METODOLOGIA PARA DESARROLLO DE SOFTWARE BASADA EN ONTOLOGIA

Ing. Adriana Quero Bastidas

Tutor: Isabel Besembel Carrera

COMO REQUISITO PARA OBTENER
EL GRADO DE
MAGISTER SCIENTIAE EN COMPUTACIÓN
DE LA
UNIVERSIDAD DE LOS ANDES
MÉRIDA, VENEZUELA
NOVIEMBRE 2007

Resumen

El desarrollo de software generalmente se realiza en diferentes contextos, puntos de vista y suposiciones acerca de un área de estudio determinada. Es por esta razón que con frecuencia se crean problemas de comunicación entre las partes involucradas, reduciéndose así el potencial de reutilizar y compartir información. Una manera de resolver esto, es crear un entendimiento compartido a través del uso de las ontologías, las cuales permiten establecer correspondencia y relaciones entre diferentes dominios. El objetivo fundamental de esta investigación es definir una ontología basada en el conocimiento del área de la Ingeniería de Software, que contenga un conjunto de procedimientos, técnicas y ayudas de documentación, y que pueda ser utilizada como apoyo a nuevas metodologías o metodologías existentes de desarrollo de productos de software. Se presenta el proceso de creación de una ontología de la Ingeniería de Software, cuyo objetivo primordial es compartir y organizar todos los conocimientos

acumulados hasta ahora en esta área, además de servir de inicio a investigaciones relacionadas con la interpretación automática de estos conceptos, usando sistemas de software o agentes de software inteligentes. Se inicio la investigación y análisis de la guía SWEBOK. Posterior a este análisis, se realizó la abstracción de estos conceptos en un diagrama de clases UML, el cual fue transformado en una ontología representada en el lenguaje OWL (Ontology Web Language) utilizando la herramienta Protégé. Finalmente, se realizó una aplicación Web basada en esta ontología que soporta los conceptos estudiados; para demostrar la utilidad en la Ingeniería de Software, se planteó un caso de estudio ficticio en el que un Ingeniero de esta área se hace interrogantes acerca de diferentes tópicos relacionados con el ciclo de desarrollo de software las cuales son respondidas de manera exitosa por el sistema, logrando que el Ingeniero pueda obtener la información requerida en cada etapa del ciclo de desarrollo.

A **Dios Todopoderoso** por colocar en mi camino a las personas adecuadas que me impulsaron a culminar con éxito este sueño.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1. Introducción.

El desarrollo de los sistemas de software generalmente se realiza en diferentes contextos, puntos de vista y suposiciones acerca de un área de estudio determinada. Es por esta razón que con frecuencia se crean problemas de comunicación por falta de entendimiento entre las partes involucradas, lo cual trae como consecuencia falta de interoperabilidad, reduciéndose de esta forma el potencial de reutilizar y compartir información. Los nuevos sistemas de información deben apuntar hacia entender el modelo de usuarios en cualquier parte del mundo y su significado, además de comprender los modelos procedentes de distintas fuentes de información. Una manera de resolver esto es crear un entendimiento compartido a través del uso de las ontologías, las cuales permiten establecer correspondencia y relaciones entre los diferentes dominios de entidades de información.

La Ontología es una antigua disciplina que se define como un esquema específico de categorías que refleja una visión específica del mundo. Desde el punto de vista informático, las ontologías especifican un vocabulario relativo a un cierto dominio. Este vocabulario define entidades, clases, propiedades, predicados y funciones, además de las relaciones entre estos componentes. Las ontologías se encargan de definir los términos utilizados para describir y representar un área de conocimiento, son utilizadas por los usuarios, las bases de datos y las aplicaciones que necesitan compartir información específica, es decir, en un campo determinado, como puede ser el de las finanzas, medicina, deporte, etc. Además juegan un papel clave en la resolución de la interoperabilidad semántica entre sistemas de información y su uso.

1.1 Antecedentes.

Actualmente existe cierta cantidad de trabajos que involucran el uso de técnicas ontológicas con las etapas del proceso de desarrollo de software. Se seleccionaron los artículos más resaltantes y se presenta a continuación un resumen de los mismos.

1.1.1 Uso de Ontologías Formales en los Sistemas de Información

Nicola Guarino en [2] expone de forma concisa la manera como se ha venido incrementando el uso de las ontologías en los sistemas de información (ingeniería de conocimientos, diseño e integración de bases de datos, extracción y recuperación de información). Se presenta un enfoque del uso de las ontologías utilizando sus peculiaridades metodológicas y arquitectónicas. Desde el punto de vista metodológico, su principal peculiaridad es la adopción de un enfoque altamente interdisciplinario donde la filosofía y la lingüística tienen un rol fundamental en el análisis a un alto nivel de la estructura de una realidad dada y la formulación de un riguroso vocabulario. Desde el punto de vista arquitectónico, el aspecto más importante se basa en el rol que puede jugar la ontología en los sistemas de información, a través de la liderización de una nueva perspectiva del desarrollo de sistemas, sistemas de información guiados por ontologías (ontology-driven information systems).

Presenta, también, terminologías básicas fundamentales para el estudio de las ontologías, como lo son: nociones de las ontologías, conceptualización y compromisos ontológicos; además de mostrar los diferentes tipos de ontologías que existen y su contribución e importancia en la integración de la información.

Como aspecto resaltante se concluye con una explicación de cómo las ontologías pueden ser utilizadas en diferentes etapas del desarrollo de los sistemas de información, a saber:

- Uso de la ontología en tiempo de desarrollo.
- Uso de la ontología en tiempo de ejecución.
- Uso de la ontología para los componentes de bases de datos.
- Impacto de las ontologías en los componentes de los sistemas de información
- Uso de la ontología para componentes de interfaces de usuario.

1.1.2 Metodologías para el procesamiento de los requerimientos y de los componentes reutilizables basados en técnicas ontológicas.

Motoshi Saeki en [3] presenta un proyecto de investigación llevado a cabo en la Universidad de Tokio, donde se busca dar soporte al manejo de requerimientos en ingeniería de software, a la reutilización de arquitecturas, frameworks y componentes de software basados en ontologías. Según el autor, el software en la actualidad es desarrollado adaptando y combinando arquitecturas, componentes, frameworks y paquetes de software. Es por esta razón que los ingenieros de software, en particular los que trabajan con manejo de requerimientos y diseños de aplicaciones, deben ser capaces de seleccionar y adaptar las distintas partes que conforman un software de acuerdo a los requerimientos de sus clientes.

La metodología presentada se basa en el procesamiento semántico de los requerimientos y de los elementos reutilizables basados en las técnicas ontológicas. Consiste principalmente en un sistema de ontologías basado en un diccionario de términos perteneciente a un dominio específico; para esta metodología los requerimientos son levantados en base a palabras relevantes que se encuentran en este sistema de ontologías. En cuanto a los elementos de software, se cuenta con una base de datos de elementos reutilizables incluyendo su información semántica. La selección de los elementos reutilizables se lleva a cabo haciendo una correspondencia entre la palabra suministrada y el sistema de ontologías. La figura 1-1 presenta gráficamente la manera como los requerimientos son procesados utilizando las técnicas ontológicas.

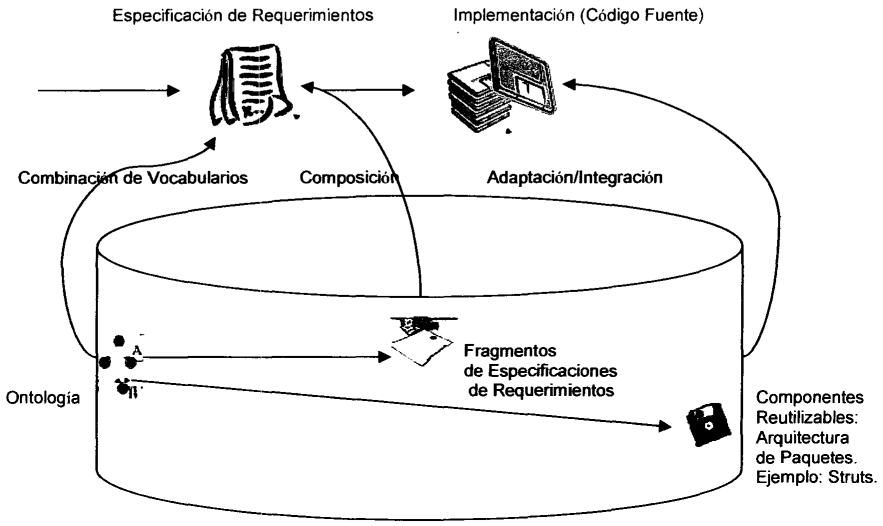

Figura 1-1. Sistema de Ontologías

1.1.3 Desarrollo de herramienta de integración de los procesos de software basado en el enfoque ontológico ODE (Ontology Software based software Development Enviroment).

Ricardo de Almeida en [4] expone la necesidad de la elaboración de una infraestructura basada en modelos conceptuales, que permita la integración de todas las herramientas involucradas en el proceso de desarrollo de software. Se utilizan las ontologías para lograr un entendimiento común y como formas básicas de comunicación y de representación de la información.

Expone además los papeles que pueden representar las ontologías en el desarrollo de los procesos de software, tales como: gestión de requisitos, calidad de software, análisis de riesgos, etc. Sus autores desarrollaron un sistema basado en el enfoque ontológico, que permite integrar estas diferentes etapas del desarrollo de software cuyo nombre es ODE (Ontology Software based software Development Environment). Este sistema, a través del uso de una ontología que se diseñó para tal fin, permite soportar el proceso de la definición de software, seguimiento e integración de un software.

1.2 Conceptos Relacionados

El uso de las ontologías en los procesos de desarrollo de software debe estar relacionado de alguna u otra forma con la definición de una metodología de desarrollo que se adapte a este tipo de técnicas y que aproveche todos sus beneficios. Se presentan a continuación algunos conceptos básicos relacionados con el tema.

1.2.1 Metodologías de Software:

Conjunto de procedimientos, técnicas, herramientas y soporte documental que apoyan el proceso de desarrollo de software, permitiendo producir de manera económica y organizada software de alta calidad. Puede seguir uno o varios modelos de ciclo de vida, los cuales indican qué es lo que hay que obtener a lo largo del desarrollo de proyecto pero no cómo hacerlo.

1.2.2 Clasificación de los modelos de software:

Existe una gran diversidad de metodologías definidas para el desarrollo de software, se realizó una selección de las más utilizadas en la actualidad, para hacer una comparación entre las mismas. Se presenta a continuación una definición breve de cada una de ellas, para luego mostrar una tabla comparativa (Tabla1) donde se resumen las ventajas y desventajas más relevantes de las mismas.

- **Modelo en cascada:** se basa en un enfoque sistemático secuencial, el cual se inicia con la especificación de requerimientos por parte del cliente y continúa con la planificación, el modelado, la construcción y el despliegue, para culminar en el soporte y mantenimiento del software terminado. Este modelo es útil en situaciones donde los requerimientos no cambian y donde el trabajo se realiza hasta su conclusión, de manera lineal [5].

- **Modelo de procesos incrementales:** combina elementos del modelo cascada aplicado en forma iterativa, este modelo aplica secuencias lineales de manera escalonada conforme avanza el tiempo en el calendario. Generalmente, al utilizar este modelo, el primer "incremento" es un producto esencial, al cual es sometido a una evaluación por parte del cliente, de los resultados de esta evaluación se genera un plan para el próximo incremento. Dicho plan contiene las modificaciones del producto esencial con el fin de satisfacer las necesidades del cliente y la entrega de funcionalidades adicionales. Este proceso se repite después de la entrega de cada incremento hasta que se haya elaborado el producto completo. Este modelo se concentra en la entrega de un producto operacional por cada incremento. Este modelo de desarrollo es útil en ocasiones cuando el personal necesario para una implementación completa no está disponible. Los primeros incrementos se pueden implementar con menos gente. Si el producto esencial es bien recibido, se incorpora, en caso de ser necesario, más personal para continuar con el desarrollo de los incrementos sucesivos [5].

- **Modelo DRA (Desarrollo rápido de aplicaciones):** es un modelo de software incremental que tiene como característica principal un ciclo de desarrollo corto. Con el uso de este modelo se logra un desarrollo rápido mediante un enfoque basado en componentes reutilizables. Si los requisitos están bien entendidos y se limita el alcance del proyecto, el proceso DRA permite que un equipo de desarrollo cree un sistema completamente funcional dentro de un periodo de tiempo muy corto. Este modelo cumple con las etapas comunes de los antes descritos: La comunicación o

levantamiento de requisitos, la planificación, el modelado, la construcción y el despliegue. En la etapa de construcción, se destaca el uso de componentes y la aplicación de la generación automática de código. Es útil para aplicaciones cuyas funcionalidades puedan ser divididas en módulos desarrollables en menos de tres meses, de esta manera cada función se puede abordar mediante un equipo de DRA por separado, para después hacer una integración y formar el sistema final [5].

- **Modelo de procesos evolutivos:** se refieren a modelos iterativos cuya característica principal es que permiten a los ingenieros de software el desarrollo de varias versiones cada vez más completas del software.
  - **Construcción de prototipos:** este modelo es empleado comúnmente como una técnica dentro del contexto de los modelos presentados anteriormente. Este modelo ayuda al ingeniero y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos. El prototipo debería servir como un mecanismo para identificar los requerimientos del software y puede servir como primer sistema.

  - **El modelo en espiral:** es un modelo de software evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos sistemáticos y controlados del modelo en cascada. En este caso, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, las entregas se pueden basar en documentación del modelo o en un prototipo, mientras que durante las últimas iteraciones se generan versiones más acabadas del sistema desarrollado. El modelo en espiral mejora el modelo en cascada enfatizando la naturaleza iterativa del proceso de diseño.

  - **El modelo de desarrollo concurrente:** este modelo se representa en forma esquemática como una serie de actividades, acciones y tareas de la ingeniería de software y sus estados asociados. Todas las actividades existen

de forma concurrente, pero se encuentran en diferentes estados. Este modelo define una serie de eventos que dispararán transiciones de estado a estado para cada una de las actividades, acciones o tareas de la ingeniería de software. Este modelo es útil para todos los tipos de desarrollo de software y proporciona una visión exacta del estado actual de un proyecto [5].

- **Modelos especializados de proceso:**
  - **Desarrollo basado en componentes:** incorpora muchas de las características del modelo en espiral. Es evolutivo por naturaleza y exige un enfoque iterativo para el desarrollo de software. Sin embargo, el modelo configura aplicaciones a partir de componentes de software empaquetados en forma previa. Para iniciar las etapas de modelado y construcción se deben identificar los componentes candidatos. Estos componentes se pueden diseñar como módulos de software convencionales o como clases o paquetes de clases orientados a objetos.
  - **El modelo de métodos formales:** comprende un conjunto de actividades que conducen a la especificación matemática del software de computadora. Estos métodos permiten al ingeniero de software especificar, desarrollar y verificar un sistema basado en computadoras al aplicar una notación matemática rigurosa. Cuando estos modelos se usan durante el diseño sirven como base para la verificación de programas y, por consiguiente, permiten descubrir y corregir errores que de otra manera podrían no haberse detectado[5].

