Universidad de las Ciencias Informáticas

Facultad #10

Modelos Semánticos para la Gestión de Requisitos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Dayra Iris Hechavarría Rodríguez

Tutores:

MSc. David Leyva Leyva

Ing. Daymy Tamayo Avila

Ciudad de la Habana, julio 2007

“… lo que da al hombre el poder, no es el mero conocimiento que viene del uso de los sentidos, sino, ese otro conocimiento más profundo que se llama Ciencia.”

José Martí

Declaración de Autoría

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

Para que así conste firmo la presente a los _2_ días del mes de _Julio_ del año 2007.

_____ _____

Firma del Autor Firma del Tutor

(Dayra I.Hechavarría Rodríguez) (**MSc. David Leyva Leyva)**

_____

Firma del Tutor

(**Ing. Daymy Tamayo Avila**)

Datos de Contacto

Tutor: David Leyva Leyva

MSc. David Leyva Leyva, Vicedecano de Producción e Investigación, Facultad 10, Universidad de Ciencias Informáticas. Teléfono: (53) (07) 837 2503. davidl@uci.cu Licenciado en Cibernética-Matemática, Universidad de Las Villas (UCLV), 1988. Máster en Computación Aplicada, Universidad de Las Villas (UCLV), 1995. Profesor de la Universidad de Holguín desde 1990. Jefe de Departamento de Informática (2003-2005). Participó en un Proyecto de Educación a Distancia en la Universidad del 2001 hasta el 2005, año en el que comenzó a trabajar en la Universidad de Ciencias Informáticas. Ha participado en un gran número de eventos nacionales e internacionales. Le fueron concedidas sendas becas Intercampus en La Universidad de Castilla-La Mancha (1998) y La Universidad Autónoma de Madrid (2000). Trabajó como profesor invitado en la University of Belize (Belice, de 2001 a 2003) y de la Universidad Nacional de Ingeniería (Managua, Nicaragua, 2005). Actualmente es miembro de un grupo de investigación y desarrollo orientado a la extensión de la plataforma Moodle y el desarrollo de otras herramientas para la teleformación.

Tutor: Daymy Tamayo Ávila.

Ing. Daymy Tamayo Ávila, Especialista de la Dirección de Teleformación, Profesora de Ingeniería de Software Facultad X, Universidad de Ciencias Informáticas. Carretera a San Antonio de los Baños Km. 2 ½, Torrens, Boyeros, Ciudad de La Habana. Cuba. Teléfono: (53) (07) 837 2453. daymy@uci.cu Ingeniera Informática, Universidad de Holguín, 2005. Culminó sus estudios con índice académico de 4.94 puntos, obteniendo Título de Oro. Fue Alumna de Alto Aprovechamiento Académico y Alumna Ayudante en la Disciplina de Técnicas de Programación de Computadoras desde el segundo año de su carrera. Participó en un Proyecto de Educación a Distancia en la Universidad del 2001 hasta el 2005, año en el que comenzó a trabajar en la Universidad de Ciencias Informáticas. Ha participado en un gran número de eventos nacionales e internacionales. Ha sido tutora de varios Trabajos de Curso y Diploma, y de Alumnos Ayudantes. Actualmente es líder de un grupo de investigación y desarrollo orientado al desarrollo y soporte de herramientas para la teleformación.

Agradecimientos

Les agradezco de todo corazón a mis tutores, por su gran ayuda y dedicación para que la realización de esta tesis fuera satisfactoria.

A David Leyva L. por su apoyo permanente durante todo el desarrollo de la investigación, mejor tutor de tesis no podía haber sido.

A todos mis compañeros de estudio, especialmente a los cercanos, por sus preocupaciones y solidaridad.

A mis amistades, (sin mencionar nombres para evitar omitir a alguno de ellos), por su compañía y por darme ánimo para seguir adelante. Dicen que las amistades se apagan cuando mueren los recuerdos… esa llama seguirá viva.

A la Universidad de las Ciencias Informáticas (UCI), y en general a todos los miembros de este gran proyecto de la Revolución, nacido de la Batalla de Ideas, por los recursos que han puesto a mi disposición.

A mi Amparito y a Lissette por su atención y dedicación en estos años vividos junto a ellas.

A mi novio Lannie O. Herrera Pérez por su ayuda, su compañía incondicional, y su gran valentía al defenderme en cada minuto de mi estancia en la Universidad, sin su apoyo no hubiera sido posible haber dejado plasmada mis letras en ésta tesis y no hubiera llegado a la meta.

A mi familia, gracias por todo el amor y confianza que siempre me han otorgado. Sin ustedes, este sueño no significaría nada. Gracias sobre todo a ti, Denia, por todos los sacrificios y horas de desvelo.

A todos los que no he mencionado, pero que consciente o inconscientemente hicieron posible esta aventura.

Dedicatoria

Este trabajo está dedicado a mi familia y en especial a mi madre Denia y a mi hermana Dayrisa, las cuales quiero mucho. A mi novio Lannie Octavio Herrera, por su paciencia y días dedicados a mí, a todas las personas que han confiado en mí y a nuestro comandante en jefe, Fidel Castro Ruz.

Resumen

La Gestión de Requisitos, es el proceso encargado de la identificación, asignación, verificación, y modificación de los requisitos a lo largo del ciclo de vida del software. Además es considerada como uno de los procesos más importantes dentro de la Ingeniería de Requisitos. Con una buena Gestión de Requisitos se logra crear software de buen rendimiento que satisface realmente las necesidades del usuario. En este trabajo se abordan los aspectos teóricos necesarios sobre la Gestión de Requisitos, las características más importantes de la misma y las herramientas más utilizadas para su realización. Además se hace referencia a características de la semántica, los conceptos referentes al proceso de metamodelado y herramientas para metamodelado. Se abordan además las ventajas y desventajas de la Reutilización de Requisitos como una buena alternativa para la reducción del tiempo del trabajo de los proyectos. Se dan a conocer casos de estudio donde se vincula el enfoque semántico a la Gestión de Requisitos. Se sugiere la creación de un repositorio utilizando estas ideas para ser aplicado en proyectos productivos.

Palabras claves: Semántica, Modelo Semántico, Requisitos, Metamodelado, Ontologías, Repositorio, Reutilización.

Índice

Introducción

En la actualidad en la Industria de Software existe una tendencia al crecimiento del volumen y complejidad de los productos, y se exige mayor calidad y productividad en menos tiempo.

El proceso de desarrollo de software se encarga de traducir las necesidades del usuario en requerimientos de software (Richard , 1997), estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo.

La Ingeniería de Software, se considera la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas a los problemas de desarrollo de software, es decir, permite elaborar consistentemente productos correctos, utilizables y costos-efectivos. La misma requiere llevar a cabo varias tareas, una de ellas es el análisis de requisitos. El análisis de requisitos permite extraer los requisitos de un producto de software.

La Ingeniería de Software es una tecnología que indica "CÓMO" construir técnicamente un software: económico, fiable y que funcione eficientemente. (Jacobson, 2000)(Booch; 2000)

La ingeniería de requisitos es una disciplina de la Ingeniería de software. Esta disciplina considera diferentes líneas de trabajo, pero una de las más importantes es la gestión de requisitos, la cual se encarga de proveer la dirección y alcance del proyecto.

Los requisitos deben ser la base de cualquier desarrollo de software. La obtención de una especificación de requisitos de alta calidad es fundamental para asegurar que el software se corresponde con las necesidades del cliente.

En el análisis de requisitos se investiga la parte del mundo real (también llamado universo de discurso o minimundo) que se va a modelar para tener en cuenta todas las necesidades de los usuarios finales y así dejarlas documentadas de la forma más completa posible.

Los requisitos una vez establecidos y documentados, sufren cambios continuos, en este sentido, no se trata la obtención ni el análisis de los mismos, se trata de su gestión, es decir, el seguimiento respecto a los cambios que se generan durante el ciclo de vida del proyecto y las herramientas de gestión de requisitos que auxilian y/o automatizan estas tareas.

El uso de herramientas para auxiliar la gestión de requisitos se ha convertido en un aspecto importante de la Ingeniería de Sistemas y el diseño. Considerando el tamaño y la complejidad del desarrollo, las herramientas vienen siendo algo esencial. Las herramientas que los gestores de requisitos utilizan para automatizar los procesos de Ingeniería de Requisitos han disminuido el trabajo duro en el mantenimiento de requisitos, añadiendo beneficios significativos al reducir errores. (Loucopoulos, 1995)

El punto de partida en el proceso de desarrollo de software es la construcción de un modelo, el cual actúa como una especificación precisa de los requerimientos que el sistema debe satisfacer. Un modelo del sistema consiste en una conceptualización del dominio del problema. El modelo se focaliza sobre el mundo real: identificando, clasificando y abstrayendo los elementos que constituyen el problema y organizándolos en una estructura formal. Actualmente todos los métodos de desarrollo de software han adoptado formas diferentes. Lo que varía de un método a otro es la clase de modelos que deben construirse, la forma de representarlos, manipularlos, etc.

El modelo de un problema es esencial para describirlo y entenderlo, independientemente de cualquier posible sistema informático que se use para su automatización. El modelo constituye la base fundamental de información sobre la que interactúan los expertos en el dominio del problema, los analistas y los desarrolladores de software.

Por lo tanto es de fundamental importancia que exprese la esencia del problema en forma clara y precisa. Por otra parte, la actividad de construcción del modelo es una parte crítica en el proceso de desarrollo. Los modelos son el resultado de una actividad compleja y creativa y son propensos a contener errores, omisiones e inconsistencias. La verificación del modelo es muy importante, ya que los errores en esta etapa tienen un costoso impacto sobre las siguientes etapas del proceso de desarrollo de software.

En todas las disciplinas de la Ingeniería se hace evidente la importancia de los modelos ya que describen el aspecto y la conducta de "*algo*". Ese "*algo*" puede existir, estar en un estado de desarrollo o estar, todavía, en un estado de planeación. Es en este momento cuando los diseñadores del modelo deben investigar los requerimientos del producto terminado y dichos requerimientos pueden incluir áreas tales como funcionalidad, y confiabilidad. Además, a menudo, el modelo es dividido en un número de vistas, cada una de las cuales describe un aspecto específico del producto o sistema en construcción.

El modelado sirve no solamente para los grandes sistemas, aun en aplicaciones de pequeño tamaño se obtienen beneficios de modelado, sin embargo es un hecho que entre más grande y más complejo es el sistema, más importante es el papel que juega el modelado.

El modelo es una descripción de todo o parte de un sistema escrito en un lenguaje bien definido. El hecho de que un modelo esté escrito en un lenguaje bien definido tiene una gran importancia para Arquitectura Dirigida por Modelos (Model Driven Architecture, o MDA), ya que supone que el modelo tiene asociada una sintaxis y una semántica bien definida. Esto permite la interpretación automática por parte de transformadores o compiladores de modelos, fundamentales en MDA.

Según OMG[1], MDA proporciona una solución para los cambios de negocio y de tecnología, permitiendo construir aplicaciones independientes de la plataforma e implementarlas en plataformas como CORBA (Common Object Request Broker Architecture), J2EE o Servicios Web. Además permite mejorar la productividad, la portabilidad, la interoperabilidad y la reutilización de los sistemas. (OMG, 2003)

El metamodelado es un mecanismo que permite definir formalmente lenguajes de modelado. Así, un metamodelado de lenguaje es una definición precisa de sus elementos mediante conceptos y reglas de cierto metalenguaje, necesarias para crear modelos en ese lenguaje. Básicamente se trata de usar modelos para describir otros modelos. Por ejemplo el metamodelado de UML, define los conceptos y reglas que se necesitan para crear modelos UML.

[1] OMG: Object Management Group, consorcio de empresas de informática.

UML se ha convertido en el estándar para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación, sea o no de software.

Las aproximaciones metodológicas de hoy en día (Hilera, 2003), (Larma, 2003), están altamente influenciadas por el enfoque de orientación por objetos; dichas aproximaciones permiten entender y analizar los fenómenos de la realidad e identificar abstracciones en el espacio de la solución que son consistentes con los elementos del espacio del problema (Booch, 2002). El lenguaje de especificación o modelado, por su parte, representa el vehículo indispensable para la comunicación entre los participantes y para la representación de conceptos a lo largo de todo el proyecto, utilizando una semántica común.

Vincular los lenguajes de modelos al proceso de desarrollo del software, es un buen punto de partida, los sistemas complejos serían fáciles de entender y de describir, ya que estos lenguajes aportan un metamodelado y una semántica. Por ejemplo el UML no es un método Orientado a Objeto, sino que propone una notación y una semántica universal. Es un lenguaje para especificar, construir y documentar artefactos software.

En la Universidad de las Ciencias Informáticas (UCI), los proyectos productivos han presentado algunos problemas, por ejemplo, se ha observado falta en la organización de los mismos, se han utilizado prácticas pobres de Ingeniería, el índice de productividad no ha sido el esperado, y en ocasiones el proceso de análisis de requisitos no se ha hecho con todos los pasos requeridos.

Además no se ha hecho una vinculación de los términos de metamodelo, y lenguajes de modelado en el plan de estudio de la Ingeniería del Software y dentro de esta en la Gestión de Requisitos.

Dada esta situación, se plantea como **problema científico** lo siguiente: ¿Cómo vincular los conceptos de semántica y metamodelado a la Gestión de Requisitos en los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI)?

De acuerdo al problema planteado anteriormente se propone como **objetivo general,** hacer un estudio de la relación entre la semántica y la Gestión de Requisitos, y proponer formas prácticas de su vinculación.

Para dar cumplimiento al objetivo general es necesario responder las siguientes **preguntas científicas**:

I. ¿Cuáles son las definiciones más importantes asociadas a la Gestión de Requisitos y las herramientas que se utilizan para desarrollarlas?

II. ¿Cuáles son los conceptos fundamentales del enfoque semántico vinculados a la Ingeniería del Software?

III. ¿Se podrá mejorar la Gestión de Requisitos utilizando el enfoque semántico?

Este trabajo tiene como **objeto de estudio** la Gestión de Requisitos, y como **campo de acción** la Gestión de Requisitos utilizando el enfoque semántico.

Para responder a estas preguntas se propone desarrollar las siguientes **tareas:**

1- Realizar un estudio acerca de los conceptos asociados a la Gestión de Requisitos.

2- Identificar las herramientas más utilizadas para la Gestión de Requisitos.

3- Estudiar los aspectos relativos al enfoque semántico, y al proceso de metamodelado.

4- Estudiar los estándares asociados al proceso de metamodelado.

5- Analizar la relación entre el enfoque semántico y la Gestión de Requisitos.

6- Identificar casos de estudio en los que se utilice el enfoque semántico para la gestión de requisitos.

Para la realización de este trabajo investigativo se utilizó el método teórico Analítico-Sintético, para comprender la esencia de los procesos de la Ingeniería de Requisitos, la Gestión de Requisitos, y el enfoque semántico, así como la vinculación de los mismos.

También se utilizó el método teórico histórico-lógico, para determinar la evolución de conceptos como semántica, modelo de datos, y metamodelado.

El método empírico revisión de documentos permitió encontrar los términos más importantes en la bibliografía consultada.

El presente trabajo consta de Introducción, tres Capítulos, Conclusiones, Recomendaciones, Referencias bibliográficas, Glosario de abreviaturas, Glosario de términos y Anexos.

En el Capítulo 1, Estado del Arte, se abordan aspectos teóricos necesarios para la investigación, se explica qué es la Ingeniería de Requisitos (IR), las técnicas más usadas de la Ingeniería de Requisitos y las herramientas de la Gestión de Requisitos. Se analizan así las características más importantes de las mismas.

El Capítulo 2, Enfoque semántico, hace referencia a características de la semántica, se tratan conceptos referentes al proceso de metamodelado, las ontologías y se analizan las herramientas y estándares para el metamodelado. También se aborda la utilidad de los modelos en el proceso de desarrollo del software, así como

En el Capítulo 3, Reutilización de Requisitos, se plantean las características, ventajas y desventajas de la reutilización de requisitos. Se dan a conocer casos de estudio que utilizan el enfoque semántico, la gestión de requisitos y su vinculación.

Capítulo I .Estado Del Arte.

Introducción

En este capítulo se hace referencia a conceptos tales como: requisitos, Ingeniería de Requisitos, Gestión de Requisitos entre otros. Se analizan las herramientas de la Ingeniería de Requisitos, así como algunas técnicas de la misma. De la Gestión de Requisitos se hace énfasis en las tareas o actividades.

1.1 ¿Qué es un requerimiento/requisito?

¿Qué es un Requerimiento?

Normalmente, un tema de la Ingeniería de Software tiene diferentes significados. De las muchas definiciones que existen para requerimiento, a continuación se presenta la definición que aparece en el glosario de la IEEE. (Richard ,1997)

(1) Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. (2) Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal. (3) Una representación documentada de una condición o capacidad como en (1) o (2). (Richard , 1997)

1.1.1 Características de los requerimientos

Los requerimientos deben especificarse antes de intentar comenzar la construcción del producto, sin ellos no podrá ser posible llevar a cabo las etapas de diseño y construcción correctamente. Los mismos pueden verse como una declaración abstracta de alto nivel de un servicio que el sistema debe proporcionar, como una definición matemática detallada y formal. Los requisitos cumplen una doble función ya que son la base para una oferta de contrato, por lo tanto deben estar abiertos a la interpretación. Además son la base para redactar el contrato en sí mismo.

Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. A continuación se presentan las más importantes.

Necesario: Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.

Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.

No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Verificable: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación, inspección, demostración o pruebas.

1.1.2 Tipos de requisitos

Los requerimientos pueden dividirse en varios tipos dentro de ellos, se hará referencia a los siguientes:

- Requisitos de usuario
- Requisitos del sistema
- Requisitos funcionales
- Requisitos no funcionales

Requisitos de usuario

Declaraciones en lenguaje natural y en diversos diagramas de los servicios del sistema y de las restricciones bajo las que debe operar.

1.- El sistema debe permitir representar y acceder a archivos externos creados por otras herramientas.

2. Sentencias muy generales sobre lo que el sistema debería hacer.

Requisitos del sistema

Un documento estructurado que determina las descripciones detalladas de los servicios de sistema. Escrito como contrato entre el cliente y el contratista.

1.- El usuario deberá poder definir el tipo de un nuevo archivo externo.

