Universidad de las Ciencias Informáticas

Facultad 6

Aplicación Informática para establecer semejanzas entre ontologías representativas del conocimiento en la Web.

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

Autores: Yahíma Matos Galvez
Elio Luis Toledo García

Tutor: Ing. Edgar Rojas Ricardo

Junio 2011

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Yahíma Matos Galvez

_____
Firma del Autor

Elio Luis Toledo García

_____
Firma del Autor

Edgar Rojas Ricardo

_____
Firma del Tutor

DATOS DE CONTACTO

Autores:

Yahíma Matos Galvez.

Universidad de las Ciencias Informáticas

La Habana, Cuba

Correo Electrónico: ymgalvez@estudiantes.uci.cu

Elio Luis Toledo García.

Universidad de las Ciencias Informáticas

La Habana, Cuba

Correo Electrónico: eltoledo@estudiantes.uci.cu

Tutor:

Ing. Edgar Rojas Ricardo.

Correo Electrónico: erojas@uci.cu

Ingeniero Informático, Universidad de Ciencias Informáticas, 2006.

Categoría Docente: Instructor.

AGRADECIMIENTOS

A mi mamá y a mi papá por su amor y comprensión, por guiarme y apoyarme en cada paso… a ellos mi vida entera.

A mi hermanita del alma por crecer junto a mí.

A toda mi familia, en especial a Nuri, Nuvi y mi tía Luisa por brindarme todo su cariño.

A mi dúo de tesis por trabajar junto a mí estos meses y lograr el fruto de nuestro trabajo.

A nuestro tutor por su constante preocupación y dedicación, por exigirnos y ayudarnos a cambio de nada.

A Eny por ser esa personita especial durante estos cinco años, por ayudarme a seguir mis sueños y hacer de mí una mejor persona.

A Lianet por ser quien es, por compartir risas y lágrimas en estos cinco años y ayudarme cuando más la necesité.

A mis amigos de Holguín, los de verdad, en especial a Yani que hubiese deseado compartir hoy aquí conmigo.

A Martica, Tecky, Milo, a las niñas del apto por compartir algún momento de nuestras vidas.

A todas las personas que de alguna forma u otra contribuyeron a realizar este sueño.

A todos ustedes….Gracias.

Yahi

A nuestro Comandante y a la Revolución por la creación de esta Universidad.

A todas las personas que han contribuido con mi formación profesional y para la vida, profesores, amigos, familia, les agradezco sinceramente con el mayor deseo de estar a la altura de las enseñanzas y principios que compartieron conmigo.

A Edgar nuestro tutor que me ayudó mucho profesionalmente con sus críticas y consejos constantes, muchas gracias por su paciencia, comprensión y apoyo.

A Yordania y Yahima por permitirme trabajar junto con ellas y haber hecho este trabajo realidad.

A Yaimara, Yeimy, María, Luis Rúben, Enrique y Renier de no ser por ellos no me encontrara aquí.

Elio

DEDICATORIA

A mi mamá y mi papá por su amor y dedicación.

A mi hermanita por traer a Sergitín a nuestras vidas.

A mi abuelita Lula por su cariño incondicional.

A mi abuelito Walter, que no pudo verme graduada.

Yahi

A mis padres por haber sido tan rectos, por todo el fundamento que sembraron en mi conciencia y por su infinita paciencia.

A mis abuelos por ser los más lindos del mundo entero. Le doy gracias por sacarme de esos hoyos tan profundos y sin salidas, por brindarme luz cuando más oscura está mi vida.

A Gina la tía más comprensible y hermosa por la sencillez de su alma y su carita de diosa, por su valerosa fuerza por las hermanas que me dio.

A Yanisel, Yoandra y Grober por permitirme ser su hermano, nunca me olviden, que donde quiera que yo esté siempre hay un corazón para ustedes que no es muy grande pero es suyo que dudas no quede.

A todos mis amigos por ser reales, leales y especiales conmigo, gracias por cada abrazo, gracias por cada beso, mil veces gracias por sus consejos y esfuerzos.

Elio

RESUMEN

Actualmente se construyen ontologías en diferentes partes del mundo, sin tener en cuenta que se puede reutilizar el conocimiento almacenado en otras existentes debido a que se tratan conceptos que son iguales en diferentes terminologías, pues no existe un vocabulario común entre los especialistas de un mismo dominio en diferentes escenarios. En la actualidad se busca una forma de definir las semejanzas entre los conceptos, predicados y acciones que conforman una ontología. El presente trabajo tiene como objetivo fundamental desarrollar una aplicación informática para establecer semejanzas entre ontologías representativas del conocimiento en la Web y que a su vez cumpla con los requisitos funcionales y no funcionales del cliente. Con el desarrollo de esta aplicación informática, la cual permite establecer un índice de semejanza entre ontologías representativas del conocimiento, le brinda al usuario saber cuán semejantes pueden ser dos ontologías, así como comparar sus conceptos. Para el desarrollo de la aplicación informática se empleó la metodología XP, el lenguaje de programación Java, como frameworks de desarrollo Jena y PostgreSQL como gestor de base de datos.

PALABRAS CLAVE

Comparar, conocimiento, dominio, ontologías, semejanza.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Desde que existe la humanidad, uno de sus mayores retos lo ha constituido la necesidad de almacenar información. Con el avance de las tecnologías se busca la forma más viable de almacenar los datos para preservar, representar y adaptar el conocimiento de tal manera que los ordenadores puedan interpretarlos.

Internet es una herramienta con la capacidad de interconectar ordenadores situados alrededor del mundo, basa su funcionamiento en la disponibilidad de brindar diferentes servicios que pueden ser utilizados por los usuarios pertenecientes a su dominio. Uno de estos servicios es la *World Wide Web (WWW),* presentada por Tim Berners-Lee y Robert Cailliau en 1989. Esta creación tiene su esencia en los enlaces de hipertextos y los documentos escritos en lenguaje HTML (HyperText Markup Language), permitiendo el acceso a diversas fuentes de datos en Internet. A pesar de los importantes avances aportados por las nuevas tecnologías de la Web, los usuarios de la web aún carecen de sistemas que permitan procesar y acceder a la información de un modo rápido y conciso. Aunque los buscadores actuales brindan un grupo de facilidades al usuario para realizar sus consultas en la red de redes, tienen también sus limitaciones en cuanto a la escasa precisión en los resultados y una alta sensibilidad al vocabulario empleado en la búsqueda, donde algunos generan páginas que carecen de interés.

Entre las iniciativas tecnológicas destinadas a crear una futura WWW que permita la interoperabilidad[1] entre ordenadores para procesar la información, representarla, encontrarla y gestionarla de una manera más eficiente se encuentra la denominada Web Semántica. Esta idea es impulsada a finales de la década del 90 por Tim Berners-Lee, quien la define como “una extensión de la Web actual donde la información viene dotada de significado bien definido que permitirá a las computadoras y a las personas trabajar en cooperación”. La Web Semántica propone superar las limitaciones de la web actual mediante la introducción de descripciones explícitas del significado, a través de las ontologías, cuyo objetivo es lograr un entendimiento entre las partes que intervienen en la construcción y explotación de la web: usuarios, desarrolladores y programas de diversos perfiles.

Las ontologías surgen de la necesidad de proveer una comprensión compartida y consensuada de un dominio determinado, facilitan la reutilización del conocimiento y el aprendizaje de personas ajenas al

[1] **Interoperabilidad:** Capacidad de un programa para acceder a múltiples sistemas diferentes.

mismo. Brindan además una comunicación entre personas y sistemas heterogéneos[2]. Desde la aparición de la inteligencia artificial, cobran una especial importancia en este campo, le añaden conocimientos especializados y métodos de razonamientos específicos, propios de las tareas que el sistema pretende resolver. Representan de forma conceptual el conocimiento, donde los conceptos están relacionados unos con otros a través de la semántica. Teniendo como objetivo, que los sistemas tengan la capacidad de razonar a partir de ellas. (1)

Con la creación de ontologías se pretende tener una base del conocimiento con el dominio del mundo real estructurado en un modelo que lo represente. De esta forma las computadoras serán capaces de comprender el dominio representado.

Actualmente se construyen ontologías en diferentes partes del mundo, sin tener en cuenta que se puede reutilizar el conocimiento almacenado en otras existentes debido a que se tratan conceptos que son iguales en diferentes terminologías, pues no existe un vocabulario común entre los especialistas de un mismo dominio en diferentes escenarios. En la actualidad se busca una forma de definir las semejanzas entre los conceptos, predicados y acciones que conforman una ontología.

Por lo anteriormente planteado se define como ***problema de la investigación*****:** ¿Cómo establecer semejanzas entre ontologías representativas del conocimiento en la Web?

La investigación tiene como ***objeto de estudio:*** Gestión de la información de las ontologías representativas del conocimiento en la Web, enmarcadas en el ***campo de acción:*** Aplicaciones informáticas para establecer semejanzas entre ontologías representativas del conocimiento en la Web.

Para dar solución a la situación planteada anteriormente, la investigación tiene como ***objetivo general:*** Desarrollar una aplicación informática que establezca semejanzas entre ontologías representativas del conocimiento en la Web para facilitar la toma de decisiones en el proceso de reutilización de los conceptos en diferentes escenarios, el cual se desglosa en los siguientes ***objetivos específicos:***

- Realizar el análisis de la aplicación informática.
- Realizar el diseño de la aplicación informática.

[2] **Sistemas heterogéneos**: Un sistema heterogéneo es aquel que se encuentra compuesto por hardware con características físicas distintas entre sí, y software con características operativas distintas entre sí, pero que se pueden comunicar utilizando medios comunes.

- Realizar la implementación y pruebas a la aplicación informática.

A los cuales se les dará cumplimiento a través de las siguientes ***tareas de la investigación:***

- ✓ Análisis de los conceptos fundamentales de las ontologías.
- ✓ Análisis y definición de las métricas de comparación.
- ✓ Selección de las herramientas para implementar la aplicación informática.
- ✓ Definición de los requerimientos no funcionales de la aplicación informática.
- ✓ Elaboración de las historias de usuarios de la aplicación informática.
- ✓ Elaboración de las tarjetas CRC como parte del diseño de la aplicación informática.
- ✓ Implementación de la aplicación informática.
- ✓ Evaluación de la aplicación informática mediante pruebas de aceptación.

Estructura de la tesis

Capítulo 1: Ontologías y Métricas de Comparación: En este capítulo se abordarán los principales conceptos que serán de utilidad para tener un dominio sobre el campo de las ontologías. Además se definirá la métrica a utilizar para la comparación de las ontologías, se realizará una caracterización sobre los referentes teóricos de la investigación y la selección de las herramientas, lenguajes y metodologías para el desarrollo de la investigación.

Capítulo 2: Planificación y Diseño de la Aplicación Informática: En este capítulo se describe la propuesta de solución para la situación problémica anteriormente planteada. Se definirán las principales características que poseerá la aplicación informática a desarrollar, comenzando con el estudio de la planificación, la definición de los requisitos no funcionales y el diseño logrando enfocar la implementación de la aplicación informática cumpla con los objetivos especificados. Se generan todos los artefactos de estas fases durante el ciclo de vida del proceso de desarrollo de software seleccionado.

Capítulo3: Implementación y Pruebas de la Aplicación Informática: En este capítulo se realizará la representación del código fuente de los principales componentes así como los mecanismos de implementación utilizados. Se obtendrá resultados del diseño implementando la aplicación informática en términos de componentes y garantizar la calidad de software mediante la realización de pruebas de aceptación.

CAPÍTULO 1: ONTOLOGÍA Y MÉTRICAS DE COMPARACIÓN

1. Introducción

En el presente capítulo se abordarán los principales conceptos que serán de utilidad para tener dominio sobre el campo de las ontologías. Además se definirá la métrica con la cual se establecerá el índice de semejanzas entre las ontologías, se realizará una caracterización sobre los referentes teóricos de la investigación y se seleccionará las herramientas, lenguajes y metodología para el desarrollo de la investigación.

1.1. Conceptos asociados al dominio del tema

Para entender el significado del término es necesario realizar un estudio de los conceptos asociados al tema.

Dominio

En el campo de la ontología un dominio denota un área específica de interés (La Habana, por ejemplo) o un área de conocimiento (física, aeronáutica, medicina, contabilidad, fabricación de productos, entre otros.). (2)

Semántica explícita

Semántica: Proviene de un vocablo griego que puede traducirse como **"significativo"**. Se trata de aquello perteneciente o relativo a la significación de las palabras expresiones o símbolos. Todos los medios de expresión suponen una correspondencia entre las expresiones y determinadas situaciones o cosas, ya sean del mundo material o abstracto. En otras palabras, la realidad y los pensamientos pueden ser descritos a través de las expresiones analizadas por la semántica. (3)

Explícito: Es aquello que expresa una cosa con claridad y determinación. Cuando algo es explícito, puede ser apreciado o advertido de manera evidente. (4)

Teniendo en cuenta las definiciones anteriormente expuestas, se puede concluir que una semántica explícita no es más que un lenguaje, mediante el cual se puede describir la realidad y los pensamientos, a través de una expresión con claridad y determinación.

1.2. Ontología

El término ontología proviene del campo de la filosofía, por lo que una ontología se considera una explicación sistemática de la existencia y está relacionada con el estudio del ente, la comprensión y la existencia del ser. Existen diversas definiciones de ontología, a continuación se citan algunas de ellas.

Según el diccionario de la Real Academia de la Lengua Española (*DRAE*) el término Ontología es la "*Parte de la metafísica que trata del ser en general y de sus propiedades transcendentales*"

Una ontología es una jerarquía de conceptos con atributos y relaciones, que define una terminología consensuada para definir redes semánticas de unidades de información interrelacionadas. (5)

Es una teoría lógica que corresponde al significado intencional de un vocabulario formal, o sea, un comportamiento ontológico con una conceptualización específica del mundo. Los modelos intencionados de un lenguaje lógico que usa este vocabulario son controlados por su comportamiento ontológico. Este comportamiento en la conceptualización subentendida es reflejado en la ontología por la aproximación desde modelos intencionados. (6)

Es una especificación explícita de una conceptualización, es decir, que proporciona una estructura y contenidos de forma explícita que codifica las reglas implícitas de una parte de la realidad; estas declaraciones explícitas son independientes del fin y del dominio de la aplicación en el que se usarán o reutilizarán sus definiciones. (7)

Es una jerarquía de un conjunto estructurado de términos para describir un dominio que puede ser usado como principio, como un esqueleto firme para una base de conocimiento. (8)

Es un instrumento de organización y representación del conocimiento que permite hacer explícitas las reglas implícitas de una parte de la realidad. Idealmente su presentación formalizada permite que estas declaraciones explícitas sean independientes del sistema que las usa y que a su vez pueda ser reutilizada por otros sistemas. (9)

Cada autor tiene su forma de conceptualizar una ontología, pero de forma general todos tienen puntos comunes que permiten hacer un resumen de su definición: Una ontología no es más que un lenguaje para la representación de conceptos, donde se encuentran establecidas relaciones e inferencias. Estudian los tipos de objetos que conforman o describen una realidad, así como sus propiedades y relaciones.

