UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
MAESTRÍA EN SISTEMAS DE INFORMACIÓN

TRABAJO DE GRADO DE MAESTRÍA

MODELO ONTOLÓGICO PARA DETERMINAR EL USO DE PATRONES DE SOFTWARE ORIENTADO A DEPARTAMENTOS DE DESARROLLO DE SISTEMAS EN EMPRESAS DEL SECTOR BANCARIO

Presentado por:
Adriano Da Silva Abreu
Para optar al título de Magíster en Sistemas de Información.

Tutor:
Dr. Pedro Nolasco Bonillo Ramos

Caracas, Julio de 2014

ACEPTACIÓN DEL ASESOR

Por la presente hago constar que he leído el trabajo de grado de maestría presentado por el estudiante Adriano Da Silva Abreu. C.I.: V-11.677.444 para optar al título de Magíster en Sistemas de información, cuyo título es "Modelo ontológico para determinar el uso de patrones de software orientado a departamentos de desarrollo de sistemas en empresas del sector bancario" y manifiesto que cumple con los requisitos exigidos por la Dirección de Postgrado de la Universidad Católica Andrés Bello, y que por lo tanto, lo considero apto para ser evaluado por el jurado que se decida designar para tal fin.

En la ciudad de Caracas, a los 02 días del mes de abril de 2014.

Pedro Nolasco Bonillo Ramos
C.I. V-10.868.538

DEDICATORIA

Al Nazareno de San Pablo, a quien dediqué esta maestría, en quien me apoyo en los momentos de dificultad para salir adelante.

A mi esposa Susana y mis hijos Juan Miguel e Isabel Victoria por el apoyo, la comprensión y el ánimo que me dan para lograr cada día más éxitos.

RECONOCIMIENTOS

El presente reconocimiento va dedicado a:

Dr. Pedro Bonillo, tutor de la presente investigación por su labor de apoyo y orientación desde el inicio de la maestría en el desarrollo de este proyecto.

A la Profesora Lourdes Ortiz por ser la guía metodológica y brindarme no solo su experiencia sino también por sus palabras de aliento y ánimo en los momentos difíciles.

A mis compañeros de trabajo por su valioso aporte en conocimientos y experiencias en el campo de la presente investigación.

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
MAESTRÍA EN SISTEMAS DE INFORMACIÓN

TRABAJO DE GRADO DE MAESTRÍA

MODELO ONTOLÓGICO PARA DETERMINAR EL USO DE PATRONES DE SOFTWARE ORIENTADO A DEPARTAMENTOS DE DESARROLLO DE SISTEMAS EN EMPRESAS DEL SECTOR BANCARIO

Autor: Adriano Da Silva A.

Asesor: Pedro Nolasco Bonillo Ramos

Fecha: Junio de 2014

RESUMEN

Los patrones se han utilizado con éxito en el diseño de software para reutilizar soluciones a problemas recurrentes, sin embargo, la cantidad de patrones existentes y la complejidad de combinarlos en distintas soluciones de manera adecuada, puede llegar a convertirse en un producto que no cumple con los requerimientos de los clientes, o bien traducirse en una solución con bajo nivel de adaptación a los cambios y con alta complejidad para su mantenimiento. Este trabajo hace énfasis en desarrollar un modelo que permita optimizar el proceso de selección de los patrones y creación de una base de conocimiento compartida a través del estudio de las ontologías de los patrones y los conceptos de diagramas de modelado. El ciclo metodológico se basa en la investigación-acción, que se implementó en cada una de sus cinco fases y complementándose con la fase de evaluación del método a través de la metodología de evaluación de ontologías como procedimiento de evaluación del producto final. El producto final se desarrolló con el objetivo de hacer más eficiente el proceso de construcción de productos de software de calidad haciendo uso de técnicas, metodologías y herramientas basadas en las buenas prácticas comúnmente aplicadas en el campo del proceso de desarrollo de sistemas.

Palabras clave: Patrones, Ontologías, Antipatrones, *frameworks*, desarrollo orientado a modelos. Línea de investigación: Ingeniería del Software

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

Figura **Página**

ÍNDICE DE TABLAS

LISTA DE ACRÓNIMOS Y SIGLAS

ACM	Association of Computer Machinery
ABAS	Attribute-Based Arquitectural Styles
ADL	Architectural description languages
BD	Base de Datos
BPM	Bussines Process Managament
CC	Componentes conectados
Coh	Cohesion
CMMI	Capability Maturity Model Integration
DESMET	A method for evaluating Software Engineering methods and tools
CU	Class Utilization
ETL	Extract, Transform and Load
FS	Fábrica de Software
GoF	Gang of Four
IEC	International Electronic Commission
IEEE	Institute of Electrical and Electronics Engineers
IR	Inheritance Richness
ISO	International Organization os standardization
ITIL	Information Technology Infrastructure Library
J2EE	Java 2 Enterprise Edition
KB	Knowledge base
LISI	Laboratorio de Investigación en Sistemas de Información
MOSEFPIV	Modelo Sistémico De Evaluación Financiera De Proyectos de Inversión de Tecnología de la Información en Venezuela
OWL	Web Ontology Language
OntoQA	Onto Quality Assurance Software para evaluar ontologías a través de métricas.
RD	Relationship Diversity
RDF	Resource Description Framework
SIBO	Sistema de Información basado en ontologías
SQuaRE	Software Product Quality Requirements and Evaluation
TIC	Tecnologías de información y comunicaciones
UP	Unified Process
VCBU	Venezolano de Crédito Banco Universal
XML	Extensible markup language
XP	Extreme Programming
.NET	Framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones.

INTRODUCCIÓN

El desarrollo de software en toda empresa, debe estar enfocado en la necesidad de garantizar la calidad en dos aspectos claves como son: sus procesos y sus resultados. Para ello, debe buscar optimizar sus procedimientos y reorientar sus estrategias que permiten en primer lugar lograr la madurez de sus procesos y en segundo lugar obtener ventajas en un entorno muy competitivo para aprovechar al cien por ciento los conocimientos y experiencias de las personas de manera que el conocimiento adquirido pueda persistir en el tiempo y a su vez ir perfeccionándose con nuevas técnicas y estrategias.

Lograr un conocimiento compartido y que éste pueda ser representado en componentes que pueden ser implementados de forma repetitiva en diferentes situaciones de manera eficiente es una tarea difícil, esto motivado al creciente número de patrones y componentes de software que se han venido creando a lo largo del tiempo. Este es uno de los principales problemas que intervienen en el proceso de desarrollo de software de calidad y que se busca lograr con esta investigación para obtener herramientas de apoyo que permitan evaluar y tomar las mejores alternativas que puedan presentarse en una determinada situación.

Este modelo surge como resultado de cumplir con cada una de las actividades y fases enmarcadas en el *framework* metodológico propuesto por el Laboratorio de Investigación en Sistemas de Información LISI-USB dentro de un tiempo planificado donde se llevó a cabo el desarrollo del proyecto para representar el conocimiento obtenido aplicándolo a la práctica con la finalidad de obtener el modelo ontológico de referencia.

Las ventajas más importantes de esta investigación son: en primer lugar ofrecer un conjunto de recomendaciones para mejorar la base de conocimientos empresariales para el uso eficiente de los *frameworks*, componentes y patrones desarrollados y de esta manera

contribuir con la calidad del proceso y del producto en el desarrollo de software y en segundo lugar la aplicación del modelo en una organización.

Este trabajo se estructura de la siguiente manera, el capítulo I establece el contexto del tema de investigación, el capítulo II describe un Marco teórico y el capítulo III establece los antecedentes de otras investigaciones que apoyaron este proyecto. El proceso de desarrollo de la investigación comienza con el análisis de una situación real a través de una encuesta que se esboza en el capítulo IV. En Los capítulos V y VI se desarrolla la propuesta y su validación con el uso de los métodos y herramientas propuestas. Finalmente se presentan las conclusiones, recomendaciones, referencias bibliográficas de la investigación y los anexos para complementar y dar soporte a la información que se detalla en cada capítulo.

CAPÍTULO I. EL PROBLEMA DE INVESTIGACIÓN

El objetivo de este capítulo es dar a conocer el planteamiento del problema, objetivos, justificación, limitaciones del proyecto, esto con el propósito de conocer el contexto del tema de investigación.

1.1 Planteamiento del problema

Los patrones de software representan soluciones comprobadas a problemas recurrentes, esto se enmarca en un conjunto de buenas prácticas que garantizan el correcto desempeño en las distintas fases del ciclo de desarrollo de sistemas y que establecen métodos y herramientas de trabajo común para desarrollar, definir y compartir el conocimiento.

Se trata de utilizar eficientemente el conocimiento a través de los distintos patrones identificando sus interacciones y la manera en que pueden acoplarse las distintas soluciones en el desarrollo del software lo que se traduce finalmente en el producto requerido en el tiempo estimado y según los costos esperados, tal como se refleja en Grimán, Pérez, Mendoza, Domínguez . (2005). En lo que refiere a calidad del software, el producto final debe cumplir en primer lugar, con los requerimientos funcionales exigidos y en segundo lugar con los principios de la ingeniería de software indicados en IEEE Computer Society, (2004, pp. 2,3) relacionados con la abstracción, el bajo acoplamiento y alta cohesión, descomposición y modularización, lo que garantiza la calidad en el producto final

Actualmente los patrones y los *frameworks*, estos últimos referenciados por Flores (2005) como uno de los más complejos artefactos orientados a objetos en el desarrollo de software (p.2), son poco usados o son usados de manera aislada por los grupos de trabajo en las organizaciones, esto se debe en parte, a que su implementación resulta costosa y requiere de mayores esfuerzos de los grupos de desarrollo para lograr componentes reutilizables. Como resultado se tiene que no existen las condiciones para un uso adecuado de los componentes y que no se transmite el conocimiento entre los grupos de trabajo, esto

conlleva inevitablemente a que no exista desde un punto de vista general, métodos, procesos y herramientas adaptadas a las necesidades de las organizaciones, que se puedan ir perfeccionando en el tiempo y la no reutilización de componentes de software existentes, con todo lo que esto implica en tiempo y costos para las empresas en el proceso desarrollo de los proyectos. Todo lo anterior indicado por Hall, Hung,, (2001 cit. en González, 2005) como algunos de los problemas típicos en el desarrollo de software.

Por otra parte, existe un número considerable de patrones que, en la práctica dificultan la identificación adecuada de las soluciones a implementar, adicionalmente existen similitudes y a su vez leves diferencias en las especificaciones de los patrones, esto implica que fácilmente puedan llevar a los especialistas a una mala elección del patrón o modelo a usar, originando un antipatrón. Según Budgen (2003 quién cita a Long, 2001) "Un antipatrón es una solución obvia pero errada a problemas recurrentes en el desarrollo de software, (p.225)". Esto implica que el producto final pueda representar un problema mayor a mediano plazo o bien a una solución poco escalable. De igual manera, estos antipatrones también deben ser documentados y relacionados en su contexto para evitar ser recurrentes en situaciones similares.

Citando a Wohed, (1998) y Hamza (2004), uno los retos de usar los patrones de análisis es que los patrones necesitan ser lo suficientemente generalizados o abstractos para ser aplicados en muchos tipos de modelos y por otra parte, requieren ser lo suficientemente sencillos para ser aplicados desde el inicio. (pp. 3,4). Agrega Hamza (2004) cuatro razones presentes en el proceso de desarrollo que influyen en el uso adecuado de los patrones estos son 1) Inconsistencia en la estructura en cada uno de los diferentes tipos de patrones, 2) Redundancia de patrones similares, 3) Exceso de formalismo en la descripción de los mismos y 4) límites de los problemas a los cuales son aplicables los patrones. (p.4).

Por otra parte, definir la base de conocimientos de los distintos patrones implementados por las áreas de desarrollo, representaron además un avance significativo en cuanto a los tiempos requeridos para el desarrollo de los sistemas, la optimización en el uso de los recursos en función del beneficio y la definición de un conjunto de herramientas

reutilizables que van a ir perfeccionándose de los conocimientos del grupo. En Medina, (2010) se hace énfasis en la necesidad en la reutilización de los componentes y gestionar ese conocimiento como activo de la información. (p.6)

Mantener el conocimiento en el tiempo es un requerimiento clave en el problema que se plantea en este estudio a través de un repositorio de ontologías de patrones. Entendiéndose una Ontología como: "Una especificación formal y explícita de una conceptualización" Toward (2011, p1) y un repositorio de ontologías según Ding (2006) actúa como un repositorio corporativo de conocimiento que apoya el registro de la semántica del conocimiento implícito o tácito de la información, tales como los modelos de datos, modelos de negocio, lógica de negocio y las políticas.

1.2 Formulación de las interrogantes de la investigación.

1. ¿Cómo seleccionar de manera adecuada el marco trabajo en el proceso de desarrollo de software que mejor se adapta al problema planteado?
2. ¿Es posible unificar conocimientos en un conjunto de procesos, técnicas y buenas prácticas para determinar el uso de patrones en el proceso de desarrollo de software en una organización?
3. ¿Lograr que la especificación desarrollada no esté abierta a múltiples interpretaciones y restringir la libertad de criterios, puede incrementar las posibilidades de una mejor comprensión y la reducción de los llamados antipatrones?
4. ¿Cuál es la problemática existente en relación a selección de los patrones y *frameworks* de desarrollo de software que mejor se adaptan al proyecto según sus características?
5. ¿Es posible representar la estructura de los sistemas, su comportamiento y el modelo de datos a usar y además describir la lógica de negocio y restricciones de software en un mismo modelo ontológico?
6. ¿Cómo se puede validar el modelo ontológico obtenido mediante su aplicación en un caso de estudio?