2.- Cada tipo de archivo tendrá una herramienta asociada, que se le aplicará.

3.- Cada tipo de archivo se representará con un icono específico.

4.- El usuario deberá poder definir el icono que representa un tipo de archivo externo.

5.- Cuando el usuario selecciona un icono que representa un archivo externo, el efecto es aplicar la herramienta asociada con este tipo de archivo al archivo representado por el icono seleccionado.

Requisitos funcionales

Declaración de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares. Describen la funcionalidad del sistema, y dependen del tipo de software, del sistema a desarrollar y de los usuarios del mismo.

Por lo general se describen mejor a través del modelo de Casos de uso y los Casos de uso como tal. Por lo tanto los requerimientos funcionales especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto.

Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Restricciones que afectan a los servicios o funciones del sistema, tales como restricciones de tiempo, sobre el proceso de desarrollo, estándares, etc.

Clasificación de los requisitos no funcionales

Requisitos del producto: Especifican el comportamiento del producto obtenido, velocidad de ejecución, memoria requerida, y porcentaje de fallos aceptables.

Requisitos organizacionales: Son una consecuencia de las políticas y procedimientos existentes en la organización, procesos estándar utilizados, de fechas de entrega, y documentación a entregar.

Requisitos externos: Presentan factores externos al sistema y a su proceso de desarrollo, interoperabilidad del sistema con otros, requisitos, legales, y éticos.

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, etc.

Algunas propiedades de los requerimientos no funcionales que hacen al producto atractivo, usable, rápido o confiable, son las siguientes:

Requerimientos de apariencia o interfaz externa

Ejemplo: Muy legible, Simple de usar, Profesional o tipo ejecutivo.

Requerimientos de Usabilidad

Ejemplo: Facilidad de uso por personas que hablen otros idiomas distintos al del país donde el producto fue creado, Accesibilidad para personas discapacitadas, Consistencia en la interfaz de usuario, Documentación de usuario.

Requerimientos de Rendimiento

Ejemplo*:* Velocidad de procesamiento o cálculo, Eficiencia, Disponibilidad, Tiempo de respuesta.

Requerimientos de Soporte

Ejemplo*:* Adaptabilidad, Mantenimiento*.*

Requerimientos de Portabilidad

Ejemplo*: El producto podrá ser usado bajo el sistema operativo Linux*

Requerimientos de Seguridad

Confidencialidad: La información manejada por el sistema esta protegida de acceso no autorizado y divulgación.

Integridad: la información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes.

Disponibilidad: Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

Requerimientos de confiabilidad

Frecuencia y severidad de los fallos, Protección contra fallos, Recuperación, Predicción de fallos, Tiempo medio entre fallos.

Requerimientos de Software.

Ejemplo: Sistema Operativo Windows 95 o Superior; Maquina Virtual de Java versión 1.3 o Superior; etc.

Requerimientos de Hardware.

Ejemplo: se requiere disponer de un MODEM estándar o una tarjeta digitalizadora de video, etc.

A pesar de las diferentes características que nos brindan los requerimientos, existen dificultades para recolectar los requisitos, las cuales no nos permiten elegir los requerimientos con la calidad necesaria; ya que estos pueden relacionarse unos con otros y a su vez con otras partes del proceso. Pero aun así, se plantea que sin el levantamiento de requisitos no se podrían desarrollar procesos que son de vital importancia para el desarrollo del software. Los requisitos constituyen el enlace entre las necesidades reales de los clientes, usuarios y otros participantes vinculados al sistema.

1.2 ¿Qué se entiende por Ingeniería de Requisitos?

La Ingeniería de Requisitos es definida como:

La disciplina de la Ingeniería de Software que trata con actividades y intenta comprender las necesidades exactas de los usuarios del sistema software, para traducir tales necesidades en instrucciones precisas y no ambiguas las cuales podrían ser posteriormente utilizadas en el desarrollo del sistema. (Loucopoulos,1995)

Ingeniería de Requerimientos es el proceso en el cual se transforman los requerimientos declarados por los clientes , ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones. Es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. (Richard, 1997).

1.2.1 Ingeniería de Requisitos (IR). Sus Características

La Ingeniería de Requisitos en una disciplina de la Ingeniería de Software**,** en ésta, se identifica el propósito del sistema, dirección y alcance. Abarca un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada siguiendo un determinado estándar.

La Ingeniería de Requerimientos cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.

El proceso de Ingeniería de Requisitos tiene como objetivos, descubrir, modelar, validar y mantener un documento de requisitos, utilizando una combinación de métodos, herramientas y actores.

1.2.2 Actividades de la Ingeniería de Requerimientos.

Las actividades de la Ingeniería de Requisitos más comunes son:

- Estudio de Viabilidad
- Elicitación de Requisitos
- Análisis de Requisitos
- Validación de Requisitos
- Gestión de Requisitos

Estudio de viabilidad: El estudio de viabilidad permite decidir si el sistema propuesto es conveniente. Es un estudio rápido y orientado a conocer. Además tiene en cuenta si el sistema contribuye a los objetivos de la organización, si el sistema se puede realizar con la tecnología actual y con el tiempo y el coste previsto, y si el sistema puede integrarse con otros existentes.

Elicitación y análisis de requisitos: Elicitación (o extracción o determinación) de requisitos, es el proceso mediante el cual los usuarios descubren, revelan, articulan y comprenden los requisitos que desean. En esta etapa, se trata de descubrir los requisitos y personal técnico trabaja con los clientes y usuarios para descubrir el dominio de la aplicación, los servicios que se deben proporcionar y las restricciones. Puede implicar a usuarios finales, encargados, ingenieros implicados en el mantenimiento, expertos del dominio, etc. Son los llamados participantes (stakeholders).Ver figura 1.

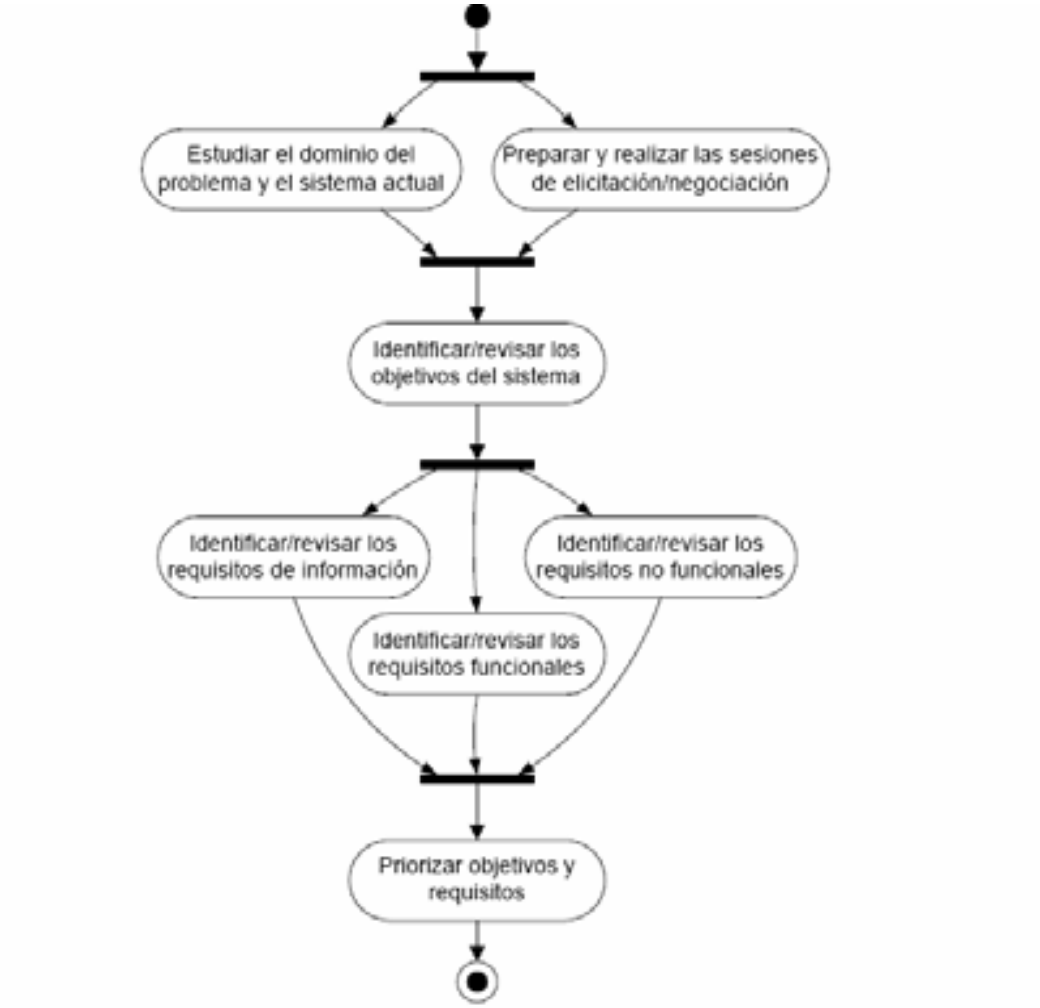


Figura 1. Las etapas en Elicitación de Requisitos

Análisis de requisitos: El proceso de razonamiento sobre los requisitos obtenidos en la etapa anterior, detectando y resolviendo posibles inconsistencias o conflictos, coordinando los requisitos relacionados entre sí, etc.

Validación de requisitos: El proceso de confirmación, por parte de los usuarios, de que los requisitos especificados son válidos, consistentes, y completos.

Gestión de Requisitos: Es el proceso de manejar los requisitos que cambian durante el desarrollo del sistema.

El proceso de Ingeniería de Requisitos se adapta a los diferentes modelos de procesos de Ingeniería de Software como pueden ser, de **cascada, espiral, prototipazo, transformacional,** etc. Ver figura 2.

Figura 2. Modelo Espiral del Proceso de la Ingeniería de Requisitos. (Sommerville, 1997)

Validación de Requisitos

La validación es la actividad de la IR que permite demostrar que los requerimientos definidos en el sistema son los que realmente quiere el cliente; además revisa que no se haya omitido ninguno, que no sean ambiguos, inconsistentes o redundantes.

En este punto es necesario recordar que la SRS debe estar libre de errores, por lo tanto, la validación garantiza que todos los requerimientos presentes en el documento de especificación sigan los estándares de calidad. No debe confundirse la actividad de evaluación de requerimientos con la validación de requerimientos. La evaluación verifica las propiedades de cada requerimiento, mientras que la validación revisa el cumplimiento de las características de la especificación de requisitos. Durante la actividad de validación pueden hacerse preguntas en base a cada una de las características que se desean revisar. La validación de requerimientos es importante pues de ella depende que no existan elevados costos de mantenimiento para el software desarrollado.

Evolución de los requerimientos

Los requerimientos son una manera de comprender mejor el desarrollo de las necesidades de los usuarios y cómo los objetivos de la organización pueden cambiar, por lo tanto, es esencial planear posibles cambios a los requerimientos cuando el sistema sea desarrollado y utilizado. La actividad de evolución es un proceso externo que ocurre a lo largo del ciclo de vida del proyecto. Los requerimientos cambian por diferentes razones. Las más frecuentes son:

- Porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas.
- Porque cambió el problema que se estaba resolviendo.
- Porque los usuarios cambiaron su forma de pensar o sus percepciones.
- Porque cambió el ambiente de negocios.
- Porque cambió el mercado en el cual se desenvuelve el negocio.

Cambios a los requisitos involucra modificar el tiempo en el que se va a implementar una característica en particular, modificación que a la vez puede tener impacto en otros requerimientos. Por esto, la administración de cambios involucra actividades como establecer políticas, guardar historial de cada requerimiento, identificar dependencias entre ellos y mantener un control de versiones. Tener versiones de los requerimientos es tan importante como tener versiones del código, ya que evita tener requerimientos emparchados en un proyecto.

Entre algunos de los beneficios que proporciona el control de versiones están:

- Prevenir cambios no autorizados.
- Guardar revisiones de los documentos de requerimientos.
- Recuperar versiones previas de los documentos.
- Administrar una estrategia de "releases".
- Prevenir la modificación simultánea a los requisitos.

En vista que las peticiones de cambios provienen de muchas fuentes, las mismas deben ser enrutadas en un solo proceso. Esto se hace con la finalidad de evitar problemas y conseguir estabilidad en los requerimientos.

1.2.3 Personal involucrado en la Ingeniería de Requerimientos.

Realmente, son muchas las personas involucradas en el desarrollo de los requerimientos de un sistema. Es importante saber que cada una de esas personas tienen diversos intereses y juegan roles específicos dentro de la planificación del proyecto; el conocimiento de cada papel desempeñado, asegura que se involucren a las personas correctas en las diferentes fases del ciclo de vida, y en las diferentes actividades de la IR.

No conocer estos intereses puede ocasionar una comunicación poco efectiva entre clientes y desarrolladores, que a la vez traería impactos negativos tanto en tiempo como en presupuesto. Los roles más importantes pueden clasificarse como sigue:

Usuario final: Son las personas que usarán el sistema desarrollado. Ellos están relacionados con la usabilidad, la disponibilidad y la fiabilidad del sistema; están familiarizados con los procesos específicos que debe realizar el software, dentro de los parámetros de su ambiente laboral. Serán quienes utilicen las interfaces y los manuales de usuario.

Usuario Líder: Son los individuos que comprenden el ambiente del sistema o el dominio del problema en donde será empleado el software desarrollado. Ellos proporcionan al equipo técnico los detalles y requerimientos de las interfaces del sistema.

Personal de Mantenimiento: Para proyectos que requieran un mantenimiento eventual, estas personas son las responsables de la administración de cambios, de la implementación y resolución de anomalías. Su trabajo consiste en revisar y mejorar los procesos del producto ya finalizado.

Analistas y programadores: Son los responsables del desarrollo del producto en sí; ellos interactúan directamente con el cliente.

Personal de pruebas: Se encargan de elaborar y ejecutar el plan de pruebas para asegurar que las condiciones presentadas por el sistema son las adecuadas. Son quienes van a validar si los requerimientos satisfacen las necesidades del cliente.

Otras personas que pueden estar involucradas, dependiendo de la magnitud del proyecto, pueden ser: administradores de proyecto, documentadores, diseñadores de base de datos, entre otros.

1.2.4 Análisis comparativo de las técnicas de ingeniería de requerimientos.

En la Ingeniería de Requisitos se describen técnicas que permiten la captura requisitos de software, que permiten la recopilación de la información y en qué casos es adecuada usar cada cual. A continuación se hace un análisis de estas técnicas. (Sommerville, 1997)

Técnica: Entrevistas. Características.

- Forma de conversación, no de interrogación.
- Ocupan un lugar preponderante de acuerdo al tiempo que ocupan y el objetivo que tienen.
- Mayor fuente de información del analista.
- Basadas en un cuestionario rígido o una guía que las orienta hacia puntos bien definidos.

Ventajas

- Se presenta necesidades de forma directa y se verifica si las preguntas fueron interpretadas correctamente.
- Oportunidad para conocer el grado de aceptación o no entre los usuarios hacia el sistema que se desea diseñar**.**

- Mediante ellas se obtiene una gran cantidad de información correcta a través del usuario.
- Pueden ser usadas para obtener un pantallazo del dominio del problema.
- Son flexibles.
- Permiten combinarse con otras técnicas.

Desventajas

- La información obtenida al principio puede ser redundante o incompleta.
- Si el volumen de información manejado es alto, requiere mucha organización de parte del analista, así como la habilidad para tratar y comprender el comportamiento de todos los involucrados.

Realización de las Entrevistas

Los pasos:

- Preparación
- Ejecución
- Recapitulación

¿Cómo lograr una entrevista exitosa?

- Acordar una cita por anticipado con las personas que se entrevistarán.
- Avisar a los entrevistados sobre la naturaleza de la entrevista.
- Planear una entrevista común por no más de una hora
- Prepararla conociendo de antemano a los individuos que se van a entrevistar.
- Familiarizarse con el tema y preparar un conjunto apropiado de preguntas.

Durante la entrevista:

- Presentarse, subrayando el tema y la naturaleza del proyecto.
- Comenzar con preguntas generales que establezcan el marco de trabajo

- Continuar con los temas y aspectos que surjan de quienes responden.
- Asegurarse de encontrar por qué quienes responden creen que es tan importante el tema como para comentarlo
- Cuando todos los temas vistos se hayan discutido, realícense otras preguntas específicas que se crea deban discutirse.
- Al finalizar, resumir la información recabada durante la misma.
- Si se considera apropiado, indicar que se preparará un resumen escrito de la entrevista.
- Considerar la posibilidad de continuar con la entrevista en otro momento**.**

Técnica: Cuestionarios. Características

Permiten obtener información de un gran número de personas en corto tiempo, sin que estas deban estar presentes.

Son recomendables cuando:

- Se requiere una pequeña cantidad de información de un gran número de personas en un corto periodo de tiempo.
- La información se desea consolidar en tablas estadísticas.
- Usuarios geográficamente dispersos

¿Cómo desarrollar un Cuestionario?

- Determinar qué datos se necesitan y qué personas están calificadas para proporcionarlos.
- Seleccionar el tipo de cuestionario (abierto, cerrado).
- Incluir preguntas redundantes, cuando sea necesario, para verificar consistencia.

Examine el cuestionario para detectar errores en preguntas que:

- Puedan ser mal interpretadas.
- No se puedan responder.
- Se interpretarán en forma diferente dependiendo de cada entrevistado.
- No proporcionan opciones adecuadas de respuesta.
- No estén ordenas adecuadamente

Desventajas

- Información suministrada por escrito.
- Los encuestados pueden objetar preguntas, interpretarlas a su forma o no tomarlas en serio.
- Difíciles de diseñar.

Antes de aplicar un Cuestionario:

- Probarlo en un grupo pequeño para detectar otros problemas.
- Analizar las respuestas de prueba para asegurar que el análisis se pueda llevar a cabo con los datos recopilados.

Técnica: Lluvia de Ideas

Ventajas

- Los diferentes puntos de vista y las confusiones en cuento a terminología, son aclaradas por expertos.
- Ayuda a desarrollar ideas unificadas basadas en la experiencia de un experto.

Desventaja

- Es necesaria una buena compenetración del grupo participante.