1.2.1. Componentes de una ontología

La identificación de los elementos que componen una ontología se torna en ocasiones difícil, debido a que las diferentes disciplinas que se encargan del estudio de las ontologías no identifican con exactitud los mismos elementos, y existe además, cierta ambigüedad en los términos que los designan.

Partiendo de esta premisa, se pueden identificar, cinco componentes esenciales: conceptos, instancias, relaciones, funciones y axiomas.

A continuación una breve descripción de los componentes de una Ontología (10):

Conceptos: Son las ideas básicas que se intentan formalizar. Pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento. Las clases en una ontología suelen organizarse en taxonomías a las que se le pueden aplicar los mecanismos de herencia.

Instancias: Contienen elementos o datos que describen o ejemplifican un concepto. Por lo general son propiedades que no se recogen en el concepto o nociones que sirven para aclarar la connotación de este en un contexto determinado. Se utilizan para representar objetos determinados de un concepto.

Relaciones: Son las relaciones que establecen el tipo de interacción semántica entre los conceptos, de manera que no solo se hace explícito lo que denotan, sino también lo que connotan. Entre las principales relaciones que pueden ser establecidas se encuentran las relaciones lógicas. Representan la interacción y enlace entre los conceptos de un dominio, suelen formar la taxonomía del dominio. Por ejemplo subclase-de, parte-de.

Funciones: Constituyen un tipo especial de relación, a través de las cuales se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase y asignar-fecha. Este componente es imprescindible cuando las ontologías son usadas para modelar sistemas y procesos.

Axiomas: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Son los elementos que permiten hacer inferencias que no están explícitas en la taxonomía de conceptos. Una inferencia es un proceso mental por el cual se extraen conclusiones a partir de premisas más o menos explícitas, o bien por aplicación de reglas. Por ejemplo “Si A y B son de la clase C, entonces A no es subclase de B”.

Los axiomas junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente.

1.2.2. Clasificación de Ontologías

Una vez conocido los diferentes componentes así como las características que integran una ontología, a continuación se exponen las diversas clasificaciones atendiendo a:

Dependencia y relación con una tarea específica. (11)

- **Ontologías de Alto Nivel o Genéricas:** Describen conceptos más generales. En relación con los Sistemas de Información estas ontologías describirían conceptos básicos.
- **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. Ellas representan las necesidades de los usuarios relacionados con una aplicación específica.

Ontologías refinadas y no-refinadas, o lo que es lo mismo, off-line y on-line. (11)

- **Refinada:** Precisa de un lenguaje de alta expresividad y tiene un gran número de axiomas. Deberían ser usadas off-line y solamente para referencia.
- **No refinada**: Tiene un número mínimo de axiomas y su objetivo es ser compartida por usuarios que concurren sobre una determinada visión del mundo. Tienen una mayor capacidad de ser compartidas y deberían ser utilizadas on-line para dar soporte en funcionalidad de sistemas de información.

Cantidad y tipo de estructura de la conceptualización. (12)

- **Ontologías terminológicas:** Especifican los términos que son usados para representar el conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado.
- **Ontologías de información:** Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.

- **Ontologías de modelado de conocimiento:** Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen.

Como se puede apreciar se clasifican de acuerdo a criterios de diferentes autores pero entre todas se destacan las ontologías de dominio por su mayor utilidad, encargadas de modelar el conjunto de conceptos de un dominio y las relaciones que existen entre ellos. Otro tipo importante lo constituyen las ontologías de aplicación, usadas con éxito en el análisis de sistemas y para modelar procesos empresariales.

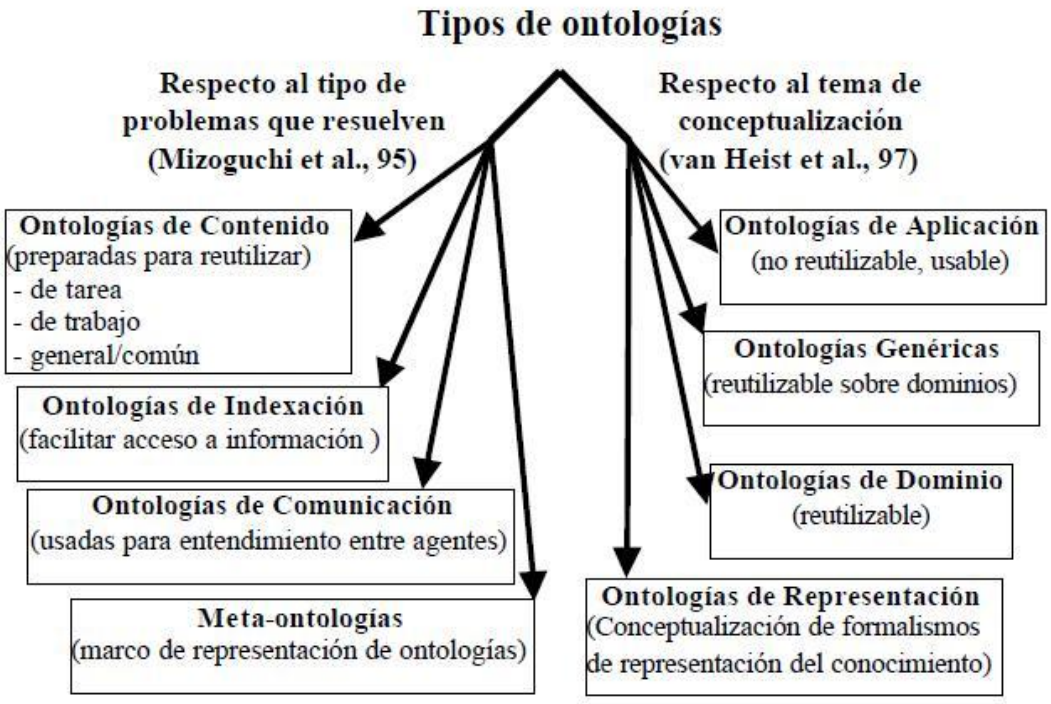

Figura 1. Clasificaciones de ontologías respecto al tipo de problemas que resuelven y respecto al tema de conceptualización.

1.2.3. Ontologías en los sistemas de información

El desarrollo de los Sistemas de Información (SI), normalmente se hacen en diferentes contextos, con distintos puntos de vista y suposiciones acerca del dominio de estudio. Provocando problemas de comunicación por falta de entendimiento compartido.

Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. (13)

Los (SI) son esencialmente artefactos de conocimiento que capturan y representan el conocimiento sobre ciertos dominios. (14)

Los paradigmas que han sustentado el desarrollo de los SI se han basado en diferentes núcleos de interés, sin embargo hoy no son suficientes para abordar los fenómenos y las situaciones problémicas que surgen frente a los avances de las tecnologías de la información y la comunicación.

Por otra parte, los (SI) necesitan usar representaciones, tan generales como sea posible, para responder a los atributos de calidad del software y aumentar al máximo la posibilidad de reusabilidad. Pero al mismo tiempo, las representaciones deben corresponderse, lo más estrechamente posible a los procesos que ellos representan. Es así como, las cuestiones relacionadas con la gestión de datos son cuestiones efectivamente ontológicas. (15)

Las ontologías generalmente se usan para especificar y comunicar el conocimiento del dominio de una manera genérica y son muy útiles para estructurar y definir el significado de los términos. La nueva generación de los SI deberá ser capaz de resolver la interoperabilidad semántica, en la cual un hecho puede ser más que una descripción, para poder hacer un buen uso de las informaciones disponibles. (16)

La forma de resolver dicho problema consiste en crear un entendimiento compartido, como son las ontologías, que unifican los diferentes puntos de vista y se utilizan para (17):

- Entender cómo diferentes sistemas comparten informaciones.
- Descubrir ciertas distorsiones presentes en los procesos cognitivos de aprendizaje en un mismo contexto.
- Formar patrones para el desarrollo de SI.

Es así como, el uso de ontologías en el desarrollo de los SI permite establecer correspondencia y relaciones entre los diferentes dominios de entidades de información. El uso de ontologías en el desarrollo de los SI contribuye a mejorar la calidad del producto final. De esta forma las ontologías pueden proveer los mecanismos para organizar y almacenar *ítems* que incluyen esquemas de las bases de datos, objetos de interfaz de usuario, y programas de la aplicación. Es decir, las ontologías están llegando a ser una herramienta fructífera en la investigación y desarrollo de la disciplina de los SI. (18)

Cada Sistema de Información tiene su propia ontología implícita, sin embargo de manera explícita las ontologías pueden tener roles diferentes dentro de los Sistemas de Información.

Teniendo en cuenta los beneficios que ofrecen las ontologías, se aborda su rol en los SI desde dos perspectivas (19): Como un soporte para el análisis conceptual de métodos y técnicas de los SI.

Como un soporte para el diseño, desarrollo y uso de los SI. En esta perspectiva se analizan dos dimensiones, una la visión de los desarrolladores, concerniente a la manera en que una ontología ayuda o se usa para desarrollar un Sistema de Información y la otra desde la visión del usuario, relativa a la manera en que una ontología facilita la tarea del mismo al interactuar con los Sistemas de Información. (14)

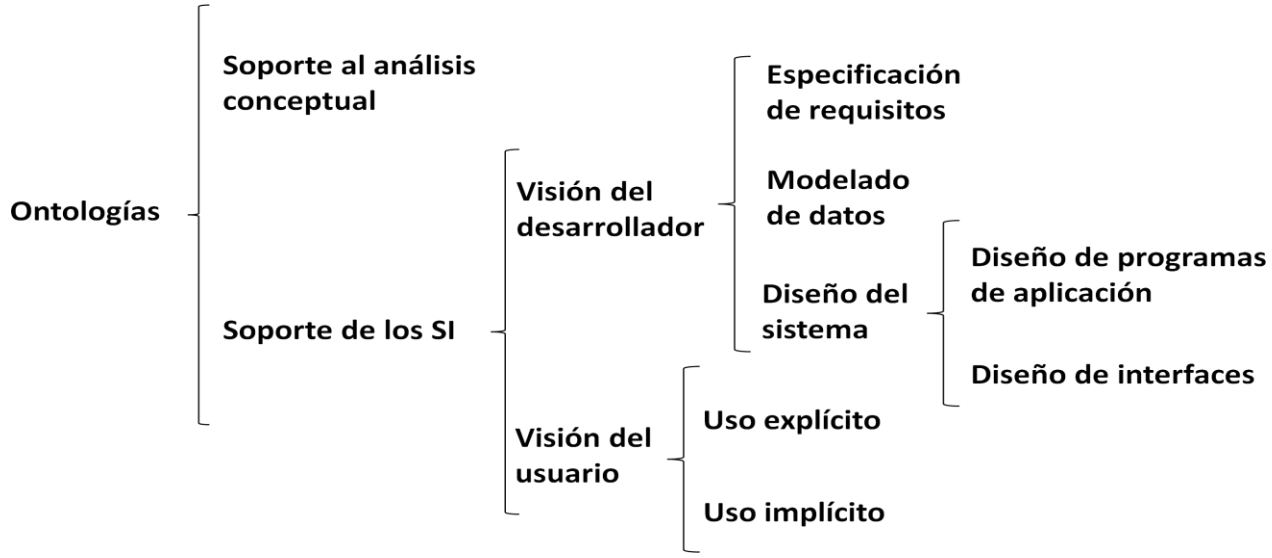

Figura 2 Rol de las ontologías dentro de los SI

1.3. Web Semántica

La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante. (20)

1.3.1. Capas de la Web semántica

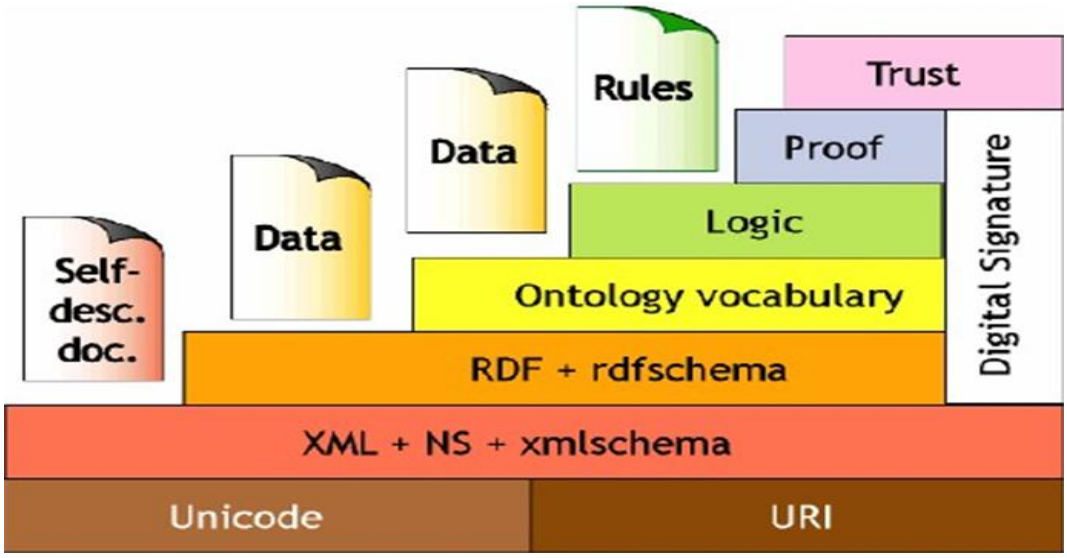

Fig. 3 Capas de la Web Semántica

- **Unicode:** Es un estándar cuyo objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático. El mismo permite mostrar información en cualquier idioma y con la certeza de que no aparezcan símbolos extraños.
- **URI:** Son cadenas que permiten acceder a cualquier recurso de la Web. En la Web Semántica las *URIs* son las encargadas de identificar objetos. Todos los objetos pueden ser identificados mediante una URI. Si dos objetos cuentan con la misma URI pueden existir colisiones. El grupo de trabajo del W3C está intentando resolver este problema.
- **XML+NS+xmlschema:** Esta es la capa más técnica de la Web Semántica. En ella se encuentran agrupadas las diferentes tecnologías que posibilitan la comunicación entre agentes.
- **RDF+rdfschema:** Está basada en la capa anterior, define el lenguaje universal con el que se pueden expresar diferentes ideas en la Web Semántica. RDF es un lenguaje que define un modelo de datos para describir recursos mediante tripletas sujeto-predicado-objeto.

Las dos primeras capas serán URIs y la tercera puede ser URI o un valor literal. RDF Schema es un vocabulario RDF que permite describir recursos mediante una orientación a objetos. Esta capa no sólo ofrece una descripción de los datos, sino también cierta información semántica.