- **Metodologías Ágiles:** representan una alternativa a los procesos de desarrollo de software tradicionales caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas; están constituidas por un conjunto de principios que permiten a los equipos desarrollar software rápidamente, respondiendo a los cambios que puedan surgir a lo largo del proyecto. Existe una organización llamada *"The Agile Alliance"* cuyo objetivo es

promover los conceptos relacionados con el desarrollo ágil de software y con la implementación de los mismos en las empresas, para ello crearon la llamada filosofía "ágil" la cual se resume en doce principios :

- ✓ *La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.*
- ✓ *Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.*
- ✓ *Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.*
- ✓ *La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.*
- ✓ *Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.*
- ✓ *El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.*
- ✓ *El software que funciona es la medida principal de progreso.*
- ✓ *Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.*
- ✓ *La atención continua a la calidad técnica y al buen diseño mejora la agilidad.*
- ✓ *La simplicidad es esencial.*
- ✓ *Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.*
- ✓ *En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.*

En cuanto a la aplicabilidad de estas metodologías, las organizaciones deben identificar el área de negocio donde estas son aplicables, evaluando el grado de

dificultad que tendrá su implantación y la permanencia de estas prácticas, sus efectos colaterales (tanto positivos como negativos) y lo más importante, la satisfacción de sus empleados en cuanto a la adopción de este tipo de métodos. Entre las metodologías ágiles se encuentran: SCRUM, Crystal Methodologies, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Feature-Driven Development (FDD), Lean Development (LD) y Extreme Programming (XP) [6].

- **RUP (Rational Unified Proccess) :** es un proceso de desarrollo de software que junto con el lenguaje UML (Unified Modeling Language) constituye una de las metodologías estándar más utilizadas para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. RUP divide en 4 fases el proceso de desarrollo de software:
  - **Inicio**: abarca la comunicación con el cliente y las actividades de planificación, en esta etapa se propone una arquitectura aproximada para el sistema y se desarrolla un plan inicial. A través de la definición de casos de uso son descritos los requerimientos fundamentales del negocio.
  - **Elaboración**: en esta etapa se refinan los casos de uso y se expande la representación arquitectónica para incluir cinco visiones diferentes del software: el modelo de casos de uso, el modelo de análisis, el modelo de diseño, el modelo de implementación y el modelo de despliegue.
  - **Construcción**: en esta etapa el objetivo es llegar a obtener la capacidad operacional inicial. Se utiliza el modelo arquitectónico como entrada, esta fase desarrolla o adquiere los componentes de software que harán que cada caso de uso sea operativo para los usuarios finales. Se realizan además actividades de integración (ensamblaje de componentes y pruebas de integración). Los casos de uso son utilizados para generar un conjunto de pruebas de aceptación que deben ser ejecutados antes de la siguiente fase.

- **Transición**: su objetivo es la obtención de la versión funcional del proyecto. Abarca las últimas etapas de construcción del proyecto y la primera parte de la actividad de despliegue. En esta etapa el software es entregado al usuario el cual tiene la posibilidad de reportar tantos defectos como cambios sean necesarios. Además el equipo de proyecto crea la documentación relacionada con el soporte del proyecto para el lanzamiento final del mismo [5].

- **MSF (Microsoft Solution Framework):** Es un método flexible e interrelacionado con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo, dejando en un segundo plano las elecciones tecnológicas. Las fases que se deben cumplir en un proyecto según esta metodología son:

  - **Estrategia y Alcance**: esta fase incluye la elaboración del documento de alcance y estrategia, la formación del equipo de trabajo y distribución de competencias, la elaboración del plan de trabajo y la elaboración de la matriz de riesgos y plan de contingencia.
  - **Planificación y Prueba de Concepto**: en esta fase deben alcanzarse los siguientes objetivos: elaboración del documento de planificación y arquitectura, el cual contiene el detalle de los aspectos funcionales y operativos, el documento del plan de laboratorio, que contiene la descripción de la prueba de concepto, los diversos escenarios a simular, el control de incidencias y las matrices de calidad.
  - **Estabilización**: en esta etapa la solución real previamente implantada se pasa a un entorno real restringido por número de usuarios y en condiciones tales, que se pueda llevar un control efectivo. Los objetivos fundamentales de esta fase son: selección del entorno de pruebas pilotos, gestión de incidencias, revisión de la documentación final de arquitectura, elaboración

de la documentación de formación y operaciones, elaboración del plan de despliegue y del plan de formación.

- **Despliegue**: en esta fase se ejecutan los planes relacionados con despliegue y formación elaborados en la etapa anterior. Los principales objetivos de esta fase son: implantación de la plataforma, puesta en servicio de todas las funcionalidades del sistema, formación de usuarios y administradores, continuación de las labores de recepción de incidencias, registros de mejoras y sugerencias, control de cambios, entrega de los documentos definitivos, revisión de la matriz de riesgos y matrices de calidad, entrega del proyecto y cierre del mismo [5].

Tabla 1-1. Tabla comparativa de los modelos de procesos.

Modelo	Ventajas	Desventajas	Iterativa
Cascada	Es ideal en situaciones donde los requerimientos están fijos y donde el trabajo se realiza, hasta su conclusión de manera lineal, por ejemplo en el caso de mejoras o adaptaciones bien definidas a un sistema ya existente.	Puede llegar a producir estados de bloqueo en los cuales algunos miembros del equipo de proyecto deben esperar a otros para terminar tareas independientes.	No
Procesos Incrementales	Puede ser utilizado cuando todo el personal necesario para una aplicación completa no está disponible. Mediante las entregas parciales del producto permite validar progresivamente la	Algunas empresas no se ajustan a este tipo de modelos por su complejidad de administración.	Si

	aplicación y así lograr un producto mejor adaptado a los requerimientos proporcionados por el cliente.		
DRA	Puede ser utilizada para aplicaciones cuyos módulos puedan completarse en menos de tres meses. Puede ahorrarse dinero, tiempo y esfuerzo humano. Se concentra en los elementos esenciales del sistema, desde el punto de vista del usuario.	Mayor velocidad y menores costos pueden tener influencia en la calidad del sistema. Pueden producirse inconsistencias entre diseños internos y entre sistemas. Posibles violaciones de los estándares de programación. Peligrosa incoherencia entre el sistema desarrollado y el negocio, debido a la falta de información o a procesos del negocio sobreentendidos.	Si
Evolutivos	Se realizan varias entregas evolutivas desde documentación hasta productos de software permitiendo así llevar un mejor control del desarrollo del proyecto.	En ocasiones en este tipo de modelos se presentan problemas de planificación del proyecto debido al número incierto de ciclos requeridos para construir el producto final. Adicionalmente este tipo de modelo se	Si

		enfoca en la flexibilidad y extensibilidad por encima de la alta calidad, prioriza la velocidad del desarrollo sobre los cero defectos.	
Basados en Componentes	Puede resultar costosa en algunos casos. Poco acoplamiento entre los componentes desarrollados y los necesarios para el desarrollo de una aplicación determinada. Limitación en cuanto a los servicios para almacenar, localizar, distribuir y recuperar componentes.	Reutilización de Software. Simplifica las pruebas. Simplifica el mantenimiento del sistema. Mayor Calidad. Ciclos de Desarrollo más cortos.	Si
Formales	Es utilizado para sistemas que deben estar libres de errores y en sistemas de alta seguridad.	Es muy costoso y consume mucho tiempo. Adicionalmente requiere una capacitación detallada y no es un mecanismo de comunicación efectivo con clientes que no tienen muchos conocimientos técnicos.	Si
Ágiles	No es conveniente para	Sencillez en cuanto su	Si

	ser utilizados en proyectos donde el entorno del sistema es muy cambiante. Generalmente, usado sólo para proyectos pequeños. Generalmente, los equipos tradicionales de trabajo presentan resistencia al cambio.	aprendizaje y aplicación. Entrega rápida de un software que satisfaga al cliente. Está basada en la comunicación y colaboración entre los miembros del equipo y entre los profesionales y sus clientes.	
RUP	Está guiado por la representación de los requisitos mediante el uso de UML (Unified Model Language) el cual es una excelente guía para los equipos de software en el desarrollo de una tecnología.	En ocasiones estos modelos tienen un nivel de complejidad que requiere mucha experiencia administrativa por parte del gerente de proyecto y su equipo de trabajo para manejarlo adecuadamente.	Si
MSF	Debido al hecho de que este modelo requiere un equipo organizado, código estructurado y procesos sistemáticos, los riesgos se minimizan y se maximiza la capacidad de toma decisiones inteligentes.	Requiere una curva de aprendizaje de parte de los equipos de trabajo para que pueda ser aplicada de manera efectiva. Mal interpretada puede convertirse en un obstáculo en los procesos de desarrollo.	Si

El propósito de la tabla comparativa anterior es identificar y documentar los diferentes modelos de software que existen, con el objetivo de comprender las ventajas y desventajas

de cada una y así poder entender como la construcción de una ontología podría apoyar modelos con este tipo de características.

1.3 Justificación

Existen muy buenas razones para usar las técnicas ontológicas en el proceso de desarrollo de software. En la actualidad, tanto las empresas como las personas, se enfrentan continuamente a una gran cantidad de información, por lo cual el acceso a la información relevante se ha convertido en un factor crítico, es por esto que se hace necesario el uso de nuevas técnicas que permitan el manejo eficiente de la misma.

Las ontologías facilitan la comunicación entre sistemas informáticos, ya que mejoran la calidad de las respuestas por medio de la incorporación de una conceptualización del dominio basada en seres humanos. Entre los aspectos más resaltantes relacionados con el desarrollo de sistemas de información, en los cuales pueden ser utilizadas las técnicas ontológicas procesables automáticamente, se tienen:

- Para mejorar la calidad del proceso software compartiendo la experiencia en proyectos de desarrollo.
- Para garantizar la consistencia entre todos los artefactos de un proyecto de software (casos de uso, diagramas de clase, código fuente, etc.) y facilitar la navegación a través de todos ellos.
- Para facilitar el mantenimiento y la reutilización del software.
- Para realizar los "modelos de dominio" del proceso de ingeniería de dominio en las metodologías de Ingeniería del Software basada en la reutilización de componentes.
- Para el diseño de interfaces de usuario de calidad.
- Para establecer un método sistemático de generación automática de objetos, patrones y frameworks (p.ej. en Java) a partir de la ontología de dominio.
- Para facilitar la generación y validación de modelos UML en proyectos de software.

- Para facilitar la transición de la Ingeniería del Conocimiento a la Ingeniería del Software en proyectos de desarrollo de sistemas software basados en conocimiento.
- Como forma de representar meta-modelos en la Ingeniería del Software basada en Modelos.

1.4 Objetivos

Esta investigación tiene como objetivo general la creación de una ontología basada en la guía de conocimiento SWEBOK, la cual contenga información acerca de un conjunto de procedimientos, técnicas y ayudas de documentación para el desarrollo de productos de software. Se pretende que la ontología resultante cubra por completo el ciclo de desarrollo de software y que pueda ser utilizada para mejorar la productividad del desarrollo de sistemas y mejorar la confianza de los clientes finales en los mismos. Adicionalmente, se espera demostrar la aplicabilidad de esta ontología a través de la creación de un sistema que incluya todas las etapas de un ciclo de desarrollo y que pueda ser consultado a través de un navegador Web por los Ingenieros de Software o personal relacionado con esta área.

Como objetivos específicos basados en la utilización de las ontologías como base para compartir conocimiento de una forma más sencilla y confiable, se consideran los siguientes:

- Mejorar la calidad del proceso software a través del uso de una fuente de conocimiento común para el uso de los Ingenieros de Software.
- Convertirse en el punto de partida para el desarrollo de aplicaciones de usuario que contribuyan a la toma de decisiones en la definición y ejecución del proceso software.
- Mejorar la integración del conocimiento en Entornos de Ingeniería del Software.
- Convertirse en punto de partida para impulsar la investigación en esta área en nuestra industria de software.

1.5 Plan de Trabajo

Para el desarrollo de este trabajo las actividades propuestas a seguir son las siguientes:

1. Como primera etapa se realizará la recopilación de la información relacionada a las ontologías.
2. Realizar un estudio de las técnicas ontológicas.
3. Investigación del uso de estas técnicas en el proceso de desarrollo de software.
4. Realizar una identificación de las diferentes áreas relacionadas con el proceso de desarrollo de software.
5. Definir para cada una de las áreas identificadas la manera como serán utilizadas las técnicas ontológicas en cada una de ellas.
6. Definir los procedimientos y técnicas relacionados con cada una de las etapas del proceso de desarrollo de software.
7. Aplicación de la metodología planteada a un proyecto de software de un dominio seleccionado.

1.6 Alcance

El alcance de este proyecto se fundamenta en la definición de una metodología basada en técnicas ontológicas. Se puede plantear como una extensión del mismo la creación de un software que permita soportar la metodología creada y apoyar a cada una de las etapas del desarrollo de software.

1.7 Organización de la Tesis

Se presenta a continuación un resumen de los siguientes capítulos y secciones en los que está dividido este trabajo:

Capítulo 2, Conceptos Teóricos: En este capítulo se presenta en profundidad la definición de Ontologías, sus características, tipos y lenguajes utilizados para su representación. Además se presenta la relación existente entre las Ontologías y el desarrollo de procesos de software.

Capítulo 3, Construcción de una Ontología para la Ingeniería de Software: En este capítulo se presenta, inicialmente los conceptos relacionados con la creación de las ontologías y los lenguajes que soportan su creación. Posteriormente se muestra el análisis realizado sobre los conceptos existentes en la guía ***SWEBOK*** (*Guía para la Ingeniería de Software*), obteniéndose como resultado una representación de los mismos en un diagrama de clases UML, el cual representa el conocimiento contenido en esta guía para finalmente llevar este diagrama a una ontología representada en el lenguaje ***OWL*** (*Ontology Web Language*). Esta ontología será utilizada para la creación de un sistema que servirá de apoyo a consultas relacionadas con el área de conocimiento de la Ingeniería de Software.

Capítulo 4, Uso de la aplicación como apoyo a un proyecto de un dominio específico: En este capítulo, se presenta la manera como la ontología creada en el capítulo 3 puede ser utilizada para apoyar a una metodología en específico, esto se logra mediante la selección de un proyecto en un dominio determinado, para el cual se plantean interrogantes relacionadas con cada una de las etapas al desarrollo del software, demostrando como la aplicación desarrollada da respuesta a cada una de ellas.

Capítulo 5, Conclusiones y Recomendaciones: Este último capítulo se refiere a las conclusiones obtenidas en cada una de las partes del manuscrito y las posibles recomendaciones que se tuvieron al respecto.

Capítulo 2. Conceptos teóricos.