Técnica: Prototipos

El uso de prototipos para recoger requisitos o comprobar si se han entendido perfectamente es una práctica cada vez más extendida, especialmente en sistemas que suponen un elevado grado de interactividad.

En este caso los prototipos a evaluar no serán mas que maquetas no operativas o especificaciones formales que un grupo de expertos deberán evaluar.

Ventajas

- Ayudan a validar y desarrollar nuevos requerimientos.
- Permite comprender aquellos requerimientos que no están muy claros y que son de alta volatilidad.

Desventajas

- El cliente puede llegar a pensar que el prototipo es una versión del software que será desarrollado.
- A menudo, el desarrollador hace compromisos de implementación con el objetivo de acelerar la puesta en funcionamiento del prototipo

Técnica: Análisis Jerárquico

Ventajas

- Permite determinar el grado de importancia de cada requerimiento.
- Ayuda a identificar conflictos en los requerimientos.
- Muestra el orden en que deben ser implementados los requerimientos.

Desventaja

- Debe construirse un estándar claro de evaluación, que incluya la participación del cliente.

Técnica: Casos de Uso

Ventajas

- Representan los requerimientos desde el punto de vista del usuario.
- Permiten representar más de un rol para cada afectado.
- Identifica requerimientos estancados, dentro de un conjunto de requerimientos.

Desventaja

- En sistemas grandes, toma mucho tiempo definir todos los casos de uso.
- El análisis de calidad depende de la calidad con que se haya hecho la descripción inicial.

Técnica: Estudio

Estudio de documentación: En esta técnica se estudia documentación o estándares que puedan informar sobre las actividades de las tareas a realizar, puesto que en muchas

ocasiones algunos procedimientos ya están sujetos a algún tipo de regulación que es preciso tener en cuenta.

Estudio de la literatura: Otra valiosa fuente de información, especialmente adecuada si el equipo de desarrollo no tiene mucha experiencia en el dominio de aplicación del producto, es buscar en la literatura ejemplos de productos similares.

En base a las ventajas y desventajas mostradas anteriormente, se hace una comparación entre algunas de las técnicas.

Entrevistas vs. Casos de Uso: Un alto porcentaje de la información recolectada durante una entrevista, puede ser usada para construir casos de uso. Mediante esto, el equipo de desarrollo puede entender mejor el ambiente de trabajo de los involucrados. Cuando el analista sienta que tiene dificultades para entender una tarea, pueden recurrir al uso de un cuestionario y mostrar los detalles recabados en un caso de uso. De hecho, durante las entrevistas cualquier usuario puede utilizar diagramas de casos de uso para explicar su entorno de trabajo.

Entrevistas vs. Lluvia de Ideas: Muchas de las ideas planteadas en el grupo, provienen de información recopilada en entrevistas o cuestionarios previos. Realmente la lluvia de ideas trata de encontrar las dificultades que existen para la comprensión de términos y conceptos por parte de los participantes; de esta forma se llega a un consenso.

Casos de Uso vs. Lluvia de Ideas: La lista de ideas proveniente del brainstorm puede ser representada gráficamente mediante casos de uso.

Las Técnicas de la Ingeniería de Requisitos son de gran importancia, nos permiten conocer las diferentes alternativas que existen para identificar requerimientos.

1.2.5 Gestión de Riesgos en la Ingeniería de Requerimientos

Para evitar el riesgo hay que definir las estrategias necesarias para que este no se produzca y tomar las medidas encaminadas para que, aún cuando se produzca, se minimicen sus efectos. Para el monitoreo del riesgo hay que definir los indicadores que influyen en la probabilidad de que este se produzca y revisar periódicamente dichos factores, además de vigilar la efectividad real de las acciones encaminadas a evitarlo. (Charette, 1989)

La gestión del riesgo y plan de contingencia asumen que la evitación y la monitorización han fallado y el riesgo se ha producido. Por ello se definen las estrategias y acciones a tomar para lograr que los efectos se minimicen. Nunca se podrá reducir a cero el coste del plan de contingencia, ya que él puede implicar algunos costes en sí mismo, por lo cual se ha de valorar el beneficio que se espera obtener de éste.

Un riesgo es aquel factor que influye negativamente en el éxito del proyecto. El riesgo en un proyecto de desarrollo de software incluye componentes técnicos y de conocimiento del mismo. Los temas de naturaleza organizacional constituyen los factores dominantes de los riesgos del proyecto, a la vez que son los que se tratan satisfactoriamente en menos de la tercera parte de los proyectos de desarrollo, entre ellos los conflictos entre departamentos, entre usuarios, el cambio del responsable ejecutivo del proyecto,

volatilidad del personal, número de unidades de la organización implicadas y proyectos que involucran a múltiples proveedores.

El riesgo acompaña a todo cambio porque implica elección e incertidumbre. Si a la vez que se inicia la actividad de elicitación de los requisitos del software a construir, se inicia la identificación de los riesgos asociados a los requisitos individuales y a grupos de ellos, será posible gestionarlos tempranamente para minimizarlos, evadirlos y controlarlos. El jefe o administrador de proyectos anticipa riesgos que pueden afectar al desarrollo o a la calidad de los requisitos y emprende acciones para evitarlos.

Esta actividad garantiza que, desde el inicio del proceso de desarrollo del software, se realicen las tareas encaminadas a garantizar la calidad del producto.

En (Pressman, 2002), se describe los siguientes procedimientos:

***Paso 1**: Identificación de riesgos*. Problemas potenciales que pueden ocurrir en el proceso de IR o en los requisitos, o en la Especificación de los Requisitos del Software (ERS), como de presupuesto, de personal, del usuario, de organización, técnicos, de comunicación u otros. Debe comenzar con el análisis de los riesgos genéricos, que constituyen una amenaza potencial para todo el proyecto de software. Después se deben identificar los riesgos específicos, que implican un conocimiento profundo del proyecto, y están relacionados con el entorno de desarrollo, la tecnología, la experiencia y el tamaño del equipo.

Para los requisitos y ERS se debe comenzar esta tarea dando respuesta a la pregunta: ¿qué características especiales tiene este requisito, o grupo de ellos, que pueden estar amenazadas?

Un método probadamente efectivo para identificar riesgos es la creación de una *lista de comprobación de elementos de riesgo*. Esta lista debe centrarse en los riesgos relacionados con: tamaño del producto, impacto en el proyecto y en la organización, características del cliente, definición del proceso, el entorno de desarrollo, la tecnología a construir, el tamaño del equipo y la experiencia del personal.

Los riesgos serán considerados a partir del comportamiento de sus características individuales y grupales (ambigüedad, claridad, completitud, consistencia, rastreabilidad, entre otras.). En dependencia de la intensidad de ese comportamiento de la característica de calidad del requisito, podrá controlarse de forma efectiva el riesgo en el propio proceso de IR, antes de que pase a la siguiente etapa del ciclo de vida del proyecto de desarrollo

del software. De no gestionarse el riesgo en esta etapa inicial, puede hacerse difícil su control efectivo una vez iniciado el proceso de desarrollo.

Desequilibrios en el comportamiento de las características de calidad de los requisitos se traducen en que la aparición de riesgos haga vulnerable el proyecto e impráctica la aplicación de cualquier plan de calidad, razón por la cual conviene su detección y tratamiento temprano. En esta situación, se recomienda la redefinición de los requisitos afectados (o la ERS), que es posible por iteraciones en este procedimiento.

Paso 2*: Evaluación de los riesgos.* Determinar en qué indicador se verá reflejado que un problema se presente, se deben establecer puntos de referencia para cada riesgo, que permita decidir si el riesgo, según su prioridad de atención, se sale del manejo aceptable.

Proyección de los riesgos. Consiste en determinar la probabilidad de que un riesgo ocurra y las consecuencias que puede tener, por ejemplo: incremento de costos, cancelación del proyecto, insatisfacción del cliente. Implica ordenar la lista de riesgos teniendo en cuenta la probabilidad de que ocurra y el impacto de cada riesgo. Se asigna el nivel de probabilidad, que puede ser alta, media o baja. Se valora el impacto (consecuencias) en cuanto al alcance (cuánto se afecta) y la duración (por cuánto tiempo se manifiesta).

Paso 3: *Planificación de riesgos.* Este paso tiene como objetivo desarrollar una estrategia para tratar los riesgos. Si el equipo de trabajo adopta un enfoque proactivo frente al riesgo, evitarlo será siempre la mejor estrategia. Esto se consigue desarrollando los planes de reducción del riesgo y de contingencia.

En la planificación de riesgos se considera cada uno de los riesgos claves identificados y las estrategias para administrarlos, que vendrán dadas por el juicio y la experiencia del administrador del proyecto.

Las estrategias de anulación intentan reducir la probabilidad de que surja el riesgo, las estrategias de disminución: intentan reducir el impacto del riesgo. Los planes de contingencia se elaboran para estar preparados por si el riesgo ocurre poder actuar con una estrategia determinada.

Paso 4: *Supervisión de los riesgos*. Consiste en hacer el plan de supervisión, dado el caso de que se acepte continuar con el proyecto. Indicar que acciones y decisiones se tomarán ante un problema que ya ha sido identificado, proyectado y evaluado.La supervisión de riesgos valora cada uno de los riesgos identificados para decidir si es más

o menos probable y cuándo han cambiado sus posibles efectos. Hay que controlar factores que pueden indicar cambios en la probabilidad y el impacto.

- La tarea principal del administrador consiste en minimizar riesgos. Involucrar a todos los implicados/afectados y al equipo de desarrollo del proyecto en la identificación de los riesgos.
- El riesgo inherente en una actividad se mide en base a la incertidumbre que presenta el resultado de esa actividad. Las actividades con alto riesgo elevan los costos.
- Comunicar los riesgos a todos los niveles de la organización.
- El riesgo es proporcional al monto de la calidad de la información disponible. Cuanto menos información, mayor el riesgo.
- Monitorear constantemente los factores que propician la materialización de los riesgos y la efectividad de las acciones definidas encaminadas a su prevención y/o minimización.
- Definir plantillas o tablas de riesgos valorados para diferentes tipos de proyecto que puedan servir como punto de partida para un nuevo proyecto.
- Las salidas de esta actividad son las listas (tablas o taxonomías) de riesgos en sus diferentes acepciones, el plan de supervisión de riesgos y el plan de contingencia.

Un adecuado proceso de ingeniería de requisitos tiene implicaciones positivas en la calidad del producto final, por ende, en la satisfacción del cliente. Debido a esto el proceso de IR tiene que estar bien definido y ser desarrollado de forma disciplinada, coherente y repetitiva, garantizando la obtención de experiencias que permitan aplicar las mejores prácticas.

El tratamiento proactivo de los riesgos asociados a los requisitos del software permite al gestor adoptar, desarrollar e implementar adecuadamente las actividades de gestión de estos, en función de obtener productos de calidad que satisfagan las necesidades del cliente, manteniendo el equilibrio de plazo y costo del proyecto en virtud de lograr un mejor desempeño del proceso de IR en la pequeña y mediana empresa de software. El manejo de los riesgos asociados a los requisitos, organizados y gestionados a través de las diferentes taxonomías propuestas puede constituirse en una útil herramienta para los gestores y equipos de desarrollo.

1.2.6 Importancia de la Ingeniería de Requerimientos.

Los principales beneficios que se obtienen de la Ingeniería de Requerimientos son:

- **Permite gestionar las necesidades del proyecto en forma estructurada**: Cada actividad de la IR consiste de una serie de pasos organizados y bien definidos.
- **Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados**: La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- **Disminuye los costos y retrasos del proyecto**: Muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro.
- **Mejora la calidad del software**: La calidad en el software tiene que ver con cumplir un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).
- **Mejora la comunicación entre equipos**: La especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.
- **Evita rechazos de usuarios finales**: La ingeniería de requerimientos obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto.

La ingeniería de requerimientos es una de las disciplinas de la Ingeniería de Software de mayor importancia pues la misma depende de una intensa comunicación entre clientes y analistas de requerimientos. La Ingeniería se encarga de establecer y mantener un acuerdo en qué el sistema debe hacer. Proporciona al equipo de desarrollo un entendimiento de los requisitos, hasta definir los límites del sistema. Además se controlan los cambios a los requisitos.

1.3 Gestión de Requisitos. Principales características.

La Gestión de Requisitos es un componente vital en el desarrollo de un proyecto software ya que es una de las actividades de la Ingeniería de Requisitos más importantes. Los requisitos se inician cuando empieza un proyecto en las etapas de análisis y especificación de requisitos, posteriormente, dichos requisitos en el ciclo de vida de un proyecto pueden ser modificados por lo que se establece el concepto de Gestión de Requisitos, que es el tratamiento y control de las actualizaciones y cambios a los mismos.

Debido a que un proyecto informático es susceptible de cambios, habría que proceder a su actualización o a la incorporación de nuevas funcionalidades o eliminar otras, esto obliga a mantener controlado y documentado el producto. Los cambios de requisitos deben ser gestionados para asegurar que la calidad de los mismos se mantenga, los problemas suscitados por los cambios de requisitos podrían incurrir en altos costos, siendo el requisito factor crítico de riesgo.

Más formalmente el **Manejo de Requisitos** es una forma sistemática de descubrir, organizar y documentar los requisitos del sistema. Además es el proceso que establece y mantiene un consenso entre el cliente y el grupo del proyecto en el cambio de los requisitos del sistema.

El término Gestión de Requisitos incluye:

- ✓ Técnicas para Descubrimiento/Recogida de Requisitos.
- ✓ Recoger las peticiones del usuario y determinar las verdaderas necesidades de éste.
- ✓ Técnicas de Análisis
- ✓ Especificación y verificación
- ✓ Manejo de Requisitos

1.3.1 Tareas principales de la Gestión de Requisitos.

Durante el proceso de la gestión de requisitos, hay que planear algunas actividades, dentro de las que se pueden mencionar las siguientes: la identificación de los requisitos, en proceso de gestión de los cambios, las políticas de trazabilidad, la cantidad de información sobre las relaciones entre los requisitos que se mantiene, entre otras.Ver figura 3.

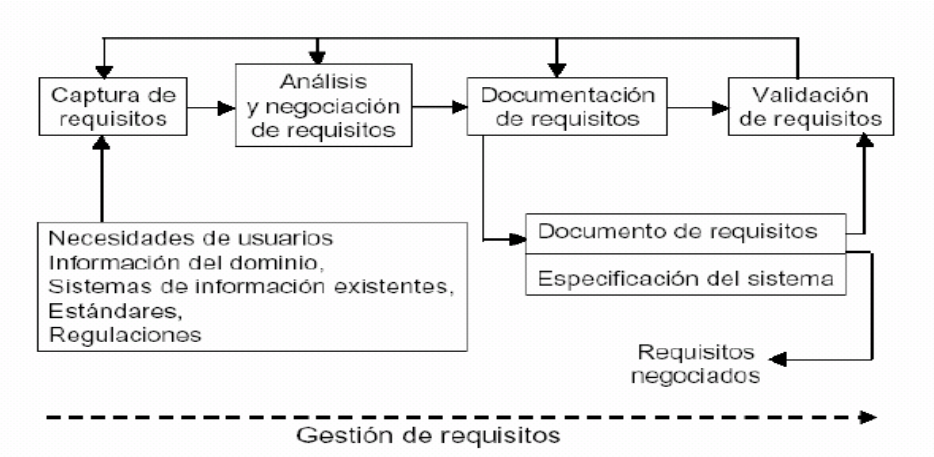


Figura3. Gestión de Requisitos.

Actividades y su descripción:

Recolección: Recolección y documentación de requisitos es una actividad de comunicación iterativa entre clientes, gerentes y practicantes, para descubrir, definir, refinar y registrar una representación precisa de los requisitos del producto. Varios métodos son utilizados para la recolección de requisitos. Algunos análisis iniciales como es la agrupación, categorización, priorización son desarrollados durante esta actividad.

Documentación: Después que los requisitos han sido recolectados, hay que analizarlos a detalle y documentarlos en una especificación de requisitos. El resultado de la especificación de requisitos y de cualquier especificación de requisitos de componentes hardware/software derivado sirve como registro de convenio con el cliente y compromiso con el proveedor. Estas especificaciones son rastreadas utilizando una matriz de trazabilidad de requerimientos y son sujetos a verificación y gestión de cambio a través del ciclo de vida del producto.

Verificación: Una vez que la especificación de requisitos ha sido desarrollada, los requisitos son verificados. La verificación de requisitos es un proceso para asegurar que la especificación de requisito del producto es una representación exacta de las necesidades del cliente. Este proceso también asegura que los requisitos sean trazados y verificados a través de varias fases del ciclo de vida; particularmente en el diseño, implementación y pruebas. Además, todos estos requerimientos deben ser trazados al diseño, implementación y pruebas para asegurarse que los requerimientos han sido satisfechos.

Gestión de Cambios: Gestión de cambios es un proceso formal para identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto. Como el proyecto va evolucionando, los requerimientos pueden cambiar o expandirse para ajustar algunas modificaciones en el alcance o diseño del proyecto. Un proceso de gestión de cambios proporciona un rastreo completo y preciso de todos los cambios que son pertinentes al proyecto.

El proceso del ciclo de vida de la Gestión de Requisitos, debe ser flexible y adaptable para reunir las necesidades del proyecto. Las características del alcance e implementación del proceso del ciclo de vida de la Gestión de Requisitos en un proyecto, variará dependiendo de algunos factores claves.

- Tamaño y complejidad del proyecto
- Experiencia del personal del proyecto
- Experiencia de los clientes del proyecto
- Dominio de la aplicación
- El propósito y uso de esta aplicación

1.3.2 Especificación de Requisitos de Software

La especificación de requisitos de software es la actividad en la cual se genera el documento, con el mismo nombre, que contiene una descripción completa de las necesidades y funcionalidades del sistema que será desarrollado; describe el alcance del

sistema y la forma en como hará sus funciones, definiendo los requerimientos funcionales y los no funcionales. En la SRS se definen todos los requerimientos de hardware y software, diagramas, modelos de sistemas y cualquier otra información que sirva de soporte y guía para fases posteriores.

Es importante destacar que la especificación de requisitos es el resultado final de las actividades de análisis y evaluación de requerimientos; este documento resultante será utilizado como fuente básica de comunicación entre los clientes, usuarios finales, analistas de sistema, personal de pruebas, y todo aquel involucrado en la implementación del sistema.