1.3 Objetivo general.

Desarrollar un modelo ontológico para determinar el uso de patrones software orientado a departamentos de desarrollo de sistemas en empresas del sector bancario

1.4 Objetivos específicos.

1. Diagnosticar el problema relacionado a la calidad del software desarrollado y la necesidad de recurrir a mejores prácticas para la reutilización de los componentes de software y optimización de los procesos de desarrollo de software.
2. Elaborar la taxonomía de los patrones de manera ontológica para agrupar el conocimiento de dominio según área de aplicación en el uso de patrones, componentes y *frameworks* en base a la metodología para elaboración de ontologías que indica McGuinness (2005).
3. Desarrollar el modelo del repositorio de patrones donde se definirá la base de conocimientos obtenidos por los modelos a representar en ellos.
4. Definir un marco de trabajo que proporcione una estrategia que describa los procesos para el uso, reutilización y difusión del conocimiento en el proceso de desarrollo de sistemas en una organización.
5. Evaluar el modelo en un caso de estudio de una empresa del sector financiero.
6. Evaluar el modelo propuesto para especificar el aprendizaje.

1.5 Justificación

El desarrollo de un modelo de ontologías permitirá representar el conocimiento del dominio en un lenguaje formal, usando los lenguajes apropiados los análisis de patrones que se representen podrán ser fácilmente identificados dentro del contexto y el modelo del sistema al cual se quiere aplicar como una solución factible.

Permitir la reutilización del conocimiento del dominio en el proceso de desarrollo de sistemas va a orientar a los grupos de trabajo a la construcción de software basada en componentes, según se indica en Bonillo, (2006 citando a Dsousa et al. 99. p.146):

"Cuando se habla de componentes, se pueden encontrar algunas definiciones relacionadas con la implementación, donde se encuentran aquellas que entienden por componente un paquete coherente de código que: (i) puede ser desarrollado y distribuido independientemente, (ii) tiene interfaces explícitas y bien especificadas para el servicio que ofrece, (iii) tiene interfaces explícitas y bien especificadas para el servicio que espera de otros componentes, y (iv) puede ser compuesto junto con otros componentes, quizás extendiendo alguna de sus propiedades, pero sin modificar al componente propiamente dicho". Estos componentes se pueden acoplar con otros patrones y formar soluciones más complejas como los *framework*.

Otro aspecto importante es que permite a las organizaciones catalogar su experiencia en las fases de desarrollo de manera que pueden detallarla en bloques de conocimientos que no son más que el conocimiento acumulado originado por el trabajo de sus colaboradores transformado en ideas, aprendizajes, experiencias en una estructura natural e independiente de la estructura tecnológica. Este aspecto se menciona en Hamza (2004 p.6)

Lo anterior conlleva necesariamente a facilitar las decisiones relacionadas con la elección de un conjunto de componentes entre varias opciones y las posibles combinaciones entre ellos. Contribuye a disminuir los tiempos en el desarrollo al mantener información accesible para todos, relacionada con los distintos patrones y soluciones comprobadas de software.

Evitar los antipatrones. Según el autor, Gamma et al. (1994), un antipatrón es un patrón de diseño que invariablemente conduce a una mala solución para un problema. Al documentarse los antipatrones, además de los patrones, se dan argumentos a los diseñadores de sistemas para no escoger malos caminos, partiendo de documentación disponible en lugar de simplemente la intuición.

Por todo lo anteriormente descrito, se considera que el estudio exhaustivo de patrones y su documentación explicita en ontologías para la construcción de repositorios de ontologías, contribuyendo en gran parte a incrementar el uso de los patrones, beneficiando los procesos

de desarrollo, contribuye a la reutilización y agilización de los procesos de construcción de software por ensamblaje, donde un 30% del desarrollo sea realizado por los expertos y el 70% sea completado por componentes probados y escalables. Esto aporta una alto grado de estandarización (Greenfield & Short, 2004), ya que el uso de patrones y componentes están asociados a las características especificadas en el estándar ISO/IEC 9126 y especificados en el modelo de calidad de software de Grimman, (2006).

1.6 Limitaciones.

El Modelo desarrollado aporta soluciones a los problemas ya indicados en el presente documento; este modelo está asociado al proceso de desarrollo de sistemas en empresas del sector financiero en sus áreas de desarrollo de software, ya que en estas instituciones es donde existe la producción de software a gran escala, existe un alto nivel de estandarización y seguridad, y se presenta el concepto de distribución de trabajo y automatización.

No obstante es aplicable en menor grado de complejidad en empresas de otros sectores como industrial, comercial, educativos, de salud, etc. No es aplicable esta investigación en instituciones bancarias donde en el proceso de desarrollo de sistemas, se usan tecnologías que no soportan el concepto de patrones y reutilización de componentes, esto en vista que estas organizaciones tienen otras metodologías, herramientas y procedimientos y conceptos para los cuales no aplica esta investigación.

Desde un punto de vista de tiempo y recursos económicos disponibles para el desarrollo de este proyecto, se determina que en la organización donde se desarrolló el caso de estudio, objeto de la a presente investigación, se realizó en un lapso de tiempo no mayor a un año, una vez iniciado el proceso. Este proyecto se desarrolló con el uso del presupuesto disponible en la V.P. de Sistemas del Venezolano de Crédito Banco universal para cubrir gastos de investigación, gastos operativos y de logística durante el tiempo de permanencia en la empresa.

1.7 Alcance

El desarrollo de este proyecto se realizó a través de un caso de estudio enfocado a tres empresas del sector financiero, una de las empresas donde se aplica el modelo y se evalúan los resultados es el Venezolano de Crédito B.U. como una de las instituciones estudiadas y donde se identifica como una institución financiera de amplia trayectoria en Venezuela donde sus desarrollos de software se caracterizan por ser productos de software adaptados a los nuevos requerimientos de seguridad, calidad y buenas prácticas en la construcción de software. Es una organización donde se hace uso de nuevas tecnologías que le permiten ser más competitivos en el mercado, su elección entre otras razones, se debe a que en esta empresa se desarrolla el 80% de sus aplicaciones bajo ambientes de sistemas abiertos en lenguajes orientados a objetos, el 20 % restante está desarrollado en ambientes Mainframe y software de terceros y que está en proceso de ser migrado a las nueva plataforma de sistemas abiertos. De tal manera que se dispone de un amplio grupo de profesionales en el área de arquitectura y desarrollo de sistemas y distintos expertos en el área que se desempeñan como consultores externos de la organización. La investigación se amplía a distintas empresas del sector con la finalidad de obtener los conocimientos y prácticas usadas en el proceso de desarrollo de software lo que da un excelente punto de arranque para obtener información clave en el proceso de captura, análisis y resultado del estudio que se está presentando en este documento.

Este proyecto está orientado a elaborar un modelo ontológico que abarque el proceso de desarrollo de la ontología de los distintos patrones para agrupar el conocimiento de dominio del área de aplicación para los patrones de análisis, patrones de arquitectura, de diseño o interacción.

Esto quiere decir que la elaboración de las distintas herramientas de software de ayuda y/o *frameworks* de trabajo que permitirán hacer uso de los conocimientos plasmados en el repositorio de ontologías, podrían ser elaborados en otro proyecto basándose en los resultados obtenidos en éste y otros estudios relacionados con el tema de ontologías en la construcción de software.

CAPÍTULO II. MARCO TEÓRICO

En este capítulo se define el contexto teórico dentro del cual se desarrolló esta investigación. Se inicia con el mapa mental del modelo ontológico que permitirá obtener una idea generalizada de los distintos elementos que conforman las tecnologías que se concentran en la propuesta y como se relacionan cada una de ellas. Posteriormente se enmarca toda la investigación en la teoría que sustenta el uso de patrones de diseño en el proceso de desarrollo de software y los antecedentes relacionados con investigaciones realizadas en el campo de las ontologías y patrones de software, así como también se detalla una representación estadística en el uso de patrones en el proceso de desarrollo de software.

2.1. Mapa mental del marco teórico del modelo ontológico para determinar el uso de patrones de software orientado departamentos de desarrollo de software del sector bancario.

En las figuras 1,2 y 3 se presentan los mapas mentales con los tópicos más importantes relacionados con la teoría que fundamenta la investigación. El tema central es el desarrollo de modelo ontológico para determinar el uso de patrones de software. En el sentido de izquierda a derecha y de arriba hacia abajo, se esbozan los antecedentes relacionados a la presente investigación, los objetivos de esta investigación y los dos grandes aspectos que detallan la información, los patrones de software, su clasificación, elementos que lo conforman, beneficios y características. Finalmente las ontologías se detallan en cuanto a su definición, tipos, lenguajes usados, los criterios de diseño y los pasos para su desarrollo. Temas ampliamente desarrollados en la presente investigación.

