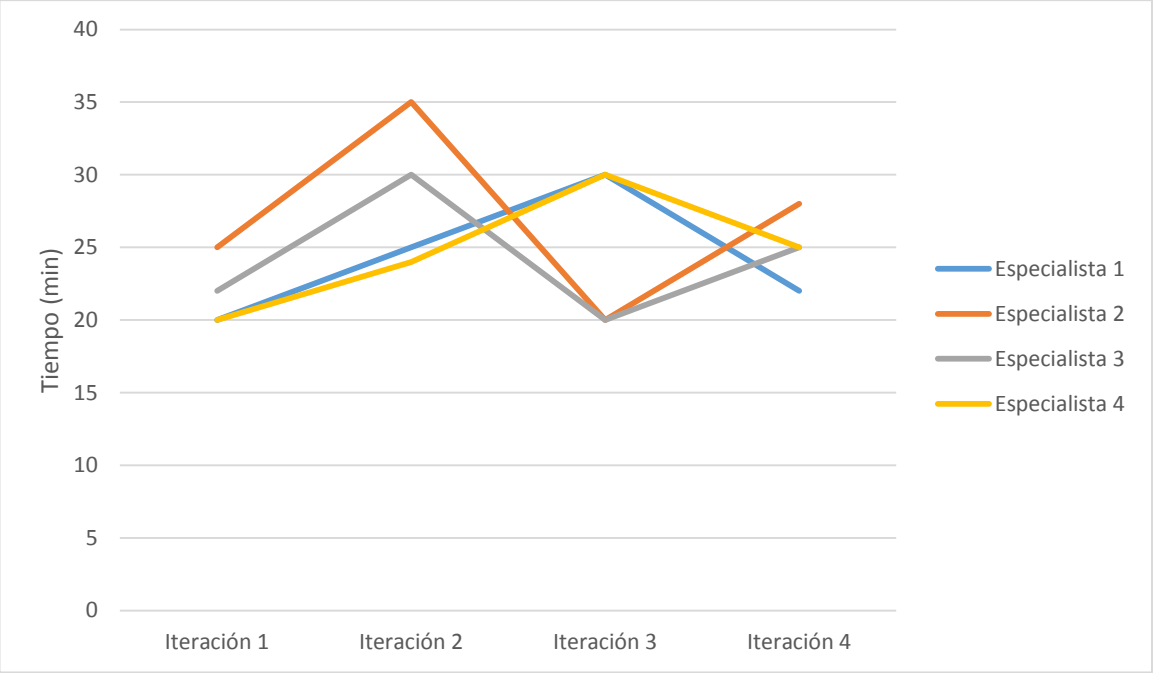


Fig. #13: Cantidad de instancias insertadas de forma manual.

Como se pudo observar el proceso de poblado de la ontología para la creación de una base de conocimiento puede resultar en un proceso largo y complejo inclusive para una persona experimentada en el campo llegándose a demorar un promedio de 25 minutos. Para contrastar este valor con los resultados obtenidos con el desarrollo del Plugin se aplicaron Pruebas de Rendimiento al sistema haciendo uso de una computadora con un procesador AMD a 1 GHz y 4 GB de memoria RAM como servidor y varias PC cliente con distintos navegadores web y sistemas operativos, con el objetivo de medir el tiempo de renderizado.

En el primero de los casos se hizo por el método convencional o común, utilizándose el siguiente código empleando funciones del API de la consola de Firebug que sirven para calcular el tiempo de ejecución (benchmark en inglés) de código JavaScript**:**

```
function test(){
var num = '0';
var count = 5;
console.time(num);
while(count--){
var radio = Math.random()*1000;
                    Genesig.ajax.triggerLight('ExtractorSemantico.Insertar', {
                        opcion: Ext.encode('1'),
                        codigo: Ext.encode(data[1][Math.random()*data.size()]),
                        param: Ext.encode(radio)
                    });
console.timeEnd(num);
}
}
```

Fig. #14: Código de la prueba de rendimiento por el método convencional.