Los clientes y usuarios utilizan la SRS para comparar si lo que se está proponiendo, coincide con las necesidades de la empresa. Los analistas y programadores la utilizan para determinar el producto que debe desarrollarse. El personal de pruebas elaborará las pruebas funcionales y de sistemas en base a este documento. Para el administrador del proyecto sirve como referencia y control de la evolución del sistema. La SRS posee las mismas características de los requerimientos: completa, consistente, verificable, no ambigua, factible, modificable, rastreable, precisa, entre otras.

Para que cada característica de la SRS sea considerada, cada uno de los requerimientos debe cumplirlas; por ejemplo, para que una SRS se considere verificable, cada requerimiento definido en ella debe ser verificable; para que una SRS se considere modificable, cada requerimiento debe ser modificable y así sucesivamente. La estandarización de la SRS es fundamental pues ayudará, entre otras cosas, a facilitar la lectura y escritura de la misma. Será un documento familiar para todos los involucrados, además de asegurar que se cubran todos los tópicos importantes. Existen plantillas creadas para la SRS, sin embargo, cada uno tiene la potestad de crear su propia plantilla.

Especificación de requisitos del sistema (SyRS)

De acuerdo con el estándar IEEE 12207.0, el SyRS debe incluir: funciones y capacidades del sistema; requisitos del negocio, organizativos y de usuario; requisitos de seguridad, seguridad a terceros y privacidad; requisitos de ingeniería de factores humanos; requisitos de operaciones y mantenimiento; y restricciones de diseño.

A partir de estas características, el estándar IEEE 12207.1 , detalla el contenido específico de la plantilla del SyRS haciendo referencia al estándar IEEE Std 1233 ,Guide

for Developing System Requirements Specifications, donde se pueden encontrar guías para especificar los requisitos del sistema y un esquema del SyRS.

Especificación de requisitos del software (SRS)

La mayoría de los requisitos del software se obtienen directamente a partir de los requisitos del sistema. El SRS se basa en el estándar IEEE Std 830-1998 y la plantilla VOLERE en las que se recogen requisitos acerca de la funcionalidad del sistema, interfaces externas, rendimiento, restricciones de diseño y atributos del software (portabilidad, mantenimiento, seguridad, disponibilidad y fiabilidad).

Especificación de pruebas de sistema y software (SyTS y STS)

Para poder validar los requisitos, estos deben cuantificarse en el SyRS y SRS. En los documentos de pruebas (SyTS o STS) se especificarán criterios de prueba para asegurar que el sistema o el software cumplen los requisitos especificados. Este plan incluirá una lista de cuestiones relacionadas con la seguridad (en cuanto al personal, organización o funciones) que, después, podrían comprobarse fácilmente.

Especificación de requisitos de interfaz (IRS)

Los requisitos relacionados con los interfaces entre los elementos del software y entre el usuario y el sistema podrían incluirse en el SRS. No obstante, con el objeto de no producir documentos muy extensos, en algunas ocasiones será conveniente hacer uso de un documento separado denominado IRS. Por tanto, se deben establecer las relaciones de trazabilidad adecuadas entre el SyRS, el SRS y las interfaces descritas. La plantilla del IRS tiene la misma estructura que el apartado del SRS dedicado a la especificación de requisitos de interfaz. Ver figura 4

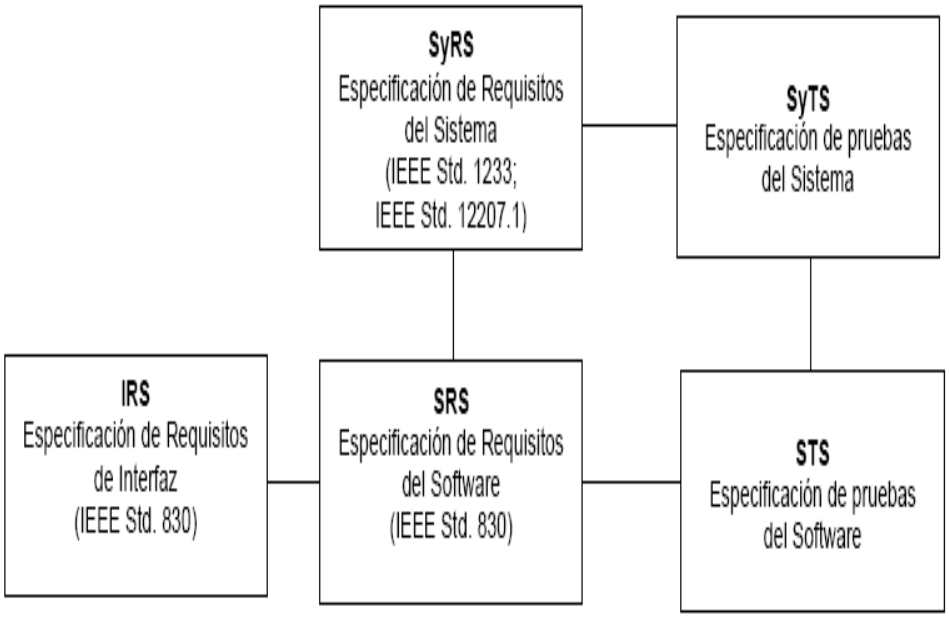


Figura 4.Especificación de Requisitos.

1.3.3 Trazabilidad de Requisitos

La existencia de distintos tipos de requisitos y la derivación de unos a partir de otros, hace necesario que se establezcan mecanismos de trazabilidad. Además los requisitos no son independientes entre si, para poder determinar el impacto de un cambio en un requisito, las relaciones entre requisitos deben ser entendidas, documentadas y mantenidas.

Los mecanismos básicos para mantener la trazabilidad son las **matrices de seguimiento**. En ellas se muestra la relación de un requisito de un tipo con un requisito de otro tipo. Las trazas son direccionales y se establecen como un atributo más de los requisitos

La trazabilidad de requisitos es un proceso clave para la exitosa gestión de los requisitos de un sistema de información. A pesar de ello, no existe un consenso acerca de los tipos de especificaciones y enlaces entre éstos que deben usarse en un proceso de trazabilidad. Además, a pesar de la existencia de herramientas específicas para la

gestión de requisitos, éstas no proveen mecanismos adecuados para la configuración de la trazabilidad de acuerdo con las necesidades específicas del proyecto.

La trazabilidad de requisitos se define como la habilidad para describir y seguir la vida de un requisito en ambos sentidos, hacia sus orígenes o hacia su implementación, a través de todas las especificaciones generadas durante el proceso de desarrollo de software.

A continuación se indica la información asociada a la trazabilidad de requisitos y sus posibles usos. (Ramesh, 2001)

Las actuales de herramientas para la trazabilidad de requisitos son muy excasas, proviniendo mayoritariamente del ámbito universitario. Entre las más importantes se encuentra TOOR (Traceability of Object-Oriented Requirements). Dicha herramienta presenta un tratamiento de artefactos especialmente orientado a la gestión de requisitos textuales. En (Ramesh,2001), propone un metamodelado para Trazabilidad de Requisitos.

SharpTrace, es una herramienta que implementa mediante un perfil UML una propuesta de metamodelo de trazabilidad presentada en (Letelier, 2002). En dicho metamodelo, teniendo en cuenta el extenso uso de UML como notación OO, se marca como objetivo la integración de artefactos UML y no-UML. (Anaya, 2002)

SharpTrace se beneficia de los mecanismos de extensión UML, permitiendo adaptar el marco de trazabilidad a las necesidades específicas del proyecto. Con tal finalidad, la herramienta proporciona un proceso de configuración de trazabilidad compuesta de un conjunto de tareas que guían al usuario. *SharpTrace* permite definir nuevos tipos de artefactos y enlaces de trazabilidad. La herramienta es independiente del proceso de desarrollo empleado.

Para demostrar la adaptabilidad e independencia respecto al proceso de desarrollo de software, en el artículo se ha mostrado el uso de *SharpTrace* en el desarrollo de una tienda de venta online siguiendo, alternativamente, un proceso de desarrollo RUP o XP. Se ha ilustrado como *SharpTrace* es fácilmente adaptable a las necesidades específicas del proyecto. En el ejemplo se muestran los mecanismos para establecer enlaces de trazabilidad entre artefactos de una forma intuitiva y guiada gracias al profile UML de trazabilidad.

SharpTrace dota a Rational Rose de mecanismos para crear tipos de artefactos no UML y trabajar junto con los UML en un mismo repositorio, proporcionando un único contexto.

1.4 Las Herramientas de Gestión de Requisitos

El uso de herramientas de la gestión de requisitos es alentado para mejorar tanto la productividad como la calidad en el desarrollo de un proyecto software. Existen varias herramientas tanto hechas en casa como en el mercado que auxilian a las tareas de gestión.

Rational RequisitePro, es una herramienta centrada en documentos, que almacena los requisitos asociándolos a documentos (aunque también permite guardarlos directamente en la base de datos), mientras que las otras herramientas están orientadas a requisitos.

Se auxilia especialmente en el control de cambio de requisitos, con trazabilidad para especificaciones de software y pruebas. La herramienta permite el uso de Oracle sobre Unix o Windows y también soporta SQL Server sobre Windows.

Beneficios de RequisitePro

- Permite el trabajo en equipo por medio de un repositorio compartido de información.
- Permite la clasificación de requerimientos, en base a las necesidades de cada empresa.
- Define atributos para todos los tipos de requerimientos especificados.
- Ayuda a manipular el alcance del proyecto mediante la asignación de prioridad de desarrollo a cada uno de los requerimientos planteados.
- Permite características avanzadas de rastreo, por medio de matrices, que permiten visualizar las dependencias entre requerimientos dentro de un proyecto o en diferentes proyectos.
- Administración de cambios mediante el rastreo y la visualización histórica de los cambios efectuados al requerimiento, cuándo y quién los realizó.

- Manejo de plantillas creadas por el usuario, o creadas por otras empresas.
- Interactúa con los demás productos Rational para el ciclo de vida, así como con herramientas de Microsoft Office.
- Ayuda a determinar en forma automatizada cuántos requerimientos tiene el proyecto.
- Ayuda a determinar responsables y actores en cada uno de los requerimientos.
- RequisitePro, permite organizar los requerimientos, establecer y mantener relaciones padre/hijo entre ellos.

Visual Paradigm es una herramienta CASE que da soporte al modelado visual con UML 2.0, permitiendo representar gráficamente el sistema software, resaltando los detalles más importantes. Se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos, de esta forma mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.

Beneficios de Visual Paradigm

Visual Paradigm ofrece:

- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.

El uso de una herramienta de gestión de requisitos proporciona al proyecto:

- Ahorro en costes de especificación y de desarrollo minimizando el impacto de errores.
- Mejora la calidad mediante un adecuado análisis y gestión de los requisitos.
- Facilita la reutilización real.
- Mejora la productividad facilitando la reutilización real desde la especificación.
- Reduce las no-conformidades del sistema.
- Permite controlar la especificación.
- Permite administrar más fácilmente la especificación.
- Ayuda a cumplir con estándares de calidad.
- Permite centralizar toda la información del problema.
- Permite especificar sistemas de una forma estructurada y gráfica.
- Proporciona una trazabilidad completa de la especificación, etc.

Otras Herramientas de la Gestión de Requisitos en el Mercado

Las herramientas seleccionadas proporcionan casi todas las necesidades básicas exigibles. Además, estas herramientas están ampliamente difundidas y son muy reconocidas.

Todas las herramientas asumen que la estructura de los requisitos es jerárquica, de forma que un requisito puede estar formado o tener asociados otros requisitos de nivel inferior, y la mayoría permite extraer párrafos de ficheros generados por procesadores de texto comerciales y convertirlos en requisitos. Otras de las características comunes a la mayor parte de las herramientas es la posibilidad de realizar consultas sobre los requisitos en función de determinados valores de sus atributos.

Estas herramientas son: IRqA 3.0, CaliberRM y DOORS

IRqA 3.0 es una herramienta de ingeniería de requisitos especialmente diseñada para soportar el proceso completo de ingeniería de requisitos. En IRqA el ciclo de especificación completo incluye la captura de requisitos, análisis, especificación de sistema, validación y la organización de requisitos es soportada por modelos estándares. (Lan A, 1994)

CaliberRM es para sistemas grandes y complejos y proporciona una base de datos de requisitos con trazabilidad. Se ve a los requisitos como parte del proceso al igual de gestión de la calidad del software, las pruebas (testing) y el trazado de defectos (defect tracking). Caliber está basado en internet y maneja referencia de documentos, responsabilidad de usuario, trazabilidad, prioridad y estado entre otras características.

DOORS a diferencia del resto de las herramientas, considera los requisitos como objetos y los documentos como módulos. Tiene una orientación basada en objetos, frente a RequisitePro y Caliber-RM, que manejan solamente requisitos y sus atributos. Es una herramienta para organizaciones grandes que necesitan controlar complejos conjuntos de usuarios y requisitos de sistemas con una completa trazabilidad. Proporciona buena visualización de tales documentos como jerárquicas, y su lenguaje de extensión permite una gran variedad de soporte de herramientas a ser construidas.

1.4.1 Funcionalidades de las Herramientas de Gestión de Requisitos

Estas herramientas permiten a los desarrolladores del sistema importar grandes documentos de una variedad de formatos estándar de procesadores de palabras. Los elementos del documento están sujetos a rigurosos cambios y a un régimen de control de versiones. Se puede establecer una relación entre los elementos del documento, y los atributos pueden ser asociados con los elementos del sistema y a menudo relacionados.

Pueden ser generados una variedad de vistas de documentos utilizando tanto los atributos como las relaciones, generalmente vistas específicas de trazabilidad tales como matrices de trazabilidad. De la misma manera, plantillas de documentos pueden ser configuradas para crear nuevos documentos compuestos.

Las herramientas de gestión de requisitos son genéricas, esto es que necesitan ser configurados para soportar ingeniería de requisitos específicos y procesos de desarrollo de sistemas. Dichas configuraciones son soportadas por la creación de plantillas de documentos, esquemas/diseño de atributos y tipos de relación y vistas de documentos. La mayoría de las herramientas son vendidas incluyendo algunos procesos tales como aquellos que son establecidos por el estándar IEEE. (Finkelstein, 2000)

Funciones Básicas

Se han identificado las siguientes características básicas necesarias para una herramienta para ser considerada como una herramienta de Gestión de Requisitos:

- Identificación de requisitos "individuales"
- Asignación a un destino y clasificación de requisitos
- Grupo de requerimientos (recopilación), revisión, identificación / punto de arranque, proveer una interfaz de datos básicos.

Una herramienta de Gestión de Requisitos puede soportar la disciplina de ingeniería y la disciplina de gestión para gestionar requisitos, así mismo, una herramienta debe poder coleccionar y gestionar los requisitos técnicos y programáticos. Funciones comunes realizadas por la herramienta, consisten en la identificación de requisitos, revisión y edición, rastreo de requisitos a su origen, y generación de informes. Funciones técnicas requeridas por las herramientas incluye análisis del impacto del cambio. Cuando un requisito es cambiado, se deben de identificar todos los requisitos afectados.

Otra función beneficiosa que debe ser utilizada es la verificación de la integridad y la consistencia. La función de gestión requerida por las herramientas consiste en la recopilación de métricas y supervisión de la estabilidad de los requerimientos a través del control de cambio, donde este último, consiste en mantener la pista de añadir, borrar o cambiar cualquier requisito existente.

Funciones Auxiliares

Algunas herramientas incluyen como funciones auxiliares, análisis funcional y capacidades de simulación, integradas a la interfaz, o como un producto acompañante. Cuando las funciones auxiliares son incluidas en una herramienta las interfaces asociadas están generalmente ocultas. Se garantiza la integridad de la interfaz porque todos los aspectos son controlados por un mismo proveedor.

Capítulo II. Enfoque Semántico

Introducción

A los efectos de esta investigación, interesa la conexión de la semántica a la Ingeniería de Software y específicamente a la Gestión de Requisitos. Se hace necesaria la introducción de conceptos relacionados con esta área y en particular analizar términos similares que se han utilizado en intereses investigativos diferentes pero que han tenido una amplia divulgación en la literatura científica y pudieran llevar a confusión al lector. Tal es el caso de Modelos de Objetos Semántico (SOM) que no es más que un modelo de datos alternativo al enfoque relacional y que abordaremos en las secciones que siguen. Por otra parte, también se presenta el proceso de metamodelado así como herramientas y características significativas de éste.

2.1 La Semántica

La Semántica Proviene del griego "semantikos", que quiere decir "significado relevante", derivada de "sema", lo que significaba "signo" .La semántica se dedica al estudio del significado de los signos lingüísticos y de sus combinaciones, desde un punto de vista sincrónico o diacrónico. (Piek; 1995). El término semántica, se refiere a los aspectos del significado o interpretación de un determinado código simbólico, lenguaje o representación formal. En principio cualquier medio de expresión (código y lenguas), admite una correspondencia entre expresiones de símbolos o palabras y situaciones o conjuntos de cosas encontrables o inferibles en el mundo físico o abstracto que puede ser descrito por dicho medio de expresión. (Berners-Lee, 2001)

2.1.1 Características de la Semántica

La semántica, aplicada a la web, tendría que ver con el significado de las cosas, y de las relaciones de unas con otras. Parecería que esto no tiene nada que ver con un diseñador o desarrollador, sino de quien se encargue del contenido. Pero el contenido no es lo único que tiene un significado. El código que se usa para estructurar y presentar este contenido también lo tiene, y es importante que esté de acuerdo con el contenido.

Así, las técnicas conducidas por los modelos enriquecidos por la semántica permiten la flexibilidad y la variabilidad en medios de representación.

Tipos de semánticas:

1. Semántica Lingüística.

2. Semántica en matemáticas y lógica.

Semántica Lingüística

La lingüística es la disciplina donde originalmente se introdujo el concepto de semántica. La semántica lingüística es el estudio del significado de las expresiones del lenguaje. La semántica lingüística contrasta con otros dos aspectos que intervienen en una expresión con significado: la sintaxis y la pragmática.

La semántica es el estudio del significado atribuible a expresiones sintácticamente bien formadas. La sintaxis estudia sólo las reglas y principios sobre como construir expresiones interpretables semánticamente a partir de expresiones más simples, pero en sí misma no permite atribuir significados.