- ***Ontology (Ontologías):*** Permite clasificar la información. Esta capa permite extender la funcionalidad de la Web Semántica agregando nuevas clases y propiedades para describir los recursos.
- **Logic (Lógica)** Además de ontologías se precisan reglas de inferencia.
- **Proof (Pruebas):** Se intercambiarán "pruebas" escritas en el lenguaje unificador de la Web Semántica. Este lenguaje posibilita las inferencias lógicas realizadas a través del uso de reglas de inferencia.
- **Trust (Confianza):** Hasta que no se haya comprobado de forma exhaustiva las fuentes de información, los agentes deberían ser muy escépticos acerca de lo que leen en la Web Semántica.
- **Digital Signature (Firma digital):** Utilizada por los ordenadores y agentes para verificar que la información ha sido ofrecida por una fuente de confianza. (21)

1.3.2. Las Ontologías como base de la Web Semántica. Componentes.

El objetivo de las Web semánticas es que los datos puedan ser utilizados por los ordenadores sin la necesidad de supervisión humana, se trata de convertir la información en conocimiento. Los usuarios de la web no solo encontrarán la información de manera rápida y precisa, si no que podrán realizar inferencias automáticamente, buscando información relacionada con la que se encuentra situada en las páginas y con las peticiones realizadas por los usuarios; para ello se necesita que el conocimiento de la web esté representado de forma legible, consensuado y reutilizable por los ordenadores. Las ontologías proporcionan la vía para representar este conocimiento. (20)

En epígrafes anteriores se vieron las ventajas que brindan las ontologías para representar el conocimiento, pueden ser utilizadas como bases o dominios de conocimientos gracias a los diferentes componentes con los que cuenta como: relaciones, axiomas y herencia, los cuales permiten realizar búsquedas inteligentes ya que puede realizar inferencias entre conceptos.

Existen muchas ventajas en la utilización de ontologías dentro de la Web Semántica (22):

- Constituyen una herramienta para la adquisición de conocimiento.
- Permite compartir conocimiento.
- Crear una red de relaciones entre conceptos.

- La reutilización de conocimiento.

Se hace necesaria la utilización de mecanismos de anotación semántica, que permitan realizar una correspondencia entre la información de un sitio web y los conceptos de las ontologías. Para esto la Web Semántica usa algunos de sus componentes como OWL, a continuación listamos los componentes por lo que está compuesta la Web Semántica (23):

- **XML:** Provee una sintaxis elemental para las estructuras de contenidos dentro de documentos.
- **XML Schema:** Es un lenguaje para proporcionar y restringir la estructura y el contenido de los elementos contenidos dentro de documentos XML.
- **RDF:** Es un lenguaje simple para expresar modelos de los datos, que refieren a los objetos "recursos" y a sus relaciones. Un modelo de RDF se puede representar en sintaxis de XML.
- **RDF Schema**: Es un vocabulario para describir propiedades y clases de recursos RDF, con semántica para generalizar jerarquías de las propiedades y clases.
- **OWL:** Es un mecanismo para desarrollar temas o vocabularios específicos en los que se pueden asociar esos recursos.

El desarrollo de la Web semántica requiere la utilización de estos componentes de manera tal que puedan dotar a cada página, archivo y a cada recursos o contenido de la red, de una lógica y un significado, y que permitan a los ordenadores conocer el significado de la información que manejan con el fin de que esta información pueda no sólo ser presentada en pantalla, sino también que pueda ser integrada y reutilizada.

1.4. Análisis y definición de la métrica para establecer el índice de semejanzas entre ontologías.

Según la definición de Fenton, "una medición es el proceso por el cual números o símbolos se asignan a atributos de entidades del mundo real, de forma que lo describen de acuerdo a reglas claramente definidas". (24)

Aunque los términos medida, medición y métricas se utilizan a menudo indistintamente, existe gran confusión a la hora de referirse a ellos.

”Las métricas son medidas que se le aplica a un producto, estas medidas proporcionan una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto”. (25)

La *IEEE* siglas en ingles de *The Institute of Electrical and Electronics Engineers (Instituto de ingenieros eléctricos y electrónicos)* en “*Standard Glossary of Software Engering Terms*” (Glosario Estándar de Términos de Ingeniería de Software.) define métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. (26)

Atendiendo a los conceptos antes mencionados se concluye que una métrica es una forma de evaluar y una escala, definidas para realizar mediciones de uno o varios atributos.

Para la investigación se propone una métrica que permita dado una ontología definir cuantas existen semejantes a ella en un repositorio determinado, basándonos en el esquema Noy y Hafner, proponen un marco para comparar ontologías con el propósito de estudiar la diversidad de los diseños de ontologías e indicar las fortalezas y debilidades de cada uno. Las características las clasifican como se muestra en la figura 4.

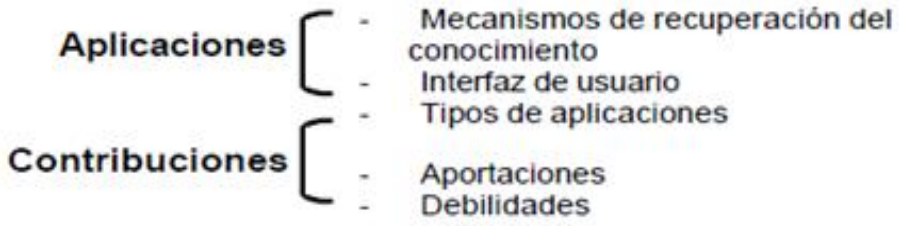

Figura 4: Taxonomía de características de Noy y Hafner

El marco de comparación, expuesto en la figura 4, fue aplicado a las siguientes ontologías: *CYC*, *Dahlgren's ontology*, *GUM*, *Gensim*, *KIF 27*, *Plinius*, *Sowa's Ontology*, *TOVE*, *UMLS* y *Word net*. (24)

Las características seleccionadas para conformar la métrica definida para el desarrollo de la aplicación informática son:

- Su tamaño, identificado por el número de conceptos, propiedades, instancias y propiedades de las instancias.
- Los términos representados: qué conceptos incorpora la ontología.
- Si los conceptos tienen estructura interna (si tiene un conjunto de propiedades que le dan significado a cada categoría).
- Los tipos de atributos que están presentes en un concepto.

Según la métrica confeccionada dos ontologías se pueden definir como semejante atendiendo a un porciento determinado, de acuerdo al número de características que contengan similitud entre ellas. El porciento de semejanza será igual al número que represente la similitud de las características entre el total de características definidas en la métrica por cien, quedando la fórmula representada de esta forma:

$$Ps=Ns/Tc*100$$

Donde:

Ps = es por ciento de semejanza.

Ns = número que represente la similitud de las características.

Tc = total de características definidas.

1.5. Metodología de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayuda a la documentación para el desarrollo de productos de software. Una metodología puede seguir una o varios modelos del ciclo de vida, el ciclo de vida indica que es lo que hay que obtener a lo largo del desarrollo del proyecto pero no como hacerlo. (27) La metodología indica cómo hay que obtener los distintos productos parciales y finales. Entre los procesos de desarrollo más conocidos se obtienen Programación Extrema (*Extreme Programming*, XP), clasificados como un método ligero y Proceso Unificado de Desarrollo (*Rational Unified Process*, RUP), que se clasifica como método pesado.

1.5.1. Proceso Unificado de Desarrollo (Rational Unified Process, RUP)

Un proceso de desarrollo de software define quién hace qué, cómo y cuándo. RUP define cuatro elementos trabajadores (roles), que responden a la pregunta ¿Quién?, las actividades que responden a la pregunta ¿Cómo?, los artefactos (productos), que responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo?

El ciclo de vida de RUP se caracteriza por estar dirigido por casos de usos ya son los que guían el proceso de desarrollo, estar centrado en la arquitectura, la arquitectura muestra la visión común del sistema completo, por lo que describe los elementos del modelo más importantes para su construcción y ser iterativo incremental, propone que cada fase se desarrollo en iteraciones, las iteraciones hacen referencia a pasos en el flujos de trabajo, y los incrementos al crecimiento del producto.

Esta metodología divide en cuatro fases el desarrollo de software:

- Inicio: la cual tiene como objetivo determinar la visión del proyecto.
- Elaboración: la cual tiene como objetivo determinar la arquitectura óptima.
- Construcción: el objetivo es obtener la capacidad operacional inicial.
- Transición: el objetivo es obtener el *release* del proyecto. (27)

1.5.2. Programación Extrema (Extreme Programming, XP)

La Programación Extrema es una de las metodologías de desarrollo de software más exitosa en la actualidad, utilizadas para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito de proyecto. (27)

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva de proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierten en miembro del equipo.

Lo fundamental en este tipo de metodología es:

- La comunicación entre los usuarios y los desarrolladores.
- La simplicidad al desarrollar y codificar los módulos del sistema.
- La retroalimentación concreta y frecuente del desarrollo, el cliente y los usuarios finales. (28)

La metodología XP se divide en 4 fases.

- Planificación: Plantea la planificación como un permanente diálogo entre las partes la empresarial (deseable) y la técnica (posible).
- Diseño: Plantea que el diseño adecuado para el software es el que funciona con todas las pruebas, no tiene lógica duplicada, manifiesta cada intención importante para los programadores, tiene el menor número de clases y métodos.
- Desarrollo: Plantea la recodificación, programación por parejas, la propiedad colectiva, integración continua del código.
- Pruebas: Plantea que no debe existir ninguna característica en el programa que no haya sido probada, los programadores escriben pruebas para chequear el correcto funcionamiento del programa, los clientes realizan pruebas funcionales. (28)

1.5.3. ¿Por qué XP?

Se seleccionó XP en busca de simplificar el desarrollo del software y reducir el costo del proyecto. Al mismo tiempo combina las que han demostrado ser las mejores prácticas de desarrollo de software, y las lleva al extremo. Conjuntamente se rediseñará todo el tiempo (refactoring), dejando el código siempre en el estado más simple posible. Se harán pruebas no sólo de cada nueva clase (pruebas unitarias), sino que

también los clientes comprobarán que el proyecto va satisfaciendo los requisitos (pruebas funcionales). Las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, esto permite beneficiarse de la retroalimentación tan a menudo como sea posible.

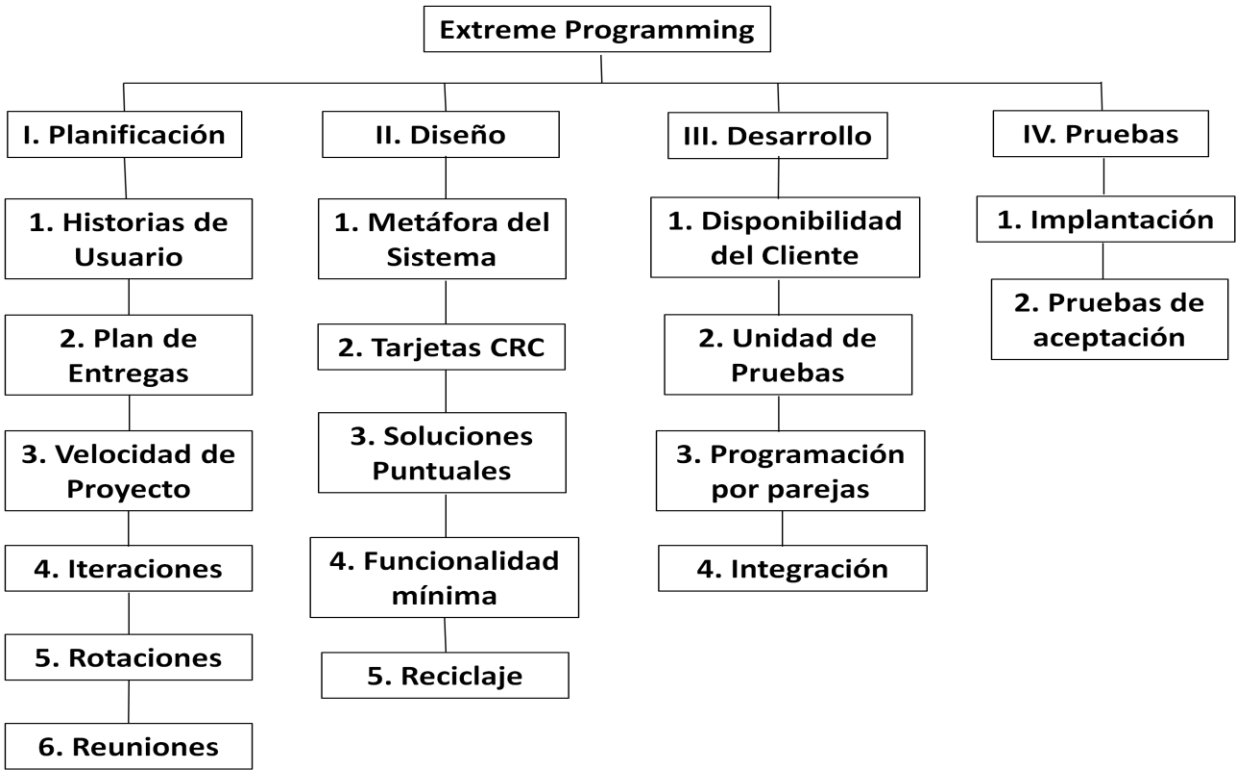

Figura 5 Fases de desarrollo de XP.

1.5.4. Roles y artefactos

Un rol es una definición abstracta de un conjunto de actividades realizadas y artefactos obtenidos, son realizados típicamente por un individuo, o un conjunto de individuos, trabajando juntos en un mismo equipo.

De acuerdo con las necesidades y características de esta investigación los roles que se desarrollarán serán: programador, cliente y encargado de pruebas.

Programador: En XP los programadores diseñan programan y realizan pruebas. Son los responsables de tomar decisiones técnicas y construir el sistema. Los artefactos principales serán elaborados por él.

Artefactos generados por el rol de programador:

Tarjetas CRC: El uso de tarjetas CRC (Clase, Responsabilidades y Colaboraciones) sirven para diseñar el sistema en conjunto entre todo el equipo. Estas tarjetas representan objetos, la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha las clases que colaboran con cada responsabilidad.

Cliente: Es parte del equipo determina que construir y puede establecer pruebas funcionales.

Artefactos generados por el rol de cliente:

Historias de usuario: Las historias de usuario tienen el mismo propósito que los casos de uso. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema. Existen diferencias entre estas y la tradicional especificación de requisitos. La principal diferencia es el nivel de detalle. Las historias de usuario solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario.

Encargado de Pruebas: Es el responsable de ayudar al cliente a escoger y escribir pruebas funcionales. Es responsable también de hacer funcionar las pruebas regularmente y comunicar sus resultados al resto del equipo. Debe ser una persona con capacidad de abstracción y mucho tacto y facilidad de comunicación, dado que está en un lugar intermedio entre los programadores y el cliente.

1.6. Lenguaje de programación y tecnología

Un lenguaje de programación es un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un lenguaje le da la capacidad al programador de especificarle al computador, qué tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano. Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, interpretación o intermedio, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador. (29)

Siguiendo los lineamientos del proyecto KAHOS de la Universidad de Málaga, se utilizará el lenguaje de programación Java y la tecnología JSP.