El objetivo fundamental de este capítulo es presentar de manera detallada los conceptos más importantes relacionados con las ontologías y su relación con la Ingeniera de Software. El capítulo comienza con una descripción del significado de las ontologías, su importancia, utilidad, relación con otras áreas del conocimiento como: Web Semántica, Software Enginering Enviroments (SEE), Ingeniería de Dominios (Domain Engineering), Inteligencia Artificial, tipos, características fundamentales, metodologías para su definición; adicionalmente se describen los lenguajes que se utilizan para definirlas, y algunas herramientas existentes para la definición de las mismas, se introduce además el concepto de Ingeniería Ontológica, los criterios de diseño y las diferentes etapas que deben ser tomadas en cuenta para el desarrollo de las ontologías; finalmente, se presenta la relación que existe en la actualidad entre las ontologías y los procesos de desarrollo de software y como pueden las mismas contribuir con mejoras sustanciales en el ciclo de vida del desarrollo de un proyecto.

2.1 Definición de ontología.

Este concepto tiene sus orígenes en la Filosofía, cuyo significado es el de una ***explicación sistemática de la existencia,*** proviene del griego, en el que significa "***estudio del ser***". No hay una definición aceptada unánimemente para el concepto de ontología, sino que cada una de las definiciones que aparecen en la literatura aporta diferentes y a la vez complementarias visiones del significado de esta palabra. Filósofos e Ingenieros de Software tienen puntos de vistas diferentes en cuanto a la definición de las ontologías. Para

Guarino [2], ontología describe una cierta realidad con un vocabulario específico, por otro lado *Gruber* [10] define una ontología como una especificación que proporciona una estructura y contenidos de forma explícita. Apartándonos del aspecto filosófico, podemos definir a una ontología como un consenso necesario sobre cómo expresar y entender información de un determinado dominio [11]. Mas profundamente una ontología o también llamada base de conocimiento (siglas en inglés KB) es una especificación explícita de una o parte de una conceptualización que incluye vocabulario de términos y la especificación del significado de cada uno de ellos, define un vocabulario común para los que necesitan compartir información acerca de un dominio; haciendo una analogía al desarrollo de sistemas, es lo que una especificación es a un programa. Entendiéndose conceptualización como una interpretación estructurada de una parte del mundo que usan los seres humanos para pensar y comunicarse sobre ella. En el área de la informática, una conceptualización podría ser, por ejemplo, la clasificación de sistemas informáticos atendiendo su naturaleza física en sistemas hardware y software.

Desde el punto de vista informático, las ontologías especifican un vocabulario relativo a un cierto dominio. Este vocabulario define: entidades, clases, propiedades, predicados y funciones y, la relación entre estos componentes.

Podemos tomar como ejemplo el área de la ingeniería de diseño, supongamos que un grupo de ingenieros desea definir un vocabulario de entendimiento común en cuanto a los componentes eléctricos, para lograr su objetivo se define una ontología del dominio de los componentes eléctricos, asumiendo que se tiene un vocabulario donde se discuten elementos conceptuales como: transistores, amplificadores operacionales, voltajes y las relaciones existentes entre estos componentes, los amplificadores operacionales son un tipo de componentes eléctricos y los transistores están compuestos de amplificadores operacionales. Identificar este tipo de vocabulario y su conceptualización generalmente requiere un análisis cuidadoso, que se explicará más adelante en este capítulo, del tipo de objeto y las relaciones que pueden existir en un determinado dominio.[1]

2.1.1 Importancia de las ontologías.

La importancia de las ontologías se puede resumir en dos aspectos fundamentales:

- El análisis ontológico clarifica la estructura del conocimiento. Dado un dominio, su ontología constituye el corazón de cualquier representación de un sistema de conocimiento para ese dominio. Sin ontologías o sin las respectivas conceptualizaciones que soportan el conocimiento no puede existir un vocabulario para representar el conocimiento. Así el primer paso para obtener un efectivo sistema de representación del conocimiento y su vocabulario es llevar a cabo un efectivo análisis ontológico del campo o dominio de estudio, un análisis pobre conlleva a bases de conocimiento incoherentes.
- Las ontologías permiten compartir conocimiento. Supongamos que se realiza un análisis y se obtiene un conjunto satisfactorio de conceptualizaciones y la representación de sus términos para un área en específica, por ejemplo el dominio de los componentes electrónicos, la ontología resultante debe incluir términos específicos del dominio como: transistores y diodos; términos generales como funciones y modos; y términos que describen el comportamiento como, voltajes. Para construir un lenguaje de representación del conocimiento basado en el análisis previo, se necesita asociar los términos con los conceptos y las relaciones, y definir una sintaxis de codificación del conocimiento en término de los mismos. Es posible compartir un lenguaje de representación de conocimiento con otros que tengan similares necesidades para el mismo dominio, de tal modo que se elimine la necesidad de replicar el proceso de análisis del conocimiento. Compartir ontologías puede de este modo significar la base de la definición de un lenguaje de representación del conocimiento [11].

2.1.2 Utilidad de las ontologías.

Se mencionan a continuación el uso que, en la actualidad, tienen las ontologías:

- Sirven para entender como diferentes sistemas comparten información.
- Se utilizan para descubrir distorsiones que puedan presentarse en los procesos cognitivos de aprendizaje en un mismo contexto.
- Sirven para formar patrones para el desarrollo de Sistemas de Información. En el ámbito del software se viene utilizando hace algunos años para describir las propiedades del software (componentes, arquitecturas, lenguajes de definición), este punto se verá con más detalle en la última parte de este capítulo.
- Permiten compartir y reutilizar conocimiento común.
- Ayudan a establecer comunicación entre personas y organizaciones con el fin de unificar diferentes áreas de investigación
- Permiten la interoperatividad entre sistemas de software usando ontologías como un lenguaje intermedio para unificar diferentes lenguajes y herramientas.
- Aumentan los beneficios de la Ingeniería de Sistemas ya que el uso de ontologías facilita la construcción de software clásico o basado en el conocimiento porque permite que los sistemas se puedan reutilizar [11].

2.1.3 Relación con otras áreas del conocimiento.

Las ontologías se relacionan y son aplicables en la actualidad a diversas áreas del conocimiento, en las cuales contribuyen a la realización de una actividad más efectiva, se seleccionaron algunas áreas consideradas importantes para describir la forma en que las ontologías trabajan en conjunto con las mismas:

- **Web Semántica:** la Web Semántica es una Web extendida, dotada de mayor significado en la cual los usuarios de Internet pueden encontrar respuestas a

sus preguntas de manera más rápida y sencilla gracias a una información bien definida. Esto permite compartir, procesar y trasmitir información de forma sencilla. Cuando se habla de que es una Web dotada de significado, quiere decir, de más semántica, lo cual le permite procesar sus contenidos, razonar con este, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente. Es una infraestructura basada en metadatos los cuales permiten razonar en la Web extendiendo de esta forma su capacidad, proporcionando a las máquinas la capacidad de resolver problemas bien definidos, a través de operaciones bien definidas que se llevan a cabo sobre datos bien definidos. La manera como se relaciona este nuevo enfoque de la Web con las ontologías es que estas últimas son utilizadas como estándar de la Web Semántica para obtener una adecuada definición de los datos a través del lenguaje OWL (Ontology Web Language), el cual permite definir ontologías estructuradas que pueden ser utilizadas a través de diferentes sistemas, usuarios, bases de datos y aplicaciones que necesitan compartir información específica, es decir, en un campo determinado como por ejemplo, el de las finanzas, la medicina, deportes, etc; en la sección 4 de este capítulo se expondrá una explicación más detallada de este lenguaje. [12]

- **Ingeniería de Dominio (Domain Engineering):** un dominio es un área funcional diferenciable con requisitos y rasgos similares (features) que puede ser soportada por algún tipo de sistema de software. La Ingeniería de Dominios se encarga de capturar, organizar y representar la información útil para el desarrollo de sistemas de software, de manera que esta pueda ser reutilizada para crear nuevos sistemas [13]. La idea principal es lograr desarrollar un conjunto de productos de software que comparten información y tienen aspectos comunes entre si, pudiendo de esta forma ser reutilizados por los ingenieros de software. Su propósito es identificar, modelar, construir y catalogar un conjunto de productos de software que pueden ser aplicados a existentes o futuros sistemas en un dominio de

aplicaciones particular. Un proceso de Ingeniería de Dominios debe contemplar por lo menos tres actividades principales: análisis de dominio, especificación de la infraestructura e implementación de la infraestructura. En la Ingeniería de Dominio, las ontologías pueden ejecutar distintos roles, en esta área el interés principal se basa en el uso de las ontologías como una especificación, en la cual la ontología de un dominio proporcione un vocabulario para requerimientos específicos de una o mas aplicaciones.[14]

- **Software Engineering Environments (SEEs):** representan colecciones integradas de herramientas que facilitan las actividades de la ingeniería de software a través de todo su ciclo de vida. En este contexto las ontologías son utilizadas para mejorar el proceso de integración de los SEEs, ya que permiten conceptualizar un dominio a través de la definición de clases de objetos y sus relaciones. El enfoque que se sigue para el caso de los SEEs se centra principalmente en que las herramientas que se desean integrar estén basadas en ontologías, esto con el objetivo de facilitar este proceso [15].
- **Gestión del Conocimiento (Knowledge Management):** es un concepto utilizado para la transmisión de conocimiento y experiencia entre los empleados dentro de una organización. Usualmente este proceso requiere capturar, organizar y almacenar el conocimiento para transformarlo en un activo intelectual que se comparte. La actividad de la gestión de conocimiento es amplia y compleja, esta puede ser manejada como un conocimiento individual o como un conocimiento empresarial. Para lograr estos objetivos, las ontologías son consideradas una metodología adecuada para soportar la variedad de actividades de la gestión del conocimiento, como la recuperación del conocimiento, su almacenamiento y distribución. Las ontologías pueden ser vistas como la clasificación de conocimiento, definiendo el vocabulario que facilita la comunicación del conocimiento, almacenamiento, búsqueda y distribución de los sistemas basados en gestión del conocimiento. La definición de una ontología es una tarea laboriosa que consume gran cantidad de tiempo. En general, la aplicación e identificación

de una ontología es sólo para dominios específicos, por ejemplo: medicina, turismo, la industria de metal, etc. [16]

- **Inteligencia Artificial (IA):** es una de las áreas de la ciencia de la computación encargadas de que la creación de hardware y software tenga comportamientos inteligentes. Una de las principales áreas de estudio de la Inteligencia Artificial está representada por los Agentes Inteligentes, los cuales son entidades de software con una arquitectura robusta que puede funcionar en distintos entornos o plataformas de computación y que son capaces de realizar de forma "inteligente" y autónoma distintos objetivos, intercambiando información con el entorno, o con otros agentes humanos o computacionales. En esta área, las definiciones de las ontologías juegan un papel importante ya que permiten establecer un consenso de comunicación sobre cómo se expresa y entiende la información de un determinado dominio en el cual se desenvuelven los agentes de software. [17]

2.1.4 Formas de utilizar las ontologías en las aplicaciones

Las aplicaciones de las ontologías pueden ser clasificadas en 4 categorías principales. Se debe tener presente que una aplicación puede incluir una o más de estas categorías:

- **Autor Neutral** (Neutral Authoring): una ontología es desarrollada como un simple lenguaje y es trasladada a diferentes formatos y utilizada en aplicaciones de múltiples usos.
- **Ontología como una especificación:** se crea una ontología de un dominio dado, la cual tiene la capacidad de proporcionar un vocabulario para la especificación de requisitos de una o más aplicaciones. En este caso, una ontología puede ser vista como un modelo del dominio. La ontología es utilizada como una base para la especificación y desarrollo de aplicaciones en el dominio seleccionado, permitiendo así la reutilización de conocimiento.

- **Para acceder a información:** una ontología es utilizada para permitir a una aplicación de múltiples usos (o a los humanos) tener acceso a diversas fuentes heterogéneas de información, la cual normalmente es expresada usando un vocabulario diverso o formatos inaccesibles.
- **Búsquedas basadas en ontologías:** una ontología es utilizada en la búsqueda de recursos en un repositorio de información, mejorando la precisión y reduciendo la cantidad de tiempo invertido en realizar la búsqueda. [14]

2.2 Elementos que componen las ontologías

Los elementos de los que se compone una ontología son:

- **Conceptos**: cualquier entidad sobre la que se puede decir algo, a la que se le puede poner un nombre. Además de esto también la descripción de una tarea, función, acción, estrategia, etc. Por ejemplo: coche, hombre, árbol.
- **Relaciones**: representan una interacción entre conceptos del dominio. Por ejemplo: subclase-de, parte-de, etc.
- **Funciones**: son un caso especial de las relaciones, en el que el elemento n-ésimo de la relación es único para los n-1 restantes. Por ejemplo: madre-de, precio-de, etc.
- **Axiomas**: son sentencias siempre verdaderas para dicho dominio. Por ejemplo: "*el lunes va después del domingo*".
- **Instancias**: son conceptos concretos. Por ejemplo: David, Olmo.

No siempre todos estos elementos deben aparecer en una ontología. En el caso de que se tenga una ontología que posee solo conceptos y relaciones se llama taxonomía u ontología ligera [17].

2.3 Tipos de ontologías

Las ontologías se pueden clasificar teniendo en cuenta varios criterios, se toman en cuenta dos de ellos:

a) El alcance de su aplicabilidad

- **Ontologías de Dominio:** Proporciona el vocabulario necesario para describir un dominio dado. Incluye términos relacionados con:

  .- Los objetos de dominio y sus componentes (quirófano, estetoscopio).

  .- Un conjunto de verbos o frases que dan nombre a actividades y procesos que tienen lugar en ese dominio (anestesiar, dar a luz).

  .- Conceptos primitivos que aparecen en teorías, relaciones y formular que regulan o rigen el dominio (después de nacer el mejor valor para el Test del bebe es el 10).

- **Ontologías de Tareas:** Proporcionan el vocabulario para describir términos involucrados en los procesos de resolución de problemas los cuales pueden estar relacionados con tareas similares en el mismo dominio o en dominios distintos. Incluye nombres, verbos, frases y adjetivos relacionados con la tarea (objetivo, planificación, asignar, clasificar).
- **Ontologías Generales:** Representa los datos generales que no son específicos de un dominio. Por ejemplo, ontologías sobre el tiempo, el espacio, ontologías de conducta, de capacidad, etc.

b) Cantidad y tipo de conceptualización

- **Terminológicas:** Especifican los términos que son usados para representar conocimiento en el universo de discurso. Son utilizadas para unificar vocabulario en un dominio determinado.

- **De información:** Especifican la estructura de almacenamiento de bases de datos. Ofrece un marco para el almacenamiento estandarizado de información (estructura de registros de la BD).
- **De modelado del conocimiento:** Especifican conceptualizaciones del conocimiento. Poseen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen (términos y semántica) [11].

2.4 Lenguajes utilizados para la definición de Ontologías

Los primeros lenguajes para la representación de las ontologías estaban basados en representación del conocimiento, entre ellos se tienen:

1.- **CycL**: lenguaje utilizado en la ontología Cyca basado en marcos y en lógica de primer orden.

2.- **Ontolingua**: lenguaje construido sobre KIF (Knowledge Interchange Format-sintaxis formal utilizada para expresar conocimiento) y basado en marcos y en lógica de primer orden.

3.- **LOOM**: lenguaje basado en lógica de descripciones y reglas de producción.