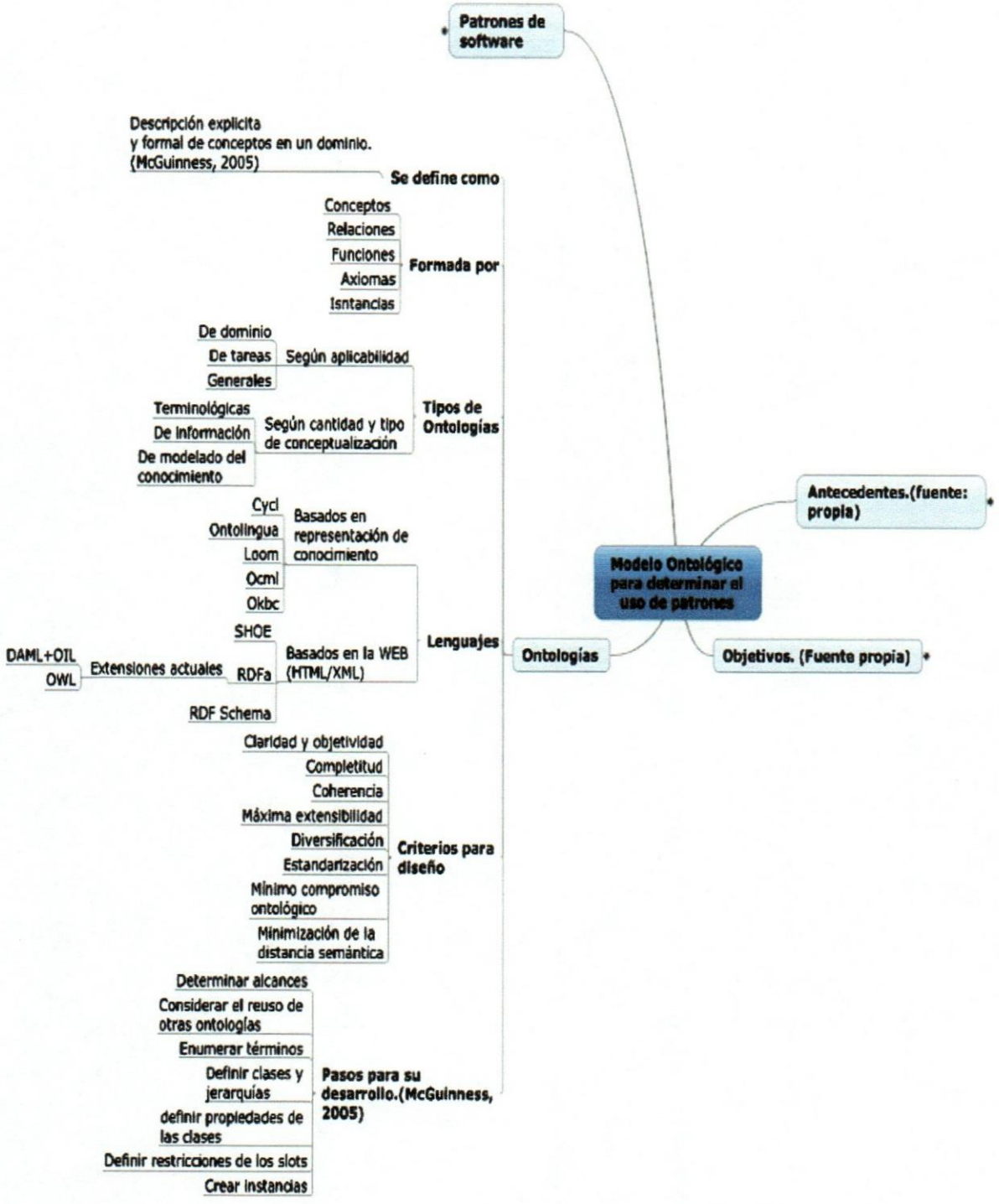


Figura 1. Mapa mental del marco teórico, parte 1

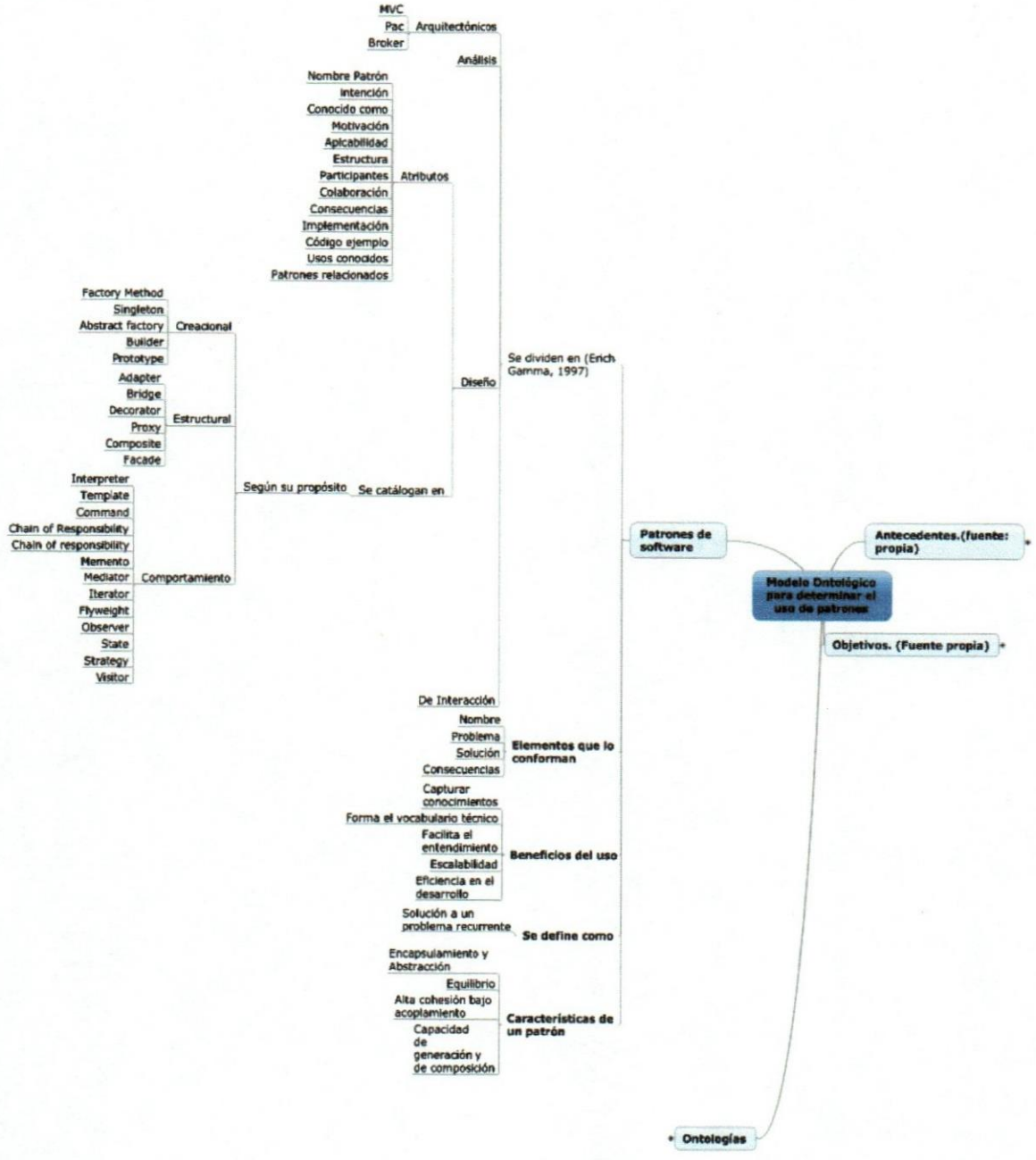


Figura 2. Mapa mental del marco teórico, parte 2

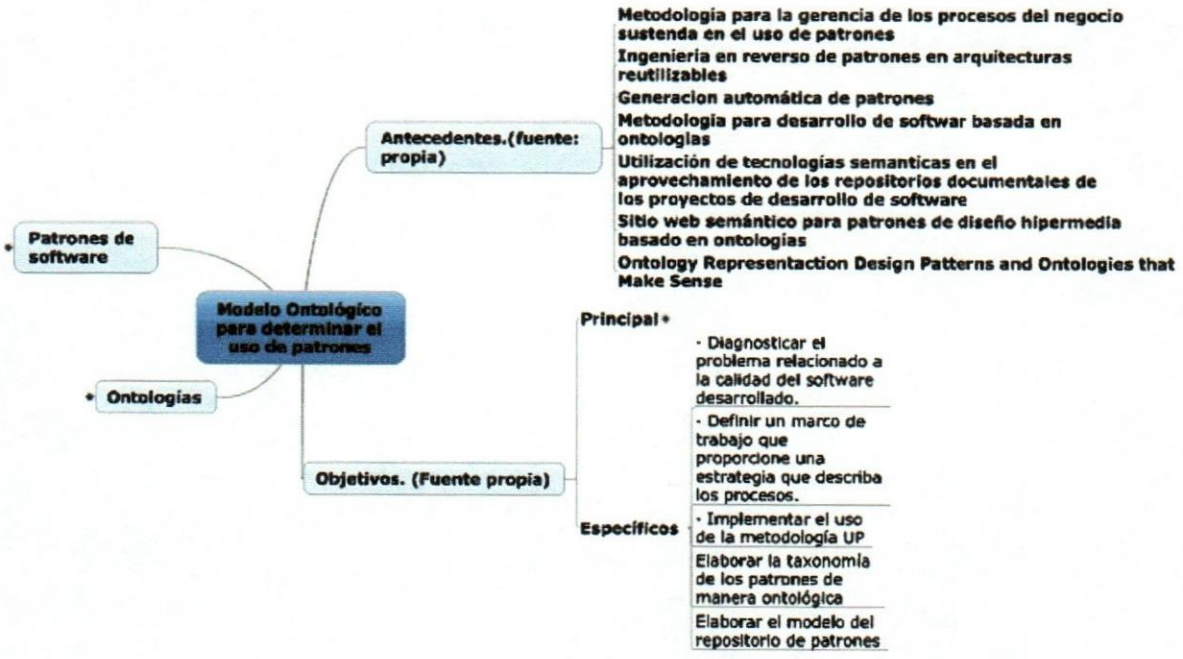


Figura 3. Mapa mental del marco teórico, parte 3

2.2. Antecedentes.

A continuación se detallan las distintas investigaciones consideradas en la presente investigación como aportes relacionados con el tema de este proyecto:

Tabla 1. Investigaciones y antecedentes considerados en el desarrollo del modelo propuesto

Fuente de información:	Hacia una ontología para fábricas de software. (Griman et al. 2005)
Tipo:	Artículo
Resumen:	Se retoma el concepto de Fábricas de Software (FS) donde la reutilización juega un rol protagónico. Debido a la existencia de visiones diferentes en el área y a pesar de que el concepto de FS no es nuevo en la Ingeniería de Software, aún no se cuenta con la madurez conceptual necesaria como para identificar claramente el tratamiento de ciertas variables dentro del proceso. Por ello, en este artículo se realiza una revisión histórica del concepto FS y se propone una ontología basada en las definiciones más recientes. Este artículo forma parte de una investigación en progreso que pretende precisar la calidad sistémica en compañías desarrolladoras de software que decidan implantar una FS.
Aspectos considerados en esta investigación	*Se destaca el concepto de fábricas de software y las características que debe tener para que pueda cumplir con una línea de producción de software, según los autores citados en el artículo, una fábrica de software contiene desde el punto de*

	vista tecnológico tres ideas claves llamados esquemas, plantillas y ambientes de desarrollo. Posteriormente se incluyen los elementos relacionados con los términos o conceptos de economía en una fábrica de software y en tercer lugar se incluyen las mejores prácticas asociadas a las fábricas de software. La idea presentada de mostrar los modelos descritos de forma agrupada con la finalidad de buscar cuales conceptos se repiten y sirven como integradores permite a efectos de esta investigación, definir y aclarar las familias de productos y posteriormente las arquitecturas a desarrollar para tales familias de productos para obtener frameworks y componentes a ensamblar para cada caso. Aunque no se abordan los aspectos de la calidad de software, es un modelo que complementa la idea de representar el modelo ontológico que se desea elaborar en esta investigación. En otro artículo referenciado en este documento se abordan los conceptos de fábrica de software bajo el estándar ISO 15504.
Fuente de información:	Metodología para la gerencia de los Procesos del negocio sustentada en el uso de Patrones (Bonillo, 2006)
Tipo:	Artículo
Resumen:	"En este trabajo se propone una taxonomía de patrones y su representación a través de una arquitectura de procesos, componentes y aplicaciones en el dominio BPM. Además se extiende la especificación de los patrones propuesta por Acosta y Zambrano (2004), obteniendo un nuevo meta-patrón acorde con la taxonomía, con elementos que permiten medir la calidad de los patrones en el proceso y el producto de software, a través del uso de estilos de Arquitectura basados en atributos (ABAS) y los modelos de calidad ISO14-598 e ISO-9126. Tomando en cuenta esta combinación de métodos, herramientas y técnicas, se propone un conjunto de pasos que en el ámbito de BPM, permiten identificar los procesos claves, modelarlos y analizarlos, simularlos, implantarlos de forma asistida (tanto los nuevos procesos como sus versiones), evaluarlos, monitorearlos y mejorarlos. Con la finalidad de poner en práctica la metodología se presenta un caso de estudio con el proceso de Gestión de Versiones de la Librería de Infraestructura de Tecnología de la Información (ITIL)." (Bonillo, 2006)
Aspectos considerados en esta investigación	Esta tesis doctoral presenta varios aspectos a considerar, en primer lugar representa de manera clara la relación entre patrones, componentes, aplicaciones y procesos donde cada uno de estos de manera que cada uno de ellos se expresan de manera detallada y enmarcada en un entorno de calidad. En segundo lugar la propuesta busca representar una taxonomía de patrones a través del lenguaje de definición de arquitecturas ADL. Adicionalmente se extiende la especificación de los patrones con la finalidad de medir la calidad de los procesos mediante estilos de arquitectura basados en ABAS y ampliando con la creación de un meta-patrón basado en la definición de (Acosta, 2004) que permite la descripción de patrones, este enfoque es importante para esta investigación porque demuestra aspectos relacionados a patrones que se deben considerar en la creación de las facetas y los slots de la ontología a crear.
Fuente de información:	Rpao: repositório de padrões de análise mapeados em ontologias. (Tonietto, 2007)
Tipo:	Trabajo de grado de maestría
Resumen:	La aplicación de modelos de análisis tiene como objetivo promover la reutilización, eficiencia y buenas prácticas de ingeniería de software. La reutilización de componentes proveniente de la experiencia de ingenieros de software para identificar patrones y describir el modo que otros proyectos también hacen uso de los análisis estándar ya identificados. La ingeniería de requisitos tiene una gran influencia en el descubrimiento de patrones de análisis. En este trabajo se propone la creación de un repositorio de análisis de los patrones, ya que el mercado no ofrece una herramienta de almacenamiento de propicio para los patrones de análisis y mapeo de las ontologías formato OWL. Las ontologías son generadas por el software cada vez que un patrón está catalogado, por lo tanto,

	todo el estándar documentado está disponible para una ontología con la estructura del patrón. Además de los patrones de almacenamiento de análisis, la herramienta realiza un mapeo requisitos y artefactos, que potencialmente causan un defecto, lo cual representa una base de conocimientos de los patrones de análisis.
Aspectos considerados en esta investigación	Para efectos de la presente investigación, se destaca el uso de dos metodologías para el desarrollo de este proyecto. En primer lugar, basándose en el proceso unificado para todo el control del ciclo del proyecto y siguiendo todas sus fases así como también para la representación en el modelo orientado a objetos de clases para representar los patrones y sus relaciones y la representación de los modelos de negocio que son llamados en UP "workflows" base. Este enfoque es importante para este proyecto ya que ejemplifica la forma de implementar la metodología en proyectos de creación de repositorios de patrones de análisis, mostrando las disciplinas del proceso unificado y los modelos o artefactos que se generan en cada uno de ellos. Finalmente se combina con la metodología para desarrollo de ontologías que permitió elaborar todo el desarrollo de la conceptualización del estudio realizado en OWL.
Fuente de información:	Ingeniería en reversa de patrones en arquitecturas reutilizables. (Flores, 2005)
Tipo:	Trabajo de grado de maestría
Resumen:	El proceso de ingeniería en reversa que se propone en esta disertación, se basa en un enfoque de recuperación que realza los elementos que implementan la flexibilidad de un *framework*, ayudando a su reutilización y aprendizaje. Los frameworks orientados a objetos son una poderosa técnica de reutilización de software a gran escala. A través de elevados niveles de reutilización de código y diseño, los frameworks permiten aumentar la productividad, y reducir los tiempos de desarrollo aumentando la calidad de las aplicaciones. Sin embargo, antes de ser capaz de utilizar un framework de forma eficaz, es necesario dedicar tiempo y esfuerzo en aprender los detalles esenciales de la arquitectura y diseño. La información en el diseño de la estructura se pierde con el tiempo y se torna difícil de obtener. frameworks y patrones de diseño son dos conceptos fuertemente relacionados que representan niveles de abstracción diferente, el nivel de la definición del diseño y la arquitectura de un sistema de software. Por lo general, un framework está compuesto por diversos patrones de diseño. Por el hecho de que estos son extremadamente útiles en su construcción, para hacer frente a un problema recurrente con una solución considera buena al ofrecer mecanismos que promuevan su extensibilidad y flexibilidad. En esta disertación se propuso un enfoque para recuperación de los elementos de diseño y una estructura que dota a la flexibilidad al proceso. El enfoque está basado en un proceso de ingeniería inversa semiautomático dividido en fases a través de las cuales se detectan estructuras de diseño de creciente nivel de abstracción. La representación de los patrones de diseño se basan en una formalización mediante el concepto de meta-patrón, una abstracción de alto nivel que define la estructura y la relación de los elementos constitutivos de los patrones de diseño, los mismos que le dan la flexibilidad en su uso y adaptación. Este enfoque permite que el usuario de un framework tenga una rápida concientización de los puntos de variabilidad, a través de resultados que ayudan a la efectiva reutilización del código y el diseño del framework como su propósito principal.
Aspectos considerados en esta investigación	Este trabajo tiene un enfoque inverso al proceso que se desea llevar en este modelo ontológico porque parte de los sistemas existentes para identificar los componentes que lo constituyen y sus respectivas relaciones para obtener su diseño. El concepto de meta-patrón (patrón de patrones) se utiliza como la estructura esencial para definir un *framework* específico. Su aporte a este proyecto está en que precisamente estos meta-patrones brindan una forma macro para documentar relaciones de clases esenciales y la manera como fue abordado este concepto considerando los siete tipos de meta-patrones clasificados según las