Los resultados obtenidos se muestran en la Tabla#13. Se realizaron varias iteraciones utilizando distintos huracanes elegidos de forma aleatoria, usándose para esto varias plataformas: los Sistemas Operativos Windows 8 y Linux Debian 6 y los navegadores Web Firefox, Iceweasel y Chrome en varias versiones.

Tabla #13: Resultados de la prueba de rendimiento convencional.

	67 instancias	80 instancias	93 instancias	100 instancias
Iceweasel 33.0 Debian 6	0.68	0.97	1.00	1.00
Firefox 36.0 Ubuntu 14.04	0.75	1.00	1.10	1.15
Chrome 36.0 Ubuntu 14.04	0.78	1.05	1.17	1.20
Firefox 36.0 Windows 8	0.84	1.15	1.42	1.50
Chrome 38.0 Windows 8	0.80	1.23	1.43	1.45
Promedio	0.77	1.08	1.22	1.26

Por otro lado, se utilizó con el mismo fin, el software JSLitmus, el mismo determina el rendimiento de un fragmento de código ejecutándolo repetidamente hasta que ha pasado una suficiente cantidad de tiempo con el fin de obtener resultados válidos. Su ejecución permite saber cuántas operaciones por segundo está manejando la computadora igualmente con una cantidad aleatoria de instancias insertadas. Los resultados obtenidos se muestran en la Tabla #14. A continuación el fragmento de código:

```
JSLitmus.test('Extractor', function(){
var count = 5;
while(count--){
var radio = Math.random()*1000;
                    Genesig.ajax.triggerLight('ExtractorSemantico.Insertar', {
                        opcion: Ext.encode('1'),
                        codigo: Ext.encode(data[1][Math.random()*data.size()]),
                        param: Ext.encode(radio)
                    });
}
});

```

Fig. #15: Código de la prueba de rendimiento usando JSLitmus.

Tabla #14: Resultados de la prueba de rendimiento de JSLitmus.

	65 instancias	84 instancias	100 instancias
Iceweasel 33.0 Debian 6	0.80	1.10	1.20
Firefox 36.0 Ubuntu 14.04	0.75	1.00	1.10
Chrome 36.0 Ubuntu 14.04	0.78	1.05	1.25
Firefox 36.0 Windows 8	0.84	1.20	1.60
Chrome 38.0 Windows 8	0.80	1.23	1.35
Promedio	0.80	1.12	1.30

Para una mejor comprensión se pueden observar estos resultados en la Fig. #16. Es posible observar que la inserción de 100 instancias se realiza en un tiempo promedio de 1.28 minutos, con lo cual se comprueba el buen rendimiento del sistema en comparación a su realización de forma manual. También es observable como el tiempo de renderizado aumenta a medida que aumentan la cantidad de instancias que es proporcional a la cantidad de puntos en la trayectoria de un huracán determinado.