Semántica en matemáticas y lógica

La lógica de predicados de primer orden es el tipo de sistema lógico-matemático más sencillo donde aparece el concepto de interpretación semántica. Dicha lógica está formada por (Litowitz, 1988)

Un conjunto de signos (conectivas, paréntesis, cuantificadores,...).

Un conjunto de variables y constantes.

Un conjunto de predicados sobre las variables.

Un conjunto de reglas de buena formación de expresiones a partir de expresiones sencillas.

En la lógica de primer orden el conjunto de variables y consonantes juega un papel similar al lexicón de las lenguas naturales ya que bajo una interpretación semántica son los elementos que admiten referentes. A su vez, el conjunto de reglas de buena formación de expresiones hace el papel de la sintaxis en las lenguas naturales. En lógica

matemática se suelen dividir los axiomas de los sistemas formales en axiomas de dos tipos:

Axiomas lógicos, que definen básicamente las reglas de deducción y están formados por tautologías. Básicamente son válidos para cualquier tipo de sistema formal razonable.

Axiomas matemáticos, que asevera la existencia de cierto tipo de conjuntos y objetos con verdadero contenido semántico. Gracias a ello es posible introducir conceptos nuevos y probar las relaciones entre ellos. Así si se tiene un conjunto de axiomas que define la teoría de grupos, cualquier grupo matemático es un modelo en el que las proposiciones y axiomas de dicha teoría reciben interpretación y resultan en proposiciones ciertas sobre ese modelo.

2.2 Otros Enfoques

Hablar del concepto lógico o físico de Base de Datos involucra un conjunto de pensamientos concretos que hacen posible la absorción temática del significado de los datos. La abstracción de los datos como una forma o un comportamiento que hace posible concretar un "algo", se asocia con un esquema del conocimiento lógico, su semántica, condiciones y acciones, que permiten la producción de *modelos* por medio de los cuales se representa la funcionalidad de un sistema.

Inicialmente el "dato" fue trabajado desde la óptica pura de su almacenamiento a través de los "Sistemas de Archivos"; donde cada uno de los archivos que se creaban solo obedecía a una necesidad de almacenamiento más que a la utilización funcional del dato. Por este motivo surgen los esquemas conceptuales que son elaborados a través del análisis de procesos de las áreas del negocio, los cuales involucran al "dato" como una consecuencia lógica funcional de ellos. Pero para poder estandarizar este tratamiento se crea el concepto del "***Modelo de Datos***".

A continuación se habla brevemente acerca de los Modelos de datos y sus características.

2.2.1 Modelo de Datos

Los ***Modelos de Datos*** permiten definir un conjunto de elementos y símbolos que permiten estandarizar traducir dicho análisis a un lenguaje semántico y sistemático que dispone de reglas de control y evaluación del comportamiento del dato en el transcurso del tiempo, tanto en su almacenamiento como en su utilización. Estos modelos de datos, hacen posible que la lógica de un negocio pueda ser estructurada de forma tangible a través de un esquema físico que representa el almacenamiento de los datos bajo las reglas del negocio y de un sistema gestor de base de datos que permitirá la persistencia de estos a través del tiempo. Mecanismo formal para representar datos de manera general y sistemática (descripción de datos, operaciones y reglas de integridad).

Los ***Modelos de Datos***, también llamados **modelos lógicos**, se han clasificado en dos grandes grupos debido al tratamiento de los datos: Basados en Objetos y basados en Registros. Estos dos grupos de modelos representan dos estados de la interpretación de los requerimientos de usuario:

Basados en Objetos: Un problema de la vida real maneja concepciones abstractas o concretas, tangibles o intangibles, a las cuales se les ha dado el nombre de "objetos", calificados a partir de un valor significativo dentro de los parámetros de una forma o estilo de vida; dichos objetos se modelan a través de propuestas que fueron estructuradas para así poder estandarizar la forma de manipularlos. Dentro de estos modelos tenemos:

- ✓ Modelo Entidad Relación (MER)
- ✓ Modelo Orientado a Objetos (MOO)
- ✓ **Modelo Semántico**
- ✓ Modelo Deductivo

Basados en Registros: Otra forma de tratar lógicamente la información suministrada por un sistema es a través de los "Registros", originalmente concebidos por los sistemas de archivos (registro: conjunto de campos que almacenan información de diferentes tipos), lo cual dió pie a la estructuración de modelos lógicos tales como:

- ✓ Modelo Relacional (MR)
- ✓ Modelo de Red
- ✓ Modelo Jerárquico

2.2.1.1 Modelo Semántico

La historia del modelamiento semántico, apareció en 1974 y maduró con el desarrollo de muchos modelos importantes como el Semantic Data Model , General Semantic Data Model, Iris Data Model, Semantic Object Data Model (Naphtalid,2003). Los modelos semánticos fueron introducidos como herramientas de diseño de esquemas. Este modelo es sin duda una representación del mundo real.

El modelado semántico permite la adquisición, la interpretación y la adaptación de modelos que pueden variar en diferentes escenarios en los que se debe tomar una decisión**.**

El modelo de objeto semántico

El modelo de objeto semántico (Litowitz, 1988) se emplea para documentar los requerimientos de los usuarios y para desarrollar modelos de datos. El equipo de desarrollo entrevista a los usuarios; reflexiona sus reportes, formas[2] y consultas. A partir de ellos, estructura un modelo de los datos de los usuarios. Este modelo de datos se transforma en un diseño de base de datos.

Objetos semánticos

Un objeto semántico es una representación de algunas cosas identificables en el ambiente de trabajo de los usuarios. De manera formal, un objeto semántico es un conjunto de atributos que describen con eficacia una identidad bien determinada.

[2] Formulario

Los objetos semánticos se agrupan en clases. Una clase de objeto tiene un nombre que la hace diferente a otras y que corresponde a los nombres de las cosas que representa. Un objeto semántico particular es una ocurrencia de tal clase.

Atributos

Los objetos semánticos poseen atributos que definen sus características. Hay tres tipos de atributos. Los atributos simples tienen un valor único. Algunos ejemplos son: Fecha de contratación, Número de factura y Total de ventas. Los atributos de grupo están compuestos por otros atributos. Un ejemplo es Dirección, que contiene los atributos {Calle, Ciudad, Estado, Código Postal} y los atributos de objeto semántico son los que establecen una relación entre un objeto semántico y otro.

Los atributos simples se dibujan escribiendo su nombre poniendo en mayúscula las iniciales de cada palabra que componga dicho nombre. Los atributos de grupo se dibujan con un corchete derecho que muestra la agrupación desde el nombre global que se le da al conjunto de atributos hasta el último de los atributos que componen dicho grupo. Finalmente los atributos de objeto semántico se dibujan en un rectángulo sombreado con el nombre del objeto en el centro del rectángulo y en mayúsculas.

Vistas de objetos semánticos

La parte visible de un objeto en una aplicación particular se denomina la vista de objeto semántico o sólo la vista. Una vista es el nombre del objeto más una lista de todos los atributos visibles en ella.

2.3 La Web semántica

Cada vez es mayor la preocupación de las grandes entidades por la usabilidad de sus portales. El ascenso vertiginoso de las operaciones de negocio por medio de la WEB es cada vez más grande. Se pueden realizar infinidad de operaciones por medio de un ordenador y una conexión a red: inversiones bursátiles, operaciones bancarias como transferencias, asegurar vehículos, altas en contratos telefónicos y de acceso a Internet, etc.

Como no podía ser de otra forma, la captación de clientes en este ámbito viene pareja a la seguridad de las comunicaciones y a la facilidad de uso de los servicios ofrecidos. Al ser amplia la competencia, cualquier esfuerzo en la captación y fidelización de los clientes repercute en las cuentas generales. Una forma de captación y fidelización es hacer que los portales corporativos sean cómodos, ágiles, con un uso restringido a cortos períodos de tiempo por operación. Es en si, conseguir que el usuario del servicio no abandone la web, por el simple hecho de no comprender el funcionamiento o considerar que invierte un período excesivo de su tiempo.

La usabilidad trata de estudiar todo lo que concierne a las variables de factor humano en el uso de las aplicaciones. Esta, evidencia la importancia del diseño de los interfaces y en especial, el diseño de los enlaces y guías con contenido semántico. Los usuarios de la WEB y del software en general están continuamente aplicando su propio conocimiento semántico cuando navegan (Berners-Lee,2001). Por ejemplo, para realizar una búsqueda, el usuario encontrará en su camino enlaces o "links" que le indicarán la ruta a seguir. La cantidad de información y la forma en que varios enlaces solapen su significado extensional, es decir, parezcan referirse a los mismos objetivos, hará o no que los usuarios dilaten el tiempo invertido en la búsqueda a riesgo de que abandonen finalmente el portal y recalen en otro.

La proliferación de los servicios ofrecidos vía WEB hace probable que el tiempo y el número de "clicks" que un usuario considere aceptable para dar con lo que busca sea relativamente bajo. Por eso, los diseñadores de los portales WEB (en especial los de grandes compañías), son cada día más sensibles a introducir protocolos de usabilidad en los portales que construyen. En otras palabras, hacer un diseño centrado en el usuario, que seria el Modelo Semántico del Usuario. (Kaur,2005)

Además de su diseño visual, copioso es el caudal de razones que inclina a considerar protocolos que contengan como sustrato el modelo mental del usuario, es decir, el modelo semántico que cada usuario posee del mundo que le rodea. Este modelo semántico de los usuarios, expresaría cómo las palabras se relacionan unas con otras y de qué manera las agrupaciones de estas palabras (enlaces de más de un término) se parecen entre sí.

2.4 ¿Qué es una ontología?

Aunque el concepto de ontología ha estado presente desde hace mucho tiempo en la filosofía, recientemente se utiliza en Informática para definir vocabularios que las máquinas puedan entender y que sean especificados con la suficiente precisión como para permitir diferenciar términos y referenciarlos de manera precisa.

Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento, tales como medicina, fabricación de herramientas, bienes inmuebles, reparación automovilística, gestión financiera, etc.). Las ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son útiles para los ordenadores.

Muchas personas se preguntan qué es una ontología; sin embargo, no saben que cualquiera tiene en su cabeza ontologías mediante las que representan y entienden el mundo que los rodean. Estas ontologías no son explícitas, en el sentido de que no se detallan en un documento ni se organizan de forma jerárquica o matemática. Todos usamos ontologías pues sabemos que una familia se compone de varios miembros, que un hijo no puede tener más de un padre y una madre biológicos, que los padres tienen o han tenido padres. No se necesita explicitar este conocimiento, pues forma parte de lo que todo el mundo sabe.

Sin embargo, cuando se tratan términos poco comunes o cuando se quiere que estos términos sean procesados por máquinas, se precisa explicitar las ontologías; esto es, desarrollarlas en un documento o darles una forma que sea entendible para las máquinas. Las máquinas carecen de las ontologías con las que nosotros contamos para entender el mundo y comunicarse entre ellas es por eso que se necesita de las Ontologías (Miguel, 2005).

Según el Grupo de Trabajo en Ontologías del consorcio W3C,"una ontología define los términos que se usan para describir y representar un cierto dominio. Uso la palabra dominio" para denotar un área específica de interés (el río Duero, por ejemplo) o un área de conocimiento (física, aeronáutica, medicina, contabilidad, fabricación de productos, etc.) Toda ontología representa cierta visión del mundo con respecto a un dominio" (Miguel, 2005).

Con la aparición de las ontologías en la web los usuarios pueden organizar la información de forma tal que los sistemas de software interpretan el significado de la misma. Las aplicaciones extraen automáticamente datos de las páginas web, los procesan y sacan conclusiones de ellos, pueden tomar decisiones y negociar con otros agentes o personas.

Esto se debe a que las ontologías catalogan y definen los tipos de cosas que existen en un cierto dominio al igual que sus relaciones, sus propiedades y sus componentes. "*Por ejemplo, una ontología del mundo empresarial usará conceptos como Venta, Compra, Transferencia, Pago, etc.; y relaciones como "Una Transferencia corresponde a una Venta o a una Compra", "Un Pago corresponde a una o varias Transferencias", etc."* (Miguel, 2005).

2.4.1 Tipos de Ontologías

Las Ontologías se pueden clasificar de acuerdo a la cantidad y tipo de estructura de la conceptualización. (Miguel, 2005).

Ontologías terminológicas: Especifican los términos que son usados para representar conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado*.*

Ontologías de información: Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para almacenamiento estandarizado de información*.*

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

Las ontologías también son clasificadas según su dependencia y relación con una tarea específica desde un punto de vista, se clasifica las Ontologías en:

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

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

Ontologías de Tareas o Técnicas básicas: Describen una tarea, actividad o artefacto, por ejemplo componentes, procesos o funciones.

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

2.4.2 Componentes de una ontología

Las Ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio (Gruber, 1993)

Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

Relaciones: representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.

Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignar- fecha, etc.

Instancias: se utilizan para representar objetos determinados de un concepto.

Axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición C1, A es B", etc. (Adolfo, 2007)

2.4.3 Lenguaje de Ontologías Web (OWL)

OWL es un lenguaje de Ontologías Web. OWL proporciona un lenguaje que utiliza la conexión proporcionada por RDF[3] para añadir las siguientes capacidades a las ontologías:

Capacidad de ser distribuidas a través de varios sistemas

- Escalable a las necesidades de la Web

[3] Resource Description Format, extensión de XML que incluye algunos elementos semánticos. Es ampliamente usado para implementar "porciones" de la denominada Web Semántica o Web 2.0

- Compatible con los estándares Web de accesibilidad e internacionalización
- Abierto y extensible

¿Cuál es la utilidad de las Ontologías para la Web?

El Grupo de Trabajo de Ontologías para la Web identificó los principales casos de uso de ontologías en la Web y los describió en el documento de Casos de Uso y Requisitos.

El Grupo de Trabajo los clasificó en las siguientes seis áreas principales:

- ✓ Portales Web
  - Reglas de categorización utilizadas para mejorar la búsqueda
- ✓ Colecciones Multimedia
  - Búsquedas basadas en contenido para medios no textuales
- ✓ Administración de Sitios Web Corporativos
  - Organización taxonómica automatizada de datos y documentos
- ✓ Asignación entre Sectores Corporativos.
- ✓ Documentación de Diseño
  - Explicación de partes "derivadas"
  - Administración explícita de Restricciones
- ✓ Agentes Inteligentes
  - Expresión de las Preferencias y/o Intereses de los usuarios
  - Mapeo de contenidos entre sitios Web
- ✓ Servicios Web y Computación Ubicua
  - Composición y Descubrimiento de Servicios Web
  - Administración de Derechos y Control de Acceso

2.5 El proceso de desarrollo de software basado en modelos

En los finales de los años 70 se observó un cambio importante en la filosofía del desarrollo de software. Tom DeMarco en su libro Structured Analysis and System

Specification introdujo el concepto de Ingeniería de Software Basada en Modelos. DeMarco destacó que la construcción de un sistema de software debe ser precedida por la construcción de un modelo del sistema, tal como se realiza en otros sistemas ingenieriles. De esta forma, el modelo de un sistema provee un medio de comunicación y negociación entre usuarios, analistas y desarrolladores. (DeMarco,1979)

El punto de partida en el proceso de desarrollo de software es la construcción de un modelo, el cual actúa como una especificación precisa de los requerimientos que el sistema debe satisfacer. Un modelo del sistema consiste en una conceptualización del dominio del problema. El modelo se focaliza sobre el mundo real: identificando, clasificando y abstrayendo los elementos que constituyen el problema y organizándolos en una estructura formal.

2.5.1 Utilidad de los modelos

El modelo del sistema se usa básicamente para los siguientes propósitos:
Para definir las necesidades del usuario. El propósito principal de la especificación de un producto es definir las necesidades de los usuarios del producto.

Como un medio de comunicación y negociación entre los usuarios y los desarrolladores y entre los distintos desarrolladores entre sí.

Como un documento de referencia durante la corrección de errores en el producto. Luego de introducir modificaciones en el sistema, la especificación es necesaria para chequear que la nueva implementación realmente está corrigiendo los errores contenidos en la versión previa del producto.

Como un documento de referencia durante la evolución del producto. En el caso de tener que adaptar el producto debido a cambios en los requerimientos, la especificación original debe ser adaptada para reflejar estos cambios consistentemente. Se ha observado que la construcción de modelos es una técnica muy efectiva para detectar y resolver discrepancias entre los divergentes puntos de vista de los usuarios acerca de sus requerimientos, brindando así bases firmes para las siguientes etapas del proceso de desarrollo.

Modelos formales vs. Modelos no-formales

El modelo del sistema se construye utilizando un lenguaje de modelado (que puede variar desde lenguaje natural o diagramas hasta formulas matemáticas). Los modelos informales son expresados utilizando lenguaje natural, figuras, tablas u otras notaciones. Hablamos de modelos formales cuando la notación empleada es un formalismo, es decir posee una sintaxis y semántica (significado) precisamente definidos. Existen estilos de modelado intermedios llamados semi-formales, ya que en la práctica los ingenieros de software frecuentemente usan una notación cuya sintaxis y semántica están sólo parcialmente formalizadas.

Métodos Formales en Modelado

Tipos de enfoques:

No-formales, usando lenguaje natural.

Semi-formales, notaciones (en parte graficas) con ciertas reglas y cuyas construcciones tienen una semántica más o menos precisa.

Formales, usando una notación gráfica o textual basada en un sistema formal (soporte matemático.)

Los métodos formales permiten determinar y expresar con mayor rigor las propiedades del software. Sin embargo, aún no son ampliamente utilizados.

Las principales mejoras al utilizar métodos formales son:

- Mayor rigor en la especificación.
- Mejores condiciones para realizar la verificación y validación en forma más exhaustiva.
- Mejores condiciones para automatización de procesos de generación automática de prototipos y/o código final.

¿Por qué se necesita un lenguaje de modelado?

- Los sistemas complejos son fáciles de entender si se cuenta con un modelo que los describa.
- Si se dispone de un lenguaje capaz de modelar cualquier sistema software, esto sería esencial.
- El lenguaje de modelado tiene un valor añadido si dicho lenguaje es estándar.

¿Qué debe aportar un lenguaje de modelado?

- Un metamodelo y una semántica.
- Una notación gráfica.
- Un conjunto de recomendaciones.