	relaciones entre clases y como estas se conectan entre sí, estas son: conexión 1:N, conexión 1:1, conexión recursiva 1:N, conexión recursiva 1:1, unificación, unificación recursiva 1:1, unificación recursiva 1:N, estos puntos deben ser considerados dentro de la ontología para representar en ella los distintos tipos de meta-patrones y sus atributos.
Fuente de información:	Una ontología para la representación de conceptos de diseño de software (Giraldo, Gloria et.al 2011)
Tipo:	Artículo
Resumen:	En los últimos años se integraron conceptos importantes al diseño de software, tales como nuevos diagramas de modelado y patrones, creados con el fin de evolucionar el diseño de software y mejorar la calidad de las aplicaciones. Los conceptos incorporados se presentan a los desarrolladores en extensa documentación. La cantidad de información resultante se ha incrementado de tal manera que es difícil manejarla. Las investigaciones hacen énfasis en mejorar y crear nuevos conceptos de diseño, pero existen pocas iniciativas orientadas a estructurar este conocimiento. Un claro ejemplo de la necesidad de estructurar este conocimiento son los patrones GoF, pues su éxito no fue su creación sino su recopilación desde los trabajos de otros desarrolladores. En este artículo se propone una ontología para organizar el conocimiento en diseño de software, incorporando conceptos de diagramas de modelado y patrones, Con el fin de recopilar algunos conceptos de diseño a tener en cuenta en un proyecto de software.
Aspectos considerados en esta investigación	Se presenta una ontología para la representación del conocimiento de diseño de software, para todos los patrones de diseño, diagramas UML y patrones GoF (Grupo de los cuatro por sus siglas en inglés). Para elaborar este modelo ontológico los autores sugieren una serie de pasos para desarrollar una ontología que permite construir las clases, objetos y atributos. Se grafican todas las clases consideradas para el modelo, los slots y las instancias. La ontología es abierta y bajo este diseño es fácilmente integrable nuevos patrones y nuevos conceptos.
Fuente de información:	An ontology for architectural evaluation Case study: collaboration systems. (Grimman, 2006)
Tipo:	Artículo
Resumen:	Barbacci et al. (1995) a firman que el desarrollo de distintas maneras sistemáticas para relacionar los atributos de calidad de un sistema a su arquitectura, constituye la base para la toma de decisiones objetivas sobre los acuerdos de diseño, y ayuda a los ingenieros a hacer predicciones razonablemente precisas en cuanto a los atributos del sistema, libre de prejuicios y suposiciones que no son triviales. El objetivo es ser capaz de evaluar cuantitativamente la arquitectura de llegar a acuerdos entre múltiples atributos de calidad y así alcanzar a nivel mundial un sistema mejor. Sin embargo, los elementos necesarios para incorporar esta evaluación en diferentes tipos de modelos de desarrollo, no están claras. En este trabajo se propone una ontología para conceptualizar los problemas inherentes a la evaluación arquitectónica dentro de un proceso de desarrollo, lo que ayuda a identificar el alcance de la evaluación, así como los temas que se garantizan para conseguir la eficacia en los diferentes procesos de desarrollo, ágil y riguroso. La principal conclusión de la investigación permitió identificar los elementos de interacción entre el proceso de desarrollo y un método de evaluación de arquitectura, estableciendo los puntos inicial y final, así como los insumos necesarios para la incorporación en los diferentes tipos de procesos. Esta interacción fue validada a través de un estudio de caso y una metodología de desarrollo de sistemas de colaboración.
Aspectos considerados en esta investigación	Varios aspectos resaltan en esta investigación como el uso de la metodología AMENITIES creada para el análisis y diseño de sistemas inspirado en el proceso unificado para el desarrollo de sistemas distribuidos. La cual se ve asistida sistémicamente al dirigir el análisis y diseño de sistemas cooperativos facilitando

	el desarrollo. Esta propuesta comprende un conjunto específico de modelos y fases a seguir en el proceso que va desde el análisis e identificación de requerimientos, hasta el desarrollo de software. Los resultados obtenidos según los autores determinaron que el uso de la metodología permitió: La necesidad de tener un proceso de desarrollo bien definido en términos de actividades y artefactos y su seguimiento. La ventaja que representa aplicar una notación estándar y un modelo formal así como también el uso de múltiples visiones en arquitecturas de software.
Fuente de información:	Aplicación informática para gestionar repositorios, establecer semejanzas y caracterizar ontologías. (Rojas, 2012)
Tipo:	Artículo
Resumen:	El objetivo de la investigación es desarrollar una aplicación informática que gestione repositorios de ontologías, caracterice y establezca semejanzas entre ontologías representativas del conocimiento en la web. Para ello se realizó un análisis y selección de las tecnologías y herramientas adecuadas para la gestión de ontologías y el desarrollo de la aplicación. Se identificaron y documentaron las funcionalidades de la aplicación y se realizó el diseño implementación y pruebas de la misma. Esta aplicación permitirá gestionar repositorios de ontologías, obtener características y semejanzas entre las ontologías, para facilitar los procesos de almacenamiento, búsqueda y recuperación de ontologías, así como la reutilización del conocimiento almacenado en ellas.
Aspectos considerados en esta investigación	Cabe destacar en este artículo el uso de la metodología XP usada con la finalidad de agilizar los procesos de desarrollo del software. Otros aspectos muy importantes que requieren ser investigados con mayor detalle para esta investigación es el uso de las herramientas y tecnologías basadas en código abierto para el manejo del lenguaje de ontologías (OWL). La herramienta desarrollada permite la comparación de ontologías con el propósito de estudiar la diversidad de los modelos e identificar fortalezas y debilidades, para esta actividad los autores definieron un conjunto de métricas considerando semejanzas en aspectos como su taxonomía, la estructura conceptual interna (Propiedades, roles, relaciones entre ellas), lógica de axiomas, etc. basándose en un porcentaje de semejanza. Este enfoque de comparación puede ser abordado en esta investigación para comparación de patrones, y componentes.

2.3. Sistemas de Información

Según definición de Laudon (2012) es un conjunto de componentes interrelacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control de una organización, ayudar a gerentes y trabajadores a analizar problemas, visualizar asuntos complejos, crear productos nuevos.

2.4. Diseño de software

Para Pressman (2010) el diseño de software es un proceso iterativo por medio del cual se traducen los requerimientos en un "plano" para construir el software. El proceso de diseño está relacionado con distintas características vinculadas al uso de herramientas y la comunicación. Desde el punto de vista de las herramientas, son artefactos empleados para

la creación de otros artefactos que hoy en día en el proceso de diseño de software puede ser un compilador o un conjunto de componentes que sirven de apoyo para la construcción de piezas de software más complejas y adaptables a nuevos requerimientos. Desde el punto de vista de la comunicación, es una herramienta esencial para poder llevar requerimiento del solicitante que parte de un prototipo extraído del proceso de captura de información hacia el equipo de trabajo. Este último aspecto comunicacional es usado por los grupos de desarrollo para transmitir la experiencia desde el experto al novato, así como también para compartir el conocimiento y fomentar la reutilización de ideas y aprovechar el trabajo realizado por otros, todo esto es posible si se combinan las herramientas adecuadas, herramientas que se detallarán a continuación y que van desde lo más esencial de un patrón de software hasta lograr la creación de una amplia de una base de conocimientos, que puedan ser usados de manera eficiente a través de la especificación exhaustiva de sus características para su uso y así realimentar el conocimiento en una propuesta de repositorio de ontologías.

2.5. Calidad de software

La calidad de Software para Pressman (2010) es "Proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan." (p.344). No obstante la ISO/IEC (International Standard Organization) define la calidad de software como la totalidad de rasgos y atributos de un producto de software que le apoyan en su capacidad de satisfacer sus necesidades explícitas o implícitas (ISO/IEC 9126, 1998).

2.6. Calidad del Producto y del Proceso de desarrollo de software

Al intentar definir el concepto de calidad del software se debe diferenciar entre la calidad del Producto de software y la calidad del Proceso de desarrollo del mismo. No obstante, las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto (Scalone, 2006).

Hay un vínculo claro entre la calidad del proceso y del producto en producción debido a que el proceso es relativamente fácil de estandarizar y monitorizar.

Cada sistema de producción se calibra, y debe producir una y otra vez productos de alta calidad. Sin embargo, el software no se manufactura, sino que se diseña. El desarrollo de software es un proceso más creativo que mecánico, donde las 11 experiencias y habilidades individuales son importantes. La calidad del producto, sea cual fuere el proceso utilizado, también se ve afectada por factores externos, como la novedad de una aplicación o la presión comercial para sacar un producto rápidamente.

La Calidad del Software debe implementarse en todo el ciclo de vida del mismo. Las distintas actividades para la implantación del control de calidad en el desarrollo de software son: (1) Aplicación de metodología y técnicas de desarrollo, (2) Reutilización de procesos de revisión formales, (3) Prueba del software, (4) Ajustes a los estándares de desarrollo, (5) Control de cambios, mediciones y recopilación de información; y (6) Gestión de informes sobre el control de calidad. (Scalone, 2006).

Para poder implementar un modelo de calidad se debe aplicar un modelo o estándar que requiere de una Gestión de la Calidad del Software. La Calidad se logra a través de la Gestión de la Calidad, la cual, según ISO 9000:2000, consiste en la realización de actividades coordinadas que permiten dirigir y controlar una organización en lo relativo a la calidad.

2.7. El Standard ISO / IEC.

La ISO (Organización Internacional de Normalización) y la IEC (La Comisión Electrotécnica Internacional) forman el sistema especializado para todo el mundo estandarización. Los organismos nacionales que son miembros de ISO e IEC participan en el desarrollo de las Normas Internacionales a través de comités técnicos establecidos por la organización respectiva, para hacer frente a determinadas esferas de interés mutuo. Otras organizaciones internacionales, gubernamentales y no gubernamentales, en coordinación con ISO e IEC, también participan en el trabajo.

Según Pressman (2010) la calidad del software es "la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implicíticas que se espera de todo software desarrollado profesionalmente"

La calidad del producto junto con la calidad del proceso son los aspectos más importantes actualmente en el desarrollo de Software. En calidad del producto recientemente ha aparecido una nueva versión de la norma ISO/IEC 9126: la norma ISO/IEC 25000. Esta proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE). Es la unión de las dos normas ISO 9126 Y ISO 14598 (Evaluación del Software). El objetivo principal de esta norma es guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad. Establece criterios para la especificación de requisitos de calidad de productos software, sus métricas y su evaluación.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1 que ha sido desarrollado en un intento de identificar los atributos clave de calidad para el software. El estándar identifica 6 atributos clave de calidad:

a. Funcionalidad

Es la capacidad del producto de software para producir funciones que cumplen con los requerimientos establecidos e implícitos cuando el software es usado bajo condiciones específicas. (Lo que hace el software para cumplir los requerimientos)

b. Fiabilidad

La capacidad del producto de software para mantener su nivel de desempeño bajo condiciones establecidas por un período determinado de tiempo

c. Usabilidad

La capacidad del producto de software que permite al usuario entender si el software es adecuado, y la forma en que se puede utilizar para las tareas y las condiciones particulares de uso.

d. Eficiencia

La capacidad del producto de software para proporcionar un tiempo de respuesta adecuado, el tiempo y las tasas de rendimiento durante el procesamiento al realizar su función en las condiciones indicadas. Se trata de un atributo que puede ser medida para cada funcionalidad del sistema.

e. Mantenibilidad

La capacidad del producto de software que se desea modificar para incluir correcciones, mejoras o adaptaciones del software a los cambios en el medio ambiente, en los requisitos y especificaciones funcionales (El esfuerzo necesario para ser modificado).

f. Portabilidad

La capacidad del producto de software para ser transferido de un ambiente a otro. El entorno puede incluir la organización, el entorno de hardware o de software.

2.8. Desarrollo de software basado en componentes

El desarrollo de software basado en componentes de acuerdo Bachmann (2000) se relaciona con la construcción rápida de sistemas a partir de elementos donde los componentes y los *frameworks* tienen propiedades certificadas, estas propiedades proveen la base para predecir las propiedades de los sistemas construidos. Una definición simple pero completa de componentes se expresa en Bachmann (2000) donde se refiere a un componente como una implementeación opaca de funcionalidad sujeta a composición por tercersos y que cumple con un modelo de componentes. En Montilva, Arapé y Colmenares (2003) se detalla un poco este concepto: Un componente se considera una implementeación opaca

debido a que su distribución predominante es en formato binario y sus consumidores lo utilizan como una "caja negra" a través de su interfaz. Dicho aspecto esta alinaeado con el prinicpio de encapsulamiento expresado en IEEE (2004). Por otra parte la implementación por terceros implica que los componentes son intrínsecamente reutilizables debido a que un sistema basado en componentes puede ser ensamblado con relativa facilidad y finalmente la coordinación e interacción entre componentes exige un marco regulatorio estandarizado (modelo de componentes) que establece la infraestructura de software requerida (*frameworks*) y las comvenciones y restricciones de diseño de los mismos.

La construcción de software basado en componentes persigue tres objetivos principales: la reutilización, la adaptación y la extensión (la reutilización implica la adaptación y extensión):

1. Un componente es reutilizable en la medida en que sus servicios pueden ser utilizados por otro software.

2. Un componente es adaptable si su proveedor ha previsto los posibles cambios que puede sufrir dicho componente.

3. Un componente es extensible si su proveedor proporciona los mecanismos para modificar los servicios que ofrece el componente.

2.9. Patrones y colecciones de patrones

Los patrones de software no son más que un conjunto de soluciones a problemas habituales en el diseño de software orientado a objetos. Una definición más formal podría ser: "Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos". Un patrón captura la experiencia y conocimiento de expertos, quienes han producido soluciones exitosas a problemas, a fin que esas soluciones queden a disposición de personas con menos experiencia; sin embargo, los patrones no proveen siempre las soluciones definitivas, algunas veces, los usuarios de patrones deben tener creatividad para utilizar, instanciar o implementar un patrón. (Bonillo, 2006).

Las colecciones de patrones son grupos de patrones débilmente acoplados situados en una ubicación común (depósito, papel, libro, sitio web). La mayoría de las colecciones suelen abordar un conjunto bastante homogéneo de temas y utilizar sistemáticamente una forma de patrón común, un conjunto de atributos utilizados para describir la colección de patrones, Sin embargo la forma de patrones puede variar ampliamente entre las colecciones. Ejemplos de colecciones de patrones son: el Gang-de-Four (GoF) patrones de diseño, los cinco volúmenes Pattern-Arquitectura series Oriented Software (POSA), la colección de usabilidad van Welie, el Portland Pattern Repository, etc. (Henninger y Corrêa, 2007).

Teniendo en cuenta la definición general de patrón se puede afirmar que los mismos, pueden expresarse a través de componentes y estos a su vez representan funcionalidades que se implementan a través de las diferentes aplicaciones.