4.- **OCML**: sistema de representación de conocimiento para el desarrollo de ontologías Ontolingua y métodos de resolución de problemas (PSMs –Problem Solving Methods) ejecutables.

5.- **OKBC** (Open Knowledge Base Connectivity): protocolo de acceso a bases de conocimiento construidas con diferentes sistemas de RC.

Con el auge de Internet surgen lenguajes de ontologías para la web con sintaxis basada en HTML o XML:

1.- **SHOE** (Simple HTML Ontology Extension): lenguaje basado en marcos y reglas diseñado como extensión del HTML.

2.- **RDFa**: modelo para la descripción de recursos Web basado en redes semánticas y con sintaxis XML.

3.- **RDF Schema**: lenguaje construido sobre RDF que incorpora primitivas de marcos (clases y propiedades).

Los lenguajes presentados anteriormente sirven tanto para la descripción de ontologías como para la publicación de contenidos web procesables por máquinas, de ahí que se les llame también lenguajes para el marcado ontológico.

Los lenguajes de ontologías más recientes son extensiones de RDF(S):

1.- **DAML+OIL**: fusión de DAML (DARPA Agent Markup Language) y OIL (Ontology Inference Layer) que amplía RDF(S) con primitivas de lógica de descripciones.

2.- **OWL (Web Ontology Language):** propuesta del W3C a partir de DAML+OIL para responder a las necesidades de la web semántica. Es un mecanismo para desarrollar temas o vocabularios específicos en los que asociar esos recursos. Lo que hace OWL es proporcionar un lenguaje para definir ontologías estructuradas que pueden ser utilizadas a través de diferentes sistemas. [8]. Incluye tres niveles: OWL Lite, OWL DL y OWL Full. Para los efectos de esta investigación y para soportar la continuidad de la misma, fue seleccionado este lenguaje para el desarrollo de la Ontología, ya que está diseñado para ser usado en aplicaciones que

necesitan procesar el contenido de la información en lugar de únicamente representar la información para los humanos. OWL facilita un mejor mecanismo de interpretabilidad del contenido Web que los mecanismos admitidos por XML, RDF y esquema RDF (RDF-S) proporcionando vocabulario adicional junto con una semántica formal [36].

A la hora de elegir un lenguaje para la definición de una Ontología deben tenerse en cuenta los siguientes aspectos:

- El lenguaje debe poseer una sintaxis bien definida para poder leer con facilidad la ontología definida.
- Debe tener una semántica bien definida para comprender perfectamente el funcionamiento de la ontología.
- Debe tener suficiente expresividad para poder capturar varias ontologías.
- Debe ser fácilmente mapeable desde/hacia otros lenguajes ontológicos.
- Debe ser eficiente a la hora de realizar razonamiento.

2.5 Herramientas para definir ontologías:

Entre las herramientas más destacadas utilizadas en la actualidad para la construcción de las ontologías se encuentran:

- **Protégé 2000**: herramienta a través de la cual el usuario puede construir ontologías de dominio, generar usuarios de entrada de datos y efectuar la propia entrada de datos. Es una herramienta que permite acceso a aplicaciones externas basadas en conocimiento. Además es una biblioteca a la que otras aplicaciones pueden acceder, permitiéndoles acceder a las bases

de conocimiento de las cuales se dispone. Está disponible en: http://protege.stanford.edu/.

- **Ontology Server**: herramienta que permite al usuario la construcción de ontologias que comparten grupos geográficamente distribuidos. Fue desarrollado en el laboratorio de Sistemas de Conocimiento en la Universidad de Stanford. Este servidor es una extensión de Ontolingua. Al comienzo el término Ontolingua se usaba para referirse tanto al lenguaje para representar ontologías como a la herramienta utilizada para construirlas. Hoy, el término se utiliza para referirse al lenguaje proporcionado por el Ontology Server .Su arquitectura permite el acceso a una librería de ontologías, traductores de lenguajes y a un editor para crear y navegar por una ontología. Su servidor se encuentra disponible en la URL: http://www-ksl-svc.stanford.edu:5915
- **OntoEdit**: es un ambiente para soportar las actividades involucradas en los procesos relacionados con la Ingeniería Ontológica. Está basado en un poderoso modelo de ontologías, el cual puede ser serializado utilizando XML (Extensible Markup Language). Se encuentra disponible es la siguiente URL: http://www.ontoknowledge.org/tools/ontoedit.shtml
- **OilEd**: es un editor de ontologías gratuito que permite al usuario construir ontologias utilizando el lenguaje DAM+OIL. Aunque no posee todas las funcionalidades de otros editores de ontologías, proporciona suficiente funcionalidad para permitir a los usuarios construir ontologías. Se encuentra disponible en la siguiente dirección URL: http://oiled.man.ac.uk

2.6 Ingeniería Ontológica y Modelado de Ontologías

- **Definición de la Ingeniería Ontológica**: la Ingeniería Ontológica se refiere al proceso de construir y mantener una ontología. A través de ella se pretende elaborar representaciones conceptuales explícitas dentro de un determinado dominio y tiene su campo de aplicación en el estudio de todo tipo de problemas que no están claramente estructurados, como, el procesado del lenguaje natural, recuperación inteligente de información, organización de bases de datos, modelado y simulación de sistemas, etc.
- **Criterios de Diseño**: cuando se pretende representar algo en una ontología, se deben realizar decisiones de diseño. Como consecuencia de esto se propone un conjunto preeliminar de criterios para el diseño de las ontologías, cuyo propósito es compartir conocimiento y lograr así la interoperabilidad entre programas basados en una conceptualización compartida[10]:

    - **Claridad y objetividad**: la ontología debe proporcionar al usuario el significado de los términos de forma efectiva y en lenguaje natural para facilitar su compresión.
    - **Completitud**: las definiciones han de expresarse en términos necesarios y suficientes.
    - **Coherencia**: debe permitir hacer inferencias que sean consistentes con las definiciones.
    - **Máxima extensibilidad monótona**: una ontología debe ser diseñada para anticipar el uso de un vocabulario compartido. Las especializaciones o generalizaciones deben poder incluirse en la ontología sin necesidad de revisar las definiciones ya existentes.
    - **Diversificación**: se deben poder diversificar las jerarquías incluidas para aumentar la potencia de los mecanismos de herencia múltiple.

- **Estandarización**: se debe tratar de utilizar un vocabulario lo más universal posible. [11]
- **Minimización de la distancia semántica:** conceptos similares deben estar agrupados y representados utilizando las mismas primitivas.
- **Mínimo compromiso ontológico:** debe existir al menos una cantidad posible de suposiciones acerca del mundo modelado.

- **Proceso de desarrollo de las ontologías:** el proceso de desarrollo de las ontologías incluye las tareas que se deben llevar a cabo para la construcción de las mismas. Existe un estándar establecido por la IEEE en 1991 para el desarrollo de software en general llamado *"Standard for Developing Software Life Cycle Proccesses"* , las cuales adaptadas al proceso de desarrollo de las ontologías dividen este proceso de desarrollo es tres etapas en general:
  - Actividades ligadas al manejo del proyecto: Planificación, control del seguimiento de la planificación y asegurar la calidad del producto.
  - Actividades orientadas al desarrollo de la ontología: especificación, conceptualización, formalización, implementación y mantenimiento.
  - Actividades integrales: adquisición de conocimiento, integración con otras ontologías, evaluación y documentación.

Como un enfoque general basado en un enfoque iterativo, se pueden ver las diferentes etapas que deben seguirse para el desarrollo de las ontologías, en la figura 2-1 se observan las diferentes etapas de este proceso [20]:

Figura 2-1. Etapas del proceso de desarrollo de ontologías.

- **Determinar el ámbito**: en esta etapa se debe definir, cuál es el dominio que va a cubrir la ontología, para qué va a ser utilizada, y cuáles son los tipos de preguntas que la ontología va a responder.
- **Considerar la reutilización**: es siempre de utilidad considerar lo que otros han realizado en cuanto a una ontología determinada y chequear si se puede redefinir y extender las fuentes existentes para nuestro dominio y tarea particular. Esta reutilización permite, ahorrar esfuerzos e interactuar con otras herramientas que utilizan ontologías ya definidas y validadas. Muchas ontologías están disponibles en formato electrónico y pueden ser importadas en un ambiente de desarrollo de ontologías. Existen librerías de ontologías reutilizables en la literatura y en la Web, por ejemplo: Ontolingua (http://www.ksl.stanford.edu/software/ontolingua/), DAML (http://www.daml.org/ontologies/). Existe también un determinado número de ontologías comerciales públicas, por ejemplo: UNSCSP (United Nations Standard Products and Services Code, http://www.unspsc.org/), RossetaNet http://www.rosettanet.org.
- **Enumerar los términos**: en esta etapa debe especificarse, cuales son los términos de los cuales se va a hablar, cuales son las propiedades de estos términos, que se quiere expresar acerca de los mismos. Es de utilidad en esta etapa escribir toda la lista de términos bien sea para hacer oraciones sobre ellas o para explicárselas al usuario.
- **Definir las clases**: se deben definir las clases que son una representación del dominio, representa una colección de elementos con propiedades

similares. En esta definición se debe tomar en consideración los siguientes aspectos importantes:

- Manejar el concepto de herencia que pueda existir en las relaciones entre las clases,
- Descripción de las clases en lenguaje natural, debe existir una documentación detallada.
- Listar las suposiciones relevantes que se hayan hecho acerca del dominio.
- Listar los sinónimos.

Adicionalmente existen posibles enfoques en los cuales se puede basar la definición de la jerarquía de las clases:

- **Top-down**: comienza con la definición de los conceptos más generales en el dominio y continúa con la subsiguiente especialización de los conceptos.
- **Botton-Up**: comienza con la definición de las clases más específicas y continúa con la inclusión de las mismas en conceptos más generales.
- **Combinación de los dos anteriores**: representa una combinación de ambas técnicas, **top-down** y **botton-up** .

- **Definir las propiedades**: las clases por si mismas no proporcionan suficiente información, es por esto que se hace necesario describir la estructura interna de los conceptos y asociarlos a las clases previamente definidas. Las propiedades de las clases son conocidos como *slots* , las cuales describen los atributos de las instancias de una clases y las relaciones con otras instancias. En esta etapa deben definirse dichas propiedades con sus tipos; con respecto a las propiedades y las clases debe tomarse en cuenta lo siguiente:
  - Una subclase hereda todas las propiedades de la superclase.

- Si una clase tiene múltiple superclases, ésta hereda las propiedades de cada una de ellas.

- **Definir los restricciones asociadas a cada una de las propiedades**: estas restricciones describen el límite del conjunto de valores para una propiedad determinada; tipo de los valores, valores permitidos, la cardinalidad y otras características que los valores puedan tomar.
- **Crear las instancias**: se deben definir las instancias de cada una de las clases establecidas, las clases se convierten en un tipo directo de una instancia, las propiedades deben ser asignadas a estas instancias tomando en cuenta las restricciones previamente determinadas. Definir una instancia de una clase requiere: (1) seleccionar la clase, (2) Crear una instancia individual de esa clase, (3) Completar los valores de las propiedades asociadas a esas clases. [18] [19].

- **Metodologías existentes para el modelado de ontologías:** las metodologías para la construcción de las ontologías se dividen en dos grupos. Por un lado metodologías para construir ontologías desde cero. Por otro lado, metodologías para construir ontologías a través de un proceso de reingeniería.

  - **Metodologías para construir ontologías a partir de cero:**

    - ✓ **Metodología Cyc:** la metodología Cyc (Lenan, 1990), define una ontología como la extracción del conocimiento común que está implícito en distintas fuentes; recomienda seguir los siguientes pasos:
      - Codificación manual del conocimiento implícito y explícito de diferentes fuentes.
      - Codificación del conocimiento utilizando herramientas de software.
      - Delegación de la mayor parte de la codificación en las herramientas [20].

- ✓ **Metodología de construcción de ontologías de Uschold y King:** los pasos propuestos en esta metodología son:
  - o Identificar el propósito.
  - o Capturar la ontología.
  - o Codificación.
  - o Identificar ontologías existentes.
  - o Evaluación.
  - o Documentación [20].
- ✓ **Metodología de construcción de ontologías de Gruninger y Fox:** para esta metodología el primer paso es identificar las aplicaciones posible en las que se utilizará la ontología, usando posteriormente un conjunto de preguntas, llamadas preguntas de competencia, para determinar el ámbito de la ontología; sugiere los siguientes pasos:
  - o Identificar los escenarios motivantes.
  - o Preguntas informales de competencia.
  - o Terminología formal.
  - o Preguntas formales de competencia.
  - o Axiomas formales
  - o Teoremas de computación [20].

- ✓ **Metodología Kactus:** para esta metodología la ontología se construye sobre una base de conocimiento por medio de un proceso de abstracción; los pasos propuestos son:
  - o Especificación de la aplicación.
  - o Diseño preeliminar basado en categorías ontológicas.
  - o Refinamiento y estructuración de la ontología [20].

- ✓ **Methontology:** es una metodología para construir ontologías tanto partiendo desde cero como a través de un proceso de reingeniería; se proponen los siguientes pasos:
  - Especificación
  - Conceptualización.
  - Formalización.
  - Implementación.
  - Mantenimiento [20].

- ✓ **Metodología On-To-Knowledge:** este proyecto aplica ontologías a la información disponible electrónicamente para mejorar la calidad de la gestión del conocimiento en organizaciones grandes y distribuidas, La metodología proporciona vías para introducir conceptos y herramientas de gestión de conocimiento en empresas, logrando presentarlo así de manera más eficiente y efectiva. Esta metodología está basada en el análisis de escenarios de uso, los pasos que recomienda son los siguientes:
  - Estudio de la viabilidad.
  - Comienzo.
  - Refinamiento.
  - Evaluación [20].

- **Metodologías para construir ontologías a través de un proceso de reingeniería:** la reingeniería ontológica es el proceso de recuperar y mapear un modelo conceptual de una ontología implementada en otro modelo más adecuado, el cual sería reimplementado, existe un método basado en el esquema de la reingeniería de Chikosky que contempla 3 actividades principales:
  - Ingeniera Inversa
  - Reestructuración
  - Ingeniería hacia delante [20].

2.7 Relación de las ontologías con la Ingeniería de Software

Las ontologías desempeñan un papel en el desarrollo de sistemas relacionado con el modelado semántico, metadatos, análisis y diseño de patrones y creación de librerías de software reutilizable.

Se mencionan, a continuación, un conjunto de aspectos que representan los vínculos entre la Ingeniería de Software y la representación del conocimiento mediante Ontologías:

- Pueden utilizarse como mecanismos para facilitar el desarrollo de proyectos de software.
- Se pueden crear ontologías para conceptualizar el área de conocimiento de la Ingeniería de Software.
- Modelado y meta-modelado de sistemas.
- Comunicación entre participantes del proceso.
- Para mejorar la calidad del proceso de software compartiendo la experiencia en proyectos de desarrollo.
- Para crear sistemas de ayuda a la toma de decisiones en la definición y ejecución del proceso de software.
- Para garantizar la consistencia entre todos los artefactos de un proyecto de software (casos de uso, diagramas de clase, código fuente, etc) y facilitar la navegación a través de todos ellos.
- Para mejorar la integración del conocimiento en entornos de Ingeniería de Software compuestos por varias herramientas CASE.
- Para realizar los modelos de dominio del proceso de ingeniería de dominio en las metodologías de software basada en componentes.
- Para el diseño de interfaces de usuario de calidad.
- Para establecer un método sistemático de generación automática de objetos, patrones, y frameworks a partir de la ontología de dominio.