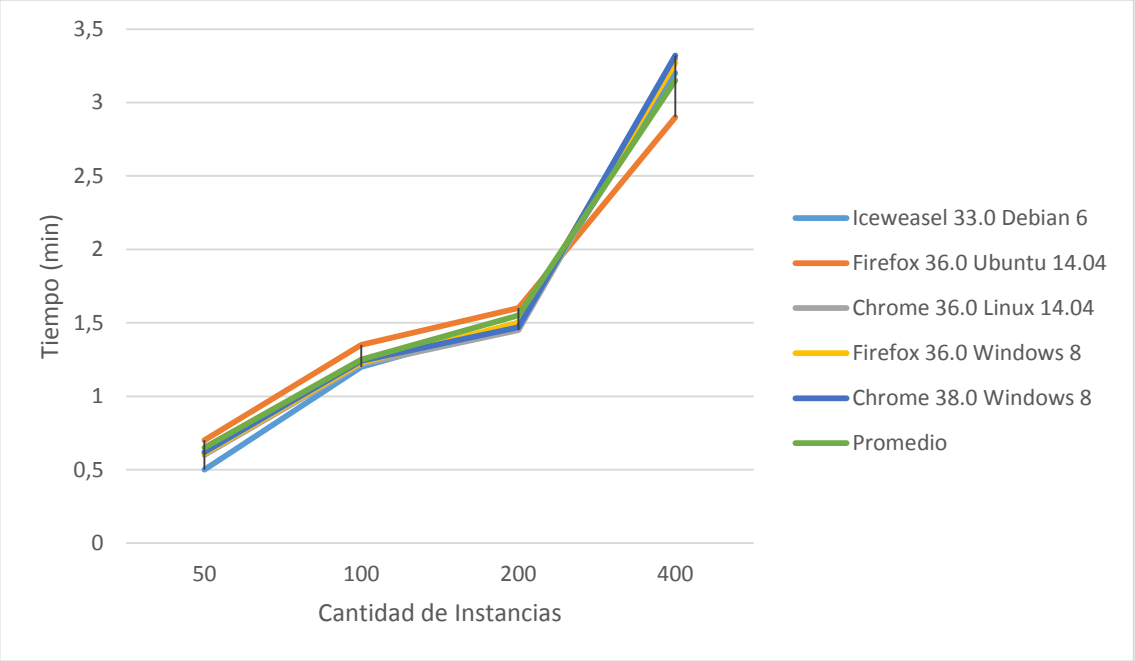


Fig. #16: Rendimiento del Plugin en distintos entornos.

Por otra parte, para adentrarse en el tema de los errores en la inserción de datos se debe partir de la siguiente afirmación: “Una de las circunstancias más frustrantes en cualquier entorno de trabajo son los procesos manuales repetitivos y largos que a menudo conducen a errores humanos que requieren extensa dedicación de recursos para tratar de identificar y rectificar” (Dekker, 2000). Desde la perspectiva de la Seguridad se identifican a los errores humanos como riesgos potenciales y alude al comportamiento de las personas que excede el límite de tolerancia definido para la seguridad de un sistema. Según (Swain, y otros, 1983) se distinguen las cinco categorías siguientes:

- Error de omisión.
- Error de ejecución.
- Error de corrección.

- Error de secuencia.
- Error de demora

Sin embargo este término continúa presentando cierta ambigüedad, pues puede encubrir aspectos relativos a la organización del trabajo, al diseño del sistema, la variabilidad de conducta de la persona (en función de su resistencia a la fatiga, sus características psicológicas, afectivas y cognitivas, sus actitudes, su formación y experiencia) y la variabilidad interpersonal, que pueden manifestarse como elementos de infiabilidad difícilmente controlables a través de la selección y formación de personal según (de Arquer , y otros, 1994). A esta variabilidad intra e interpersonal se suma la imposibilidad de intervenir en medios ambientales "extremos" (radiactivos, grandes profundidades, condiciones climáticas límites, etc.). En este caso específico la cantidad de errores en el poblado de la ontología se ven multiplicados al añadir una extensa cantidad de información técnica, poca familiarización con la herramienta y otros factores externos como el sueño y la fatiga. Además, es necesario tener en cuenta que una persona a pesar de sus características personales intrínsecas sigue un grupo de pasos para el procesamiento de la información que dan cabida a la existencia de errores según (de Arquer , y otros, 1994):

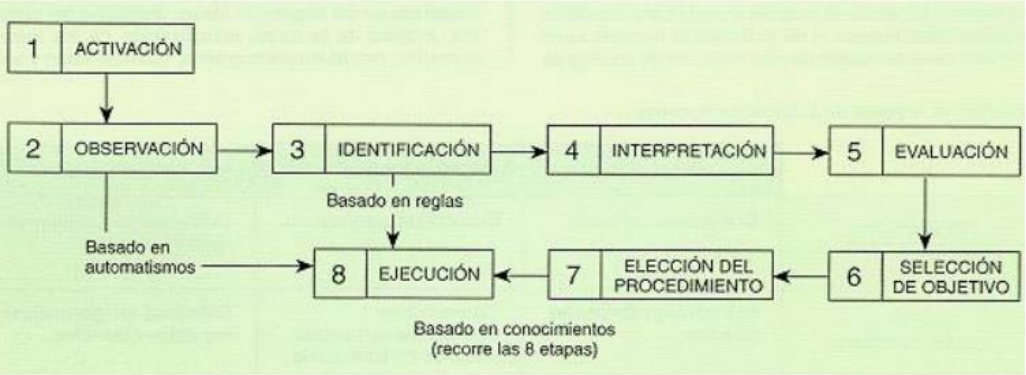


Fig. #17: Etapas de tratamiento de información y toma de decisiones de la persona.