2.10. Taxonomía de patrones

Los patrones de software no son más que un conjunto de soluciones a problemas habituales en el diseño de software orientado a objetos (Bonillo, 2006). Como un patrón de diseño es una abstracción de una solución en un nivel alto, entonces se considera que patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del proceso de construcción de software, es por esto que en el ámbito de la ingeniería de Software actualmente los patrones pueden aplicarse a nivel del: análisis de requisito, el diseño de la arquitectura, el diseño detallado, la interacción con el usuario y el código. Con lo que puede establecerse la siguiente clasificación. (Bonillo, 2006)

Patrones de análisis: Son grupos de conceptos que representan una construcción común en el mundo del modelado conceptual. Pueden ser relevantes a un dominio o ser adaptados a muchos dominios. La idea central es la construcción de escenarios utilizando patrones. Se pretende tener una visión más conceptual y estructural de las situaciones, con el fin de identificar la naturaleza intrínseca de las mismas. Con esa visión, es posible determinar el tipo de escenario correspondiente a cada situación y así, elegir un patrón de un catálogo, rehusando su estructura con el fin de derivar el escenario más fácil y directamente.

Consisten en un texto guía, que para cada componente del escenario incluye pautas acerca del contenido que deberá tener el mismo. Son presentados como escenarios descritos a los que se ha agregado un reducido número de reglas de conformación como meta componentes del escenario. Cada componente del escenario, según la estructura definida en (Bonillo, 2006 quien cita a Leite et al. 2000), ha sido completado con un texto nominal que se espera sea reemplazado en el escenario real generado al usar el patrón pero que a su vez guíe en la redacción del componente. (Bonillo 2006 citando a Ridao, 2001).

Patrones de arquitectura: Son esquemas fundamentales de organización de un sistema software. Especifican una serie de subsistemas y sus responsabilidades respectivas e incluyen las reglas y criterios para organizar las relaciones existentes entre ellos. (Bonillo 2006 citando a Buschmann et al. 96) •

Patrones de diseño: Son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema. (Gamma, Helm, Jhonson y Vlissides, 1997)

Patrones de Interacción: también conocido como Patrón de Interfaz, describe una solución exitosa a un problema recurrente concerniente a la interfaz de usuario, en un contexto dado. Un Patrón de Interacción es un medio de comunicación que se expresa en una notación sencilla, a fin de ser entendida por las personas del equipo de diseño de la interacción que generalmente es multidisciplinario. (Bonillo, 2006 citando a Mahemoff et al. 98).

2.11. Metapatrones

Un metapatrón es un patrón que conecta a otros patrones, es un patrón de patrones. En la especificación propuesta por Bonillo, (2006) se incoporan especificaciones que documentan el metapatrón como por ejemplo la especificación técnica del metapatron, historial de registro de uso, clasificación del activo según su dominio de uso: específico, general, alcance: Organizacional, de uso comercial, etc. Nivel de seguridad requerido para su uso, ubicación de librerías, etc.

Tabla 2. Metapatrón

Nombre (título), autor clasificación, dominio (usos conocidos) y rango.	**Nombre**: Idea Central por la cual se identifica el patrón. **Autor**: Nombre de la persona que creó el patrón. **Clasificación**: Se refiere al tipo de patrón de acuerdo a la taxonomía antes mencionada. **Dominio**: Indica el o los dominios en los cuales el patrón ha sido implementado. **Rango**: es el grado de confiabilidad de este patrón con respecto a los dominios en los cuales ha sido implementado.
Problema	Describe el problema que será resuelto, desde el punto de vista del usuario.
Solución	Describe, en una forma narrativa y gráfica, la solución al problema. Con base en: Objetivo, Intención, Motivación, Actores o Participantes, Recursos, Estructura, Episodios, Colaboraciones e Implementación.
Atributos de Calidad	Atributos de calidad de interés, el contexto de uso, contrastes y atributos relevantes (requerimientos específicos).
Medidas de atributos de Calidad	Un resumen de lo que es discutido en la descripción del problema, usando términos específicos relacionados con aspectos medibles de los atributos del modelo de calidad. Esto incluye una discusión de los eventos que pudieran hacer que la arquitectura responda o cambie.
Parámetros de atributos de Calidad	Un resumen de lo que será discutido en la sección de solución, pero especificando términos relevantes a los parámetros de los atributos especificados en el modelo de calidad.
Análisis del Modelo de Calidad	Una descripción de cómo los atributos del modelo de calidad serán relacionados formalmente con los elementos del patrón y las conclusiones sobre la conducta arquitectural que se obtiene con el modelo.
Contexto	Presenta las condiciones bajo las cuales el patrón es utilizado.
Fuerza	Conflictos que pudiesen restringir la solución. Incluyendo las excepciones.
Consecuencias	Describe el resultado de aplicar el patrón.
Ejemplos/Contra-ejemplos	Muestra ejemplos y contraejemplos de la solución propuesta.
Patrón Relacionado	Otros patrones que se relacionan con el patrón descrito.

Adaptado de Bonillo, (2006 citando a Acosta et al. 2004 y Kazman, Klein, 2004)

2.12. Estadísticas en el uso de patrones

En Henninger (2007) se citan unas estadísticas interesantes en cuanto al creciente número de patrones en uso. Antes del año 2000, existía un catálogo publicado de más de 1.000 patrones, para entonces se decía que "... Ahora que hay tantos modelos es muy difícil recordarlos a todos" y que "el aumento en el número de patrones de diseño hace un inmanejable vocabulario común".

Desde la fecha, el número de patrones se ha duplicado y se ha creado una diversidad cada vez mayor de colecciones de patrones de temas de desarrollo de software. La figura 4

muestra un ejemplo en términos del año en que se crearon (no se pudo determinar el año de origen de los 9 patrones). (Henninger, 2007)

El Anexo I, muestra una lista de todas las colecciones de patrones que se han utilizado. Esto debe considerarse como una subestimación en el número real de patrones disponibles, ya que es una tarea de enormes proporciones para encontrar todos los patrones en diversas fuentes impresas y electrónicas. En alineación con algunas publicaciones consultadas, se encontraron 1142 patrones hasta el año 2000. Incluyendo un estudio parcial hasta el año 2007, se encontraron otros 1092 patrones, esto evidencia que la tasa de la creación de patrones sigue siendo constante. Aunque algo inconsistente sobre los años 2002 al 2007 como uno de los años más prolíficos, con la excepción de un año en baja en 2006, en la figura 5 se aprecia de forma gráfica la relación de colecciones vs número de patrones por año de creación.

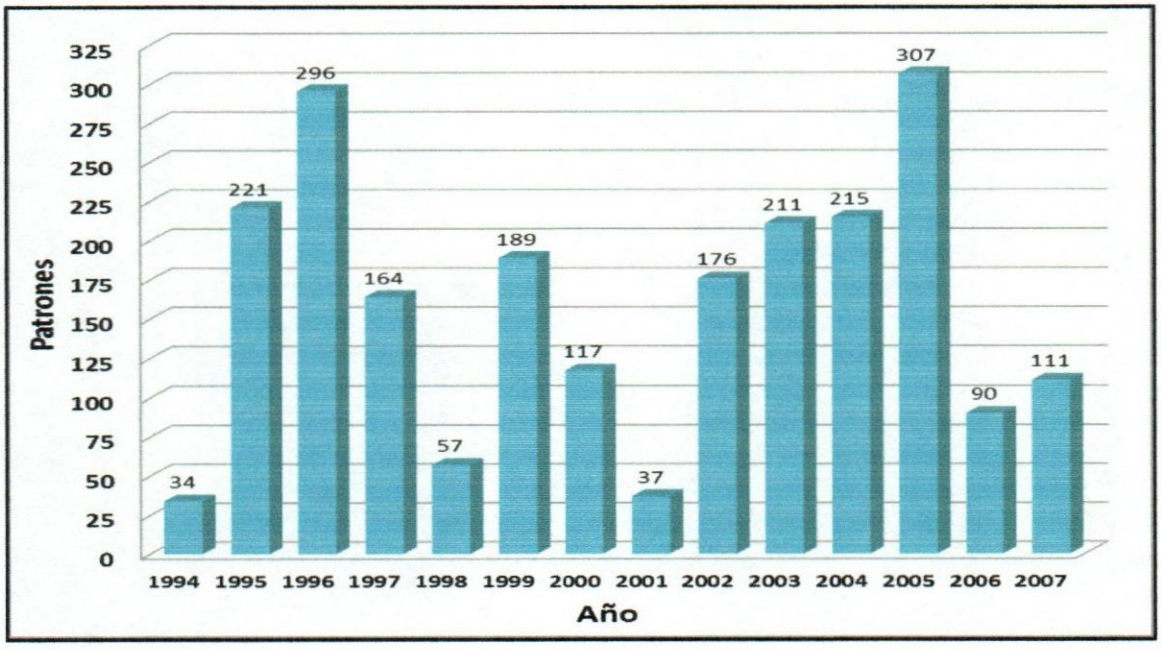


Figura 4. Número de patrones creados desde 1994 al 2007. Adaptado de Henninger (2007)

El tamaño de las colecciones de patrones va de 1 a 146. La Figura anterior revela que las colecciones tienden a ser más pequeñas. Excluyendo los 46 patrones individuales, 70 de

121 colecciones (58%) tienen entre 2 y 10 patrones. Lo más común es 5 patrones en una colección y el promedio es de 18 patrones por colección, esta sesgada por una colección con 146 y dos con más de 90 patrones. El listado de patrones en el Anexo I, se aprecian distintas colecciones de patrones usados en el desarrollo de software, estos se clasifican por el número de patrones y año de creación de la colección a la que pertenecen.

El proceso de selección de patrones es sin duda una labor compleja en vista que la lista es muy amplia y va en constante crecimiento, esto aunado al proceso en selección de la "mejor" opción para el problema que se presenta. En vista de ello se describe a continuación los inconvenientes que se pueden presentar en el proceso de selección de patrones de software:

-No todas las descripciones de patrones están disponibles en Internet. Por ejemplo, en el estudio de Henninger (2007) el 31% de los patrones de software no son inaccesibles a través de la web y sólo están disponibles en forma de libro (procedimientos, revista, libro).

-No existe un único repositorio de patrones. Henninger, (2007), contabilizó 121 colecciones de patrones relacionados con el software (es decir, los patrones de acoplamiento flexible ubicados en un lugar común, como un repositorio, papel, libro, sitio web) que contenía más de un patrón.

-Hay una variedad de formas de descripción del patrón (los utilizados en GoF (Gamma et al. 1995) o patrón de arquitectura orientado a software, libros, lenguajes de patrones, etc.). Esto hace más difícil la comparación de los patrones, por ejemplo, para determinar si dos patrones son los mismos, diferentes, o versiones del mismo patrón.

-No existen mecanismos estándar para la indexación, acceso, enlazar los catálogos de patrones.

-No hay ningún motor de búsqueda de patrón que cubra la mayor parte de los modelos disponibles y que sea universalmente aceptado por la comunidad.

-No es posible buscar usando la forma del patrón, por ejemplo, buscar una cadena específica en el problema y otra cadena para la descripción de las ventajas y desventajas de uso del patrón.

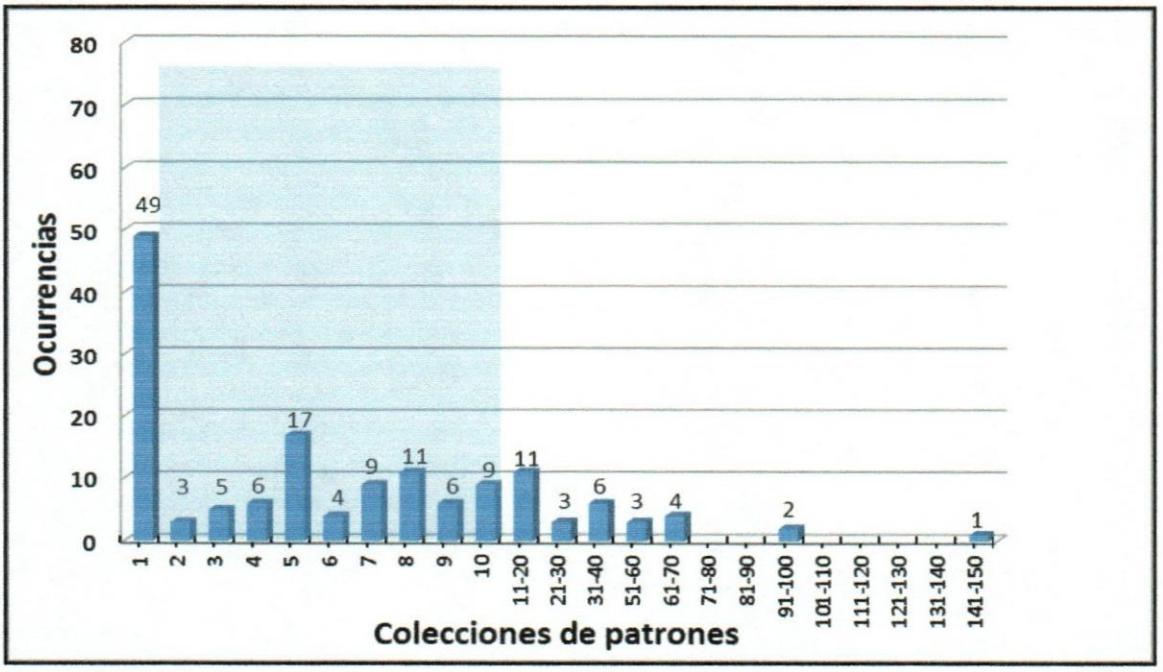


Figura 5. Colecciones y número de patrones. Adaptado de : Henninger (2007)

2.13. Desarrollo de software basado en componentes

Para Montilva et al. (2003) la reutilización de componentes de software es un proceso inspirado en la manera en que se producen y ensamblan componentes en la ingeniería de sistemas físicos. La aplicación de este concepto al desarrollo de software no es nueva. Las librerías de subrutinas especializadas, comúnmente utilizadas desde la década de los setenta, representan uno de los primeros intentos por reutilizar software.