Java: Es un lenguaje de programación de propósito general y sencillo desarrollado por *Sun Microsystems*[3]*.* Se sustenta en los siguientes pilares: la programación orientada a objetos, la posibilidad de ejecutar un mismo programa en diversos sistemas operativos, la inclusión por defecto de soporte para trabajo en red, la opción de ejecutar del código en sistemas remotos de manera segura y la facilidad de uso. La aplicación de Java es muy amplia. El lenguaje se utiliza en una gran variedad de dispositivos móviles, como teléfonos y pequeños electrodomésticos. En los navegadores web, Java permite desarrollar pequeñas aplicaciones que se incrustan en el código HTML [4]de las páginas. (30)

JSP: La tecnología Java para la creación de páginas web con programación en el servidor. JSP es un acrónimo de Java Server Pages (Páginas de Servidor Java.), es una tecnología orientada a crear páginas web con programación en Java. Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java.

1.7. Marco de trabajo o frameworks

Un *framework* (término en inglés para marco de trabajo) es una estructura de soporte mediante la cual puede ser desarrollado un proyecto de software. Generalmente incluye programas, bibliotecas y un lenguaje interpretado que optimizan y aceleran el proceso de desarrollo del software. Estos diagramas dan una vista general de lo que se desea desarrollar para una mejor comprensión.

Para el acceso a las ontologías mundialmente son utilizados tres *Frameworks* fundamentalmente, *Sesame*, Jena y RAP, todos presentan características similares y al mismo tiempo otras que los diferencian.

1.7.1. Sesame.

Es un API para Java, es decir, un entorno para el desarrollo de aplicaciones en el lenguaje de programación Java para la web semántica. Es un marco de desarrollo para almacenamiento, consulta y razonamiento con RDF y RDF *Schema.* Puede ser usado como base de datos para RDF y RDF *Schema*,

[3] *Sun Microsystems:* es un proveedor multinacional de ordenadores, software y hardware, y servicios de tecnología de la información.

[4] HTML: *HyperText Markup Language*, es el lenguaje de marcado predominante para páginas web.

o como una librería de Java para aplicaciones que necesitan trabajar internamente con RDF. De manera más general, *Sesame* proporciona a los desarrolladores de aplicaciones un conjunto de herramientas muy útil para hacer cualquier cosa por uno mismo.

Repositorio de Inferencias: Un concepto fundamental en el *framework* de *Sesame* es el repositorio. Un repositorio es un contenedor de almacenamiento para RDF. Este puede ser un simple objeto Java en memoria o incluso una base de datos relacional. Para cualquiera de las formas de almacenamiento que se elija es importante darse cuenta de que casi todas las operaciones en *Sesame* están relacionadas con un repositorio: cuando se añaden datos RDF se incluyen en un repositorio, cuando se realiza una consulta, se pregunta a un repositorio en particular. (31)

Arquitectura: Comenzando desde abajo, la capa SAIL API (*Storage And Inference Layer*), es una API interna de *Sesame* que se abstrae del formato de almacenamiento usado: si los datos son almacenados en un RDBMS, en memoria o en ficheros, por ejemplo. Esta capa también proporciona soporte de razonamiento. Encima de la capa SAIL se encuentran los módulos funcionales de *Sesame*: *SeRQL, RQL* y *RDQL Query engines*, el módulo administrador y RDF Export. A estos módulos de funcionalidad se puede acceder a través de la API de Acceso de *Sesame,* que consiste en dos partes separadas: *Repository API* y el *Graph API*. Estas dos APIs se complementan la una a la otra en funcionalidad y en la práctica se suelen usar juntas. (31)

1.7.2. JENA.

Jena es un API para Java, es decir, un entorno para el desarrollo de aplicaciones en el lenguaje de programación Java para la web semántica. Es un *framework* desarrollado por HB Labs para manipular metadatos desde una aplicación Java. En la actualidad existen dos versiones:

- **Jena 1:**
  - ✓ Principalmente soporte para RDF
  - ✓ Capacidades de razonamiento limitadas
- **Jena 2:**
  - ✓ Incluye además una API para el manejo de ontologías
  - ✓ Soporta el lenguaje OWL

Características (32)**:**

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

Componentes (32)**:**

- ARP**:** un párser de RDF
- API RDF
- API de Ontologías para OWL, DAML y RDF *Schema*
- Subsistema de razonamiento
- Soporte para persistencia
- RDQL: Lenguaje de consultas de RDF

1.7.3. RAP.

Es un conjunto de herramientas para la Web Semántica y para los desarrolladores de PHP. Ofrece funciones para analizar, manipular, almacenar, consultar y serializar gráficos de RDF. RAP se inició como un proyecto de código abierto por la *Freie Universität* de Berlín en 2002.

El núcleo de RAP son dos implementaciones de declaración RDF de almacenes que contienen gráficos, ya sea en la memoria o en una base de datos relacional. En torno a estos almacenes RAP ofrece ricas interfaces de programación para la manipulación de gráficos en los diferentes RDFS de capas de abstracción. Ofrece un conjunto de herramientas para desarrolladores lo que permite escribir aplicaciones que trabajan de forma implícita (virtual) de las declaraciones. Otra característica a la espera de ser ejecutado en el RAP es el apoyo a la firma en la sintaxis de modelos RDF o el modelo de nivel, usando Firmas XML o publicaciones en Web Semántica (SWP). (33)

Fundamentación de la selección:

Se propone utilizar Jena la versión 2.0, aunque los tres *frameworks* son utilizados para el acceso a ontologías, se descartó por la utilización de este último pues el mismo tiene más funcionalidades que los otros dos, además de más bibliografía para consultar y apoyarse.

1.7.4. Dojo.

Para el desarrollo y la creación de interfaces y la programación en la capa de presentación se propone como marco de desarrollo o *frameworks* utilizar dojo debido a que ofrece una colección completa de controles de la interfaz de usuario, dándole la capacidad de crear aplicaciones web que están altamente optimizadas para la usabilidad, rendimiento y accesibilidad. Es completamente gratuito y de código abierto sin condiciones o licencias para comprar. Ahorra tiempo y proporciona un potente rendimiento en el proceso de desarrollo. Ofrece varias redes muy flexibles, incluyendo el control DataGrid, EnhancedGrid y TreeGrid para servir a todos las necesidades de visualización de datos. Contiene una amplia gama de plataformas de soporte de plataformas, incluyendo SVG, VML, Silverlight y SVGWeb. Contiene una gran cantidad de gráficas, temas y una variedad de características en tiempo real interactivos. (34)

1.8. Herramientas para el desarrollo.

Un Entorno de Desarrollo Integrado es un software que consta de una serie de herramientas acopladas que tienen como fin fundamental la programación en uno o más de un lenguaje de programación.

1.8.1. Sistemas Gestor de Base de Datos.

Un sistema gestor de base de datos es un software dedicado a servir de interfaz entre las bases de datos, los usuarios y las aplicaciones que las utilizan, permitiendo la creación, administración y gestión de la información contenida en las propias bases de datos.

PostgreSQL 8.3 es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Berkeley Software Distribution) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. (1)

A continuación alguna de las características más importantes y que posee *PostgreSQL*: (35)

- Es una base de datos 100%

- Integridad referencial
- Replicación asíncrona / Streaming replication - Hot Standby
- Copias de seguridad en caliente (Online/hot backups)
- Juegos de caracteres internacionales
- Multi-Version Concurrency Control (MVCC)
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL
- Actualización in-situ integrada (pg_upgrade)
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit

1.8.2. Eclipse.

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador que se entrega como parte de Eclipse. Sin embargo, se puede usar para otros tipos de aplicaciones cliente, como *BitTorrent Azureus.* (36)

La versión actual de Eclipse dispone de las siguientes características: editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUNIT, control de versiones con CVS, integración con ANT, asistentes *wizard* para la creación de proyectos, clases, test, refactorización.

Los *plugins* que se le pueden añadir para soportar lenguajes adicionales están dispersos y esta herramienta presenta un alto consumo de recursos (casi 200MB de RAM se consume el Eclipse PDT), también no reconoce JavaScript y CSS, así que puede llenar la ventana de “Errors & *Warnings*”, que se consideran supuestos “errores de sintaxis” de PHP en documentos JavaScript. Contiene un buen *debugger* en aplicaciones sencillas, en un entorno MVC (Modelo Vista Controlador) es prácticamente imprescindible.

1.8.3. NetBeans.

NetBeans 6.9 es una aplicación de código abierto ("*open source*") diseñada para el desarrollo de aplicaciones, fácilmente portables entre las distintas plataformas (uso de la tecnología Java y un entorno de desarrollo integrado). Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, entre otras. Es un programa con licencia CDDL que permite su uso libremente, brinda el código del mismo y es totalmente gratuita. Permite escribir, compilar, hacer un *debug*, ensamblar y desplegar aplicaciones, y aunque está escrito en Java, brinda soporte para toda clase de lenguajes de programación. Además funciona en sistemas operativos compatibles con la máquina virtual Java (Windows XP, Vista, Windows 7, Ubuntu 9.10, Solaris, Mac OS X 10.5 o superior). (37)

Para el desarrollo de la aplicación informática se seleccionó el IDE de desarrollo NetBeans por permitir escribir, compilar, hacer un *debug*, ensamblar y desplegar aplicaciones, además de brindar soporte para toda clase de lenguajes de programación. Además funciona en sistemas operativos compatibles con la máquina virtual Java (Windows XP, Vista, Windows 7, Ubuntu 9.10, Solaris, Mac OS X 10.5 o superior).

1.8.4. Servidor Web.

Apache Tomcat 6.0

Es un servidor Web flexible, rápido y eficiente, continuamente actualizado y adaptado a las últimas versiones de protocolos. Es capaz de funcionar en la mayoría de las plataformas y entornos existente. Está entre los servidores Web más reconocidos a nivel internacional, debido a la capacidad de correr en múltiples Sistemas Operativos. Apache es una tecnología gratuita de código abierto y altamente configurable, de diseño modular capaz de interpretar diversos lenguajes de programación.

El servidor Apache está estructurado en módulos, los cuales pueden clasificarse en tres categorías específicas (38):

Módulos Base: Son funciones básicas del Apache.

Módulos Multiproceso: Responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender las peticiones.

Módulos Adicionales: Cualquier otro módulo que le añada una funcionalidad al servidor se puede añadir sin necesidad de volver a instalar el software.

Fundamentación de la selección:

Se decide seleccionar Apache como servidor Web para la aplicación informática que se desarrollará por poseer grandes ventajas que hacen que millones de servidores reiteren su confianza, además de las excelentes prestaciones y las características descritas anteriormente.

1.9. Conclusiones del capítulo 1

En este capítulo se realizó un análisis sobre los conceptos fundamentales relacionados con las ontologías, definiendo los principales conceptos y características, así como la relación que tienen con la web semántica. Se definió la métrica a utilizar para establecer el índice de semejanzas entre ontologías. Se seleccionaron las herramientas que se utilizarán para la implementación, como marco de trabajo para la gestión de ontologías se seleccionó Jena y para el trabajo en la capa de presentación de la interfaz Dojo 1.3, NetBeans 6.9 como IDE de desarrollo, PostgreSQL 8.3 como gestor de base de datos, Apache Tomcat 6.0 como servidor Web, Java como lenguaje de programación y JSP como tecnología Web, y XP como la metodología para el desarrollo de software.

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DE LA APLICACIÓN INFORMÁTICA

2.1. Introducción.

En este capítulo se describe la propuesta de solución para la situación problémica que dio origen a la presente investigación. Se definen las principales características que poseerá la aplicación informática que se implementará, comenzando con el estudio de la planificación y el diseño logrando enfocar la implementación de la aplicación informática cumpla con los objetivos especificados.

2.2. Propuesta de la aplicación informática.

El desarrollo de esta aplicación informática está orientado a establecer un índice de semejanza entre ontologías que facilite la comprensión de los conceptos en diferentes escenarios.

La aplicación informática estará constituida por una serie de funcionalidades con el objetivo de brindar a los usuarios la información referente a:

- Gestión de usuario
- Autenticar usuario.
- Índice de semejanza entre ontologías según las características definidas en la métrica a utilizar
- Índice de semejanza entre ontologías y conceptos.
- Buscar ontologías por conceptos.
- Visualizar el resultado de la comparación

2.2.1. Requerimientos no funcionales

Después que se conocen las funcionalidades que debe realizar la aplicación informática se determina como ha de comportarse, que cualidades debe tener, o cuán rápido debe ser. Estos requisitos son los que normalmente deben apuntar a la arquitectura. En la metodología XP no se definen los requisitos no funcionales como un artefacto. Para un mejor funcionamiento es necesario que el producto cumpla con cualidades y propiedades, por lo que se decide identificar los requisitos no funcionales de la aplicación informática.

Los requerimientos no funcionales identificados son:

Requerimientos de software

- Sistema operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4.0 GNU/Linux o superior, Windows XP o superior.
- Navegador Web: Mozilla Firefox 3.4 o versiones superiores, Internet Explorer 5.0 o versiones superiores.
- El sistema gestor de base de datos PostgreSQL 8.3 o una versión superior.
- El servidor de aplicaciones debe ser Apache Tomcat 6.0 o una versión superior.

Requerimientos de hardware

- Para el cliente: Procesador a 2.0 GHz, 128 MB de RAM, 100 MB libres de disco duro.
- Para el servidor: Procesador a 3.0 GHz, 1 GB de RAM, 5 GB libres de disco duro.

Requerimientos de seguridad

Para garantizar la seguridad de la aplicación informática, el nivel de acceso será de acuerdo al rol de los usuarios que se autentiquen. El administrador tendrá acceso total a la aplicación informática y el usuario autenticado solo a las funcionalidades que su rol le permite realizar. La aplicación informática estará disponible las 24 horas del día. A los usuarios autorizados se les garantizará el acceso a la información y a los dispositivos o mecanismos utilizados para lograr la seguridad, no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

Restricciones del diseño y la implementación

Será una aplicación informática para la Web, para el desarrollo de la misma se emplea la metodología XP. Se utiliza como gestor de base de datos PostgreSQL 8.3, como lenguaje de programación a utilizar será Java con la tecnología JSP para programar en el servidor y para el desarrollo se usará el framework Jena.

2.2.2. Fase de planificación.

La planificación es la fase donde se especifican los requisitos no funcionales de la aplicación informática y donde se aplicarán diferentes iteraciones. Para hacerlo será necesaria la existencia de reglas que se han de seguir por las partes implicadas en el proyecto para que todas las partes tengan voz y se sientan

realmente partícipes de la decisión tomada. En esta fase se elaboran las historias de usuarios que es uno de sus artefactos más importantes así como el plan de iteraciones y el plan de entregas.