- Para facilitar la generación y validación de modelos UML en proyectos de software.
- Para mejorar el repositorio de una herramienta CASE mediante un motor ontológico.
- Para facilitar la transición de la Ingeniería del Conocimiento a la Ingeniería del Software en proyectos de desarrollo de sistemas software basados en conocimiento.
- Como forma de representar meta-modelos en la Ingeniería de Software basada en modelos.
- Se pueden definir ontologías sobre "Calidad del software", "Mantenimiento de Software", "Proceso de Software", "Componentes de Software", "Diseño de Interfaces de usuario"
- Mantenimiento y reutilización de software.
- Educación a distancia sobre Ingeniería de Software.
- Localización de recursos en la Web Semántica.

Existen diversas ontologías ya definidas sobre "Ingeniería de Software":

- Según el IEEE *Std Glossary of Software Engineering Terminology.*
- Según el informe SWEBOK 2004. *Software Engineering Body of Knowledge.*
- Según el informe ACM/IEEE SE 2004. *Curriculum Guidelines for undergraduate Degree Programs in Software Engineering.*

2.8 Áreas de Aplicación

Las áreas de aplicación de las ontologías son muy amplias y pueden utilizarse en áreas como:

- Ingeniera del conocimiento
- Representación de conocimiento
- Procesamiento del lenguaje natural
- Sistemas de información cooperativos
- Integración inteligente de información
- Recuperación de Información
- Gestión del conocimiento
- Comercio electrónico
- Ingeniería de Software [11].

Capítulo 3. Construcción de una Ontología para la guía de conocimiento SWEBOK.

Tal como se mencionó en capítulos anteriores, la utilidad fundamental de una ontología radica en el hecho de clasificar conceptos y permitir compartir conocimiento en un área de un dominio específico. Se presenta a continuación el proceso de creación de una ontología relacionada con la Ingeniería de Software; cuyo objetivo primordial es compartir y organizar los conocimientos recopilados en esta guía, así como también servir de base para el inicio de investigaciones relacionadas con la interpretación automática de estos conceptos, usando sistemas de software o agentes de software inteligentes. Como punto de partida se llevo a cabo la investigación y análisis de la guía para la Ingeniería de Software – *SWEBOK*, por ser un proyecto que posee una reputación internacional en esta área de conocimiento, ya que surge de una revisión realizada por más de 500 expertos en este dominio relacionados con distintos ámbitos tales como: académico, agencias de gobierno, sociedades profesionales, organizaciones de estándares internacionales así como diferentes centros de investigación; una vez hecho este análisis se realizó la abstracción de estos conceptos en una diagrama de clases *UML* el cual fue transformado finalmente a la ontología de dominio en lenguaje *OWL* (*Ontology Web Language)* utilizando la herramienta para creación de ontologías, *Protégé.* [22]

Finalmente, al obtener la ontología deseada, se desarrolló una aplicación prototipo en el lenguaje de programación Java que permite la extracción de los conceptos presentes vía Web, el objetivo fundamental será utilizar esta herramienta como apoyo conceptual en los diferentes procesos y etapas definidos para llevar a cabo el proceso de desarrollo de software.

3.1 Conceptos relacionados

Para comprender mejor los temas tratados en este capítulo es importante tener en cuenta los siguientes conceptos:

- **XML (Extensible Markup Language):** es un lenguaje de marcas utilizado para estructurar información. Existen 5 caracteres especiales utilizados en XML: los símbolos menos que, <, mayor que, >, las comillas dobles, ", el apostrofe, ', y el carácter, &. Los símbolos mayor y menor son utilizados para delimitar las marcas que dan estructura al documento.
- **XMI (XML Metadata Interchange):** es una propuesta de la utilización de XML que intenta proporcionar a los programadores una forma estándar de intercambiar información sobre metadata.
- **DTD (Document Type Definition):** son archivos utilizados para definir las posibles marcas y los atributos que puede tener un documento XML. Un documento XML indica al comienzo el DTD utilizado mediante una marca <!DOCTYPE>.
- **XML Namespaces:** proporcionan un método para evitar los conflictos de nombres en un documento XML. Permite particionar el conjunto de todos los nombres posibles, de forma que se pueda definir a que zona de ese espacio corresponde una etiqueta.
- **URI (Uniform Resource Identifiers):** identifica recursos asignándole una dirección en una red dada. Son cadenas que funcionan como identificadores

globales que hacen referencias a recursos en la Web, tales como documento, imágenes, archivos descargables, servicios, buzones de correo electrónico y otros.

- **RDF (Resource Description Framework):** es un formato universal para representar datos en la Web, que permite intercambiar información entre distintas aplicaciones sin que esos datos pierdan significado, lo que facilita la reutilización de los recursos en la Web.
- **OWL (Ontology Web Language):** lenguaje estándar desarrollado por el consorcio W3C para el desarrollo de ontologías, el cual permite que los conceptos sean definidos tal y como son descritos.
- **SWEBOK (Software Engineering Body of Knowledge):** guía utilizada para proporcionar una validación de los limites de la disciplina de la Ingeniería de Software y para proporcionar acceso a los diferentes tópicos relacionados con este dominio del conocimiento.
- **UML (Unified Modeling Language):** lenguaje conformado por un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, describiendo la semántica esencial de lo que estos diagramas y símbolos significan.
- **SOAP (Simple Access Object Protocol):** es un protocolo utilizado para intercambiar estructuras de información en un ambiente descentralizado y distribuido. Es un protocolo basado en XML que está compuesto por tres componentes: un sobre (envelope) que define un marco de trabajo para describir qué es un mensaje y como procesarlo, un conjunto de reglas para expresar las instancias de una aplicación (definición de tipos de datos), y una convención para representar llamadas y respuestas provenientes de procedimientos remotos.
- **Protégé 3.2:** ambiente de desarrollo para la creación y edición de ontologías. Este ambiente fue seleccionado, debido a que presenta un conjunto de características que lo distingue de otros editores basados en conocimiento; tales como: interfaz gráfica de usuario fácil e intuitiva, herramienta escalable debido a que utiliza un sistema de cache para la liberación de memoria lo cual proporciona

un mayor rendimiento, arquitectura extensible a través de la instalación de plug-ins [37].

- **WSDL (Web Service Definition Languaje):** es un formato XML para describir servicios Web. Se encarga de definir una interfaz pública para los servicios Web. El programa cliente que se conecta a un servicio Web lee en WSDL para determinar que funciones están disponibles en el servidor.
- **Servicio Web:** es una colección de protocolos y estándares que se utilizan para realizar el intercambio de datos entre las aplicaciones.

3.2 OWL (Ontology Web Language)

Está compuesto por tres sublenguajes: OWL Lite, OWL DL y OWL Full.

3.2.1 Sublenguajes OWL:

- **OWL Lite:** es utilizado principalmente en aquellos casos de una clasificación jerárquica y restricciones simples. Su principal objetivo es proporcionar un lenguaje que sea lo suficientemente sencillo y entendible para ser visualizado por herramientas de desarrollo. Pretende capturar muchas de las características comúnmente utilizadas de OWL y DAML-OIL, adicionalmente permite describir un lenguaje útil que proporciona funciones adicionales con el objetivo de soportar aplicaciones Web.

- **OWL DL:** lenguaje diseñado para aquellos usuarios que buscan la máxima expresividad sin arriesgar el desempeño computacional. OWL DL incluye todos los constructores del lenguaje OWL con restricciones tales como el tipo de separación (un *clase* no puede ser al mismo tiempo un *individual* o una *propiedad,* igualmente una *propiedad* no puede ser a su vez un *individual* o una *clase*).

- **OWL Full:** lenguaje diseñado para aquellos usuarios que buscan la máxima expresividad combinada a su vez con la libertad sintáctica del lenguaje RDF. Por ejemplo: en el OWL Full una clase puede ser tratada simultáneamente como una colección de *individuals* y como un *individual* simple. Este lenguaje permite que una ontología incremente el significado del vocabulario predefinido en RDF u OWL.

Cada uno de estos sublenguajes es una extensión de su predecesor. El siguiente conjunto de relaciones se cumple, su inverso no:

- Una ontología legal construida en OWL Lite es legal en OWL DL.
- Una ontología legal construida en OWL DL es legal en OWL Full.
- Una conclusión legal construida en OWL Lite es legal en OWL DL.
- Una conclusión legal construida en OWL DL es legal en OWL Full.

Los desarrolladores de ontologías deben considerar cual de estos lenguajes se adapta mejor a sus necesidades. Debido a que la ontología creada será consumida por un sistema Web y que se necesita un buen desempeño de la aplicación, el tipo de OWL utilizado es el OWL-DL, sin embargo es importante destacar que el tipo de OWL puede ser cambiado utilizando la herramienta Protégé en caso de que la ontología sea modificada para futuras versiones de la aplicación.

3.3 Diagrama de clases para la representación de los conceptos relacionados con la guía SWEBOK.

Para obtener el diagrama de clases se realizó la lectura y posterior análisis de la guía *SWEBOK*, fundamentalmente se revisó la forma como están organizados los conceptos dentro de la misma, para luego representar esta estructura en un diagrama de clases *UML*.

Los objetivos principales de la guía *SWEBOK* son:

- Clasificar los conceptos y contenidos relacionados con el dominio de la Ingeniería de Software.
- Proporcionar diferentes tópicos relacionados con la Ingeniería de Software.
- Promover una manera consistente de entender a la Ingeniería de Software.
- Establecer el ámbito y el conjunto de límites de la Ingeniería de Software con respecto a otras disciplinas como: Ciencias de la Computación, Gerencia de Proyectos, Ingeniería de la Computación y Matemáticas.
- Proporcionar las bases para el desarrollo de planes de carrera y material para certificaciones en esta área.

El proyecto *SWEBOK* permite establecer un consenso en:

- Las áreas de conocimiento (*KA*) que integran los conceptos de la Ingeniería de Software.
- Los conceptos relacionados con cada dominio, así como las principales referencias relacionadas a ellas.

- Las diferentes disciplinas científicas involucradas con cada una de las áreas de conocimiento.

El proyecto *SWEBOK* está compuesto por 10 áreas de conocimiento las cuales se muestran en la figura 3-1.

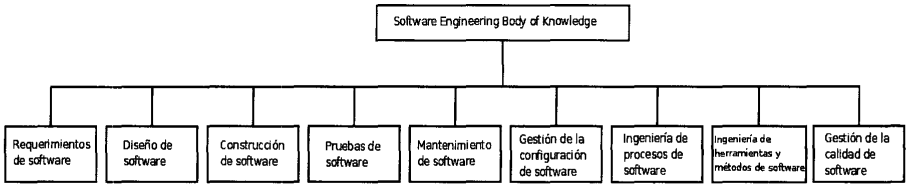

Figura3-1. Áreas de Conocimiento (KA) presentes en el proyecto SWEBOK.

En la guía *SWEBOK* la descripción de los contenidos presentes en cada una de las áreas de conocimiento es definida siguiendo una especificación preestablecida; inicialmente se presenta una introducción definiendo el área de conocimiento luego se muestran los tópicos – en la figura 3-2 se puede observar la subdivisón de tópicos correspondiente al área de *Software Requirements*- en los cuales está dividida cierta área igualmente cada uno de estos tópicos es descrito y tiene asociados un conjunto de conceptos que son definidos en esta guía. A su vez estos conceptos están relacionados con materiales de referencia en base a los cuales se puede ampliar más el concepto que está siendo consultado y a diversas disciplinas de conocimiento con las cuales guarda relación la Ingeniería de Software.

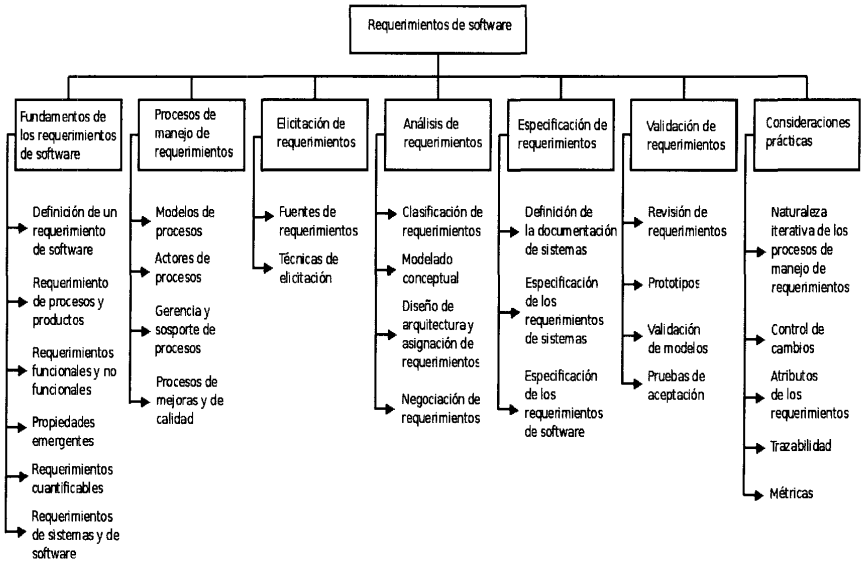

Figura3-2.Subdivisión de Tópicos para el área de conocimiento Requerimientos de Software

En base a esta organización de la guía *SWEBOK* se realizó el diagrama de clases que se muestra en la *figura 3-3*, el cual está representado con las siguientes clases:

- **Definición:** representa cada una de las definiciones utilizadas para describir un área de conocimiento.

- **Disciplina:** representa las disciplinas relacionadas con cada uno de los conceptos asociados a los tópicos de las áreas de conocimiento.

- **Tópico:** cada una de las subdivisiones presentes en cada área de conocimiento.

- **Área:** representa cada una de las áreas de conocimiento definidas en la guía SWEBOK.

- **Concepto:** representa cada uno de los conceptos que se encuentra por debajo de los tópicos.

- **Referencia Externa:** representan las referencias externas a las cuales se encuentran relacionadas cada uno de los conceptos.