Según se menciona en Bonillo (2006), la construcción de software basada en componentes persigue tres objetivos principales: la reutilización, la adaptación y la extensión (la reutilización implica la adaptación y extensión):

1. Un componente es reutilizable en la medida en que sus servicios pueden ser utilizados por otro software.

2. Un componente es adaptable si su proveedor ha previsto los posibles cambios que puede sufrir dicho componente.

3. Un componente es extensible si su proveedor proporciona los mecanismos para modificar los servicios que ofrece el componente.

2.14. Mecanismos de composición de software

Bajo el modelo de desarrollo de software basado en componentes, las aplicaciones se construyen mediante la integración o composición de componentes. Sametinger, (1997) define la composición de software como "el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces (de composición)". Nótese que se hace especial énfasis en las interfaces como elementos fundamentales para lograr la composición de componentes. La composición puede concebirse como una relación cliente-servidor entre dos componentes (Figura 6). El componente cliente solicita un servicio (operación) del componente servidor, el cual ejecuta la operación solicitada y devuelve los resultados al cliente. El servidor produce un resultado que es consumido por el cliente.

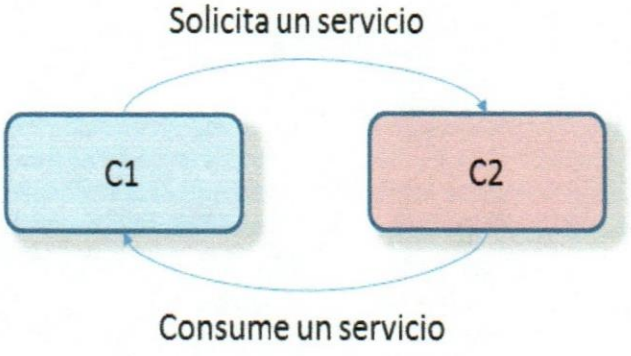


Figura 6. Interacción entre componentes.Fuente: Montilva (2003)

Además de los componentes, los frameworks también se consideran entidades sujetas a composición. En consecuencia existen tres clases principales de interacción en los sistemas basados en componentes (Montilva et al., 2003 p 6).

1. Componente-Componente (C-C): permite la interacción entre componentes. De este tipo de interacción se obtiene la funcionalidad de la aplicación, de forma tal que los contratos que especifican este tipo de interacción pueden ser clasificados como Contratos a Nivel de Aplicación.

2. Componente-*Framework* (C-F): posibilita las interacciones entre el *framework* y sus componentes. Dicha interacción permite que el *framework* administre los recursos de los componentes. Los contratos que especifican estas interacciones pueden ser clasificados como Contratos a Nivel de Sistema.

3. *Framework-Framework* (F-F): posibilita las interacciones entre framewoks y permiten la composición de componentes desplegados en framewoks heterogéneos. Estos contratos puede ser clasificados como Contratos de Interoperabilidad.

2.15. Frameworks

Un *Framework* es un conjunto de clases que engloban un diseño abstracto de soluciones de diseño para una familia de problemas relacionados. (Mattsson, 1996 citando a Johnson and Foote 1988 pp. 52). La definición del autor Mattsson (1996) es más amplia y aporta una referencia mas completa sobre un *framework*: " Una arquitectura generativa diseñada para la reutilización máxima, representada como un conjunto de clases conctretas asociadadas, que encapsulan el comportamiento potencial de especializaciones de subclases." (p.52) Así mismo el término generativo indica que es usado como el punto de arranque para el desarrollo de un número de aplicaciones específicas del dominio para el cual fue creado el *framework*.

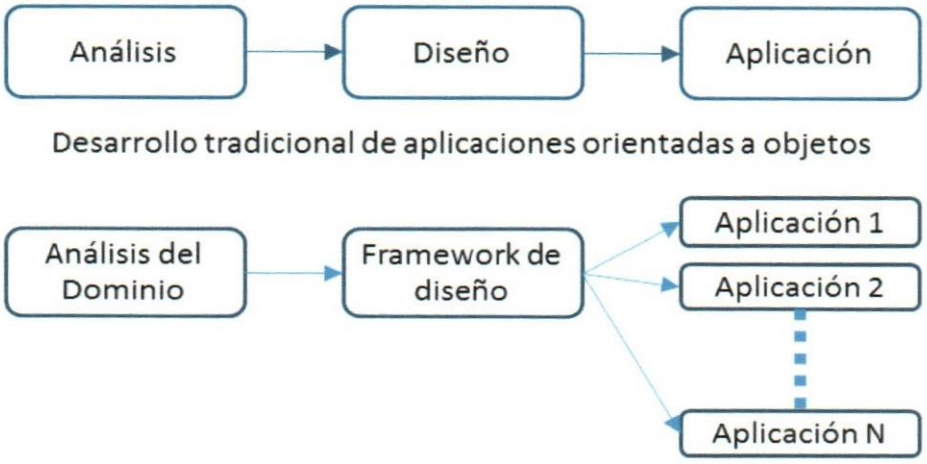


Figura 7. Comparación entre desarrollo tradicional y desarrollo de aplicacions basadas en frameworks. Fuente adaptado de Mattsson (1996).

2.16. Diferencias ente patrones de diseño orientado a objetos y frameworks.

Los patrones de diseño son más abstractos que los *frameworks*, los *frameworks* están contenidos en el código. No es el caso de los patrones de diseño, donde sólo el modelo del patrón de diseño es el que se aplica en el código. Los patrones de diseño también describen la intención, los acuerdos entre clases, las consecuencias del diseño, el cual no es el caso del *framework*.

Los patrones de diseño son arquitecturas más pequeñas que los *frameworks*. Un *framework* puede contener un número de patrones de diseño, pero lo opuesto nunca será posible. Por ello, los patrones de diseño no tienen gran impacto en la arquitectura de la aplicación.

Los *frameworks* son más especializados que los patrones de diseño ya que siempre están relacionados a un dominio específico, los patrones de diseño son generales y pueden ser aplicados a cualquier dominio de las aplicaciones.

2.17. Modelos organizacionales para la reutilización de componentes.

Según indica Sametinger (1997), existen cuatro modelos organizacionales que soportan para la reutilización de código, estos son:

- Reutilización para fines específicos entre grupos de aplicaciones.

  En este modelo el concepto de reutilización de componentes, si existe ocurre a nivel de proyectos. (Sametinger, 1997, p 55)

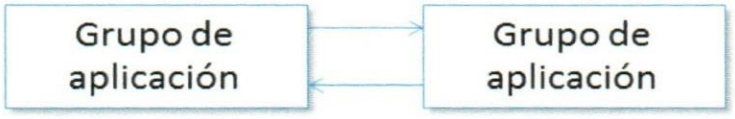


Figura 8. Modelo de reutilización de Software para fines específicos. Fuente adaptado de Sametinger (1997)

- Reutilización basada en repositorio entre los grupos de aplicaciones

  Existe en un tipo de repositorio que permite ser accedido por distintos grupos de desarrollo de software. Bajo este modelo no existen mecanismos o procedimientos establecidos para compartir los componentes en el repositorio y no existen responsables por la calidad de los componentes incluidos en el repositorio.

Figura 9. Modelo de reutilización de Software basada en repositorio. Fuente: adaptado de Sametinger (1997)

- Reutilización centralizada en grupo de componente.

  En este escenario existe un grupo responsable de administrar el repositorio, este grupo determina qué componentes se van a almacenar en el repositorio, garantiza la calidad de estos componentes y la disponibilidad de la documentación necesaria. Ayuda en la recuperación de los componentes adecuados en un escenario de reutilización particular. Esto equivale a la producción centralizada y gestión de los componentes de software reutilizables. El objetivo del grupo de componentes es reducir al mínimo la redundancia.

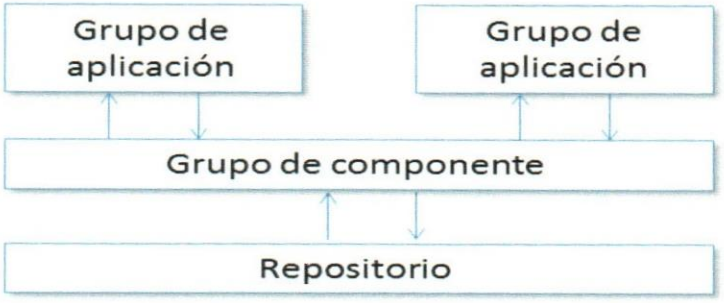


Figura 10. Modelo de reutilización de Software basada en centralización de grupo de componente. Fuente: adaptado de Sametinger (1997)

- Reutilización basada en dominio.

  En este enfoque existe un grupo responsable de un dominio específico de componentes por ejemplo: componentes de red, componentes de interfaces de usuario, componentes de base de datos, etc.

  Los grupos de desarrollo de aplicaciones pueden integrar sus aplicaciones desde distintos dominios. Estas organizaciones buscan adquirir software y herramientas y conocimientos específicos en dominios de software. Un inconveniente puede ser sobrecarga en la comunicación entre los grupos de proyecto y dominio.

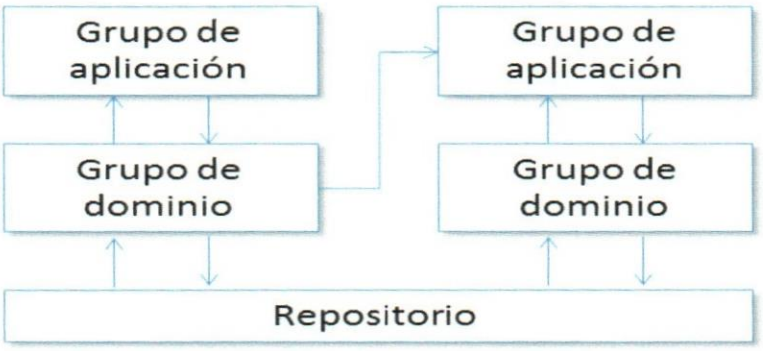


Figura 11. Modelo de reutilización de Software basada en centralización de grupo de dominio. Fuente: adaptado de Sametinger (1997)

2.18. Ontologías

Según McGuinness (2005), Una ontología define un vocabulario común para investigadores que necesitan compartir información en un dominio. Ella contiene definiciones de conceptos básicos y sus relaciones que pueden ser interpretadas por una máquina. En la Inteligencia Artificial, disciplina que trata sobre cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor, una definición de ontología ampliamente aceptada es la siguiente: "Especificación formal y explícita de una conceptualización". En este contexto, una ontología define un vocabulario común que describe en un dominio específico conceptos básicos y las relaciones entre ellos.

Tomando la idea anterior, se agrega expresividad a esta última definición complementándola de la siguiente manera:

• Conceptualización: modelo abstracto de un fenómeno, que generalmente se expresa como un conjunto de conceptos (entidades, atributos, procesos), sus definiciones e interrelaciones.

• Formal: organización teórica de términos y relaciones usados como herramienta para el análisis de los conceptos de un dominio.

• Compartida: captura conocimiento consensual que es aceptado por una comunidad.

• Explícita: se refiere a la especificación de los conceptos y a las restricciones sobre estos.

2.19. Tipos de Ontologías

Las ontologías se pueden clasificar considerando diferentes criterios. En este particular, se hace referencia a las categorizaciones proporcionadas en Guarino (1998)

En Uschold (1996) se ofrece tres dimensiones sobre las cuales varían los tipos de ontologías, a saber:

• Formalidad: referido al grado de formalismo del lenguaje usado para expresar la conceptualización, como altamente informal, informal estructurada, semi-formal y rigurosamente formal.

• Propósito: referido a la intención de uso de la ontología, entre las que se interoperabilidad entre sistemas y por último, las ontologías para beneficiar la ingeniería de sistemas.

• Materia: para expresar la naturaleza de los objetos que la ontología caracteriza. Con respecto a esta dimensión, las ontologías pueden ser: de dominio, de tarea y de representación.

Según Uschold (1996), las ontologías pueden clasificarse de acuerdo a la cantidad y tipo de estructura de la conceptualización en:

• 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 el almacenamiento estandarizado de información.

• Explicitar suposiciones de un dominio.

ser interpretables por máquinas. (McGuinnes, 2005) En este sentido, la ontología está constituida por un vocabulario específico utilizado para describir una cierta realidad.

Producto de la revisión de la bibliografía especializada se aprecia que los componentes de una ontología varían según el dominio de interés y las necesidades de los desarrolladores. A continuación, se presentan los componentes de las ontologías que se describen en McGuinnes (2005)

• Clase o tipo: son la base de la descripción del conocimiento en las ontologías ya que detallan los conceptos del dominio. La clase se refiere a un conjunto de objetos (físicos, tareas, funciones, entre otros) y cada objeto en una clase es una instancia de esa clase. Desde| el punto de vista de la lógica, los objetos de una clase se pueden describir especificando las propiedades que éstos deben satisfacer para pertenecer a esa clase. Una clase puede ser dividida en subclases, las cuales representarán conceptos más específicos que la clase a la que pertenecen. Una clase cuyos componentes son clases, se denomina superclase o metaclase.

• Relaciones: se establecen entre conceptos de una ontología para representar las interacciones entre éstos. Definidas por lo general como el producto cartesiano de n conjuntos: R: C1 x C2 x ...x Cn. Algunas de las relaciones más utilizadas son:

– (a) Instancia de: asocian objetos a clases;

– (b) Relaciones temporales: implican precedencia en el tiempo y

– (c) Relaciones topológicas: establecen conexiones espaciales entre conceptos.

• Instancias o individuos: son objetos, miembros de una clase, que no pueden ser divididos sin perder su estructura y características funcionales.

Pueden ser agrupados en clases.

• Propiedades o slots: los objetos se describen por medio de un conjunto de características o atributos que son almacenados en los slots. Éstos almacenan diferentes clases de valores.

Las especificaciones, rangos y restricciones sobre estos valores se denominan características o facetas.

Para una clase dada, los slots y las restricciones sobre ellos son heredados por las subclases y las instancias de la clase.

• Taxonomía: conjunto de conceptos organizados jerárquicamente. Las taxonomías definen las relaciones entre los conceptos, pero no los atributos de éstos.