2.2.2.1. Historia de usuarios.

Las historias de usuario tienen el mismo propósito que los casos de uso. Son escritas por los propios clientes, en estas describen las funcionalidades de la aplicación informática, conducirán el proceso de creación de los test de aceptación empleados para verificar que han sido implementadas correctamente. Proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conlleva la implementación.

A continuación se muestra una de las historias de usuario definidas para implementar la aplicación informática. El resto aparece en el documento “Plantilla de historias de usuario”.

Tabla 1: Historia de Usuario: Índice de semejanza entre ontologías según las características definidas en la métrica.

Historia de Usuario	
Número: 2	**Nombre:** Índice de semejanza entre ontologías según las características definidas en la métrica.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 2	**Iteración Asignada:** 1
Programador responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: La aplicación informática debe brindar la posibilidad de introducir las ontologías que el usuario quiera conocer su semejanza. Esta funcionalidad describe como se llevará a cabo el proceso de establecer semejanzas entre ontologías de forma general según las características definidas en la métrica a utilizar.	
Información adicional (Observaciones): Se deben tener en cuenta las características seleccionadas para conformar la métrica propuesta, el usuario solamente puede seleccionar dos ontologías para establecer semejanzas.	

Para garantizar el desarrollo de la aplicación informática propuesta, se realizó una estimación de la duración para cada una de las historias de usuarios identificadas, llegando a los resultados que se muestran a continuación:

Tabla 2: Estimación de la duración para cada una de las historias de usuario.

Historias de usuarios	Puntos de estimación
Cargar las ontologías de un repositorio.	1 semana
Índice de semejanza entre ontologías según las características definidas en la métrica.	2 semanas
Índice de semejanza entre ontologías y conceptos.	1 semana
Buscar ontologías por conceptos.	1 semana
Visualizar resultado	1 semana
Buscar ontologías por palabras	1 semana
Gestionar usuario	1 semana
Autenticar usuario	1 semana
Gestionar repositorio	1 semana

Luego de haber definido las historia de los usuarios y la estimación por puntos de la duración de cada una de ellas se decide desarrollar la aplicación informática en cuatro iteraciones. Las iteraciones tienen como objetivo que el cliente decida cuales historias de usuarios implementar según la prioridad que tenga en el negocio las mismas no deben excederse de tres semanas. A continuación se expone el plan de las iteraciones donde se describe cada una de ellos.

Plan de iteraciones

Iteración 1: En esta iteración se desarrolla las historias de usuario número uno y dos, las cuales permitirán cargar las ontologías que se encuentren en un repositorio y establecer la semejanza entre dos ontologías de acuerdo a la métrica definida, obteniendo un porciento de semejanza entre las dos.

Iteración 2: En esta iteración se corregirán los errores o disconformidades del usuario con las historias de usuario implementadas en la primera iteración. Se desarrollan las historias de usuarios tres y cuatro las

cuales permitirán establecer la semejanza entre las ontologías con un concepto o varios conceptos y buscar las ontologías por conceptos, se obtendrá el porciento en el que son semejantes sus clases, propiedades e instancias, así como ver si el concepto se encuentra en alguna de las ontologías del repositorio, el nombre de esa ontología y las ontologías ordenadas ascendentemente según el porciento de los conceptos encontrados.

Iteración 3: En esta iteración se corregirán los errores o disconformidades del usuario con la historia de usuario implementada en la segunda iteración. Se desarrollan las historias de usuarios cinco y seis las cuales permiten visualizar el resultado de la semejanza en una gráfica y buscar las ontologías según las palabras encontradas con el porciento de semejanza en las ontologías, se obtendrá una gráfica con las clases, propiedades, instancias y las propiedades de las instancias, con el porciento de semejanza del resultado de la comparación definida en la métrica y las ontologías ordenadas ascendentemente según las palabras que se encuentran en ellas.

Iteración 4: Se corrigen los errores o las disconformidades del usuario de las historias de usuarios desarrolladas en la iteración tres. Se desarrolla la historia de usuario número siete, ocho y nueve las cuales permitirán adicionar, eliminar, obtener, actualizar, autenticar un usuario y adicionar, eliminar, obtener y eliminar un repositorio.

Para garantizar un desarrollo exitoso y con calidad se realiza el plan de duración de cada una de las iteraciones, en el cual se representan las iteraciones junto con las historias de usuarios que van a hacer implementadas en cada unas de ellas y la duración total de cada iteración. A continuación se muestra el plan de duración de las iteraciones:

Tabla 3: Plan de duración de las iteraciones

Iteraciones	Historias de usuarios	Duración total de iteraciones
Iteración 1	Cargar ontologías de un repositorio Índice de semejanza entre ontologías según las características definidas en la métrica.	3 semanas
	Índice de semejanza entre ontologías y conceptos.	

Iteración 2	Buscar ontologías por conceptos.	2 semanas
Iteración 3	Visualizar resultado. Buscar ontologías por palabras.	2 semanas
Iteración 4	Gestionar usuario Autenticar usuario Gestionar repositorio	3 semanas
Total		10 semanas

Para la fase de implementación de la aplicación informática se define el siguiente plan de entrega con una propuesta de la fecha aproximada en que se realizaran las versiones de prueba a la aplicación informática al finalizar cada versión en la fase de implementación.

Tabla 4: Plan de entrega

Aplicación informática para establecer semejanzas entre ontologías representativas del conocimiento en la Web	**Entregable**
Versión 0.1	Cuarta semana de marzo
Versión 0.2	Primera semana de abril
Versión 0.3	Tercera semana de abril
Versión 1.0	Primera semana de mayo

2.3. Fase de diseño.

La metodología XP propone un diseño simple que sea fácil de entender e implementar, que cueste menos tiempo y esfuerzo al desarrollar. Para ello se apoya en las tarjetas CRC, las cuales permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica.

Las tarjetas C.R.C (*Class, Responsabilities and Collaboration*) representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

La aplicación informática contará con ocho tarjetas CRC, en las cuales se representan cada unas de las clases de la aplicación informática con sus responsabilidades y colaboraciones con las demás clases, a continuación se muestra un ejemplo de la tarjeta CRC Usuario, el resto aparece en el documento “Plantilla modelo de diseño”.

Tabla 5: Tarjeta CRC: Usuario

Tarjeta CRC	
Clase: Usuario	
Responsabilidades	**Colaboraciones**
Gestionar usuario. Autenticar usuario.	Conexión.

2.4. Patrones de Diseño.

Los patrones de diseño es un tema importante en el desarrollo de software actual. Busca ayudar a los desarrolladores de software a crear un lenguaje común para comunicar ideas y experiencia acerca de los problemas y sus soluciones.

Los patrones de diseño representan soluciones a problemas que surgen cuando se desarrolla software en un contexto particular. (39)

Patrones = par Problema/Solución dentro de un Contexto

Un patrón de diseño:

Se plantean como una buena herramienta para el diseño y la documentación de aplicaciones y *frameworks (marco de desarrollo)*.

Son descripciones de objetos y clases que se comunican y que son capaces de solucionar un problema de diseño en general, en un contexto en particular.

"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma". (39)

Los patrones de diseño deben resolver un problema, ser reutilizable y proporcionar suficiente información para la elaboración de la solución.

Un patrón está compuesto por elementos como:

Nombre: describe el problema de diseño.

El problema: describe cuándo aplicar el patrón.

La solución: describe los elementos que componen el diseño, sus relaciones, responsabilidades y colaboración.

Beneficios de uso de los Patrones de Diseño.

Proponen una forma de reutilizar la experiencia de los programadores. Contribuyen a dar flexibilidad, extensibilidad a los diseños y a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. Mejoran (aumentan, elevan) la flexibilidad, modularidad y extensibilidad. Incrementan el vocabulario de diseño, ayudando a diseñar desde un mayor nivel de abstracción. (40)

Para el desarrollo de la aplicación informática se usan los Patrones de Principios Generales para Asignar Responsabilidades (*General Responsability Assignment Software Patterns,* GRAPS.)

Los patrones *GRASP* se aplican en los primeros momentos del sistema, constituyen la base de cómo se diseñará el mismo. *"Describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones".* (39)

UML define una ***responsabilidad*** *como "un contrato u obligación de un clasificador". Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento.* (39)

Entre los patrones *GRASP* a utilizar en el diseño de la aplicación informática tendremos a los que se mencionan a continuación:

- Experto en información
- Creador
- Alta cohesión
- Bajo acoplamiento
- Controlador.

Patrón Experto en información: es el encargado de asignar la responsabilidad de **experto**, a la clase que tiene la información necesaria para implementar una responsabilidad. Este patrón presenta beneficios como: (39)

- Encapsulamiento de la información
- Distribución del comportamiento del manejo de la información

En la siguiente figura se muestra como la clase Comparar Ontología tiene asignada la responsabilidad de realizar todo el proceso de comparación de las ontologías mediante los métodos de comparación por las métricas definidas y según sus conceptos.

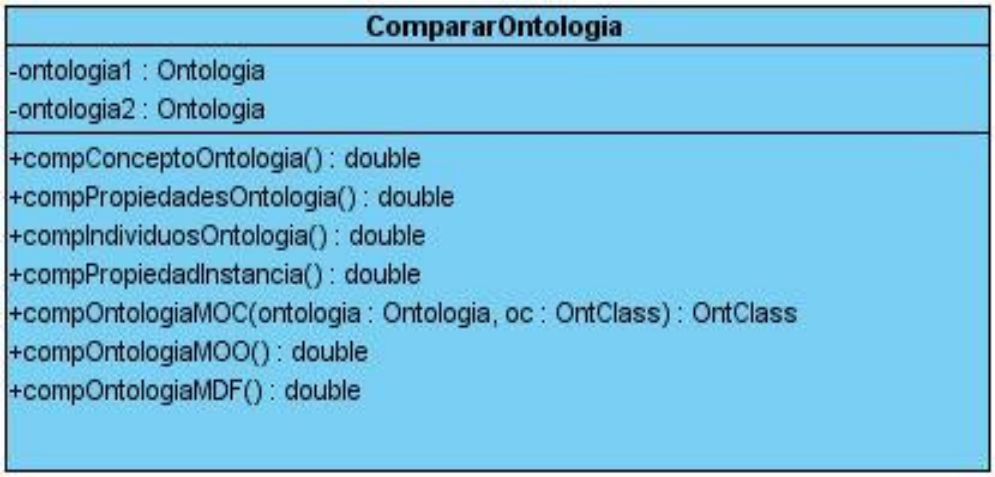

Figura 6 Patrón GRASP Experto en información.

Patrón Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea muy común).Su propósito es encontrar un creador que permita conectarse al objeto producido en cualquier evento. Tiene como beneficio el bajo acoplamiento, logrando una mejor sostenibilidad y reutilización. En la figura 7 se muestra la creación de un objeto de unas de las clases. (39)

```
String prop = request.getParameter("id2");
String inst = request.getParameter("id3");
String propinst = request.getParameter("id4");
String repositorios = request.getParameter("id5");
LinkedList<String> clases = CompConcepto.parserClases(clas);
LinkedList<String> propied = CompConcepto.parserEntrada(prop);
LinkedList<String> instanc = CompConcepto.parserEntrada(inst);
LinkedList<String> propiedInst = CompConcepto.parserEntrada(propinst);
LinkedList<CompConcepto> ccs = new LinkedList<CompConcepto>();

if(clases.size() == propied.size() && propied.size() == instanc.size() && instanc.size() == propi
    for(int i = 0; i < clases.size(); i++) {
        CompConcepto cc = new CompConcepto(clases.get(i), CompConcepto.parserEntradaE(propied.get
        ccs.add(cc);
    }
}

CompConcepto cc = new CompConcepto(clas, CompConcepto.parserEntrada(prop), CompConcepto.parserEn
CompararOntologia co = new CompararOntologia(repositorios);
double [] resultado = co.compararConceptoRepositorio(ccs);
String res = "";
res = "["; res += ""+resultado[0]; res += ","; res += ""+resultado[1]; res += ","; res += ""+resu
res += ""+resultado[3]; res += "]";
int pos = (int)resultado[4];
String nombre = co.getOntologias().get(pos).getDireccion();
out.print(nombre);
```

Figura 7 Fragmento de la creación de un objeto.

Patrón Bajo Acoplamiento: guía la asignación de responsabilidades relacionadas con el acoplamiento entre las clases, garantizando la menor dependencia entre cada unas de ellas de forma tal que de producirse alguna modificación en algunas de ellas no se afecten los cambios en otros componentes, potenciando la reutilización. En la siguiente figura se presenta un diagrama de clases donde se evidencia éste patrón. (39)

Figura 8 Patrón Bajo Acoplamiento.

Patrón Alta Cohesión: es el encargado de garantizar que las clases que responden a los eventos del sistema se mantenga con las responsabilidades que les fueron asignadas, incrementa la claridad y facilita la comprensión del diseño. En la siguiente figura se muestra un diagrama de clases donde se evidencia éste patrón. (39)

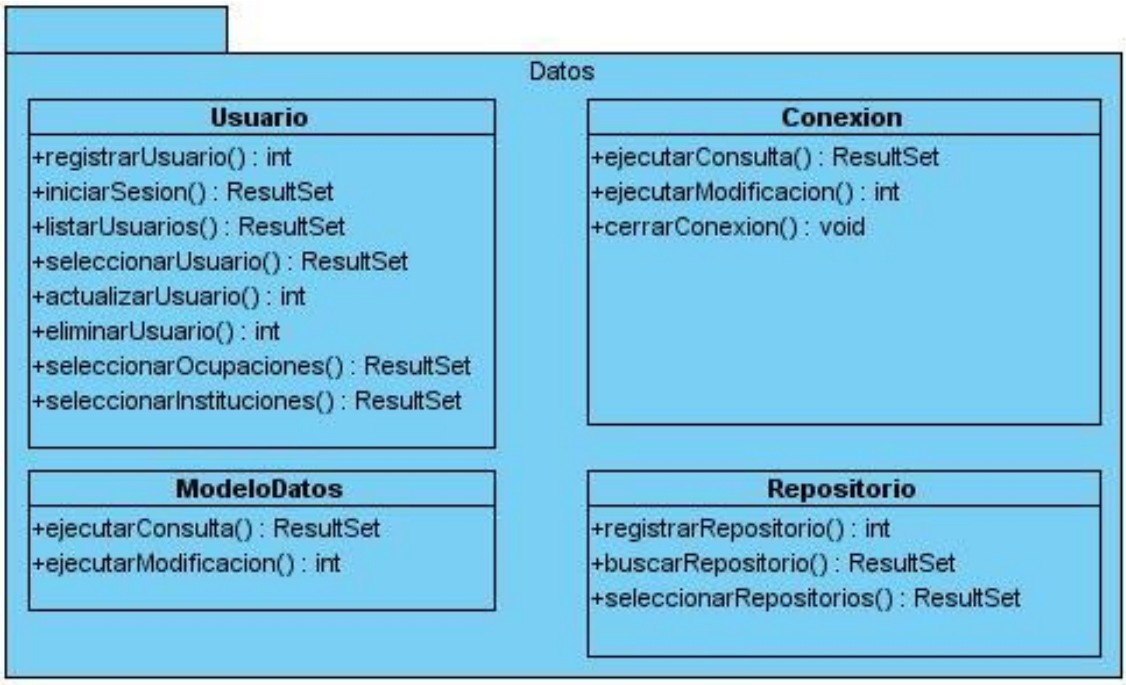

Figura 9 Patrón Alta Cohesión.