Unas de las posibles soluciones a este aspecto son diseñar sistemas tolerantes, en los que cuando se produzca un error humano, éste sea detectable y se pueda recuperar antes de que tenga consecuencias inaceptables o automatizar el proceso, de forma tal que los procedimientos logren aislarse completamente de la intervención humana. Como se pudo observar en el análisis de los distintos sistemas de creación de bases de conocimientos en el Capítulo 1, este consiste en un proceso que se puede realizar abstrayéndose

totalmente de la intervención de un usuario o con una intervención mínima. En este caso los datos a ingresar son extraídos de las distintas bases de datos espaciales y tanto su sintaxis como la información en sí son definidas por un algoritmo que no da cabida a la inserción de información errónea o imprecisa. El proceso se vuelve semi-automatizado, ya que el usuario es el encargado de proporcionar los datos que definen la forma en la que se realizará la extracción de la información, sin embargo las validaciones en la entrada de datos y la simpleza de la interfaz de usuario dotan al sistema de una infalibilidad casi total.

Con el fin de corroborar esta afirmación se analizaron los resultados del experimento realizado con los 4 especialistas de la facultad y la revisión a las instancias insertadas en la ontología arrojaron distintos errores en su mayoría de tipo gramatical (Omisiones de signos o errores ortográficos) así como errores en la inserción de los distintos códigos de los huracanes debido a la extensa cantidad de información. Sin embargo la información insertada mediante la utilización del Plugin para la extracción de información semántica de objetos móviles de GeneSIG arrojó como principal resultado la construcción de una base de conocimientos libre de inconsistencias eliminando totalmente la ocurrencia de errores en el manejo de los datos.

4.4. Conclusiones Parciales

La descripción de los principales estándares de codificación permitió optimizar el proceso de implementación y favorecer futuras etapas de mantenimiento de software. Además, mediante la aplicación de Pruebas de Caja Negra mediante la técnica de Participación Equivalente se pudieron corregir las principales no conformidades que lograron satisfacer los requisitos funcionales identificados. Una vez corroborada la hipótesis, por cuanto logra mejorar el proceso de poblado de la ontología tanto en tiempo como en la veracidad de la información insertada, se obtiene el producto final a entregar al cliente como resultado principal de la investigación.

Conclusiones Generales

Fue culminado satisfactoriamente el desarrollo del Plugin para la extracción de información semántica de objetos móviles para GeneSIG, componente que mejora el proceso de poblado de la ontología asociada a un SIGGO especializado en el análisis de la trayectoria y nivel de afectaciones de los huracanes en Cuba, por lo cual se considera cumplido el objetivo principal del presente trabajo.

- El análisis de los principales sistemas existentes para el poblado de ontologías empleadas como bases de conocimientos, demostró en primer lugar lo poco que se ha explorado el campo referente al trabajo con objetos móviles y en segundo lugar reafirmó la viabilidad del proyecto a pesar de la incapacidad de adaptar dichos sistemas a la situación particular del presente trabajo debido a que su funcionamiento responde a las características específicas su dominio.
- Se emplearon herramientas y tecnologías correspondientes al desarrollo de aplicaciones web adoptando las características especiales de la plataforma así como aquellas que permitieron el consumo de servicios desplegados en java. Fue seleccionado PRODESOFT como proceso de desarrollo de software capaz de regir el proyecto mediante la generación de artefactos en cada etapa de su ciclo de vida. Al estar concebido como un Plugin de GeneSIG asume su arquitectura así como los estándares de codificación, de diseño de interfaces y de mensajería establecidos para su implementación que favorecen sus futuras etapas de mantenimiento.
- La realización de las distintas pruebas de software permitió comprobar el cumplimiento de los requisitos funcionales y no funcionales en la solución y a partir de la información recopilada establecer una comparación que permitiera corroborar la hipótesis planteada. Cabe resaltar el buen rendimiento del sistema en los distintos entornos analizados en comparación a la alternativa de realizar el proceso de forma manual.
- La solución obtenida logró poner en práctica a OntoMov como una base de conocimiento funcional cuya utilización permite realizar un análisis semántico de la trayectoria y nivel de afectaciones de los huracanes en Cuba dentro de un SIGGO. Representa además un aporte dentro de las investigaciones referentes a la representación del dominio de objetos móviles.