• Axioma: elementos que permiten el modelado de verdades que se cumplen siempre en la realidad. Los axiomas pueden ser estructurales y/o no estructurales. Un axioma estructural establece condiciones relacionadas con la jerarquía de la ontología, conceptos y atributos definidos. Ejemplo: el concepto A no es una clase de A. Un axioma no estructural establece relaciones entre atributos de un concepto y son específicos de un dominio.

Ejemplo: la relación F=m*a, que debe cumplirse siempre entre los atributos

F (fuerza), m (masa) y a (aceleración) de un determinado concepto.

• Frame: un objeto que incluye clases, instancias y relaciones.

• Conceptualización: conjunto de conceptos, relaciones, objetos y restricciones que caracterizan un domino.

• Vocabulario: conjunto de palabras con una explicación y documentación que persigue la universalidad y el formalismo en el contexto de un dominio.

2.22. Herramientas para el desarrollo de ontologías

Actualmente existen varias herramientas que permiten representar el conocimiento consensuado de un modo general y formal, de tal manera que pueda ser compartido y reutilizado por distintos grupos de personas y aplicaciones de software. En este sentido, las ontologías conforman uno de los pilares básicos de las aplicaciones actuales y de la Web Semántica y requieren de un lenguaje lógico y formal para ser expresadas. En la

inteligencia artificial se han desarrollado numerosos lenguajes para este fin, algunos basados en la lógica de predicados, otros basados en frames (taxonomías de clases y atributos) y otros orientados al razonamiento. Con los lenguajes ontológicos se pretende alcanzar un alto grado de expresividad y uso. Dentro de los principales lenguajes destacan: XML, XML Schema, RDF, RDF Schema y OWL. En la Figura 3 se describen brevemente las herramientas y sus principales características, considerando aspectos como el tipo de software, ventajas y desventajas de uso.

Tabla 3. Herramientas usadas en la actualidad para la creación de ontologías.

Herramienta	Principales características
Protégé	Código abierto, plug-in, edición gráfica de taxonomías de conceptos y axiomas formales, chequeo de consistencia, interoperable.
KAON	Código abierto, servidor de aplicaciones, frames como formalismo de representación del conocimiento, edición colaborativa, chequeo de consistencia, escalable, eficiente e interoperable.
WebODE	Libre acceso, servidor de aplicaciones, interoperable, edición gráfica de taxonomías de conceptos y axiomas formales, chequeo de consistencia, edición colaborativa, utiliza motor de inferencias, chequeo de consistencia.
Swoop	Código abierto, extensible, edición colaborativa, utiliza motor de inferencias.
WebOnto	Libre acceso, arquitectura cliente/servidor, edición gráfica de taxonomías de conceptos, interoperable, edición colaborativa, utiliza motor de inferencias, chequeo de consistencias.
Ontolingua	Libre acceso, edición colaborativa, utiliza motor de inferencias, interoperable.
Chimaera	Flexible, robusto, ambiente distribuido, interoperable.

Fuente: Barrera et. al. (2012).

2.23. Metodología para el desarrollo de Ontologías Methontology

"Esta metodología fue desarrollada por el grupo de Ingeniería Ontológica de la Universidad Politécnica de Madrid. Methontology permite la construcción de ontologías a nivel de conocimiento. Esta metodología incluye: La identificación del proceso de desarrollo, un ciclo de vida basado en la evolución de prototipos, y técnicas particulares para apoyar las actividades". (Corcho, 2005, p.2)

Por cada prototipo, Methontology propone iniciar con las actividades de planificación para identificar las tareas a realizar, correcciones, tiempo y recursos necesarios. Luego, las actividades de especificación, administración (control y aseguramiento de la calidad) y de

soporte (adquisición de conocimiento, integración, evaluación, documentación, y manejo de configuración) comienzan al mismo tiempo. Todas las actividades de administración y soporte son realizadas en paralelo junto con las actividades de desarrollo (especificación, conceptualización, formalización, implementación y mantenimiento) durante todo el ciclo de vida de la Ontología. Una vez especificado el primer prototipo, el modelo conceptual es construido apoyado fundamentalmente por la actividad de adquisición de conocimiento. Al finalizar la conceptualización, las actividades de formalización e implementación pueden ser alcanzadas. Si algún detalle es detectado en algunas de estas actividades, se puede regresar a las actividades anteriores para realizar las modificaciones o refinamientos.

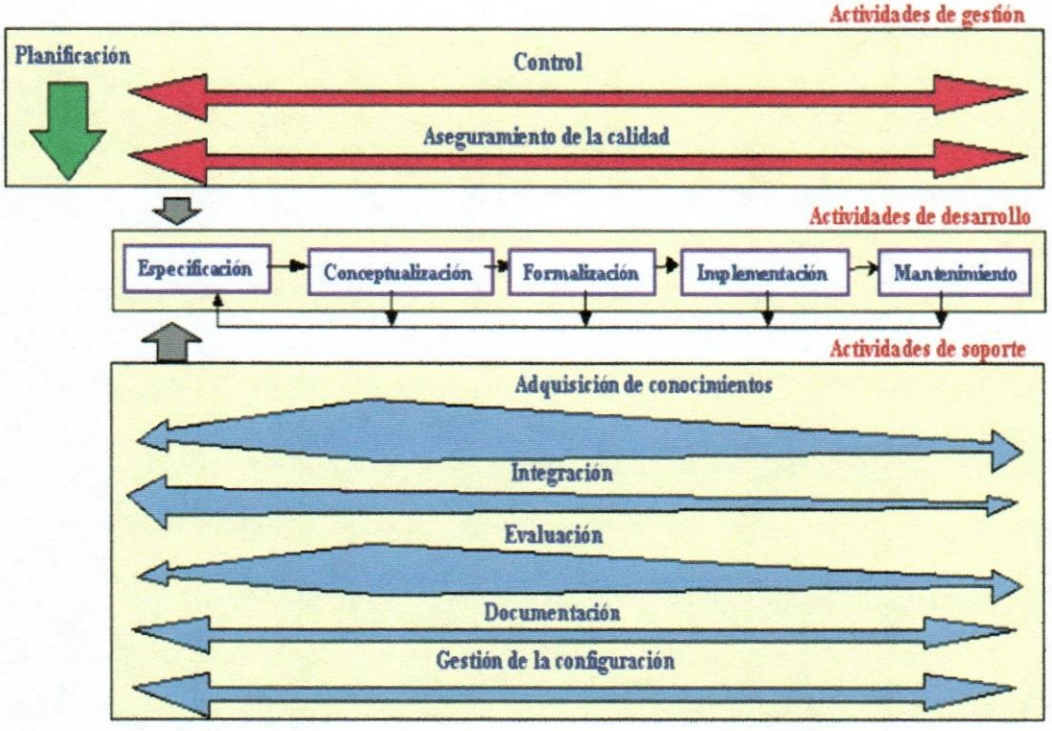


Figura 12. Relación de actividades y procesos del ciclo de vida de las ontologías. Fuente Corcho (2005)

2.24. Actividades de la metodología Methontology.

a. Especificación en Methontology

Permite determinar por qué se construye la ontología, cuál será su uso, y quiénes serán sus usuarios finales. El resultado de esta actividad consiste en elaborar un documento de especificación de requerimientos que expresa la información sobre el dominio que representará la ontología, fecha de elaboración, desarrolladores propósito, nivel de formalidad, alcance, fuentes de conocimiento. Para visualizar el detalle de esta tabla ver anexo VI

b. Conceptualización en Methontology

Esta actividad es de especial cuidado, en ella se define lo fundamental para formalizar e implementar la Ontología. METHONTOLOGY indica una serie de tareas a realizar para completar esta actividad. A continuación se describe como se realizó cada una de las tareas:

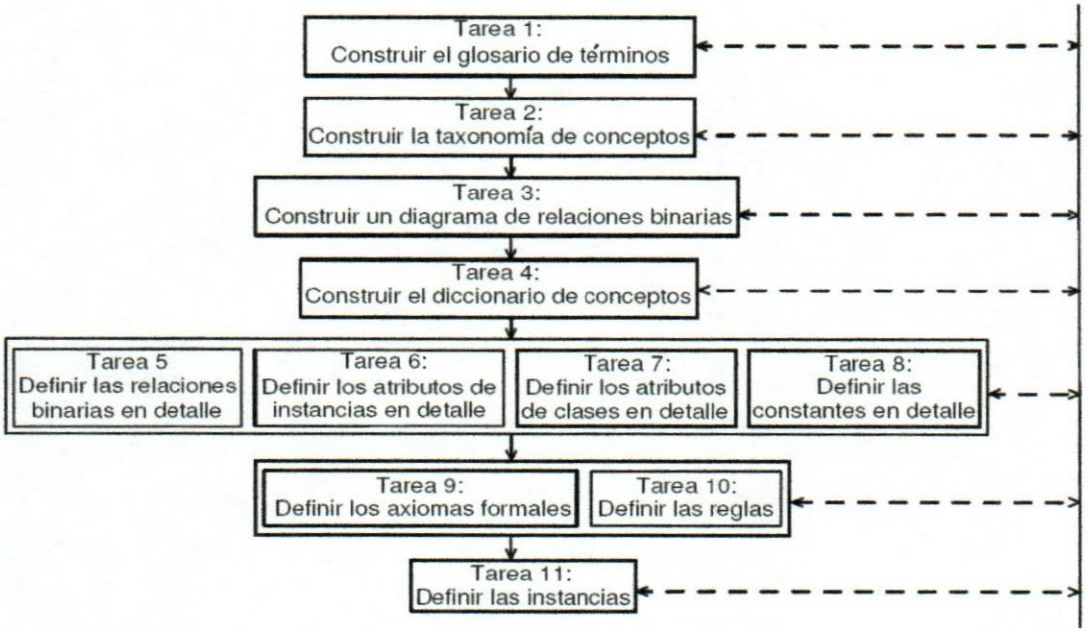


Figura 13. Pasos para la elaboración de ontologías, fuente: Corcho (2005)

A continuación se detallan las tareas para la elaboración de ontologías propuesto en Corcho (2005).

Tarea 1. Construir el glosario de términos que identifica el conjunto de conceptos a incluir en la ontología, su definición en lenguaje natural, y sus sinónimos y acrónimos.

Tarea 2. Construir taxonomías de concepto para clasificar conceptos. Los resultados de esta tarea son una o más taxonomías donde se clasifican los conceptos.

Tarea 3. Construir diccionario de conceptos para identificar las relaciones entre los conceptos de la ontología con los conceptos de otras ontologías, si existen.

Tarea 4. Describir detalladamente las relaciones binarias que aparecen en el diagrama de relaciones binarias y en el diccionario de conceptos. El resultado de esta tarea es una tabla de relaciones binarias donde se especifican nombre de los conceptos, fuentes y destino, su cardinalidad, relaciones inversas y propiedades matemáticas.

Tarea 5. Describir en detalle cada atributo que aparezca en el diccionario de conceptos. El resultado de esta tarea es una tabla donde se describen los atributos. Para cada atributo se pueden especificar los siguientes campos: nombre, concepto al que pertenece (los atributos pertenecen a los conceptos locales), tipo de valor, unidad de medida, precisión y rango de valores, valores por defecto si existen, cardinalidad máxima y mínima, atributos instancia, atributos clase, y constantes utilizadas para inferir valores de atributos, atributos que puedan ser inferidos usando valores de sus atributos, formulas y reglas que permitan inferir valores de los atributos y referencias usadas para definir el atributo.

Tarea 6. Describir en detalle cada clase que aparece en el diccionario de conceptos. El resultado es una tabla donde se describen las clases (objeto). La tabla contiene los siguientes campos: nombre, nombre del concepto donde la clase está definida, instancias de atributos cuyos valores puedan ser inferidos, etc.

Tarea 7. Describir en detalle cada constante y producir una tabla de constantes. Las constantes especifican información relacionada con el dominio de conocimiento; ellas siempre toman el mismo valor y normalmente son utilizadas en fórmulas. Cada fila de la tabla de constantes contiene una descripción detallada de la constante, donde se debe especificar: nombre, tipo de valor (número, masa, etc.), valor, unidad de medida para

constantes numéricas, y atributos a ser inferidos usando las constantes. Una vez definidos los conceptos, taxonomías, atributos y relaciones, los ontólogos pueden describir los axiomas formales (tarea 9) y reglas (tarea 10) que son utilizadas para el chequeo de restricciones y para inferir valores de atributos. Opcionalmente se puede introducir información acerca de las instancias.

Tarea 8. Describir Axiomas Formales. Se identifican los axiomas formales en la ontología y se describen de manera precisa. Para cada definición de los axiomas formales METHONTOLOGY propone especificar la siguiente información: nombre, descripción en lenguaje natural, la expresión lógica que formalmente describe el axioma utilizando lógica de primer orden, los conceptos, atributos y relaciones a los que se refiere el axioma y las variables usadas.

Tarea 10: Describir Reglas. Describir las reglas en la tabla de reglas. Para cada definición de regla METHONTOLOGY propone incluir la siguiente información: nombre, descripción en lenguaje natural, expresión que formalmente describe la regla, conceptos, atributos y relaciones a los que se refiere la regla y las variables usadas en la expresión. METHONTOLOGY propone especificar la expresión de la regla usando la plantilla: *if <condición> then <consecuencia>*.

Tarea 11: Describir Instancias. Después de definido el modelo conceptual de la ontología se deben describir instancias relevantes. Para cada instancia se debe definir: nombre, nombre del concepto al que pertenece, y los valores de sus atributos si se conocen.

c. Formalización e Implementación en Methontology

La formalización e implementación es un proceso que permite transformar el modelo conceptual en un modelo formal o semi-computable, a través del uso de herramientas para creación de ontologías como Protégé, Kaon, WeBODe, etc. En la implementación se construyen los esquemas en lenguaje de ontologías, (RDF Schema, Ontolingua, OWL) para que puedan ser accedidos y actualizados por sistemas capaces de procesar el conocimiento almacenado en la ontología.

d. Mantenimiento

Actividad que permite la actualización y/o corrección de la ontología en caso de ser necesario.