Patrón Controlador: es el encargado de asignar la responsabilidad de recibir o manejar un mensaje de **evento del sistema** a una clase. Normalmente un controlador *delega* en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. Aumenta el potencial para reutilizar las interfaces. (39)

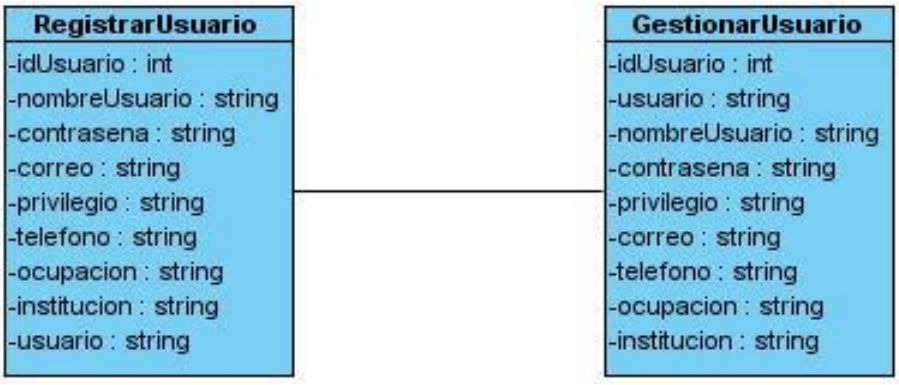

Figura 10 Patrón Controlador.

2.5. Patrones de Arquitectura.

Arquitectura de Software.

Arquitectura de software de aplicación es el proceso de definición de una solución estructurada que cumple con todos los requisitos técnicos y operacionales, al tiempo que optimiza la calidad de los atributos comunes, tales como rendimiento, seguridad y manejabilidad. Se trata de una serie de decisiones basadas en una amplia gama de factores, y cada una de estas decisiones pueden tener un impacto considerable en la calidad, rendimiento, facilidad de mantenimiento, y el éxito global de la aplicación. (41)

Un patrón de arquitectura de software describe un problema particular de diseño recurrente y presenta un esquema genérico de sus soluciones. El esquema de solución contiene componentes, sus responsabilidades y relaciones. (42)

Son patrones del software los cuales se encargan de definir la estructura de un sistema, estos a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema. Un patrón arquitectónico se enfoca a dar solución a un problema en específico, de un atributo de calidad, y abarca solo parte de la arquitectura. (43)

Atendiendo a los criterios anteriormente planteados se puede decir que los patrones de arquitectura se encargan de definir la estructura de un sistema y se enfoca a dar solución a un problema específico, apoyándose en una serie de directivas para organizar los componentes del sistema con el objetivo de facilitar el diseño del mismo.

Capas: Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtareas, las cuales se clasifican de acuerdo a un nivel particular de abstracción. (44)

La capa de presentación (interfaz de usuario): interacciona con el usuario, presenta los datos y recibe las entradas.

La capa de aplicación/negocio (lógica de aplicación): responsable de las tareas propias de la aplicación concreta aplica las reglas de negocio sobre los datos y las entradas de usuario.

La capa de datos (almacenamiento y acceso a datos): responsable de la gestión y almacenamiento permanente de los datos. (45)

Para obtener una mejor estructuración de la aplicación informática a desarrollar se hará uso del patrón arquitectónico modelo-vista-controlador.

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura empleado en aplicaciones que manejan gran cantidad de datos, garantizando un sistema mejor estructurado, facilitando una programación paralela e independiente en diferentes capas (46)

La arquitectura *MVC* separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

El modelo: se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

La vista: representa el modelo en forma gráfica disponible para la interacción con el usuario.

El controlador: se encarga de manejar y responder las solicitudes del usuario. (46)

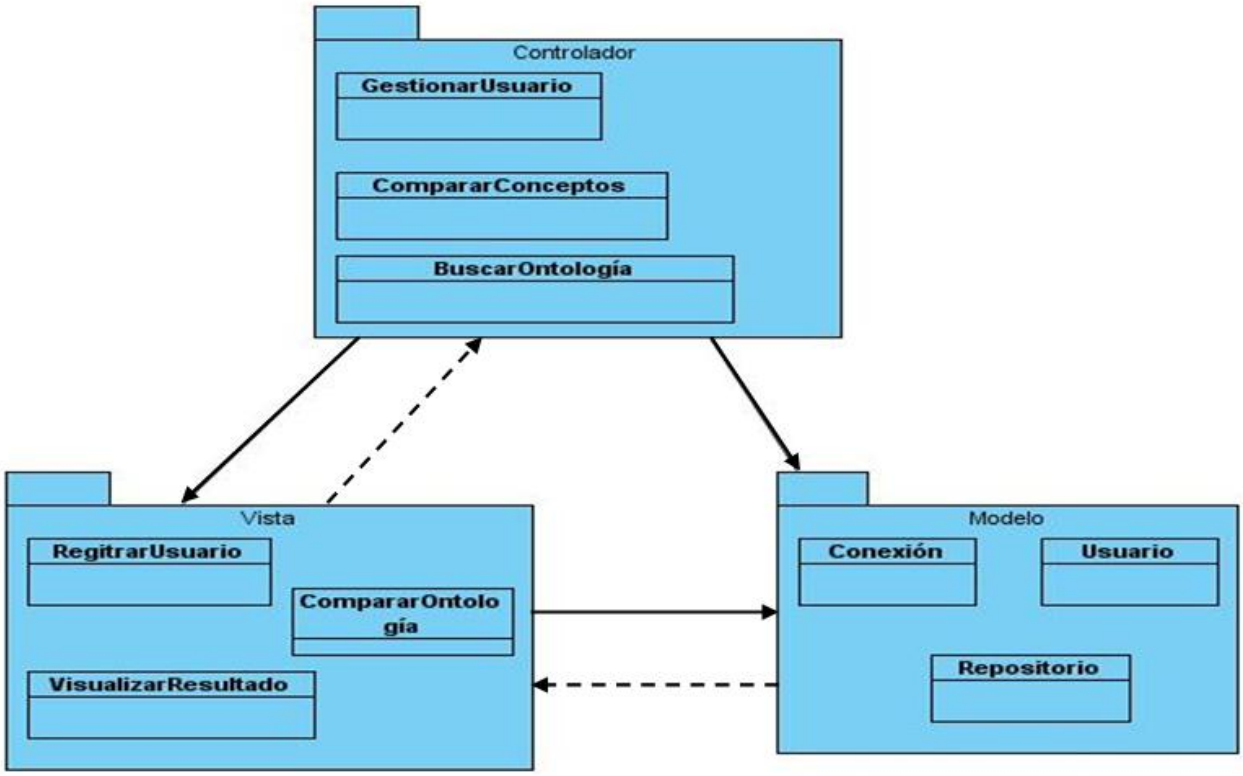

Figura 11 Patrón Controlador.

2.6. Conclusiones del capítulo 2

En este capítulo se definieron las principales características que poseerá la aplicación informática desarrollada, comenzando con la planificación de la aplicación informática donde se obtuvieron nueve historias de usuario, el plan de iteraciones y plan de entrega, la definición de los requisitos no funcionales y del diseño se obtuvieron ocho tarjetas CRC. Se redactaron todos los artefactos durante el ciclo de vida del proceso de desarrollo de software seleccionado.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DE LA APLICACIÓN INFORMÁTICA

3.1 Introducción

En este capítulo se realizará la representación del código fuente de los principales componentes así como los mecanismos de implementación utilizados. Se obtendrá resultados del diseño implementando la aplicación informática en términos de componentes y garantizar la calidad de software mediante la realización de pruebas de aceptación.

3.2 Fase de implementación

En esta fase se implementarán las historias de usuario definidas en la fase de planificación, las cuales son divididas en tareas de desarrollo para ayudar a organizar la implementación de las mismas. Las tareas de desarrollo solamente son usadas por los programadores, estas pueden estar descritas por un lenguaje técnico y no ser necesariamente entendible por el cliente.

De acuerdo a la planificación realizada en el capítulo anterior, se llevaron a cabo cuatro iteraciones, obteniéndose como finalidad un producto funcional con las características deseadas.

Iteración 1

En esta iteración se desarrolla las historias de usuario número uno y dos, las cuales permitirán cargar las ontologías que se encuentren en un repositorio y establecer el índice de semejanzas entre dos ontologías de acuerdo a la métrica definida.

- **HU Cargar las ontologías de un repositorio**

Tabla 6: Tarea 1 de la HU "Cargar las ontologías de un repositorio"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Diseño de la interfaz para cargar las ontologías de un repositorio	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 28/2/2011	**Fecha Fin:** 1/3/2011

Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García
Descripción: Se implementa el diseño de la interfaz para cargar las ontologías de un repositorio con los componentes necesarios para su funcionamiento.

Tabla 7: Tarea 2 de la HU "Cargar las ontologías de un repositorio"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Validar los datos para cargar las ontologías de un repositorio	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 2/3/2011	**Fecha Fin:** 3/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa las funcionalidades que permiten la captura de los datos, se verifican que sean correctos, en caso de que sean correctos se redirecciona a una página donde se mostrarán las ontologías de ese repositorio.	

Tabla 8: Tarea 3 de la HU "Cargar las ontologías de un repositorio"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 1
Nombre Tarea: Mostrar las ontologías de un repositorio	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 4/3/2011	**Fecha Fin:** 5/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa las funcionalidades que permiten mostrar las ontologías de ese repositorio.	

- **HU Índice de semejanza entre ontologías según las características definidas en la métrica.**

Tabla 9: Tarea 1 de la HU "Índice de semejanza entre ontologías según las características definidas en la métrica"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 2
Nombre Tarea: Validar los datos para establecer el índice de semejanza entre las ontologías.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 7/3/2011	**Fecha Fin:** 9/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa las funcionalidades que permiten la captura de los datos para establecer el índice de semejanza entre las ontologías, se verifican que sean correctos, en caso de que sean correcto se redirecciona a una página donde se mostrarán el resultado de la comparación.	

Tabla 10: Tarea 2 de la HU "Índice de semejanza entre ontologías según las características definidas en la métrica"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Comprar los conceptos que conforman las ontologías	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 10/3/2011	**Fecha Fin:** 12/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se lee el fichero de la ontología, los conceptos por los que está formada y se comparan, con los conceptos de la otra ontología para ver el porciento	

de semejanza entre estos.

Tabla 11: Tarea 3 de la HU "Índice de semejanza entre ontologías según las características definidas en la métrica"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Comprar las propiedades de los conceptos que la conforman	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 14/3/2011	**Fecha Fin:** 15/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se lee el fichero de la ontología, las propiedades de los conceptos por los que está formada y se comparan con las propiedades de los conceptos de la otra ontología para ver el porciento de semejanza entre estas.	

Tabla 12: Tarea 4 de la HU "Índice de semejanza entre ontologías según las características definidas en la métrica"

Tarea	
Número Tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea: Comprar las instancias que conforman las ontologías.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 16/3/2011	**Fecha Fin:** 17/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se lee el fichero de la ontología, las instancias de los conceptos por los que está formada y se comparan con las instancias de los conceptos de la otra ontología para ver el porciento de semejanza entre estos.	

Tabla 13: Tarea 5 de la HU "Índice de semejanza entre ontologías según las características definidas en la métrica"

Tarea	
Número Tarea: 5	**Número Historia de Usuario:** 2
Nombre Tarea: Comprar las propiedades de las instancias que conforman la ontología	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 18/3/2011	**Fecha Fin:** 19/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se lee el fichero de la ontología, las propiedades de las instancias de los conceptos por los que está formada y se comparan con las propiedades de las instancias de los conceptos de la otra ontología para ver el porciento de semejanza entre estas.	

Iteración 2

En esta iteración se desarrollan las historias de usuarios tres y cuatro las cuales permitirán establecer el índice de semejanza entre ontologías y conceptos, y buscar las ontologías por conceptos.

- **HU Índice de semejanza entre ontologías y conceptos.**

Tabla 14: Tarea 1 de la HU "Índice de semejanza entre ontologías y conceptos"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 3
Nombre Tarea: Diseño de la interfaz para establecer el índice de semejanza entre las ontologías y conceptos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 21/3/2011	**Fecha Fin:** 22/3/2011

Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García
Descripción: Se implementa el diseño de la interfaz para establecer el índice de semejanza entre las ontologías y conceptos con los componentes necesarios para su funcionamiento.

Tabla 15: Tarea 2 de la HU "Índice de semejanza entre ontologías y conceptos"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 3
Nombre Tarea: Validar los datos para establecer el índice de semejanza entre las ontologías y conceptos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 23/3/2011	**Fecha Fin:** 24/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa las funcionalidades que permiten la captura de los datos para establecer el índice de semejanza entre las ontologías y conceptos, se verifican que sean correctos, en caso de que sean correctos se redirecciona a una página donde se mostrarán el resultado de la comparación.	

Tabla 16: Tarea 3 de la HU "Índice de semejanza entre ontologías y conceptos"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 3
Nombre Tarea: Establecer el índice de semejanza entre las ontologías y conceptos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 25/3/2011	**Fecha Fin:** 26/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Esta funcionalidad describe como se llevará a cabo el proceso de	

establecer el índice de semejanza entre las ontologías según los conceptos que la conforman, con varios conceptos, se visualizarán la ontología de mayor similitud.

- **HU Buscar ontologías por conceptos.**

Tabla 17: Tarea 1 de la HU "Buscar las ontologías por conceptos"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 4
Nombre Tarea: Diseño de la interfaz para buscar las ontologías por conceptos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 28/3/2011	**Fecha Fin:** 29/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Diseño de la interfaz para buscar las ontologías por conceptos con los componentes necesarios para su funcionamiento.	

Tabla 18: Tarea 2 de la HU "Buscar las ontologías por conceptos"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 4
Nombre Tarea: Validar los datos para buscar las ontologías por conceptos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 30/3/2011	**Fecha Fin:** 31/3/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa las funcionalidades que permiten la captura de los datos para buscar las ontologías por los conceptos, se verifican que sean correctos, en caso de que sean correctos se redirecciona a una página donde se mostrarán el resultado de la búsqueda.	

Tabla 19: Tarea 3 de la HU "Buscar las ontologías por conceptos"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 4
Nombre Tarea: Buscar conceptos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 1/4/2011	**Fecha Fin:** 1/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Dado uno o varios conceptos, se verifica si existe en las ontologías que se encuentran en un repositorio y retorna las ontologías que tengan algo de similitud, quedando ordenadas ascendentemente.	