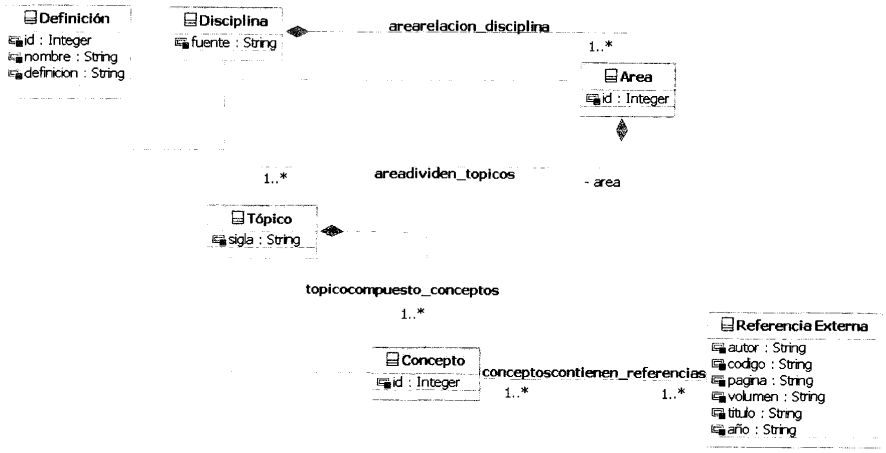

Figura3-3. Diagrama de Clases.

Se explica brevemente las relaciones existentes entre cada una de las clases del diagrama:

1- La relación existente entre las clases "Concepto" y "Referencia Externa" es una asociación binaria.

2.- Las relaciones de composición se representan con un rombo relleno del lado de la clase que contiene a la otra agregación.

En el diagrama existen relaciones de composición entre las siguientes clases:

2.1.- Relación entre las clases "Tópico" y "Concepto": Un tópico está compuesto de muchos conceptos.

2.2- Relación entre las clases "Área" y "Tópico": Un Área está compuesta de muchos Tópicos.

2.3.- Relación entre las clases "Disciplina" y "Área": Una disciplina está compuesta (se encuentra presente en diferentes áreas).

Debido a la manera en como se encuentran organizados los conceptos en la guía SWEBOK, cada una de las entidades definidas en la *figura 3-3* tiene una definición asociada, es por esta razón que la clase *Definición* es la super clase del diagrama, esto con la finalidad de no repetir los atributos de la misma en las clases asociadas a ella.

3.- Se presentan relaciones de herencia entre la clase "Definición" (clase Padre) y sus derivadas ("Disciplina", "Tópico", y "Concepto").

Finalmente se realizó una correspondencia entre el diagrama de clases y la ontología siguiendo los estándares establecidos por *Protégé* para el desarrollo de las mismas. En el siguiente apartado se explica detalladamente la manera como fue creada la ontología.

3.4 Transformación del Diagrama de clases UML a una Ontología en lenguaje OWL.

Pruebas de transformación automática de un diagrama modelado en UML a una Ontología: Esta transformación se intentó realizar de manera automática a través de diferentes pruebas e investigaciones cuyos resultados no fueron exitosos y se presentan a continuación:

- **The DAML-UML Enhanced Tool**: Proporciona un ambiente basado en UML para el desarrollo y manipulación de ontologías DAML [38].

Para este caso se realizaron pruebas con ArgoUML y RationalRose

  - **ArgoUML**: se descargó el instalador de DUET de la página señala arriba, se realizó la instalación siguiendo los pasos que aparecen documentados en *DUET Installation Guide*. La versión de Argo utilizada para hacer la prueba fue la 0.24 última versión disponible en el sitio Web del producto.

    Resultados: Al realizar los pasos de instalación indicados el Argo no volvió a cargar correctamente.

- **Rational Rose**: la versión recomendada del Rational para esta prueba es la 2000, lamentablemente no se tenía disponibilidad del producto. Aunque se dispone de la licencia del *Rational Rose 2003* no se realizó ya que se considera que el plugin *DUET* no está lo suficientemente estable para esta versión.

- **UMLBackEnd**: Representa una funcionalidad del software *Protégé* que permite importar y exportar archivos en formato UML 1.4, que hayan sido almacenados en XMI versión 1.1 ó 1.2 . Según se indica en la documentación la forma más sencilla de generar este tipo de archivos XMI es utilizando herramientas CASE. El software recomendado para hacer la exportación del archivo XMI a partir del diagrama UML es *Poseidon versión 1.6* [39].

  Los inconvenientes encontrados al realizar esta prueba fueron: al ingresar en la página del software *Poseidon* para poder obtener el producto con todas sus funcionalidades es necesario adquirir la licencia, adicionalmente la versión 1.6 no se encuentra disponible, sin embargo se realizó la prueba con una demo de una versión superior, sin obtener resultados satisfactorios.

- **Convertidor UML2DAML**: es una aplicación Open-Source creada para automatizar la transformación de los diagramas de clases UML 1.4 exportados como XMI 1.1 en DAML+ OIL o RDF schema [40].

  En la página de *UML2DAML Converter* (http://www.jdev.de/) aparecen como posibles software para realizar la exportación del modelo UML a un archivo del tipo *XMI*: *ArgoUML, Rational Rose, Together*; es importante acotar que este producto posee una documentación limitada y escasos foros de ayuda a los usuarios.

- **Prueba con ArgoUML 0.24**: Se realizó la prueba con *ArgoUML 0.24*, para esta versión se utilizó sobre *Argo UML 1.4*, siguiendo los siguientes pasos:

1.- Se realizó la exportación a *XMI 1.2* tal como se muestra en la *figura 3-4*

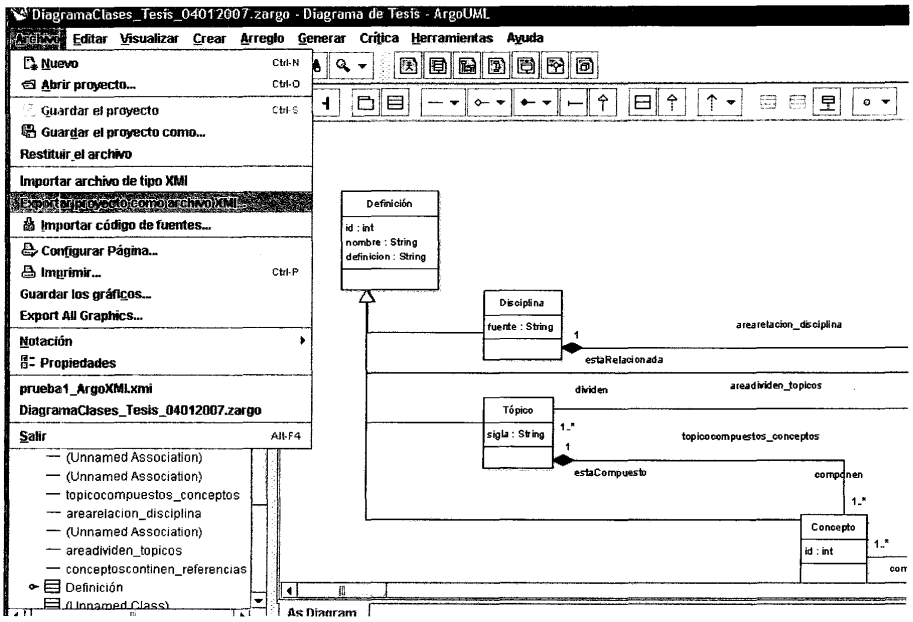

Figura3-4. Exportación archivo XMI

En el *Anexo 1* se muestra el archivo *XMI* obtenido luego de hacer la exportación.

Una vez obtenido el archivo *XMI* se realizó su importación dentro del programa *JDEV*

1.2 Para realizar la corrida del JDEV se procedió de la siguiente manera:

- Se descomprimió el *jar jdev-u2d-0.4* en la siguiente ubicación *C:\jdev-u2d-0.4*
- Se corrió utilizando el comando, ubicándose en la carpeta *C:\jdev-u2d-0.4\jar,* tal como se muestra en la *figura 3-5.*

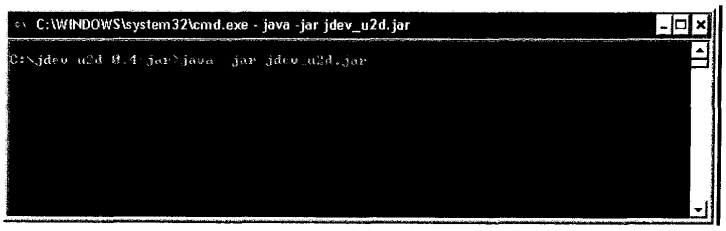

Figura3-5. Comando de ejecución programa UML2DAML Converter

- Apareciendo la aplicación en pantalla, tal como se muestra en la *figura 3-6.*

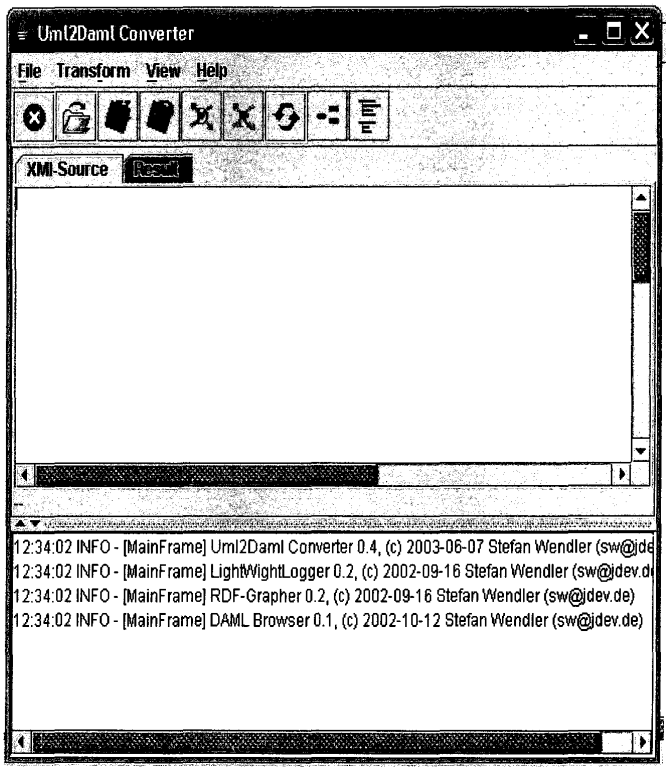

Figura3-6. Aplicación UML2DAML Converter

1.3 Se realizó la importación del archivo *XMI* generado en *ArgoUML* dentro del *UML2DAML Converter* y luego su transformación tal como se muestra en la figura 3-7, sin embargo, el archivo RDF resultante no es válido. Para verificar la validez del archivo RDF se utilizó un validador RDF suministrado por *W3C* aunque el verificador de sintaxis arrojó resultados válidos, si se observa el archivo (*Anexo 1*) se puede apreciar que las clases que provienen del diagrama de clases no se ven reflejadas haciendo que este RDF no resulte apropiado para

generar la ontología, en base a este resultado se realizó una consulta a los desarrolladores del producto sin obtener una respuesta satisfactoria [41].

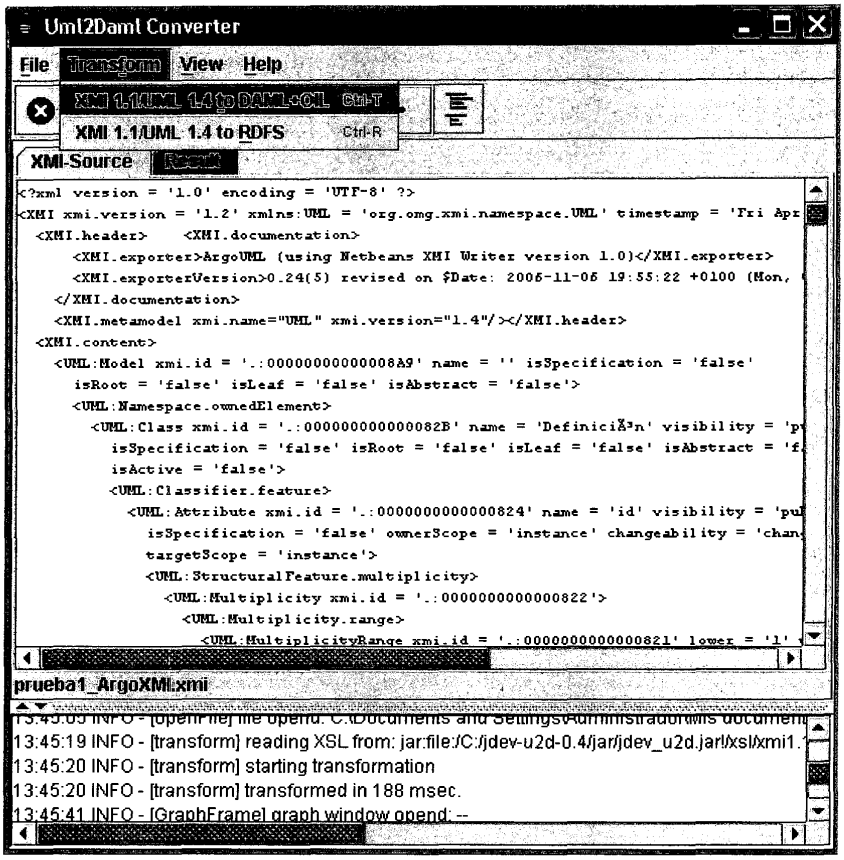

Figura3-7. Transformación del XMI en el UML2DAML Converter

3.- **Prueba realizada con Rational Rose 2003**: en el caso de Rational Rose cuando se instala el producto no contiene directamente la exportación de los diagramas UML en archivos de formato XMI, se realizó una búsqueda para esta versión del producto

de un plugin que permite realizar la exportación de este tipo de archivos, no encontrándose el plugin adecuado disponible para esta versión del producto.

4.- **Prueba realizada con Together 6.0**: se trató de realizar la prueba con el demo del producto pero no posee la opción para realizar exportaciones a XMI, ya que no se contaba con una licencia para este producto, no se hizo la prueba con el mismo.

3.5 Construcción de la Ontología utilizando Protégé:

A pesar de que existen algunas metodologías definidas para la creación de las ontologías, su proceso de creación es definido como un proceso iterativo, en el cual generalmente se define una primera ontología, la cual es luego revisada y refinada para lograr un mayor nivel de detalle.

Se realizó la creación de la ontología utilizando *Protégé versión 3.2.1,* basándose en la abstracción de los términos utilizados en la guía SWEBOK; se resumen a continuación los pasos tanto a nivel metodológico como técnico para la construcción de la ontología:

- **Determinación del ámbito:** el dominio que cubre la ontología creada está relacionado con las diferentes etapas que deben tomarse en cuenta en el desarrollo de aplicaciones de software, los conceptos relacionados con cada una de estas etapas; el objetivo fundamental se basa en proveer información a los Ingenieros de Software para que puedan realizar consultas de manera óptima.

  Se presentan a continuación posibles preguntas que pudieran encontrar respuesta al realizar una consulta a la ontología creada con el objetivo de mostrar su utilidad:

1.- ¿Cuáles son las etapas presentes en el proceso de desarrollo de software?

2.- Para una etapa en específico: ¿Cuáles son los conceptos relacionados con la misma y qué debe hacer el Ingeniero de Software para cumplir con la misma?

3.- ¿Referencias bibliográficas relacionadas con cada uno de los conceptos y etapas?

4.- ¿Cuáles son las disciplinas relacionadas con la Ingeniería de Software y con cada una de las etapas presentes?