Recomendaciones

Una vez concluida la investigación se recomienda:

- Extender el uso de OntoMov a otros SIG que se desarrollen usando la plataforma GeneSIG.
- Adaptar los algoritmos aplicados para el poblado de ontologías utilizando otras fuentes de datos que incluyan nuevos dominios de aplicación.

Referencias Bibliográficas

Palmisano , Ignazio . 2015. GitHub-Build software better, together. [En línea] GitHub, Inc., 9 de 5 de 2015. [Citado el: 12 de 6 de 2015.] http://owlapi.sourceforge.net/documentation.html.

Achour , Mehdi, y otros. 1997-2015. PHP: Hypertext Preprocessor. [En línea] the PHP Documentation Group, 1997-2015. [Citado el: 4 de 6 de 2015.] http://php.net/manual/en/intro-whatcando.php.

Alhir, Sinan Si . 2003. *Learning UML.* s.l. : O'Reilly, 2003.

Boyd, Sean , y otros. 2013. *SOFTWARE ARCHITECTURAL STYLES.* 2013.

Camptocamp SA. 2005. *CartoWeb Documentation-3.0.0 Edition.* 2005.

Candeaux Duffatt, Rafael y Díaz Cisneros, Luis R. 1994. Mappin Interactivo. [En línea] 1994. [Citado el: 2015 de 6 de 8.] http://www.mappingi nteractivo.com/plantillaante.asp?id_articulo=1184.

Castano, Silvana, y otros. 2008. *Instance Matching for Ontology Population.* Milano : s.n., 2008.

Davies, John , Grobelnik, Marko y Dunja , Mladenić. 2009. *Semantic Knowledge Management.* 2009.

de Arquer , M. Isabel y Nogareda, Clotilde . 1994. *Fiabilidad humana: conceptos básicos.* 1994.

Dekker, Sidney. 2000. *The Field Guide to Human Error.* Bedford : Cranfield University Press, 2000.

Englander , Robert . 2002. *Java and SOAP.* s.l. : O'Reilly, 2002.

Fonseca, Frederico T. y Egenhofer, Max J. . 1999. *Ontology-Driven Geographic Information Systems.* 1999.

Garea Llano, Eduardo y Larin Fonseca, Rainier. 2010. *Relaciones topológicas para generación automática de ontologías en SIG.* 2010.

Garea Llano, Eduardo y Oliva Santos, Rafael. 2009. *Integración Ontológica de Datos Metadatos y Conocimiento en Sistemas de Información Geográfica como Herramienta para la Interpretación Semántica de la Información Espacial.* 2009.

—. 2007. *Geo-ontologies as Integration Structures of Knowledge, Data, and Metadata.* 2007.

Garea Llano, Eduardo, Oliva Santos, Rafael y Maciá Pérez, Francisco. 2009. *Ideas para una arquitectura de persistencia basada en geo-ontologías para anotaciones semánticas de datos geográficos.* 2009.

Garrete, José Luis, Palomino, Carmen y Caba, Hilda Araceli. 2009. *Sistema de Información Geográfica en la Gestión Integral del Litoral.* Madrid : s.n., 2009.

Ghawi, Raji y Cullot, Nadine. 2007. *Database-to-Ontology Mapping Generation.* University of Burgundy : s.n., 2007.

Gosling, James , y otros. 2015. *The Java Language Specification-Java SE 8 Edition.* Redwood : Oracle America, Inc., 2015. Vol. 8.

Gruber, TR. 1993. *A translation approach to portable ontology specification.* 1993.

Guarino , N. 1998. *Formal Ontology in Information Systems.* s.l. : IOS Press, 1998.