2.5.2 Arquitectura del software dirigida por modelos (Model-Driven Architecture, MDA)

El MDA es una evolución de estándares definidos por OMG (*Object Management Group*) para mejorar procesos de desarrollo de sistemas de software dirigidos por modelos. Ver figura 5.

Ideas centrales en MDA

- Separar la especificación de la funcionalidad del sistema de su implementación sobre una plataforma en una tecnología específica.
- Controlar la evolución desde modelos abstractos a implementaciones tendiendo a aumentar el grado de automatización.

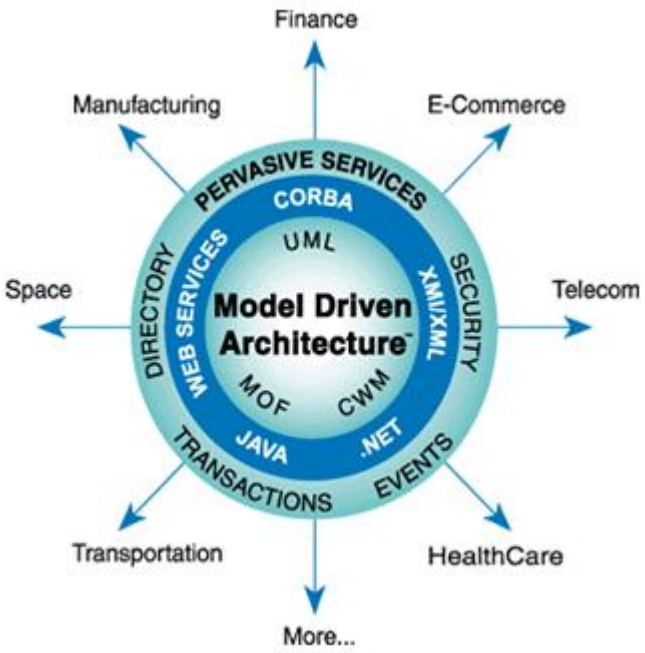


Figura 5. Representación del MDA

Plataforma en MDA

Es un conjunto de subsistemas y tecnologías que proveen un conjunto coherente de funcionalidades que pueden ser usadas en cualquier aplicación sin tener en cuenta detalles de es implementada cómo la funcionalidad.

Modelos y MDA

Distingue diferentes tipos de modelos:

- CIM (Computation Independent Model)
- PIM (Platform Independent Model)
- PSM (Platform Specific Model)
- ISM (Implementation Specific Model)

Computation Independent Model (CIM)

Es una descripción de la lógica del negocio desde una perspectiva independiente de la computación. Es un modelo del dominio.

Platform Independent Model (PIM)

Es una descripción de la funcionalidad del sistema en forma independiente de las características de plataformas de implementación específicas. Un modelo de un sistema que no contiene información acerca de la plataforma o la tecnología que es usada para implementarlo.

Platform Specific Model (PSM)

Un modelo de un sistema que incluye información acerca de la tecnología específica que se usará para su implementación sobre una plataforma específica. Es una descripción del sistema en términos de una plataforma específica. Por ejemplo, .NET, J2EE, entre otros. Modelos reusables a distintos niveles de abstracción.

Implementation Specific Model (ISM)

Es una descripción (especificación) del sistema a nivel de código. Por ejemplo, Java, C# entre otros.

MDA- Beneficios

En la actualidad, la construcción de software se enfrenta a continuos cambios en las tecnologías de implementación, lo que implica realizar esfuerzos importantes tanto en el diseño de la aplicación, para integrar las diferentes tecnologías que incorpora, como en el mantenimiento, para adaptar la aplicación a cambios en los requisitos y en las tecnologías de implementación. La idea clave que subyace al MDA es que si el desarrollo está guiado por los modelos del software, se obtendrán beneficios importantes en aspectos fundamentales como son la productividad, la portabilidad, la interoperabilidad y el mantenimiento.

Productividad: La Productividad puede definirse como la relación entre la cantidad de bienes y servicios producidos y la cantidad de recursos utilizados. En la fabricación la productividad sirve para evaluar el rendimiento de los talleres, las máquinas, los equipos de trabajo y los empleados.

Productividad=Salida/Entradas

Entradas: Mano de Obra, Materia prima, Maquinaria, Energía, Capital.

Salidas: Productos.

Índice De Productividad

Con el fin de medir el progreso de la productividad, generalmente se emplea el INDICE DE PRODUCTIVIDAD (P) como punto de comparación:

P= 100*(Productividad Observada) / (Estándar de Productividad)

Portabilidad: La portabilidad de un software es cuán dependiente es de la plataforma en la que corre. La portabilidad es mayor cuando la dependencia con el software de plataforma disminuye.

Si un software puede ser compilado en plataformas diversas, dicho software se dice que es multiplataforma. En algunos casos el software es "independiente" de la plataforma y puede ejecutarse en plataformas diversas sin necesidad de ser compilado específicamente para cada una de ellas, a este tipo de software se le llama interpretado, por que necesita de un interprete para ser ejecutado en las diferentes plataformas.

Interoperabilidad: La **interoperabilidad** (término a menudo traducido como interoperatibilidad, del inglés *interoperability*) es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos.

Así por ejemplo en el campo de la informática se habla de la interoperabilidad de la Web como una condición necesaria para que los usuarios (humanos o mecánicos) tengan un acceso completo a la información disponible. Entre las iniciativas recientes más destacadas para dotar a la Web de interoperabilidad se encuentran los Servicios Web y la Web Semántica.

Tipos de interoperabilidad

Interoperabilidad Técnica: Se refiere a la capacidad de los Sistemas de Información (SI), para intercambiar señales. Esta interoperabilidad exige de una conexión física (cable, fibra óptica, ondas electromagnéticas) entre los sistemas y un conjunto de protocolos de comunicaciones (como la pila TCP/IP). Si dos ordenadores no pueden intercambiar señales, no puede existir interoperabilidad alguna. Que dos SI interoperen técnicamente no significa que puedan intercambiar datos, pues para intercambiar datos

se precisa que los SI usen el mismo formato para el intercambio de datos… Siempre que me refiera a interoperabilidad de la Web o de las aplicaciones basadas en ella, me estaré refiriendo a la interoperabilidad sintáctica o a la semántica, o a ambas.

Interoperabilidad sintáctica: Se refiere a la capacidad de los SI para leer datos procedentes de otros SI y obtener una representación que puedan usar. Significa que en todos los SI involucrados, tanto la codificación de los datos como los protocolos de acceso son compatibles.

Interoperabilidad Semántica: Es la capacidad de los SI para intercambiar información basándose en un común significado de los términos y expresiones que usan. En otras palabras denota la capacidad de los SI para intercambiar información consistente con el significado que se le supone. La Interoperabilidad semántica no debe confundirse con la sintáctica: esta se refiere a la dificultad de procesar automáticamente los datos, no a su contenido. Aquella se refiere a la dificultad de relacionar los términos desconocidos que aparecen en los datos con los ya conocidos o definidos. La Interoperabilidad semántica no puede existir antes de la Interoperabilidad técnica y la sintáctica. (Miguel, 2005).

Evolución del software:

Durante los primeros años de la era de la computadora, el software se contemplaba como un añadido. El desarrollo del software se realizaba virtualmente sin ninguna planificación, hasta que los planes comenzaron a descalabrarse y los costes a correr. Los programadores trataban de hacer las cosas bien, y con un esfuerzo heroico, a menudo salían con éxito. El software se diseñaba a medida para cada aplicación y tenía una distribución relativamente pequeña.

La mayoría del software se desarrollaba y era utilizado por la misma persona u organización. La misma persona lo escribía, lo ejecutaba y, si fallaba, lo depuraba. Debido a este entorno personalizado del software, el diseño era un proceso implícito, realizado en la mente de alguien y, la documentación normalmente no existía.

La segunda era en la evolución de los sistemas de computadora se extienden desde la mitad de la década de los sesenta hasta finales de los setenta. La multiprogramación y los sistemas multiusuario introdujeron nuevos conceptos de interacción hombre - máquina. Las técnicas interactivas abrieron un nuevo mundo de aplicaciones y nuevos niveles de sofisticación del hardware y del software. Los sistemas de tiempo real podían

recoger, analizar y transformar datos de múltiples fuentes, controlando así los procesos y produciendo salidas en milisegundos en lugar de minutos. Los avances en los dispositivos de almacenamiento en línea condujeron a la primera generación de sistemas de gestión de bases de datos.

La segunda era se caracterizó también por el establecimiento del software como producto y la llegada de las "casas del software". Los patronos de la industria, del gobierno y de la universidad se aprestaban a "desarrollar el mejor paquete de software" y ganar así mucho dinero.

Conforme crecía el número de sistemas informáticos, comenzaron a extenderse las bibliotecas de software de computadora. Las casas desarrollaban proyectos en los que se producían programas de decenas de miles de sentencia fuente. Todos esos programas, todas esas sentencias fuente tenían que ser corregidos cuando se detectaban fallos, modificados cuando cambiaban los requisitos de los usuarios o adaptados a nuevos dispositivos hardware que se hubieran adquirido. Estas actividades se llamaron colectivamente mantenimiento del software.

La tercera era en la evolución de los sistemas de computadora comenzó a mediados de los años setenta y continuó más allá de una década. El sistema distribuido, múltiples computadoras, cada una ejecutando funciones concurrentes y comunicándose con alguna otra, incrementó notablemente la complejidad de los sistemas informáticos. Las redes de área local y de área global, las comunicaciones digitales de alto ancho de banda y la creciente demanda de acceso "instantáneo" a los datos, supusieron una fuerte presión sobre los desarrolladores del software.

La conclusión de la tercera era se caracterizó por la llegada y amplio uso de los microprocesadores. El microprocesador ha producido un extenso grupo de productos inteligentes, desde automóviles hasta hornos microondas, desde robots industriales a equipos de diagnósticos de suero sanguíneo.

La cuarta era de la evolución de los sistemas informáticos se aleja de las computadoras individuales y de los programas de computadoras, dirigiéndose al impacto colectivo de las computadoras y del software. Potentes máquinas personales controladas por sistemas operativos sofisticados, en redes globales y locales, acompañadas por aplicaciones de software avanzadas se han convertido en la norma.

La industria del software ya es la cuna de la economía del mundo. Las técnicas de la cuarta generación para el desarrollo del software están cambiando en la forma en que la comunidad del software construye programas informáticos. Las tecnologías orientadas a objetos están desplazando rápidamente los enfoques de desarrollo de software más convencionales en muchas áreas de aplicaciones.

Sin embargo, un conjunto de problemas relacionados con el software ha persistido a través de la evolución de los sistemas basados en computadora, y estos problemas continúan aumentando.

- Los avances del software continúan dejando atrás nuestra habilidad de construir software para alcanzar el potencial del hardware.
- Nuestra habilidad de construir nuevos programas no pueden ir al mismo ritmo de la demanda de nuevos programas, ni podemos construir programas lo suficientemente rápido como para cumplir las necesidades del mercado y de los negocios.
- El uso extenso de computadoras ha hecho de la sociedad cada vez más dependiente de la operación fiable del software. Cuando el software falla, pueden ocurrir daños económicos enormes y ocasionar sufrimiento humano.
- Luchamos por construir software informático que tengan fiabilidad y alta calidad.
- Nuestra habilidad de soportar y mejorar los programas existentes se ve amenazada por diseños pobres y recursos inadecuados.
- En respuesta a estos problemas, las prácticas de la Ingeniería del Software se están adoptando en toda la industria.

2.5.3 Proceso de Metamodelado. Características

El metamodelado es uno de los principales temas de investigación en el campo de las bases de datos y desarrollo de software, el concepto nace con la misma filosofía que en su tiempo motivó a desarrollar computadores (hardware) de propósito general en lugar de seguir con el esquema de construir un computador para cada aplicación en particular. Este concepto al aplicarse en el desarrollo de aplicaciones de software

permite diseñar un modelo de datos (el metamodelo) con la capacidad de almacenar otros modelos de información (Sendall, 2003). Esta tendencia genera importantes ahorros en tiempo y recursos destinados al mantenimiento de software, desarrollo de nuevas aplicaciones y en el desarrollo de nuevas características al software existente.

El Metamodelo

Metamodelo no es más que un modelo con la capacidad de almacenar diferentes modelos. Partiendo de esta definición se puede pensar en la posibilidad de construir un producto de software con la capacidad para registrar información en el metamodelo y otro producto con la capacidad de realizar consultas sobre el mismo.

El metamodelo está dividido en dos áreas:

1. Área de metadatos

Esta área almacena el metamodelo en sí, en un conjunto de tablas relacionales se almacena una descripción detallada del modelo de los datos a ser almacenados en el área de datos. Este modelo parte del principio que toda información para ser registrada en un sistema de información debe ser estructurada en uno o varios formatos siguiendo una metodología denominada**: "Metodología para la estructuración de formatos en un metamodelo"** (Griffin, 2005), la cual permite la creación de uno o varios formatos en un contexto determinado.

2. Área de datos

Esta área almacena la información acorde a la estructura definida en el área de metadatos. Es aquí donde se registra la información proveniente del mundo real, estructurada de acuerdo a los formatos que hacen parte del modelo almacenado en el área de metadatos.

2.5.3.1 Estándares para metamodelado

La mayor parte de las actividades humanas son dependientes del software, y la construcción de software se ha convertido en un sector estratégico para el desarrollo de la sociedad, siendo el software uno de los pilares para la implantación de la sociedad de la información. Desde la aparición de la ingeniería del software, su principal objetivo ha sido alcanzar la industrialización del software, crear una industria que posibilite obtener

software de alta calidad a bajo coste, de la misma manera que otras industrias crean sus artefactos.

No cabe duda que en los últimos cuarenta años se han realizado progresos importantes para la mejora del proceso de desarrollo de software. Desde la aparición de los primeros compiladores, se han sucedido un gran número de lenguajes, técnicas, tecnologías y procesos destinados a mejorar la productividad y calidad del software. Sin embargo, todavía no se puede decir que haya emergido una verdadera industria del software. (Greenfield, 2003).

La transición desde una producción artesanal a una producción industrial, es principalmente el resultado de automatizar los procesos de producción, de la fabricación de los productos mediante la integración y adaptación de componentes estándares, del uso de herramientas que permiten automatizar tareas repetitivas, y de la creación de líneas de productos. En el caso del software, es evidente que la automatización o generación automática de software, la reutilización de software basada en *assets*[4] de distinta naturaleza (patrones, componentes, frameworks, etc.), la utilización de estándares y la configuración de soluciones, han sido técnicas que a lo largo de los años han permitido mejoras significativas, pero todavía es necesario una evolución para llegar a una verdadera industria de software que permita economías de escala y de ámbito. (Greenfield, 2003).

Con el propósito de suponer un avance significativo hacia la industrialización del software, o al menos proporcionar mejoras significativas en la productividad y calidad, a lo largo de esta década ha emergido el paradigma del ***Desarrollo de Software Dirigido por Modelos*** **(DSDM,** (Molina, 2004), término que en realidad se refiere a un conjunto de técnicas que comparten algunos principios básicos, y que surgen, en su mayoría, a partir de la iniciativa MDA (Model Driven Architecture) ,(OMG, 2003), presentada en noviembre de 2001 (algunas de estas técnicas ya existían y ahora se han revitalizado). Entre ellas podemos destacar: MDA, Factorías de Software, Desarrollo Basado en Lenguajes Específicos del Dominio, y la Programación Generativa.

En un principio los perfiles UML fueron la alternativa más utilizada como lenguaje de modelado pero ahora han ganado en aceptación nuevos lenguajes definidos a partir de lenguajes de metamodelado como MOF. Junto a la automatización y al empleo de

[4] Activos

lenguajes de modelado, el uso de estándares es el tercer pilar en el que se sustenta MDA. OMG ha definido un conjunto de estándares para soportar su propuesta como son: **UML, MOF, OCL**, **XMI y QVT.**

UML

UML (*Modeling Lenguage Unified*) [UML] es un lenguaje de modelado visual que ha tenido gran aceptación en la comunidad del software y que miles de desarrolladores utilizan para crear modelos que le ayudan a razonar sobre el sistema que quieren crear y a documentar sus decisiones de diseño. UML es el lenguaje de modelado de sistemas de software más conocido en la actualidad; es el estándar internacional aprobado por la OMG (Object Management Group), consorcio creado en 1989 responsable de la creación, desarrollo y revisión de especificaciones para la industrial del software.

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

MOF

MOF (*Meta Object Facility*) [MOF] es el lenguaje de metamodelado (para definir lenguajes de modelado) propuesto por el OMG y actualmente se encuentra en su versión 2.0 Un lenguaje de modelado se define mediante un metamodelo o descripción de los elementos del lenguaje y de las relaciones existentes entre ellos. MOF es un lenguaje para crear metamodelos (por ejemplo, el metamodelo de UML ha sido definido con MOF), y es, por tanto, un elemento básico de MDA.

MOF permite a los usuarios definir totalmente nuevos lenguajes a partir de metamodelos. La alineación del metamodelo UML 2.0 con el metamodelo MOF simplificará el intercambio de modelos vía XMI y la interoperabilidad cruzada entre herramientas. La especificación del núcleo unificado MOF 2.0 debe estar arquitectónicamente alineada con la Infraestructura de UML (Booch, 2002).

Arquitectura de Lenguajes de Modelado

MOF define una Arquitectura de Lenguajes de Modelado en la que existen 4 capas o niveles:

- ✓ Nivel M3: MOF
- ✓ Nivel M2: UML
- ✓ Nivel M1: Modelo del usuario
- ✓ Nivel M0: Instancias en tiempo de ejecución.

Debido a las similitudes entre el modelo MOF y los modelos de estructura del UML, los metamodelos MOF son usualmente modelados como diagramas de clase del UML. Un estándar soportado de MOF es XMI, que define un formato de intercambio basado en XML.

MOF es una arquitectura de meta-modelado estricta; cada elemento del modelo en cada estrato es estrictamente una instancia del elemento del modelo del estrato de arriba. MOF sólo provee un medio para definir la estructura, o sintaxis abstracta de un lenguaje o dato.