- **Consideración la reutilización:** se realizó una investigación para buscar Ontologías que consideren el manejo del conocimiento relacionado con la Ingeniería de Software y las diferentes etapas asociadas al desarrollo de sistemas; a pesar de que existen varios trabajos de investigación relacionados con este tema, no se encontró una ontología para Ingeniería de Software de libre disponibilidad en la red que esté relacionada directamente con este dominio.

- **Enumeración de los términos y definición de las clases**: para elaborar el diagrama de clases que se mostró en la *figura 3-3*, tal como se mencionó anteriormente, se llevó a cabo una revisión detallada de la guía *SWEBOK*, realizando una abstracción de las clases que representan de alguna manera el conocimiento que se encuentra en la misma. De manera general, algunos de los términos extraídos de este análisis y de los cuales fueron tomados las clases son: área, concepto, definición, tópico, referencias, disciplinas.

Para realizar la definición de estas clases con la herramienta *Protégé versión 3.2.1,* se utilizó la funcionalidad de creación de clases suministrada por la herramienta; es importante destacar que la jerarquía de las clases se conservó al realizar la transformación del diagrama UML a la Ontología. Se muestran en la *figura 3-8* las clases creadas con la jerarquía correspondiente.

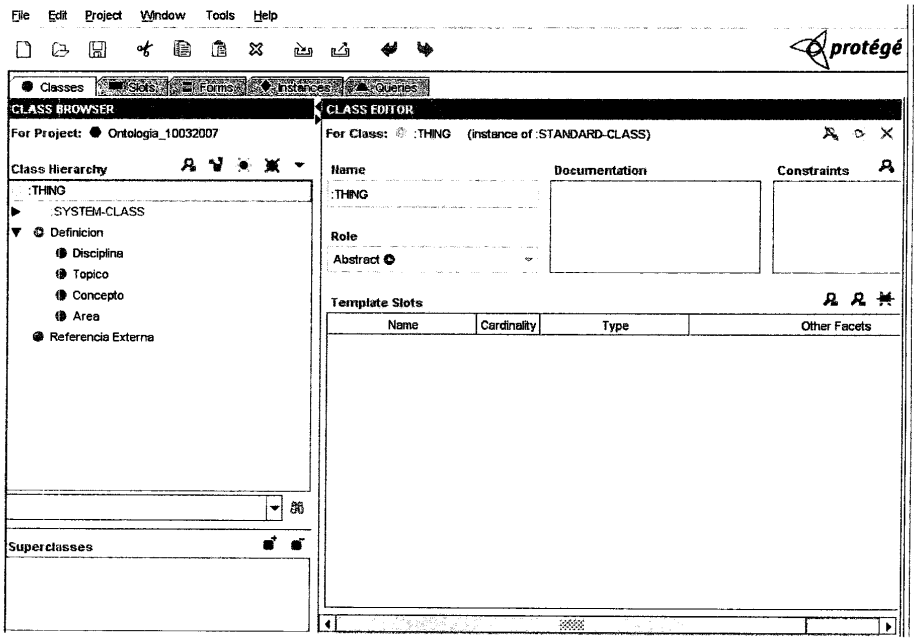

Figura3-8. Clases de la Ontología representadas en Protégé.

- **Definición de las propiedades de las clases (slots):** para cada una de las clases representadas se definieron sus propiedades *slots*; en *Protégé versión 3.2.1* existen variadas formas para crear los *slots*, una de ellas es

utilizar el *Slots Tab*, con lo cual es posible reutilizar los *slots* para todas las clases que sean creadas dentro de un proyecto.

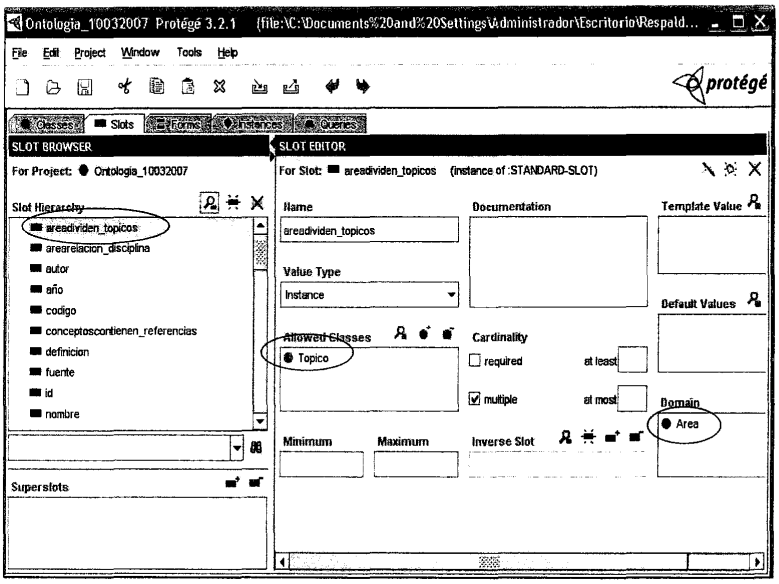

Figura3-9. Slots para la clase Área.

- **Definición de las restricciones asociadas a cada una de las propiedades:** por medio de *Protégé* se crearon las relaciones entre las clases; esto se realiza utilizando los *slots*. Por ejemplo, se creó un *slot* llamado *areadividen_topicos*, el cual representa la relación existente entras las clases *Área* y *Tópico*; en la *figura 3-9* se observa encerrado en un ovalo el *slot* y las clases involucradas en esta relación. Lo que se hizo fue crear un *slot* que almacene una o más instancias de los *Tópicos*, permitiendo que en el momento que se cree una instancia de la clase *Área* y se necesite conocer los tópicos en los cuales se subdivide esa área, se pueda obtener al cargar el valor del *slot areadividen_topicos.*

- **Creación de las instancias:** las instancias representan los datos actuales en la base de conocimiento representada por la ontología creada. Para hacer la creación de las instancias usando *Protégé*, se selecciona el *Tab Instantes,* el cual proporciona los editores y los navegadores para realizar la carga de la información. En la *figura 3-10* se muestra un ejemplo de información cargada para la clase *Área,* en la ventana "*Class Browser*" se puede observar seleccionada la clase *Área* con un número 2 encerrado entre paréntesis, lo cual quiere decir que hay dos instancias creadas, al presionar cada una de estas instancias en la ventana "*Instante Browser*" se observa los *slots* asociados a la misma con sus respectivos valores.

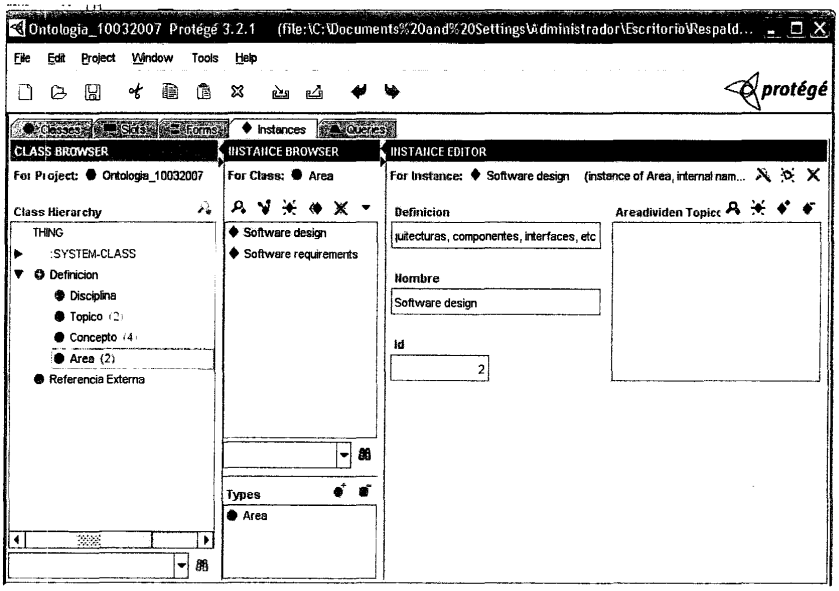

Figura 3-10 Instancias para la clase Área.

De esta forma la ontología con sus respectivas instancias quedaría preparada para ser consumida por algún sistema de manejo de conocimientos el cual permita manejar y mostrar la información contenida en la misma.

Finalmente a manera de resumen, en la *figura 3-11* donde se ilustra la equivalencia entre un extracto del diagrama de clases y su correspondiente lenguaje *OWL* generado por *Protégé*.

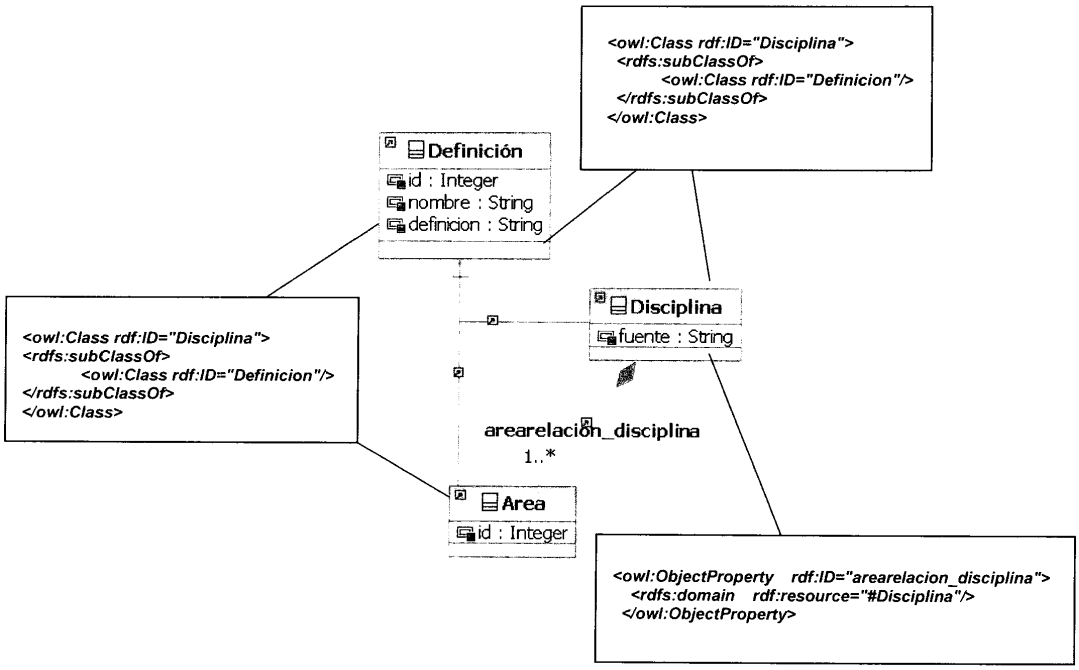

Figura 3-11 Diagrama de Clases y Ontología.

En el *Anexo 1* se encuentra la Ontología generada en lenguaje OWL.

3.6 Verificación de la Ontología creada:

Esta verificación se realizó de la siguiente manera:

- **Validador RDF:** Para realizar esta prueba se utilizó la exportación de la ontología en formato .rdf Luego este archivo fue introducido en la página del validador obteniendo los resultados exitosos mostrados en la *figura 3-12* [41].

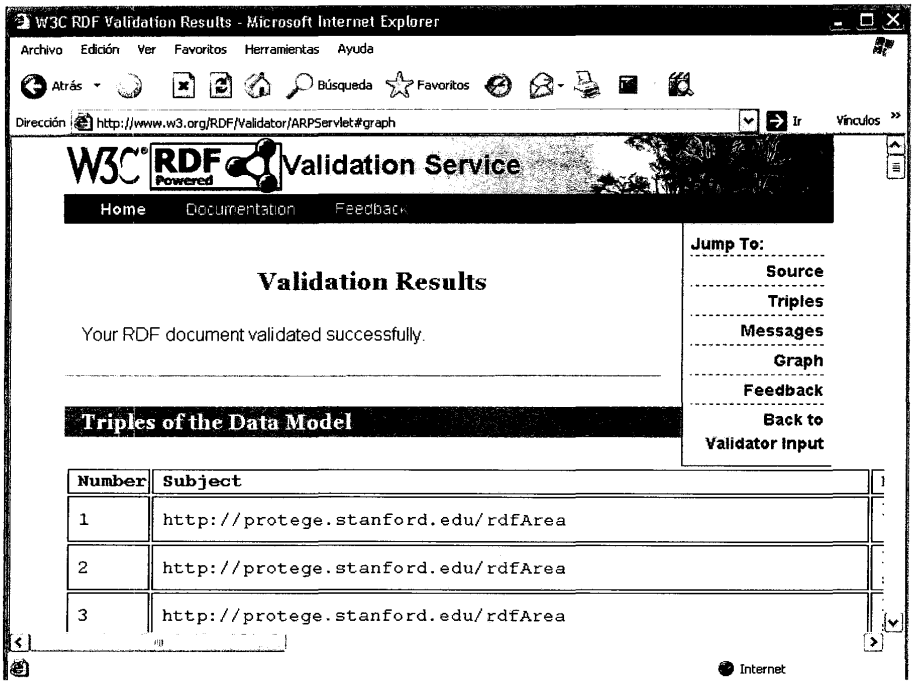

Figura 3-12 Validador RDF

3.7 Desarrollo de la aplicación Web para consultar la ontología.

El objetivo fundamental es crear un sistema Web construido sobre tecnologías de descripción de contenidos utilizando como base la ontología creada en OWL para hacer búsquedas sobre la misma. Esto permitirá aportar descripciones explícitas de los recursos Web, dejando adicionalmente accesibles los datos a través de las etiquetas inmersas en el documento tal como si fuese una base de datos.

Para el desarrollo de la aplicación se utilizaron las siguientes tecnologías:

- **Protégé API OWL**: son un conjunto de clases utilizadas para realizar la lectura del archivo ***.owl***. Básicamente se cargan las clases, sus respectivos atributos y las relaciones conjuntamente con los datos asociados. Las interfaces principales que fueron implementadas dentro de la aplicación se listan a continuación:

    - ***OWLModel:*** proporciona acceso al contenedor de recursos de la ontología. Se utiliza para crear, consultar y borrar recursos de varios tipos.
    - ***OWLNamedClass:*** corresponde a la *owl:Class* del OWL.
    - ***OWLIndividual:*** corresponde a un *Individual* dentro de la ontología que esté definida.
    - ***OWLDataTypeProperty:*** corresponde a un *Property* dentro de la ontología que esté definida.

- **GWT (Google Windows Toolkit):** utilizado para realizar la interfaz web.
- **Apache Axis versión 1.4: es** una implementación del protocolo *SOAP* utilizada para implementar servicios Web en el lenguaje de programación Java [42].
- **XStream**: es una librería utilizada para serializar objetos a XML y viceversa [43].