Tabla 20: Tarea 4 de la HU "Buscar las ontologías por conceptos"

Tarea	
Número Tarea: 4	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar ontologías	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 2/4/2011	**Fecha Fin:** 2/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se crea una interfaz mostrando las ontologías ordenadas ascendentemente que tienen similitud con los conceptos entrados.	

Iteración 3

Se desarrolla las historias de usuario cinco y seis, las cuales permitirán visualizar el resultado de la comparación en una gráfica y buscar ontologías por palabras.

- **HU Visualizar resultados**

Tabla 21: Tarea 1 de la HU "Visualizar resultados"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 5
Nombre Tarea: Visualizar la semejanza entre ontologías	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 4/4/2011	**Fecha Fin:** 6/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se crea una interfaz con una gráfica con los conceptos, propiedades, instancias y propiedades de las instancias y el por ciento de semejanza que existe en las ontologías comparadas.	

Tabla 22: Tarea 2 de la HU "Visualizar resultados"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 5
Nombre Tarea: Visualizar la semejanza entre ontología y concepto.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 7/4/2011	**Fecha Fin:** 9/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se crea una interfaz con una gráfica con los conceptos, propiedades, instancias y propiedades de las instancias y el por ciento de semejanza que existe en las ontologías en el repositorio y los conceptos.	

- **HU Buscar ontologías por palabras**

Tabla 23: Tarea 1 de la HU "Buscar ontologías por palabras"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 6
Nombre Tarea: Diseño de la interfaz para buscar ontologías por palabras	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 11/4/2011	**Fecha Fin:** 12/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Diseño de la interfaz para buscar las ontologías por palabras, con los componentes necesarios para su funcionamiento.	

Tabla 24: Tarea 2 de la HU "Buscar ontologías por palabras"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 6
Nombre Tarea: Validar los datos para buscar ontologías por palabras	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 13/4/2011	**Fecha Fin:** 14/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa las funcionalidades que permiten la captura de los datos para buscar las ontologías por los palabras, se verifican que sean correctos, en caso de que sean correcto se redirecciona a una página donde se mostrarán el resultado de la búsqueda.	

Tabla 25: Tarea 3 de la HU "Buscar ontologías por palabras".

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 6
Nombre Tarea: Buscar ontologías por palabras	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 15/4/2011	**Fecha Fin:** 15/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Describe la búsqueda de una ontología a partir de una o varias palabras determinadas entradas por el usuario.	

Tabla 26: Tarea 4 de la HU "Buscar ontologías por palabras"

Tarea	
Número Tarea: 4	**Número Historia de Usuario:** 6
Nombre Tarea: Mostrar ontologías por la búsqueda de palabras.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 16/4/2011	**Fecha Fin:** 16/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se crea una interfaz mostrando las ontologías ordenadas ascendentemente que tienen similitud con las palabras entradas.	

Iteración 4

Se desarrollan las historias de usuarios número siete, ocho y nueve la cual permitirá adicionar, eliminar, obtener, actualizar y autenticar un usuario, además de adicionar, obtener y eliminar un repositorio.

- **HU Gestionar usuario**

Tabla 27: Tarea 1 de la HU "Gestionar usuario"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 7
Nombre Tarea: Configuración de las interfaces de gestionar usuario	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 18/4/2011	**Fecha Fin:** 19/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se configuran las interfaces de usuario correspondientes para crear, modificar y eliminar cuentas de los usuarios necesarias para el administrador.	

Tabla 28: Tarea 2 de la HU "Gestionar usuario"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 7
Nombre Tarea: Captura de los datos del usuario	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 20/4/2011	**Fecha Fin:** 21/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se capturan los datos introducidos en caso de crear un nuevo usuario para posterior comprobación o se muestran los usuarios para capturar los datos de un usuario específico y realizar la acción deseada posteriormente.	

Tabla 29: Tarea 3 de la HU "Gestionar usuario"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 7
Nombre Tarea: Gestión de datos del usuario en la base de datos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 22/4/2011	**Fecha Fin:** 23/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Para ingresar el nuevo usuario, si este no existe en la base de datos sus datos se almacenan en la misma, en el caso de que exista ya se pueden realizar todas las acciones de gestionar de acuerdo con la opción escogida.	

- **HU Autenticar usuario**

Tabla 30: Tarea 1 de la HU "Autenticar usuario"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 8
Nombre Tarea: Diseño de la interfaz de autenticación	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 25/4/2011	**Fecha Fin:** 27/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementa el diseño de la interfaz de autenticación, con los componentes necesarios para su funcionamiento.	

Tabla 31: Tarea 2 de la HU "Autenticar usuario"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 8
Nombre Tarea: Validar datos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 28/4/2011	**Fecha Fin:** 30/4/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se implementan las funcionalidades que permiten la captura de los datos, se verifican que sean correctos con la base de datos de la aplicación informática.	

- **HU Gestionar repositorio**

Tabla 32: Tarea 1 de la HU "Gestionar repositorio"

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 9
Nombre Tarea: Configuración de las interfaces de gestionar repositorio.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 2/5/2011	**Fecha Fin: 3**/5/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se configuran las interfaces para gestionar repositorios para crear, obtener y eliminar los repositorios, necesarias para el administrador.	

Tabla 33: Tarea 2 de la HU "Gestionar repositorio"

Tarea	
Número Tarea: 2	**Número Historia de Usuario:** 9
Nombre Tarea: Captura de los datos del para gestionar el repositorio.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 4/5/2011	**Fecha Fin:** 5/5/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Se capturan y se validan los datos introducidos en caso de crear, obtener o eliminar un repositorio.	

Tabla 34: Tarea 3 de la HU "Gestionar repositorio"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 9
Nombre Tarea: Gestión de datos del repositorio en la base de datos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 6/5/2011	**Fecha Fin:** 7/5/2011
Programador Responsable: Yahíma Matos Galvez - Elio Luis Toledo García	
Descripción: Para ingresar el nuevo repositorio, si este no existe en la base de datos sus datos se almacenan en la misma, en el caso de que exista ya se pueden realizar todas las acciones de gestionar de acuerdo con la opción escogida.	

3.2.1 Estándar de codificación

Los estándares de codificación, también llamados estilos de programación o convenciones de código, no son más que convenios para escribir código fuente en ciertos lenguajes de programación. Estos estándares elevan la mantenibilidad del código, sirven como punto de referencia para los programadores,

mantienen un estilo de programación y ayudan a mejorar el proceso de codificación, haciéndolo, entre otras cosas, mucho más eficiente.

Las convenciones de código son importantes para los programadores por un gran número de razones: (47)

- El 80% del coste del código de un programa va a su mantenimiento.
- Casi ningún software lo mantiene toda su vida el auto original.
- Las convenciones de código mejoran la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo.
- Si distribuyes tu código fuente como un producto, necesitas asegurarte de que está bien hecho y presentado como cualquier otro producto.

Algunas de estas convenciones se pueden ver a continuación: (47)

- **Identación**

Se deben emplear cuatro espacios como unidad de indentación.

- **Longitud de la línea**

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

- **Comentarios**

Existen dos tipos de comentario, los de implementación y los de documentación, estos últimos existen solo en Java y se encuentran limitados por /**…*/.

```
/**
 * La clase Ejemplo ofrece ...
 */
public class Ejemplo { ...
```

- **Declaraciones**

Se realizará una declaración por línea, ya que facilita los comentarios. Es decir, preferiblemente

```
int nivel; // nivel de indentación
int tam;   // tamaño de la tabla
```

antes que

```
int level, size;
```

- **Inicialización**

Se inicializarán las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

- **Declaraciones de clases**

Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- No debe existir ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.
- La llave de cierre "}" empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{"
- Los métodos se separan con una línea en blanco.

```
class Ejemplo extends Object {
    int ivar1;
    int ivar2;

    Ejemplo(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int metodoVacio() {}

    ...
}
```

- **Sentencia if, if-else, if else-if else**

La clase de sentencias if-else debe tener la siguiente forma y deben usar siempre llaves {}.

```
if (condicion) {
    sentencias;
}

if (condicion) {
    sentencias;
} else {
    sentencias;
}

if (condicion) {
    sentencia;
} else if (condicion) {
    sentencia;
} else{
    sentencia;
}
```

- **Sentencias for**

Una sentencia for debe tener la siguiente forma:

```
for (inicializacion; condicion; actualizacion) {
    sentencias;
}
```

- **Sentencias while**

Una sentencia while debe tener la siguiente forma:

```
while (condicion) {
    sentencias;
}
```

- **Variables**

Todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son complkauestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje.

3.3 Fase de prueba

Una de las características fundamentales que posee la metodología XP es el proceso de prueba para comprobar las funcionalidades a medida que estas son implementadas, y a su vez mostrar al cliente los resultados obtenidos durante el desarrollo del producto. De esta forma se busca disminuir la cantidad de errores no detectados reduciendo el tiempo transcurrido entre la aparición de un error y su detección, aumentando así la calidad de la aplicación informática y la seguridad de evitar efectos colaterales a la hora de realizar modificaciones.

Esta metodología ágil divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias tienen como objetivo revisar el código de forma automática y son desarrolladas por los programadores. En cambio las pruebas de aceptación están destinadas a verificar que las historias de usuario cumplen con las funcionalidades requeridas por el cliente al final de cada iteración. Las pruebas de aceptación tienen mayor peso que las unitarias ya que representan un indicador de la satisfacción del cliente con el producto. Este tipo de pruebas debe realizarse lo más rápido posible para que los programadores puedan hacer los cambios que se necesiten con mayor rapidez.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación se realizan para asegurar que las funcionalidades cumplen su objetivo y que la aplicación informática es lo que el cliente realmente necesita. Estas pruebas son creadas a partir de las historias de usuario y es el cliente quien especifica los aspectos para comprobar que se han implementado correctamente. Una historia de usuario puede tener todas las pruebas de aceptación que necesite para garantizar su buen funcionamiento.

Estas pruebas funcionan como una caja negra ya que cada una de ellas representa una salida esperada de la aplicación informática. Los clientes son los responsables de verificar que los resultados de las pruebas sean correctos y de tomar decisiones acerca de las mismas. Una vez que todas las historias de usuario hayan pasado su prueba de aceptación, entonces se considera terminada la aplicación informática.

A continuación se muestran los casos de prueba de la historia de usuario “Cargar ontologías de un repositorio”, el resto se puede encontrar en el documento “Plantilla caso de prueba de aceptación”.

Tabla 35: Prueba 1 de la HU "Cargar ontologías de un repositorio"

Caso de prueba de aceptación	
Código: HU1_p1	**Historia de Usuario:** 1
Nombre: Entrar URL incorrecta	
Descripción: Probar que la dirección que se escriba del repositorio en la aplicación informática no sea válida.	
Condiciones de ejecución: La URL este mal conformada.	
Entrada/ pasos de ejecución: Intentar escribir una dirección en la aplicación informática que no exista un repositorio, que esté mal conformada la URL o dejar el campo vacío.	
Resultado esperado: La aplicación informática muestra al usuario que la URL entrada no es válida y se le dé la posibilidad de intentarlo nuevamente.	
Evaluación de la prueba: Satisfactoria	

Tabla 36: Prueba 2 de la HU "Cargar ontologías de un repositorio"

Caso de prueba de aceptación	
Código: HU1_p2	**Historia de Usuario:** 1
Nombre: Entrar URL correcta	
Descripción: Probar que la dirección que se escriba del repositorio en la aplicación informática sea válida.	
Condiciones de ejecución: La URL este correcta.	
Entrada/ pasos de ejecución: Escribir correctamente en la aplicación informática una dirección de un repositorio existente.	
Resultado esperado: Se cargan las ontologías existentes en el repositorio.	
Evaluación de la prueba: Satisfactoria	

3.4 Conclusiones del capítulo 3

En este capítulo se implementaron las nueve historias de usuarios definidas en la fase de planificación, las cuales fueron divididas en 29 tareas de desarrollo para ayudar a organizar la implementación de las mismas. Se definió el estándar de codificación a seguir para la implementación de la aplicación informática. Se desarrollaron las pruebas de aceptación para verificar el funcionamiento, probando individualmente la implementación de cada historia de usuario, las cuales arrojaron diez no conformidades que fueron corregidas.

CONCLUSIONES GENERALES

El análisis que se realizó arrojó los siguientes resultados: el desarrollo de la aplicación informática estuvo guiado por la metodología XP, como framework para la gestión de ontologías se utilizó Jena y como lenguaje de programación Java.

En la planificación y el diseño realizado se obtuvieron nueve historias de usuario y ocho tarjetas CRC que guiaron el proceso de implementación de la aplicación informática.

Se desarrolló la aplicación informática para establecer semejanzas entre ontologías representativas del conocimiento en la Web para facilitar la toma de decisiones en el proceso de reutilización de los conceptos en diferentes escenarios.

Se realizaron pruebas de aceptación a la aplicación informática para validar su correcto funcionamiento las cuales arrojaron diez no conformidades que fueron corregidas satisfactoriamente.

RECOMENDACIONES

- Desarrollar un componente para el trabajo con ontologías que cubran las necesidades que hoy hacen indispensable el uso de Jena en la aplicación informática, puesto que este consume muchos recursos del servidor.
- Incorporar nuevas características a la métrica definida para mejorar la exactitud del índice de semejanzas entre ontologías representativas del conocimiento en la Web.
- Se recomienda la integración de la aplicación informática para establecer semejanzas entre ontologías representativas del conocimiento en la Web con las siguientes aplicaciones: Aplicación informática para caracterizar ontologías representativas del conocimiento en la Web y Aplicación informática para gestionar repositorios de ontologías representativas del conocimiento en la Web.

REFERENCIAS BIBLIOGRÁFICAS

1. **Àngels Rius;Miguel Ángel Sicilia;Elena García.** Justificación y Descripción del Dominio de Conocimiento de una Ontología para la Formalización y Automatización de Escenarios Educativos. [En línea] [Citado el: 25 de 2 de 2011.] http://spdece07.ehu.es/actas/Rius.pdf.

2. Ontología y Web Semántica. *Ontología y Web Semántica.* [En línea] [Citado el: 25 de 2 de 2011.] http://www.info-ab.uclm.es/asignaturas/300219/pdf/WebSemantica.pdf.

3. Definición.de. *Definición.de.* [En línea] [Citado el: 25 de 2 de 2011.] http://definicion.de/semantica/.

4. Definición.de. *Definición.de.* [En línea] [Citado el: 25 de 2 de 2011.] http://definicion.de/explicito/.

5. **Gruber, T. R.** What is an Ontology? *What is an Ontology?* [En línea] 2001. [Citado el: 25 de 2 de 2011.] http://www-ksl.standford.edu/kst/what-is-an-ontology.html.

6. **Nicolas Guarino.** *Understanding, Building, and Using Ontologies.* 1996.

7. **Grauber, T. R.** What is an Ontology? *What is an Ontology?* [En línea] [Citado el: 25 de 2 de 2011.] http://www-ksl.standford.edu/kst/what-is-an-ontology.html..