Los conceptos básicos que proporciona MOF son clase, asociación, atributo, operación, generalización y el concepto de paquete como elemento organizativo. A la hora de crear un metamodelo, los conceptos del lenguaje se representan como clases, sus propiedades como atributos y operaciones, las relaciones estructurales entre ellos como asociaciones y agregaciones, y una relación de un concepto es una especialización de otro como una generalización.

OCL

OCL (*Object Constraint Language)* [OCL] es un lenguaje de especificación que permite definir modelos más precisos y completos, mediante expresiones que pueden establecer el cuerpo de una operación de consulta, un invariante de clase, las pre y postcondiciones de una operación, o especificar las reglas de derivación para atributos o asociaciones.

Es un lenguaje con el que podemos definir modelos más precisos y completos. Algunos de los usos de OCL son:

- Especificar valores iniciales de atributos.
- Definir el cuerpo de operaciones de consulta.
- Especificar las reglas de derivación para atributos o asociaciones
- Expresar restricciones sobre clases o atributos.

Las restricciones pueden expresarse tanto en lenguaje natural como en OCL. OCL es un lenguaje para consultar y restringir los elementos de un modelo, definido por OMG y es parte integrante de UML. El término Constraint que aparece en el nombre OCL perdura de la primera versión de OCL, que sólo permitía definir restricciones. Sin embargo, la versión 2.0 de OCL proporciona un lenguaje de consultas mucho más general, con una expresividad similar a la de SQL.

QVT

QVT (*Query-View Transformation*), OMG ha definido el lenguaje QVT para expresar transformaciones. La propuesta de estándar QVT, propone tres lenguajes de características diferentes: *Relations*, *Core* y *Operational Mappings*. El lenguaje *Relations* es un lenguaje declarativo, *Operacional Mappings* es imperativo, y ambos pueden implementarse sobre el lenguaje *Core*, que es un lenguaje de transformación con primitivas de bajo nivel. Actualmente la versión definitiva del estándar no ha sido publicada, y no existe ninguna implementación completa de la propuesta actual.

Es un estándar actualmente en desarrollo, que define el modo en que de lleva a cabo las transformaciones entre modelos cuyos lenguajes han sido definidos usando MOF. Este estándar se incluirá en MOF, y consta de 3 partes:

- ✓ Un lenguaje para crear vistas de un modelo.
- ✓ Un lenguaje para realiza consultas sobre modelos
- ✓ Un lenguaje para escribir definiciones de transformaciones.

XMI

XMI (*XML Model Interchange***)** es un estándar de la OMG que mapea MOF a XML. XMI define como deben ser usadas las etiquetas XML que son usadas para representar modelos MOF serializados en XML. Los metamodelos MOF son convertidos en DTD (Document Type Definitions) y los modelos son convertidos en documentos XML que son consistentes con su DTD correspondiente. XMI resuelve muchos de los problemas encontrados cuando intentamos utilizar un lenguaje basado en etiquetas para representar objetos y sus asociaciones, y además el hecho de que XMI esté basado en XML significa que tanto los metadatos (etiquetas) y las instancias que describen (elementos) pueden ser agrupados en el mismo documento, permitiendo a las aplicaciones entender rápidamente las instancias por medio de los metadatos.

Muchas de las herramientas CASE como Rational Rose, Together, Omondo, etc. soportan XMI y el estándar para importar y exportar el formato. XMI no solo puede ser usado como un formato de intercambio UML, sino que puede ser utilizado para cualquier formato descrito por medio de un metamodelo MOF. Las herramientas compatibles con MOF permiten definir metamodelos o importarlos, por ejemplo de repositorios y herramientas CASE, y empezar a editar o modificar la información del modelo. Una vez hecho esto la información del modelo podría ser intercambiada con otra herramienta compatible con MOF por medio de XMI.

UML, MOF y XMI son tres tecnologías clave para el desarrollo de software bajo el enfoque de MDA. Usadas de forma conjunta nos proporcionan grandes ventajas que hacen que los modelados sean más claros y fácilmente mantenibles. Realmente lo que definen estas tecnologías es una forma estándar de almacenar e intercambiar modelos, bien sean de negocio o de diseño. Esto permite a los constructores de herramientas CASE establecer un lenguaje común que se transformará en grandes beneficios para el desarrollador.

2.5.3.2 Herramientas para el metamodelado

En la actualidad, la construcción de software se enfrenta a continuos cambios en las tecnologías de implementación, lo que implica realizar esfuerzos importantes tanto en el diseño de la aplicación, para integrar las diferentes tecnologías que incorpora, como en el mantenimiento, para adaptar la aplicación a cambios en los requisitos.

Se han creado diferentes entornos de desarrollo basados en MDA, así como herramientas de transformación de modelos. OptimalJ de Compuware y ArcStyler de Interactive Objects son reconocidas como las herramientas MDA más maduras. Además permiten crear modelos independientes de plataforma bastante complejos.(García Molina, 2004)

El auge de MDA ha impulsado en los últimos años el desarrollo de tecnología (herramientas, lenguajes, etc.) tanto de carácter comercial como de propósito general. Algunas de estas herramientas implementan total o parcialmente los conceptos de MDA y DSDM.

Una lista bastante completa puede encontrarse en la página de OMG sobre MDA[5]. Entre ellas, destacamos las siguientes:

- **UMT** (UML Model Transformation Tool) - Herramienta para la trasformación de modelos y la generación de código basada en especificaciones UML/XMI.
- **MTL Engine** - Una implementación de QVT desarrollada por INRIA Triskel para Netbeans MDR y Eclipse EMF basada en el lenguaje MTL.
- **OpenMDX**, un entorno MDA open source que genera código para las plataformas J2EE y .Net
- **AndroMDA**, herramienta open source basada en templates para la generación de código J2EE desde modelos UML/XMI. Usa VTL (Velocity Template Engine) como lenguaje scripting.
- **XDoclet**, herramienta open source basada en atributos para la generación de código J2EE. Aunque realmente no está basada en modelos, puede ser combinada con otras herramientas, como UMT, para lograr resultados basados en modelos.
- **OMELET**, proyecto de Eclipse que trata de proporcionar un framework de carácter general que integre modelos, metamodelos y transformaciones.
- **Openmodel**, un framework basado en MOF/JMI para herramientas MDA.
- **VisualWADE**, una herramienta desarrollada por el grupo de Ingeniería Web de la Universidad de Alicante. VisualWADE permite el diseño y generación automática de aplicaciones web siguiendo una aproximación MDD (*Model Driven Development*). Combina diagramas de dominio UML con nuevos modelos para representar la interacción con el usuario sobre un entorno hipermedia. El código intermedio generado

[5] http://www.omg.org/cgibin/

es XML. En la versión actual, se proporciona un compilador de modelos que produce entregables en PHP a partir del código intermedio XML.

Herramientas comerciales:

- **ArcStyler**, herramienta comercial de Interactive Objects. Utiliza MOF para soportar estándares como XMI y UML, y además JMI para el acceso al repositorio de modelos. Integra herramientas de modelado (UML) y desarrollo (ingeniería inversa, explorador de modelos basado en MOF, construcción y despliegue).
- **MCC** (Model Component Compiler), producto comercial orientado a la generación de código para plataformas J2EE.
- **Codagen Architect,** producto comercial integrado con varias herramientas comerciales UML.
- **OptimalJ de Compuware**, usa notación de patrones para definir las transformaciones PSM y MOF para soportar estándares como UML y XMI. Se trata de un entorno de desarrollo que permite generar aplicaciones J2EE completas a partir de un PIM.

Capítulo III .Reutilización de Requisitos.

Introducción

La reutilización de requisitos es una de las tendencias actuales que está proporcionando grandes beneficios al desarrollo del software. A continuación se presentan los repositorios, repositorios semánticos, aspectos de la reutilización de requisitos y los rasgos más significativos de la misma así como las ventajas y desventajas de ésta. Por último, se muestran dos casos de estudio de formas prácticas de vinculación de la gestión de requisitos y el enfoque semántico.

3.1 Repositorio

Un repositorio es un lugar donde se almacena información la cual puede estar estructurada. Cuando un usuario busca información en un repositorio, describe de manera breve el documento que quiere obtener como respuesta. El resultado es que el usuario "navega" en el repositorio realizando consultas y refinando resultados hasta que queda satisfecho (Elizalde, 2000).

3.1.1 Repositorio Semántico

Los repositorios son semánticos cuando incluyen la propiedad de analizar la semántica de su contenido, es por eso que la web 2.0 se llamaría web semántica, entonces para que un repositorio sea semántico debe hacer uso de las ontologías

Al proporcionar un soporte para albergar los metadatos, estos repositorios desempeñan un importante papel de cara al futuro. No sólo los humanos pueden consultar y buscar información, sino también agentes software o sistemas LMS (*Learning Management Systems*) externos. Para procesar la información existente en los metadatos, es necesaria la presencia de metadatos de calidad, entendiendo por ello que cumplan unos mínimos de compleción y que los datos aportados se correspondan con un esquema de metadatos preestablecido, uniforme y, a ser posible, universal.

Actualmente, la calidad de los registros de metadatos depende, entre otros, de los siguientes factores:

- La información proporcionada en los metadatos depende de la bondad del creador del registro y del tiempo necesario para añadir dicha información.
- Las capacidades de edición o herramientas proporcionadas por el repositorio.

Los clientes del repositorio (usuarios finales, agentes software y sistemas de administración de contenidos didácticos), podrían, entre otras funcionalidades, añadir, recuperar, modificar, sin importar la conceptualización utilizada por el creador.

En (Soto, 2006) se define el prototipo SLOR (Semantic Learning Object Repository), basado en una ontología, ha sido específicamente diseñado para la creación y administración de metadatos con propósitos de integración e intercambio con otros sistemas. Esta propuesta aporta mejores y nuevas funcionalidades sobre los repositorios actuales, gracias a la posibilidad que la ontología subyacente ofrece para ejecutar inferencias sobre el conocimiento albergado en los registros del repositorio.

El método de creación obtiene una referencia a un registro de metadatos, creando una instancia de la clase correspondiente, y permite establecer las propiedades del mismo según la definición del concepto elegido en la ontología subyacente, lo que puede ocasionar variaciones.

La búsqueda semántica permite solicitar instancias de las distintas conceptualizaciones del modelo ontológico, por ejemplo, recuperar todos los objetos digitales, o todos los objetos con propósito educacional. Por otro lado, los registros de SLOR almacenan información enlazada a conceptos de otras ontologías.

3.2 Casos de Estudio

3.2.1 Aplicación de la Semántica. Semantic Access Control Model (SAC)

La Web semántica es considerada la nueva generación de la web. El objetivo de esta nueva "web de los significados" es permitir que tanto máquinas como personas trabajen en cooperación. En (Yagüe, 2003), se presenta un nuevo modelo de control de acceso al que se ha denominado SAC, *Semantic Access Control Model*, el cual usa diferentes capas de metadatos para describir la semántica de los diferentes componentes que participan en una decisión de acceso.

El diseño de este modelo se basa en información semántica para lograr que se tengan en consideración las propiedades particulares de los recursos accedidos (lo que se conoce como "introspección de contenido").

Junto con el modelo de control de acceso se diseñó un lenguaje de políticas, *Semantic Policy Language* (SPL), que se basa en el modelado semántico de la información contextual así como de los distintos recursos a proteger.

En el modelo SAC, la identificación del usuario o cliente no es obligatoria. Esto es debido a que los clientes poseen una serie de atributos, y el acceso a los recursos se basa igualmente en la especificación de un conjunto de atributos que debe reunir el cliente para poder acceder a ellos. Además, gracias a la alta expresividad semántica del modelo y a los distintos niveles de metadatos representados, se consigue una total interoperabilidad entre los distintos componentes del sistema de control de acceso.

Los conceptos de la web semántica y su infraestructura en capas pueden desempeñar un papel muy importante en muchos campos relevantes, como ocurre en este caso en las áreas de control de acceso y de autorización.

3.2.2 Ingeniería de Requisitos y Reutilización

SIREN (SImple REuse of software requiremeNts) (Nuseibeh, 2000), es un método de IR basado en la reutilización de requisitos. El propósito del desarrollo con reutilización de requisitos es identificar descripciones de sistemas que puedan ser reutilizadas (en su totalidad o en parte) con un mínimo número de modificaciones, de manera que se

reduzca el esfuerzo total de desarrollo. Este es el nivel de reutilización que aporta mayores beneficios, puesto que se pueden establecer relaciones de trazabilidad entre los requisitos de alto nivel del sistema y la arquitectura, implementación y pruebas que se construyen a partir de ellos. De esta manera, si un requisito del sistema es reutilizado, se puede acelerar el proceso de desarrollo posterior.

Los requisitos tienen formato textual, pero pueden incluir todo tipo de objetos como información complementaria del propio requisito por ejemplo, tablas, diagramas o esquemas de cualquier tipo.

Como método, SIREN incorpora guías y técnicas, un repositorio de requisitos reutilizables, un modelo de proceso, y una herramienta que soporta el repositorio (Requisite Pro). Las guías que proporciona SIREN consisten en una jerarquía de documentos de especificación de requisitos, junto con las plantillas para cada uno de ellos que determinan la estructura de un repositorio de requisitos reutilizables.

SIREN es una aproximación práctica para elicitar y especificar los requisitos de un sistema software, basada en la reutilización de requisitos y en estándares de ingeniería del software.

Tradicionalmente en el análisis de requisitos que precede al diseño de bases de datos se tienen en cuenta los requisitos de información (datos) y se presta menos atención a los requisitos funcionales o de evolución del sistema; sin embargo, es importante que la tarea de especificación de estos requisitos funcionales se realice en paralelo; esto se concreta en definir el conjunto de operaciones que el usuario del sistema prevé realizar, tanto para obtener información del sistema como para modificarla.(Nuseibeh,2000)

3.3 Beneficios e Inconvenientes de la Reutilización de Requisitos

Se pueden encontrar aspectos muy positivos en la aplicación del reuso, así como serias complicaciones y dificultades producidas precisamente por reutilizar. Este apéndice pretende presentar las nuevas tendencias en reutilización, que están consiguiendo potenciar aun más sus beneficios y reducir sus inconvenientes.

¿Qué es la reutilización?

"Se define la reutilización de software como el uso de cualquier tipo de artefacto (también llamado activo), o parte del mismo, creado con anterioridad, en un nuevo proyecto." (Addison,1996)

Un ejemplo:

- El artefacto ha sido ligeramente modificado para ajustarse a una definición de problema, o a una necesidad de solución en el nuevo proyecto.
- El artefacto fue creado para funcionar en un contexto, entorno o aplicación completamente diferente a la nueva.
- El artefacto debió ser configurado para adaptarse a las especificaciones o requerimientos del nuevo proyecto.

La reutilización es una "Buena Practica", que produce espléndidos beneficios a la organización que la pone en práctica:

La reutilización permitirá:

- Un aumento de la Productividad, mediante la mejora de los tiempos en los que se desarrollan los nuevos proyectos informáticos.
- Un aumento de la calidad de los productos.
- Una reducción de los costes de desarrollo.
- Mejoras en las actividades de Mantenimiento y soporte de aplicación.
- Mejoras en las actividades de control y planificación por la reducción de desviaciones en los desarrollos.

Pero por otro lado, los expertos en reutilización suelen advertir que también existen inconvenientes:

- Requiere una inversión inicial, por lo que resulta necesaria una seria actividad de evaluación del proceso de reutilización para estudiar cuando se produce el retorno de la inversión.
- Requiere modificaciones sustanciales en el proceso de desarrollo de Software de la organización, así como en la estructura productiva.
- Requiere formación en nuevas herramientas y métodos (algunas veces demasiado complejas)

Para mejorar la eficiencia de nuestros proyectos es necesario trabajar más rápido, de forma más inteligente o trabajar menos. Claramente, la reutilización es una alternativa óptima para estas cuestiones.

La reutilización de requisitos puede darnos aún más valor en lo relativo a costes, tiempos y calidad, es decir, **productividad**. Para lograr esto es necesario tener en cuenta las siguientes técnicas:

Búsqueda semántica de requisitos: permite **indexar** con técnicas semánticas todos y cada uno de los requisitos de los proyectos. Gracias a estas técnicas, la precisión en la recuperación se puede multiplicar por varios órdenes de magnitud y recuperar requisitos hasta ahora inalcanzables.

Haciendo uso de la **trazabilidad,** otros activos pueden aportar aún más valor al proyecto. Se debe convertir el clásico *copy-paste* en un ***find-evaluate-copy-reuse.***

Patrones de requisitos: Un patrón de requisitos puede ser visto como un conjunto de requisitos reutilizable. Un patrón de requisitos debería estar compuesto por la siguiente información:

- El conjunto de requisitos, ya sean estos horizontales o verticales.
- Los riesgos que acecharán al proyecto por la simple y sencilla razón de que tiene que abordar uno o más de esos requisitos.

- La especificación de las pruebas que han de llevarse a cabo para validar que el nuevo sistema a construir cumple con lo indicado en los requisitos del patrón.
- El conjunto de documentos que permita entender el contenido del patrón y que ayude a implementarlo correctamente.

Requisitos parametrizables: representan un tipo especial de requisitos dentro de un patrón. Contienen una parte común, aunque requieren que se indique algún dato como parte de la instanciación de este requisito en un nuevo proyecto.

La reutilización de software fue planteada como una vía complementaria para la mejora de los procesos de desarrollo de sistemas, con los objetivos de aligerar todas las tareas propias de estos procesos e incrementar la calidad de los sistemas obtenidos.

No hay ingeniería profesional sin requisitos bien gestionados (Gamma,1995). Todas estas actividades relacionadas con la ingeniería de requisitos, hasta hace no mucho tiempo, eran acometidas de manera absolutamente manual, incluyendo el uso de procesadores de texto para acometer su gestión.

Conclusiones

Con la realización de este trabajo se puede decir que se cumplieron los objetivos propuestos:

- Se estudiaron los enfoques semánticos, la gestión de requisitos y su posible vinculación.
- Se analizaron las virtudes del metamodelado y su vinculación a la IR.
- Se analizaron las ventajas de la reutilización de requisitos, y los beneficios que estas pueden tener para el éxito para la mejora de los procesos de desarrollo de sistemas.
- Se identificaron casos de estudio en los que se vincula el enfoque semántico y la gestión de requisitos.