3.7.1 Estructura del proyecto Java implementado

La programación de la aplicación fue realizada utilizando el lenguaje de programación Java, el proyecto está dividido en dos subproyectos:

- **ontologyQuery:** el cual representa el proyecto servidor ya que contiene las clases encargadas de realizar la lectura del archivo ***.owl*** y la clase que representa al servicio Web que es publicado dentro de ***Apache Axis.***

- **ontologyClient**: el cual representa el proyecto cliente que se encarga de consumir el servicio que retorna a la ontología creada y de mostrar los resultados a través de la interfaz gráfica

A continuación se resume el código utilizado para realizar la carga y lectura del archivo OWL:

En el siguiente extracto de código mostrado en la *Tabla 3-1* se puede observar la creación utilizando una dirección URL de un *OWLModel*, objeto a través del cual se podrá realizar la carga del archivo *.owl.*

Tabla 3-1 Creación del Objeto OWLModel

```
String uri = "file:///C:/owl/tesis.owl";
OWLModel owlModel=null;
owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri);
```

En la *Tabla 3-2* se puede observar el método *cargarListaAtributo* el cual recibe como parámetros el *OwlModel* previamente cargado y dos cadenas de caracteres que representan el Individual que se quiere leer y el nombre de la relación. Utilizando la clase *OWLIndividual* con los métodos *getOWLIndividual* y *getPropertyValues* es posible obtener una colección de todos los valores relacionados con un Individual determinado.

Tabla 3-2 Método cargarListaAtributo

```
public static Collection cargarListaAtributo(OWLModel owlModel, String  sIndividual, String sRelacion)
 {
  OWLIndividual ontIndividual = owlModel.getOWLIndividual(sIndividual);
    Collection collection = ontIndividual.getPropertyValues(owlModel.getRDFProperty(sRelacion));
    return collection;
 }
```

En la *Tabla 3-3* se puede observar el método *cargarAtributo* el cual permite a través de los parámetros suministrados cargar los valores de los atributos solicitados.

Tabla 3-3 Método cargarAtributo

```
public static String cargarAtributo(Iterator iterator, OWLModel owlModel, String att, OWLIndividual
individual)
  {
    OWLDatatypeProperty firstNameProperty = owlModel.getOWLDatatypeProperty(att);
   OWLIndividual myCustomer = owlModel.getOWLIndividual(individual.getBrowserText());
   String atributo = (String) myCustomer.getPropertyValue(firstNameProperty);
     return atributo;
}
```

En el Anexo II se encuentra la documentación detallada de cada uno de estos métodos pertenecientes a la clase *MainOwl.java* la cual es la encargada de realiza la carga del archivo ***.owl.***

3.7.2 Publicación del Servicio Web para la ontología creada:

Se resume a continuación la manera como se utilizó *Apache Axis* para realizar la exposición del servicio Web, que después es consumido por la aplicación cliente.

- Se codificó en el proyecto servidor la clase *WSOntology.java* cuyo objetivo fundamental es describir al servicio Web, la misma a través del llamado al método *cargarOntologia()* de la clase *MainOwl.java*, retorna un *XML* que contiene los objetos que representa los datos extraídos del archivo *.owl*. Esta clase se encuentra descrita con más detalle en el Anexo II.

- A partir de la clase *WSOntology.java* y a través de la siguiente línea de comandos se generó el archivo WSDL. Es importante destacar que para realizar esta ejecución

deben estar configuradas todas las variables de ambiente para que el contenedor Axis funcione de manera correcta, más detalles acerca de la configuración se puede encontrar en los tutoriales de Axis [43]. En este caso puede observarse en la *Tablas 3-4 y 3-5* las variables de ambiente definidas *AXISCLASSPATH y CLASSPATH* :

Tabla 3-4 Variable de ambiente AXISCLASSPATH

AXISCLASSPATH:C:\axis-1_4\lib\axis.jar;C:\axis-1_4\lib\axis-ant.jar;"C:\axis-1_4\lib\commons-discovery-0.2.jar";"C:\axis-1_4\lib\commons-logging-1.0.4.jar";C:\axis-1_4\lib\jaxrpc.jar;C:\axis-1_4\lib\saaj.jar;"C:\axis-1_4\lib\log4j-1.2.8.jar";"C:\axis-1_4\lib\wsdl4j-1.5.1.jar";

Tabla 3-5 Variable de ambiente CLASSPATH

CLASSPATH:*C:\xerces\xercesImpl.jar;C:\xerces\xmlapis.jar;C:\xerces\xmlParserAPIs.jar;C:\axis-1_4\lib\axis.jar;C:\axis-1_4\lib\axis-ant.jar;"C:\axis-1_4\lib\commons-discovery-0.2.jar";"C:\axis-1_4\lib\commons-logging-1.0.4.jar";C:\axis-1_4\lib\jaxrpc.jar;C:\axis-1_4\lib\saaj.jar;"C:\axis-1_4\lib\log4j-1.2.8.jar";"C:\axis-1_4\lib\wsdl4j-1.5.1.jar";*

Tabla 3-6 Comando para crear archivo WSDL.

*java -classpath %AXISCLASSPATH%;%CLASSPATH% org.apache.axis.wsdl.**Java2WSDL** -o wsdlontology.wsdl -l"http://localhost:8080/axis/services/ontology" -n urn:"com.ula.ws" -p" com.ula.ws" urn:*com.ula.ws com.ula.ws.WSOntology

En la línea de comando mostrada en la *Tabla 3-6* puede observarse el uso de la clase **Java2WSDL** perteneciente al paquete *org.apache.axis.wsdl* la cual es utilizada junto con la clase creada para la implementación del servicio Web (*WSOntology.java*) para crear el archivo WSDL.

-o : indica el nombre de salida del archivo WSDL.

-l : indica la localización del servicio.

-n : es el espacio de nombres del archivo WSDL.

-p : indica el espacio de nombres del paquete donde se encuentra la clase que expone al servicio Web.

El documento WSDL resultante debe contener los tipos WSDL apropiados, mensajes, puertos, enlaces, descriptores con soporte SOAP. El archivo WSDL para nuestra aplicación se encuentra en el Anexo II.

- Una vez creado el archivo WSDL a través del siguiente comando mostrado en la *Tabla 3-7* se realiza la creación de las clases necesarias que deben estar disponible para consumir el servicio Web.

Tabla 3-7 Comando para crear clases utilizadas para consumir Servicio Web.

*java -classpath %AXISCLASSPATH%;%CLASSPATH% org.apache.axis.wsdl.**WSDL2Java** -o . -d Session -s -p com.ula.servicioweb wsdlontology.wsdl*

Puede observarse el uso de la clase ***WSDL2Java*** perteneciente al paquete *org.apache.axis.wsdl,* a través de la cual es posible generar los enlaces cliente/servidor representados en forma de clases del Servicio Web.

En nuestro caso se generaron los siguientes archivos:

OntologySoapBindingImpl.java: es un archivo .java que contiene la implementación del servidor por defecto del Servicio Web.
OntologySoapBindingStub.java: es la parte cliente que actúa como la representación local o proxy del objeto remoto.
WSOntology.java: archivo nuevo generado que representa la interfaz que contiene el uso apropiado de **java.rmi.Remote.**
WSOntologyService.java: archivo .java que contiene el lado cliente de la interfaz de servicio.

WSOntologyServiceLocator.java: archivo .java que contiene la implementación de la clase del lado del cliente.

deploy.wsdd: representa el descriptor de despliegue.

undeploy.wsdd: representa el archivo para realizar la desinstalación de los Servicios Web.

- Una vez creadas las clases se consumen por parte del proyecto cliente, esto con el objetivo de cargar la información que se encuentra dentro de la Ontología y utilizarla posteriormente en la elaboración de la interfaz gráfica al usuario. Este código encuentra en el Anexo II.

3.7.3 Arquitectura de la aplicación:

En esta sección se ilustra el diseño de la aplicación, se presentan a continuación dos diagramas que muestran tanto a nivel de codificación como de diseño a más alto nivel, los distintos componentes involucrados en la elaboración del sistema y la forma como estos interactúan entre si.

En la figura 3-13 se puede observar la arquitectura planteada para la aplicación desarrollada, básicamente se cuenta con dos capas:

- **Capas de Persistencia/Ontología:** esta capa está compuesta por los archivos *.owl* que son los que contienen la información necesaria que será cargada dentro del sistema.
- **Capa de Presentación:** está compuesta por las páginas de presentación y su interacción con el Servicio Web, expuesto, que carga la información de interés para el usuario.
- **Capa de Modelo:** que contiene los componentes de dominio elaborados en el lenguaje de programación *Java* que se encargan de interactuar con los archivos *.owl* para hacer la extracción de la información.

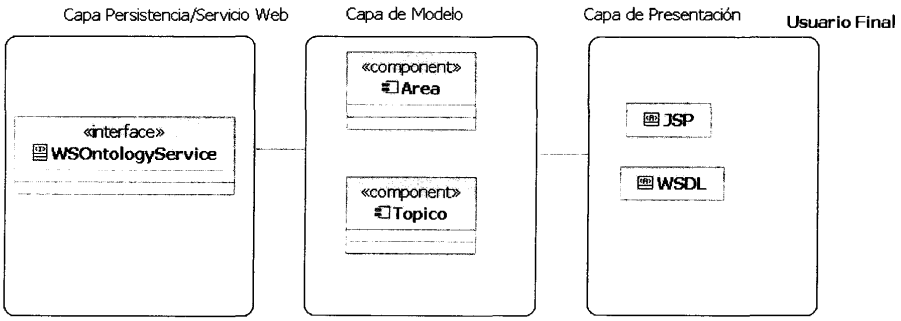

Figura 3-13 Arquitectura aplicación.

La aplicación contiene clases relacionadas con el dominio de la misma, y se encuentran representadas, por ejemplo en las clases: *Area.java, Topico.java.* La interacción con el usuario final, en este caso, está controlada por un Servicio Web que es el que envía la información final a la página JSP (*Java Server Page*). Es importante destacar que para futuros trabajos, es recomendable tomar en cuenta el uso de máquinas razonadoras inteligentes, conocidas en inglés como *Reasoners* , o máquinas para definición de reglas, como *SWRL* (*Semantic Web Rules Languages*) cuyo objetivo fundamental es lograr un comportamiento inteligente.

Adicionalmente se presenta en la figura 3-14 un diagrama que ilustra la manera como están organizados los componentes desarrollados para la implementación de la aplicación.

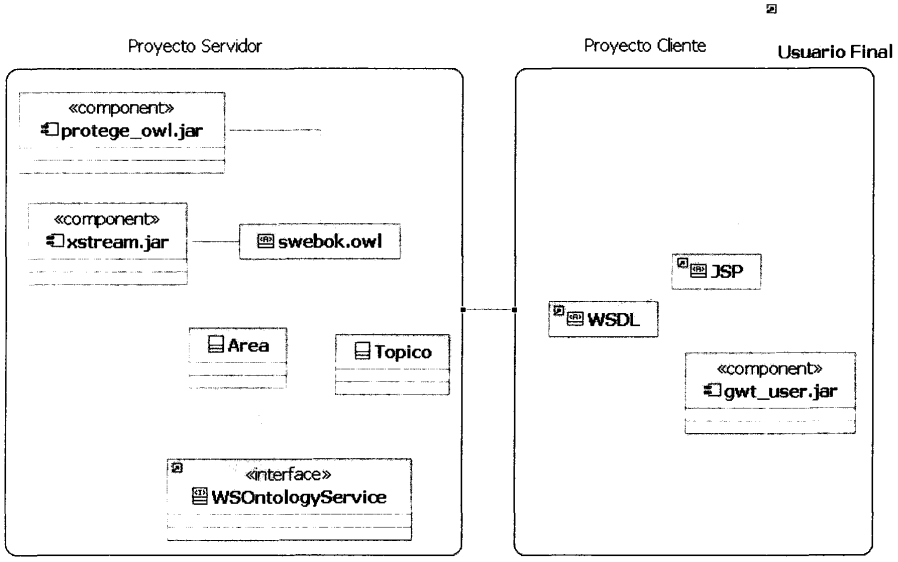

Figura 3-14 Arquitectura a nivel de la codificación.

La aplicación consta de dos proyectos desarrollados en el lenguaje de programación Java, un proyecto servidor y un proyecto cliente. En líneas generales el proyecto servidor se encarga de realizar la lectura de la Ontología desarrollada en *Protégé* a través del API destinado para este fin, esta lectura es cargada en objetos Java los cuales forman parte del dominio de la aplicación (Area.java, Topico.java, etc), estos objetos son transformados a XML a través del uso del *XStream*; la transformación es realizada ya que esta información va a ser enviada a través de la red por medio de un Servicio Web, los cuales se comunican utilizando mensajes SOAP que formaliza el uso de XML como un modo de pasar datos de un proceso a otro. Finalmente esa información es tomada del lado del cliente para ser desplegada de manera gráfica al usuario final.

3.7.4 Interfaz Gráfica:

Tal como se mencionó en el punto anterior con la información obtenida del servicio se realiza la carga de esta información gráfica a través del navegador. En la figura 3-15 se muestra cómo se podrá visualizar la interfaz gráfica una vez que la información esté cargada.

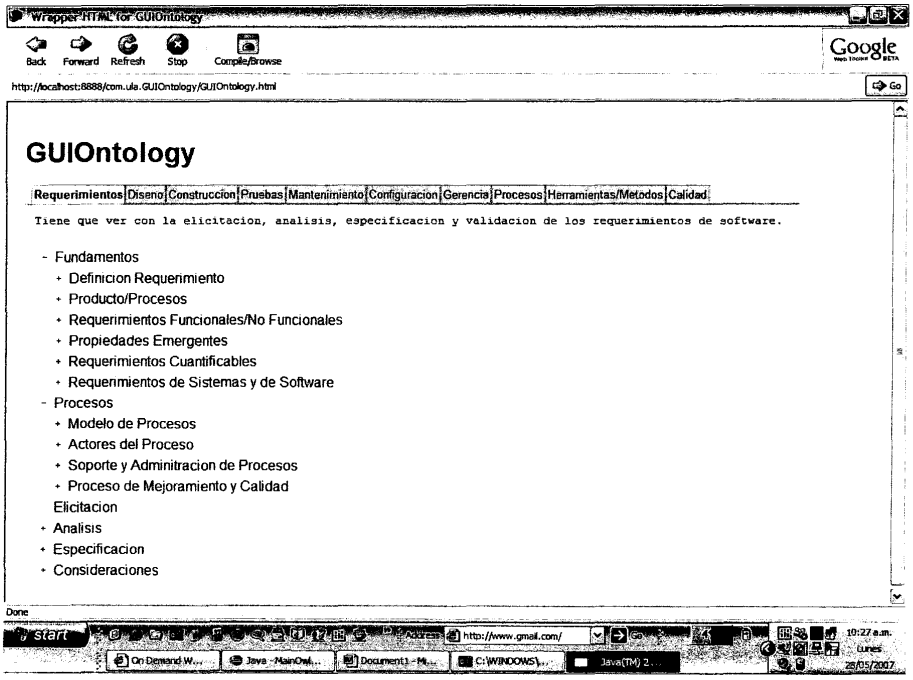

Figura 3-15. Interfaz de usuario final.

Se puede observar que cada pestaña está relacionada con un Área del conocimiento. Al presionar cada pestaña se desplegará en un árbol todos los tópicos relacionados a través del cual podrán ser consultados los conceptos asociados a los mismos. Adicionalmente se podrá cargar la información relacionada con las referencias, bien sean libros, artículos u otros, que el usuario, en su papel de Ingeniero de Software, requiera para realizar un análisis determinado.