8. **Ignacio Herrero Jiménez.** *DESARROLLO DE ONTOLOGÍAS SOBRE BBDD RELACIONALES:ONTOLOGÍA QUE SOPORTA LA BÚSQUEDA AVANZADA DE INFORMACIÓN.* España : s.n., 2004.

9. **Sowa J. F.** *Ontology, Metadata, and Semiotics.* Darmstadt, Germany : s.n., August 14, 2000.

10. **Adolfo Lozano Tello.** Ontologías en la Web Semántica. *Ontologías en la Web Semántica.* [En línea] 2001. [Citado el: 25 de 2 de 2011.] www.inf.udec.cl/~andrea/cursos/retrieval/web.pdf.

11. **GUARINO, N.** *Formal Ontology, Conceptual Analysis and Kwoledge Representation.* 1995.

12. **VAN HEIJST, G., SCHEREIBER, A.T. Y WIELINGA, B.J.** *Using Explicit Ontologies in KBS Development.* 1996.

13. **Guillermo A. Cuéllar N.** N. unicauca.edu.co. *N. unicauca.edu.co.* [En línea] [Citado el: 26 de 2 de 2011.] http://fccea.unicauca.edu.co/old/siconceptosbasicos.htm..

14. **Graciela Barchini, Margarita Álvarez, Susana Herrera y Melina Trejo.** *EL ROL DE LAS ONTOLOGÍAS EN LOS SI.* Argentina : s.n., 2007.

15. **Graciela Elisa BARCHINI; Margarita María ÁLVAREZ; Diana PALLIOTTO; Susana HERRERA; Paola BUDÁN.** *ONTOLOGÍAS EN LOS SISTEMAS DE INFORMACIÓN / CONOCIMIENTO.*

16. **Graciela Barchini;Margarita Álvarez;Susana Herrera.** *SISTEMAS DE INFORMACIÓN: NUEVOS ESCENARIOS BASADOS EN ONTOLOGÍAS INFORMATION SYSTEMS: NEW ONTOLOGY-BASED SCENARIOS.* 2006.

17. **Sheth, A.P.** Changing focus on interoperability in information systems: from system, syntax, structure to. [En línea] 20 de 11 de 2010. [Citado el: 26 de 2 de 2011.] http://lsdis.cs.uga.edu/ library/download/S98-changing.pdf.

18. **Nicolas Guarino.** Formal Ontology and Information Systems. [En línea] [Citado el: 27 de 2 de 2011.] http://citeseer.ist.psu.edu/guarino98formal.html.

19. **Wand, Y. and Weber, R.** *An Ontological Model of an Information System.* 1990.

20. Sitio Web de la Oficina Española del W3C. *Sitio Web de la Oficina Española del W3C.* [En línea] [Citado el: 26 de 2 de 2011.] http://www.w3c.es/divulgacion/guiasbreves/websemantica.

21. **Berners-Lee T., Hendler J. and Lassila O.** *The Semantic Web.* 2001.

22. **GRUBER, T. R.** *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* 1995.

23. **Castells, Pablo.** La Web Semántica. *La Web Semántica.* [En línea] [Citado el: 26 de 2 de 2011.] http://www.ii.uam.es/~castells.

24. **Adolfo Lozano Tello.** MÉTRICA DE IDONEIDAD DE ONTOLOGÍAS. [En línea] 2 de 2002. http://www.pcid.es/public.htm.

25. **Pressman.** *Aspectos y metricas de la calidad del Software.* 1998.

26. *IEEE Standard for Software Quality Management System.* 1992.

27. **Lissete González Gallo; Reynaldo Alvarez Luna.** *Solución Informática para el Centro Nacional de Balance Alimentario: Implementacion de un componente de software para la transferencia de datos a través de Internet.* Habana : s.n., 2008. Solución Informática para el Centro Nacional de Balance Alimentario: Implementacion de un componente de software para la transferencia de datos a través de Internet.

28. **Gerardo Fernández Escribano.** *Introducción a Extreme Programming.* 2002.

29. LENGUAJES DE PROGRAMACIÓN. *LENGUAJES DE PROGRAMACIÓN.* [En línea] [Citado el: 28 de 2 de 2011.] http://frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm.

30. Definición.de. *Definición.de.* [En línea] [Citado el: 27 de 2 de 2011.] http://definicion.de/java/ .

31. sesame. *sesame.* [En línea] [Citado el: 28 de 2 de 2011.] http://www.openrdf.org/doc/sesame/users/.

32. jena. *jena.* [En línea] http://jena.sourceforge.net/.

33. **Marco Villalvi.** Ontología Medio Ambiente. *Ontología Medio Ambiente.* [En línea] [Citado el: 28 de 2 de 2011.] http://ontoguate.wordpress.com/2009/01/25/ontologia-medio-ambiente/.

34. The Dojo Toolkit. *The Dojo Toolkit.* [En línea] [Citado el: 5 de 4 de 2011.] http://dojotoolkit.org/features/.

35. Sobre PostgreSQL | www.postgresql-es.org. *Sobre PostgreSQL | www.postgresql-es.org.* [En línea] [Citado el: 28 de 2 de 2011.] http://www.postgresql-es.org/sobre_postgresql.

36. Eclipse - The Eclipse Foundation open source community website. *Eclipse - The Eclipse Foundation open source community website.* [En línea] http://www.eclipse.org/.

37. Welcome to NetBeans. *Welcome to NetBeans.* [En línea] [Citado el: 28 de 2 de 2011.] http://netbeans.org/.

38. **schermbeek, Mariangela Pocaterra Van.** APACHE. *APACHE.* [En línea] [Citado el: 28 de 2 de 2011.] http://ddd.uab.cat/pub/trerecpro/2009/hdl_2072_41829/Treball+de+recerca.pdf.

39. **Prof. Ivette C. Martínez.** *DISEÑO CON PATRONES.* Universidad Simón Bolívar. Ing. de Software : s.n.

40. **Vanina Barotto;Mauricio Demonte.** *DEFINICIÓN DE PATRONES DE DISEÑO A TRAVÉS DEL METAMODELO UML.* Río Cuarto : s.n., 2005.

41. Chapter 1: What is Software Architecture? *Chapter 1: What is Software Architecture?* [En línea] [Citado el: 26 de 03 de 2011.] http://msdn.microsoft.com/en-us/library/ee658098.aspx.

42. **Jamir Antonio Avila Mojica.** *Introducción a patrones de software.*

43. Patrones Arquitectonicos. *Patrones Arquitectonicos.* [En línea] [Citado el: 27 de 03 de 2011.] http://www.buenastareas.com/ensayos/Patrones-Arquitectonicos/1069013.html.

44. ARQUITECTURAS DE SOFTWARE. *ARQUITECTURAS DE SOFTWARE.* [En línea] 04 de 2004. [Citado el: 01 de 04 de 2011.] http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.

45. Arquitectura Cliente/Servidor. *Arquitectura Cliente/Servidor.* [En línea] 27 de 09 de 2009. [Citado el: 27 de 03 de 2011.] http://ccia.ei.uvigo.es/docencia/SCS/1011/transparencias/Tema1.pdf.

46. El patrón MVC. *El patrón MVC.* [En línea] [Citado el: 28 de 03 de 2011.] http://www.librosweb.es/symfony/capitulo2/el_patron_mvc.html.

47. usuarios.multimania. *usuarios.multimania.* [En línea] [Citado el: 7 de Abril de 2011.] http://usuarios.multimania.es/manualesjava/manuales/convencionescodigojava/convencionescodigojava.pdf.

BIBLIOGRAFÍA

Adolfo Lozano Tello. MÉTRICA DE IDONEIDAD DE ONTOLOGÍAS.

Adolfo Lozano Tello. Ontologías en la Web Semántica.

Àngels Rius;Miguel Ángel Sicilia;Elena García. Justificación y Descripción del Dominio de Conocimiento de una Ontología para la Formalización y Automatización de Escenarios Educativos.

Arquitectura Cliente/Servidor. *Arquitectura Cliente/Servidor.* [En línea] 27 de 09 de 2009. [Citado el: 27 de 03 de 2011.] http://ccia.ei.uvigo.es/docencia/SCS/1011/transparencias/Tema1.pdf.

ARQUITECTURAS DE SOFTWARE. *ARQUITECTURAS DE SOFTWARE.* [En línea] 04 de 2004. [Citado el: 01 de 04 de 2011.] http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.

Berners-Lee T., Hendler J. and Lassila O. *The Semantic Web.* 2001.

Castells, Pablo. La Web Semántica. *La Web Semántica.*

Chapter 1: What is Software Architecture? *Chapter 1: What is Software Architecture?* [En línea] [Citado el: 26 de 03 de 2011.] http://msdn.microsoft.com/en-us/library/ee658098.aspx.

Definición.de. *Definición.de.* [En línea] [Citado el: 27 de 2 de 2011.] http://definicion.de/java/ .

Definición de Explícito.

Definición de Semántica.

Eclipse - The Eclipse Foundation open source community website. *Eclipse - The Eclipse Foundation open source community website.* [En línea] http://www.eclipse.org/.

El patrón MVC. *El patrón MVC.* [En línea] [Citado el: 28 de 03 de 2011.] http://www.librosweb.es/symfony/capitulo2/el_patron_mvc.html.

Entorno Virtual de Aprendizaje. Introduccion_a_la_Disciplina_de_Requisitos.pdf.

Gerardo Fernández Escribano. *Introducción a Extreme Programming.* 2002.

Graciela Elisa BARCHINI; Margarita María ÁLVAREZ; Diana PALLIOTTO; Susana HERRERA; Paola BUDÁN. *ONTOLOGÍAS EN LOS SISTEMAS DE INFORMACIÓN / CONOCIMIENTO.*

Graciela Barchini, Margarita Álvarez, Susana Herrera y Melina Trejo. *EL ROL DE LAS ONTOLOGÍAS EN LOS SI.*

Graciela Barchini;Margarita Álvarez;Susana Herrera. *SISTEMAS DE INFORMACIÓN: NUEVOS ESCENARIOS BASADOS EN ONTOLOGÍAS INFORMATION SYSTEMS: NEW ONTOLOGY-BASED SCENARIOS.* 2006.

Bibliografía

GRUBER, T. R. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* 1995.

Gruber, T. R. What is an Ontology? *What is an Ontology?*

GUARINO, N. *Formal Ontology, Conceptual Analysis and Kwoledge Representation.* 1995.

Guillermo A. Cuéllar N. N. unicauca.edu.co. *N. unicauca.edu.co.*

IEEE Standard for Software Quality Management System. 1992.

Ignacio Herrero Jiménez. *DESARROLLO DE ONTOLOGÍAS SOBRE BBDD RELACIONALES:ONTOLOGÍA QUE SOPORTA LA BÚSQUEDA AVANZADA DE INFORMACIÓN*.

Jamir Antonio Avila Mojica. *Introducción a patrones de software.*

jena. *jena.* [En línea] http://jena.sourceforge.net/.

LENGUAJES DE PROGRAMACIÓN. *LENGUAJES DE PROGRAMACIÓN.*

Lissete González Gallo; Reynaldo Alvarez Luna. *Solución Informática para el Centro Nacional de Balance Alimentario: Implementacion de un componente de software para la transferencia de datos a través de Internet.*

Marco Villalvi. Ontología Medio Ambiente. *Ontología Medio Ambiente.* [En línea] [Citado el: 28 de 2 de 2011.] http://ontoguate.wordpress.com/2009/01/25/ontologia-medio-ambiente/.

Nicolas Guarino. Formal Ontology and Information Systems.

Nicolas Guarino. *Understanding, Building, and Using Ontologies.* 1996.

Ontología y Web Semántica. *Ontología y Web Semántica.*

Patrones Arquitectonicos. *Patrones Arquitectonicos.* [En línea] [Citado el: 27 de 03 de 2011.] http://www.buenastareas.com/ensayos/Patrones-Arquitectonicos/1069013.html.

Pressman Roger S. *Ingeniería del Software. Un enfoque práctico*.

Pressman. *Aspectos y metricas de la calidad del Software.* 1998.

Prof. Ivette C. Martínez. *DISEÑO CON PATRONES.* Universidad Simón Bolívar. Ing. de Software : s.n.

schermbeek, Mariangela Pocaterra Van. APACHE. *APACHE.* [En línea] [Citado el: 28 de 2 de 2011.] http://ddd.uab.cat/pub/trerecpro/2009/hdl_2072_41829/Treball+de+recerca.pdf.

sesame. *sesame.* [En línea] [Citado el: 28 de 2 de 2011.] http://www.openrdf.org/doc/sesame/users/.

Sheth, A.P. Changing focus on interoperability in information systems: from system, syntax, structure to.

Sitio Web de la Oficina Española del W3C.

Sobre PostgreSQL | www.postgresql-es.org. *Sobre PostgreSQL | www.postgresql-es.org.* [En línea] [Citado el: 28 de 2 de 2011.] http://www.postgresql-es.org/sobre_postgresql.

Sowa J. F. *Ontology, Metadata, and Semiotics.* Darmstadt.

The Dojo Toolkit. *The Dojo Toolkit.* [En línea] [Citado el: 5 de 4 de 2011.] http://dojotoolkit.org/features/.

usuarios.multimania. *usuarios.multimania.* [En línea] [Citado el: 7 de Abril de 2011.] http://usuarios.multimania.es/manualesjava/manuales/convencionescodigojava/convencionescodigojava.pdf.

VAN HEIJST, G., SCHEREIBER, A.T. Y WIELINGA, B.J. *Using Explicit Ontologies in KBS Development.* 1996.

Vanina Barotto;Mauricio Demonte. *DEFINICIÓN DE PATRONES DE DISEÑO A TRAVÉS DEL METAMODELO UML.* Río Cuarto : s.n., 2005.

Wand, Y. and Weber, R. *An Ontological Model of an Information System.* 1990.

Welcome to NetBeans. *Welcome to NetBeans.* [En línea] [Citado el: 28 de 2 de 2011.] http://netbeans.org/.

GLOSARIO DE TÉRMINOS

IDE: Es un entorno de desarrollo integrado, programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

JSP (Java Server Pages): Es una tecnología orientada a crear páginas web con programación en Java.

Licencia Pública General de GNU: más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Metodología de Desarrollo: Marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

OWL: Es un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW.

RDF: Es un lenguaje simple para expresar modelos de los datos, que refieren a los objetos “recursos” y a sus relaciones.

Software: Equipamiento lógico o soporte lógico de un ordenador. Comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Suele ser utilizado para referirse a los programas y aplicaciones informáticas.

URL (Uniform Resource Locator): Es una secuencia de caracteres de acuerdo con un formato estándar para nombrar determinados recursos de red, para su localización o identificación.

URI: Son cadenas que permiten acceder a cualquier recurso de la Web. En la Web Semántica las *URIs* son las encargadas de identificar objetos.

XML (Extensible Markup Language): Es un metalenguaje extensible de etiquetas.