Recomendaciones

Como recomendaciones se propone:

- Insertar este tema de la vinculación de modelo semántico al proceso del desarrollo del software, en la enseñanza de la Asignatura Ingeniería del Software actual.
- Implementar un Repositorio de Requisitos Reutilizables, aplicable a cualquier proyecto, basado en el enfoque semántico.
- Insertar las ideas del enfoque semántico en la práctica de Gestión de Requisitos actual.
- Que se cree en la universidad o en la facultad un equipo de investigación que se dedique al estudio del enfoque semántico y los conceptos de metamodelado.

Referencias

1. (Adolfo, 2007) Lozano Tello, Adolfo. Ontologías en la Web Semántica. Disponible en:http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf . (Consultado el 3 de febrero del 2007).

2. (Addison, 1996) "Analysis Patterns: Reusable Object Models". Martin Fowler. Addison-Wesley Professional, 1996.

3. (Anaya, 2002) Anaya, V., Letelier, P., SmarTTrace: Una Herramienta para Trazabilidad de Requisitos en Proyectos basados en UML, Proceedings of the V Workshop on Requirements Engineering, pp. 210-224, Valencia, España, Noviembre 2002.

4. (Booch, 2000) Booch, Grady Rombaugh ,james y Jacobson, Ivar (2002.El lenguaje Unificado de modelado España: Adyson Wesley) 432 p.

5. (Booch, 2002) Grady Booch. Growing the UML. Software and System Modeling, (2002).

6. (Berners-Lee, 2001) Hendler, J. y Lassila, O. The Semantic Web. Scientific American 284, pp. 34–43.2001.

7. (Charette, 1989) Charette, R. N.: Software Engineering Risk Analysis and Management, McGraw–Hill/Intertext, 1989.

8. (DeMarco,1979) Tom DeMarco, Structured Analysis and System Specification. Englewood Cliffs,NJ:Prentice Hall, 1979.

9. (Elizalde, 2000) Elizalde Vieyra, Guadalupe. Repositorios de información (Introducción). Disponible en http://www.fismat.umich.mx/~elizalde/tesis/node7.html. (23 de mayo del 2000).

10. (Finkelstein, 2000) Anthony Finkelstein & Wolfgang Emmerich (University College London, Dept. Computer Science.)Paper "The Future of Requirement Management Tools"

11. (Gamma, 1995) "Design Patterns: Elements of Reusable Object-Oriented Software". Eric Gamma et al. Addison-Wesley Professional, 1995.

12. (Greenfield,2003), Jack Greenfield y Keith Short, "Software Factories, Assembling Applications with Patterns, Models and Tools", Companion of the 18th annual ACM SIGPLAN Conference OOPSLA, 16-27, 2003.

13. (Griffin, 2005) Catherine Griffin, Sebastien Demathieu, Shane Sendall. ModelTransformation with the IBM Model Transformation Framework, 2005,(http://www.devx.com/ibmrational/Article/28097

14. (Gruber, 1993) T. R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), pp. 199-220, 1993.

15. (Hilera, 2003) Hilera , Jose R.(2003).Metodologia Metrica .Orientada a Objetos NOVATICA (En linea)(Consulta :diciembre 12 de 2003).

16. (*Kaur, 2005*) A comparison of LSA, WordNet and PMI-IR for predicting user click behavior. Proceedings of ACM CHI 2005: Conference on Human Factors in Computing Systems, New York: ACM, pp. 51-60.

17. (Lan A., 1994) Requirements Engineering Tool Vendors and Freeware Suppliers, 1994.

18. (Larman, 2003) Larman, Craig(2003) .UML y Patrones :Una introducción al análisis y al diseño Orientado a Objetos y al Proceso Unificado .Madrid:Prentice Hall ,590 p.

19. (Letelier, 2002) Letelier, P., A Framework for Requirements Traceability in UML-based Projects.1st International Workshop on Traceability in Emerging Forms of Software Engineering. In conjunction with the 17th IEEE International Conference on Automated Software Engineering, U.K. http://ase.cs.ucl.ac.uk/, Septiembre 2002

20. (Litowitz, 1988) El modelo relacional, relación con la semántica léxica véase Vossen (1992) págs. [...] e Iris, Litowitz & Evens (1988).

21. (Loucopoulos, 1995) Karakostas, V. (1995); System Requirements Engineering McGraw-Hill, 1995. , Loucopoulos.

22. (Miguel, 2005) Abián, Miguel Ángel. ONTOLOGÍAS: QUÉ SON Y PARA QUÉ SIRVEN. Disponible en: http://www.wshoy.sidar.org/index.php?2005/12/09/30-ontologias-que-son-y-para-que-sirven (9 diciembre 2005)

23. (Molina, 2004) Jesús J. García Molina, J. Rodríguez, M. Menárguez,M.J.Ortín, J. Sánchez, Un estudio comparativo de dos herramientas MDA: OptimalJ y ArcStyler, Taller de Desarrollo de Software Dirigido por Modelos,DSDM'04 en JISBD'2004, Páginas: 88-98, Málaga, noviembre de 2004.

24. (Naphtalid, 2003) Rishe Naphtalid., *Semantic SQL: A Semantic Wrapper for Re-lational Databases*, School of Computer Science, Florida International University, Miami, 2003.

25. (Nuseibeh, 2000) Nuseibeh B. and Easterbrook, S. Requirements Engineering: A Roadmap. in 22nd International Conference on Software Engineering (ICSE'00). 2000. Limerick, Ireland:IEEE Computer Society Press.

26. [OMG, 2003] Object Management Group. MDA Guide versión 1.0.1. Omg/2003-06-01, 2003. OMG document.Especificación de MDA.

27. (Piek, 1995) Grammatical and conceptual individuation in the lexicon, Universiteit van Amsterdam, Amsterdam.Vossen , Piek.

28. (Pressman, 2002) Pressman, Roger S.: Ingeniería de Software. Un enfoque práctico. Quinta edición. McGraw-Hill. Madrid. 2002.

29. (Ramesh, 2001) B. Ramesh and M. Jarke. Toward Reference Models for Requirements Traceability.IEEE Transactions on Software Engineering, Vol. 27, No. 1, pp.58-93, January 2001.

30. (Richard , 1997) IEEE Software Requirement Engineering, Second Edition. Thayer y Merlin Dorfman, IEEE Computing Society, New York, NY. 1997.

31. (Sendall, 2003) Shane Sendall and Wojtek Kozaczynski. Model transformation: The heart and soul of model-driven software development. IEEE Software, 20(5), September/October, 2003.

32. (Sommerville, 1997) Requerimentes Engineering: A Good Practice Guide. Johm Wiley and Sons, 1997..

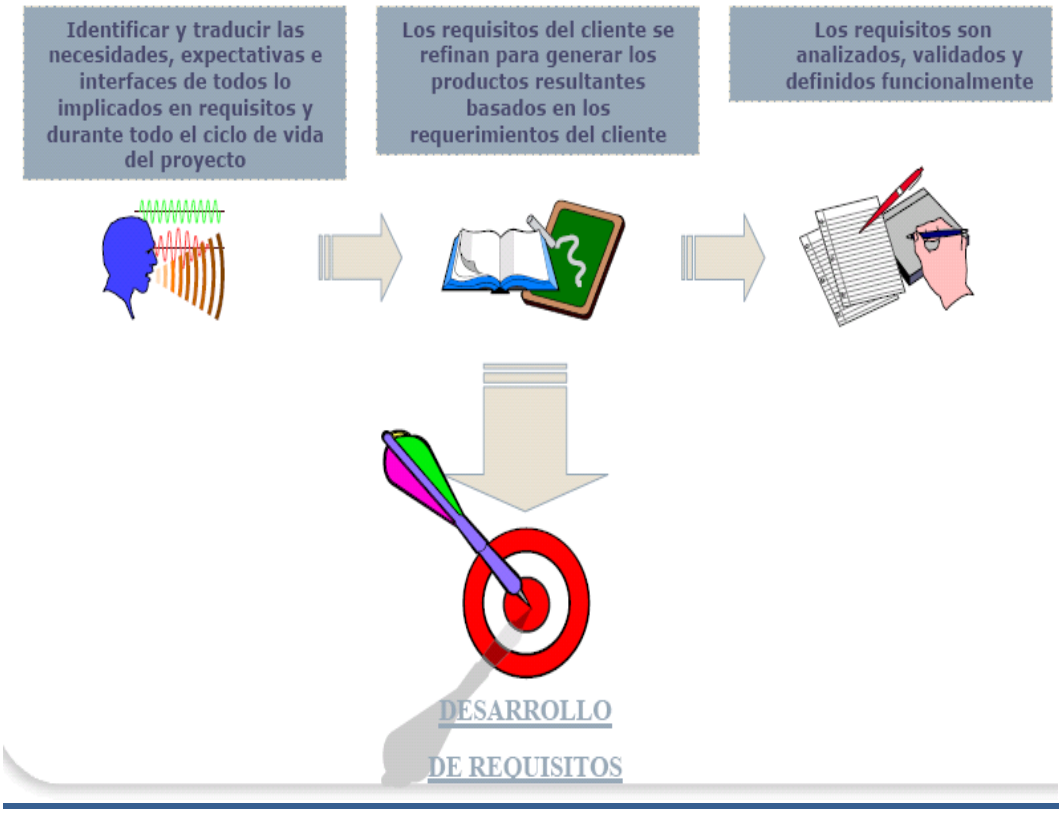
33. (Soto, 2005) Soto, J. & García, E. (2005). Sistema multiagente inteligente para la planificación organizada del estudio de un alumno. En actas del III Simposio Internacional de Sistemas de Información e Ingeniería del Software en la Sociedad del Conocimiento 1(1), 34-51.

34. (Yagüe, 2003) "Modelo basado en metadatos para la integración semántica en entornos distribuidos. Aplicación al escenario de control de accesos". Dirigida por D. José Mª Troya. Dpto. Lenguajes y Ciencias de la Computación. 2003

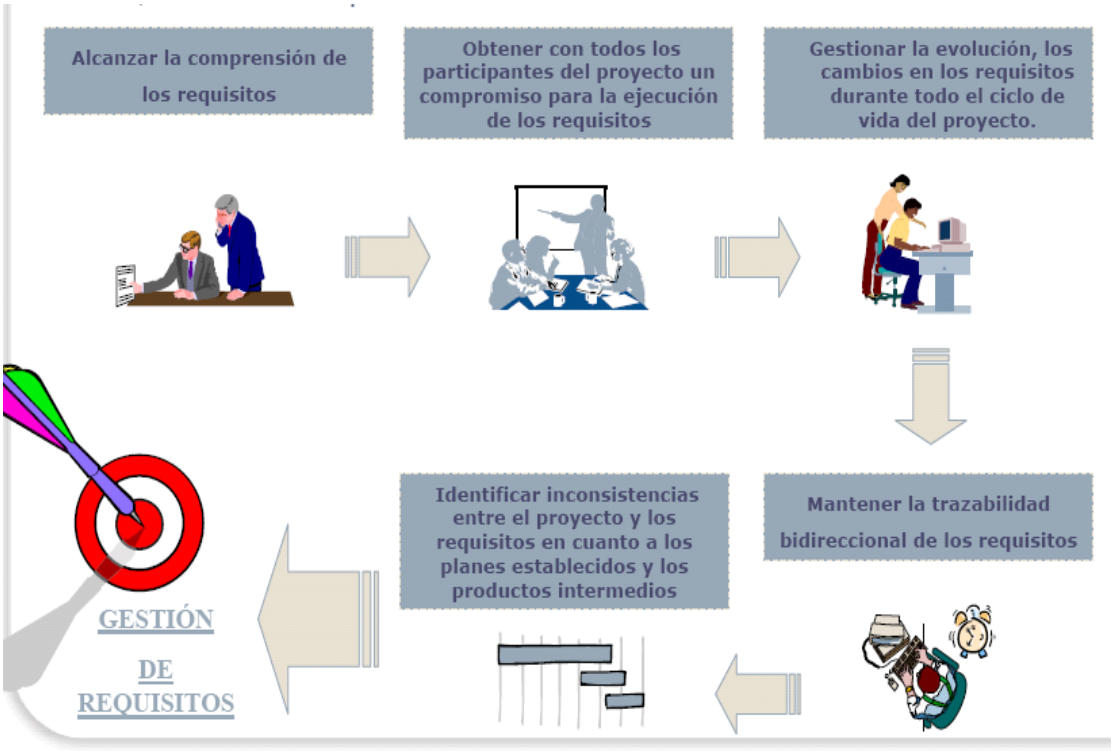
Anexos

Anexo 1. Formato de la plantilla Volere

Anexo 2.Desarrollo de Requisitos



Anexo 3.Gestión de Requisitos



Anexo 4.Reutilización de Requisitos

Glosario de Abreviaturas

Web Ontology Language (OWL): Lenguaje de Ontologías Web

Object Management Group (OMG): Grupo de Gestión de Objetos

Object Constraint Language (OCL): Lenguaje de Especificación de Restricciones

Privilege Management Infrastructure (PMI): Infraestructura de Gestión de Privilegios

Common Object Request Broker Architecture (CORBA): Arquitectura de Agentes de Consulta de Objetos Comunes

Departament of Defence(DD): Departamento de Defensa

Digital Rights Management (DRM): Gestión de Derechos Digitales (de Propiedad Intelectual)

Institute of Electrical and Electronics Engineers (IEEE)**:** Instituto de Ingenieros Eléctricos y Electrónicos.

Semantic Learning Object Repository (SLOR): Repositorio de Objeto de Aprendizaje Semántico

Model Driven Development (MDD) : Desarrollo Dirigido por Modelo

Model Driven Architecture (MDA): Arquitectura Dirigida por Modelo

Semantic Access Control Model (SAC): Control de Acceso a Dato

SImple REuse of software requiremeNts (SIREN): Simple Reuso de los requisites del software.

Vs: Contra

Ingeniería de Requisitos(IR)

Glosario de Términos

Metadatos: Información acerca de las propiedades de los datos. También puede ser información sobre la estructura de los datos o información que especifique el diseño de objetos.

Ontología: Representación formal del conocimiento mediante relaciones conceptuales dentro de un dominio dado, para facilitar la comunicación y el intercambio de la información entre diferentes sistemas.

Sentencia de comentario, directriz, [pragma]: Sentencia en un lenguaje de programación ajena al problema que se quiere resolver y que transmite una orden a una implementación particular del compilador, que es ignorada por otras implementaciones de compilador para ese lenguaje.

Sincrónico, ca Simultáneo,que ocurre o se desarrolla a la vez que otra cosa: movimientos sincrónicos;

Diacrónico: Que se desarrolla a lo largo del tiempo, evolución diacrónica.

Web Semántica:Término acuñado por Tim Berners-Lee que ve el futuro de la Web como una base de datos global. La infraestructura de la Web semántica permitirá que, tanto las máquinas como los humanos hagan deducciones y organicen la información. Los componentes de la arquitectura de la Web semántica implican semántica (significado de los elementos), estructura (organización de los elementos) y sintaxis (comunicación).

Interoperabilidad semántica: Capacidad de buscar información digital entre bases de datos distribuidas heterogéneas cuyos esquemas de metadatos se han relacionado de uno a otro.

Análisis semántico: Paso de la compilación que añade información semántica al árbol del análisis y realiza comprobaciones basadas en esa información. Se sitúa entre la creación del árbol del análisis y la generación del código ejecutable. También es posible realizar algún tipo de optimización de código. Entre la información semántica tratada están los nombres y las definiciones (firmas) de las funciones.

Semántica: Estudio del significado de las palabras.

Asset: activo, recurso Cualquier cosa que tiene un valor.

Interoperabilidad: Capacidad de dos o más sistemas para relacionarse e intercambiar información de manera útil y con sentido.

Granularidad: Cantidad mínima de almacenamiento que se puede utilizar para satisfacer una demanda de almacenamiento adicional. Nivel de modularidad de un sistema. Módulos más pequeños indican una mayor flexibilidad. Grado de especificación de la información que contiene un elemento de datos. Una tabla de hechos de granularidad fina contiene muchos hechos discretos.

Sintaxis: Parte de la gramática que estudia la forma en que se combinan y relacionan las palabras para formar secuencias mayores, cláusulas y oraciones y la función que desempeñan dentro de estas: la sintaxis estudia los tipos de oraciones.

Artefacto: En tecnología, artefacto es un dispositivo concebido y fabricado, sea de modo artesanal o industrial, por una o más personas. La característica principal de los artefactos es que cumplen una función técnica, es decir, sirven para hacer algo.

Elicitación: La Elicitación consiste en las primeras actividades a realizarse en la Ingeniería de requisitos, aunque esta etapa no se puede "divorciar" de las demás, ya que seguramente se iterará a través de las mismas durante el desarrollo de los requerimientos.

Estrategia: La estrategia es cualquier conjunto de acciones o comportamiento, sea deliberado o no. Definir la estrategia como un plan no es suficiente, se necesita un concepto en el que se acompañe el comportamiento resultante. Específicamente, la estrategia debe ser coherente con el comportamiento.

IEEE: corresponde a las siglas de The Institute of Electrical and Electronics Engineers, es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

Interactividad: Expresión extensiva que en una serie de intercambios comunicacionales implica que el último mensaje se relaciona con mensajes anteriores a su vez relativos a otros previos.

Metodología: La metodología se entenderá aquí como la parte del proceso de investigación que sigue a la propedéutica y permite sistematizar los métodos y las técnicas necesarios para llevarla a cabo.

Norma: En tecnología, una norma o estándar es una especificación que reglamenta procesos y productos para garantizar la interoperabilidad.

Sistema: Un sistema es una colección de unidades organizadas para cumplir un propósito en específico. Un sistema puede ser descrito por uno o más modelos y posiblemente desde diferente.

Terminología: Es una ciencia interdisciplinar que se nutre de un conjunto específico de conocimientos conceptualizado en otras disciplinas.

Trazabilidad: Es un conjunto de medidas, acciones y procedimientos que permiten registrar e identificar cada producto desde su origen hasta su destino final.

Usabilidad: El modelo conceptual de la usabilidad, proveniente del diseño centrado en el usuario, no está completo sin la idea utilidad.

Pragmatismo: Disciplina que estudia el lenguaje en relación con el acto de habla,el conocimiento del mundo y uso de los hablantes y las circunstancias de la comunicación.

Tautología: Repetición de un mismo pensamiento expresado de distintas maneras,pero que son equivalentes.