Guarino, N. 1998. *Formal Ontology in Information Systems.* Trento : IOS Press, 1998.

International Organization for Standardization. 2014. *ISO/IEC 25000:2014-Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)-Guide to SQuaRE.* 2014.

Larman, Craig. 2001. *UML y Patrones-Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall, 2001. Vol. 2.

Laurie , Ben y Laurie, Peter . 1999. *Apache.* [ed.] Robert Denn. s.l. : O'Reilly, 1999.

López Sarabia, Gerardo. 2008. *Búsqueda y ponderación de información contenida en Bases de datos Espaciales, utilizando jerarquía.* 2008.

Mozilla Developer Network. 2005-2015. MND-Mozilla Developer Network. [En línea] 2005-2015. [Citado el: 6 de 6 de 2015.] https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

Navarro, Juan Francisco. 2012. *Analítica Visual aplicada a la Ingeniería de Ontologías.* Salamanca : s.n., 2012.

NetBeans Community. 2015. NetBeans IDE-The Smarter and Faster Way to Code. [En línea] Oracle Corporation, 2015. [Citado el: 8 de 6 de 2015.] https://netbeans.org/features/.

Olaya, Víctor. 2010. *Sistemas de Información Geográfica.* 2010. Vol. 1.0.

Oracle . 2015. Oracle | Hardware and Software, Engineered to Work Together. [En línea] 2015. [Citado el: 3 de 6 de 2015.] http://docs.oracle.com/javase/7/docs.

Pantoja Zaldívar, Yoenis y Hernández Montero, Lidisy . 2010. *Documento Visión v1.0.* La Habana : UCI, 2010.

Petasis, Georgios, y otros. 2011. *Ontology Population and Enrichment: State of the Art.* Paraskevi : s.n., 2011.

Pressman, R.S. . 2005. *Ingeniería del Software-Un enfoque práctico.* 2005.

Ramon, Jorge. 2009. *Ext JS 3.0 Cookbook.* s.l. : Packt Publishing, 2009.

Ramsey, Paul , y otros. 2015. PostGIS-Spatial and Geographic objects for PostgreSQL. [En línea] 2015. [Citado el: 6 de 6 de 2015.] http://postgis.net/features.

Reynoso, Carlos y Kiccillof, Nicolás . 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2004.

Studer , R, Benjamins , VR y Fensel , D. 1998. *Knowledge Engineering: Principles and Methods.* 1998.

Swain, A.D y Guttmmann, H.E. 1983. *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications.* 1983.

Swartout , B, y otros. 1997. *Toward Distributed Use of Large Scale Ontologies.* 1997.

The Apache Software Foundation. 2015. Apache-HTTP SERVER PROJECT. [En línea] 2015. [Citado el: 4 de 6 de 2015.] http://httpd.apache.org/docs/.

The PostgreSQL Global Development Group. 2014. *PostgreSQL 8.4 Documentation.* Oakland : s.n., 2014.

UCID: SOFTWARE-DEFENSA. 2012. *PRODESOFT-Proceso de Desarrollo y Gestión de Proyectos de Software.* 2012. Vol. 1.5.

Uschold , M y Jasper, R. 1999. *A Framework for Understanding and Classifying Ontology Applications.* 1999.

Vera Voronisky, Francisco y Garea Llano, Eduardo. 2009. *Alineamiento de ontologías en el dominio geospacial.* 2009.

Visual Paradigm International. 2015. Visual Paradigm-Design & Managment Tool for Business IT System Development. [En línea] 2015. [Citado el: 8 de 6 de 2015.] http://www.visual-paradigm.com/features/.

Welty, C. y FIKES, R. 2006. *A reusable Ontology for fluents in owl.* s.l. : IOS Press, 2006.

Witte, René, Khamis, Ninus y Rillin, Juergen . 2010. *Flexible Ontology Population from Text: The OwlExporter.* s.l. : Concordia University, 2010.

XML Protocol Working Group. 2007. World Wide Web Consortium (W3C). [En línea] 2007. [Citado el: 2 de 6 de 2015.] http://www.w3.org/TR/soap/.