2.25. Escenarios de las ontologías en los sistemas de información.

Como indica Barchini et al. (2006), quien cita a Guarino (2005) y Uschold (1999) se tiene en cuenta los beneficios que ofrecen las ontologías en los SI y se detalla que en los sistemas de información se presentan cinco escenarios en los cuales se visualiza el rol de las ontologías.

Esos roles varían según se usen las ontologías como un soporte para el análisis conceptual de métodos y técnicas de los SI o como un soporte para el diseño, desarrollo y uso de los SI. En la tabla que se incluye a continuación se describen los escenarios detallando en cada uno de los objetivos, actores y material de soporte.

Tabla 4. Escenarios de los sistemas de información.

Escenarios de los SI	Propósito	Actores	Material de soporte
Análisis conceptual	Crear, revisar, modificar técnicas/lenguajes de modelación	Investigadores/Profesionales de los SI	Modelos ontológicos
Especificación de requisitos	Facilitar la educación y especificación de requisitos	Desarrolladores de los SI	Ontologías Bibliotecas de Ontologías
Modelado de datos	Disminuir la Heterogeneidad semántica	Desarrolladores de los SI	Biblioteca de Ontologías Ontologías de integración
Diseño de progamas e interfaces	Aumentar la calidad interna y externa del SIBO	Desarrolladores de los SI	Ontologías
Uso del SIBO	Facilitar el acceso y navegación	Usuarios	Ontologías

Fuente: Barchini et al (2006).

2.26. Las ontologías en el análisis conceptual de los SI

2.26. Las ontologías en el análisis conceptual de los SI

Existen diversos modelos ontológicos de los SI; cada uno de ellos se basa en una determinada postura ontológica-filosóficas acerca de cómo está compuesta la realidad.

La ontología filosófica qué más influencia tiene en los SI es la propuesta por Barchini et al. (2006, citando a Bunge, 1979). Ésta sostiene que el mundo está hecho de sistemas interconectados. En base a esta postura epistemológica, Wand (1990), académicos de la disciplina SI, construyeron el modelo ontológico BWW (Bunge-Wand-Weber). Este es un modelo de descomposición de los SI. Es formal, libre de contenido. A su vez, este modelo está compuesto por: el modelo de representación, el modelo de transición de estados y el modelo de buena descomposición.

El párrafo anterior describe que la investigación en los SI se esfuerza en la construcción de teorías dirigidas a determinar cómo se estructuran los SI en base a diversas perspectivas ontológicas. Ello se debe a que un SI constituye una representación de fenómenos del mundo real (Wand, 1990). Estos modelos abstractos de los SI constituyen un importante soporte teórico para el proceso de modelación y, por lo tanto, se utilizan para la evaluación de los lenguajes o técnicas de análisis y diseño de SI. En base a lo expuesto, se presentan diferentes escenarios donde se abordan los sistemas de información desde un punto de vista ontológico. En el escenario 1, en el cual las ontologías facilitan el análisis conceptual de los SI, sirven para mejorar las técnicas de modelación de los SI.

Escenario 1. Análisis conceptual ontológico

En la figura 7 se muestra el rol de las ontologías en el escenario 1, que se describe a continuación.

El investigador / profesional en SI realiza un análisis conceptual ontológico que consiste en:

– Explorar y analizar ontologías filosóficas que presentan diferentes modos de ver la realidad.

– Proponer modelos ontológicos de SI basados en ontologías filosóficas (p.e. Modelo BWW).

– Mejorar modelos ontológicos de SI.

– Explorar técnicas / lenguajes de análisis y diseño de SI (p.e. UML, OML, entidad-relación).

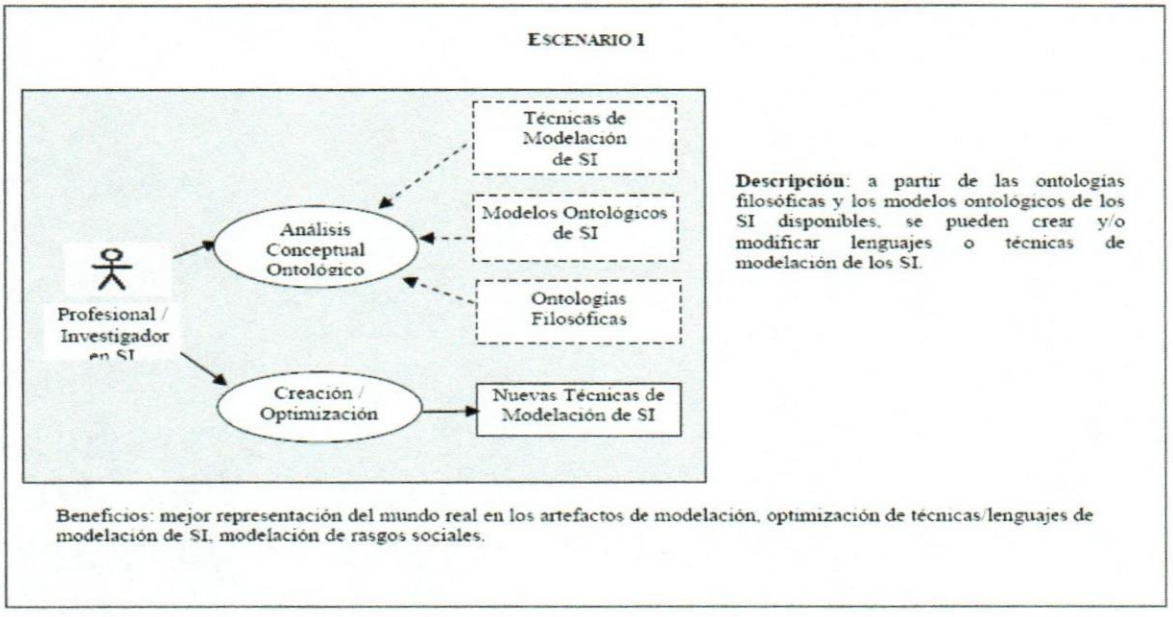


Figura 14. Análisis Conceptual Ontológico de las Técnicas de Modelación de SI. Fuente: Barchini et. al. (2006)

2.27. Las ontologías en el desarrollo de los SI

El desarrollador de los SI se enfrenta con problemas relacionados con la identificación, captura y representación del conocimiento de un dominio específico y las principales tareas que aborda son: el análisis de requisitos del sistema, la especificación funcional del sistema (arquitectura lógica) y el diseño del sistema, en estos casos se relacionan con el resto de los escenarios planteados en Uschold et al. (1999).

Escenario 2. Especificación de requisitos

Es factible usar una ontología que modele el dominio de aplicación y proporcione un vocabulario para la especificación de requisitos (ER) (Uschold et al. 1999) del sistema a diseñar.

El rol que la ontología juega en la especificación varía con el grado de formalidad y automatización de la metodología de diseño. En una aproximación informal, las ontologías facilitan el proceso de identificación de los requisitos de un sistema y el entendimiento de las relaciones entre los componentes del sistema. Esto es importante cuando en el equipo de desarrollo intervienen personas de dominios diferentes. En una aproximación formal, una ontología provee una especificación declarativa de un SI que permite razonar para qué el sistema está diseñado. (Uschold et al., 1999).

Por otra parte, es necesario destacar que las técnicas de educción, utilizadas en Ingeniería del Conocimiento, cada vez se utilizan con mayor frecuencia en la educción de requisitos en un contexto organizacional determinado. En este sentido, el usuario es considerado como un experto en su ámbito de trabajo. En la figura 5 se esquematiza el rol de las ontologías para la ER.

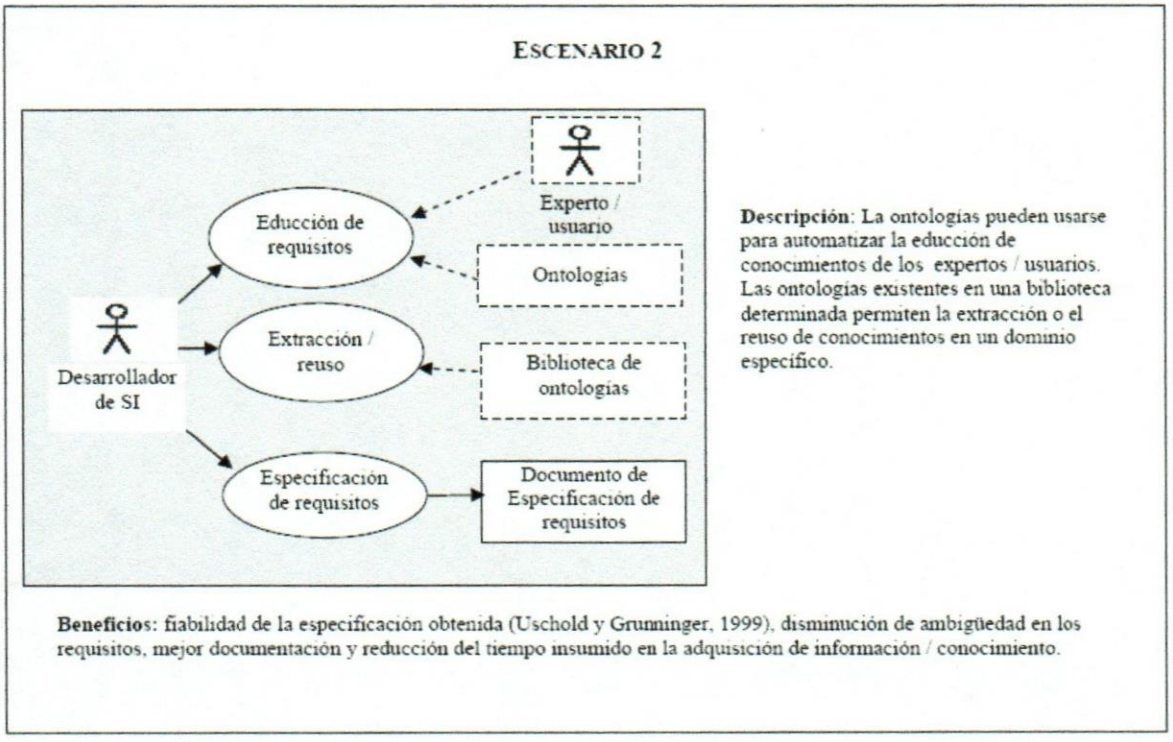


Figura 15. Especificación de Requisitos con Ontologías. Fuente: Barchini et. al. (2006)

Escenario 3. Modelado de datos

Según indica Barchini et. al (2006), un modelo de datos describe la estructura lógica de los datos y su aplicación. Se han hecho muchas extensiones del modelo entidad - relación para tratar de capturar la parte semántica de los datos, sin embargo, estos modelos aún presentan limitaciones tales como considerar un solo punto de vista del mundo y una sola posible interpretación de las instancias de interés

El uso de ontologías para el modelado de datos se muestra en la figura 8. Si las BD ya están en funcionamiento; es decir, existen otros SI o BD en el mismo contexto que necesitan interoperar se pueden utilizar ontologías para solucionar problemas de operabilidad debido a la heterogeneidad de esquemas e incompatibilidades semánticas. En este caso existen distintos enfoques de integración de BD con ontologías. Estos enfoques de integración usando ontologías permite la interoperabilidad entre múltiples aplicaciones, esto es posible

porque se accede a la misma información almacenada en BD distintas (Uschold et al. 1999).

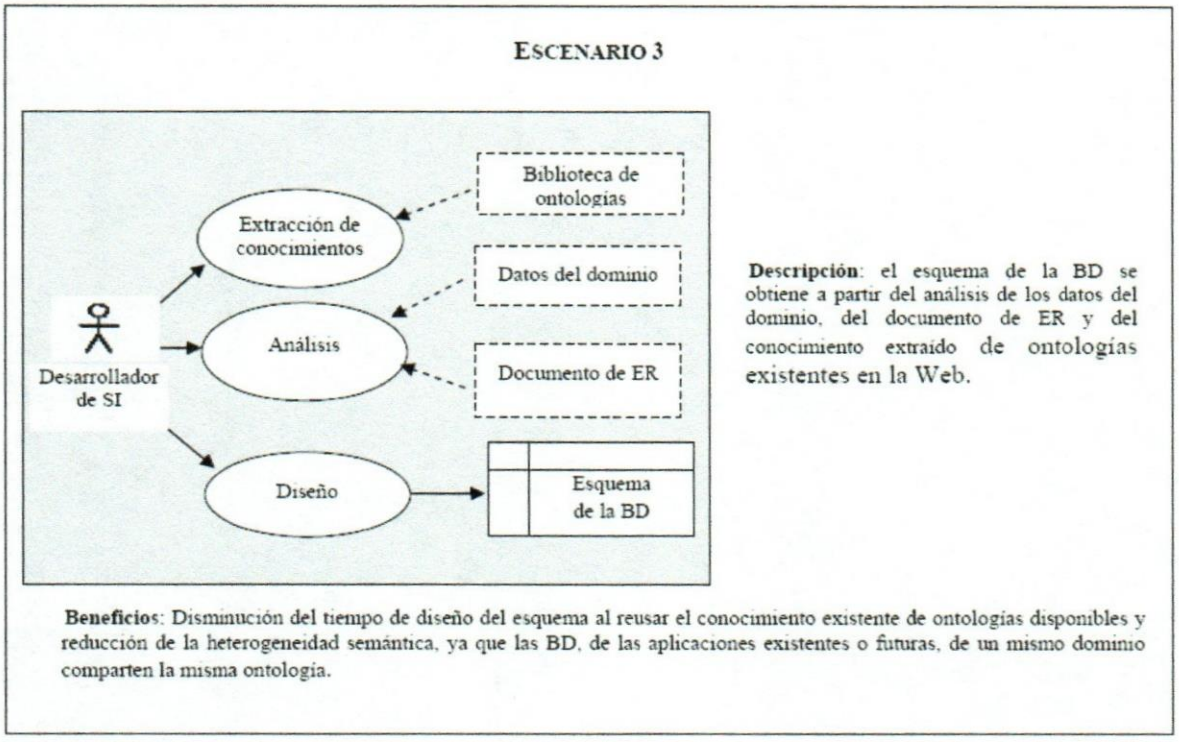


Figura 16. Modelado de datos con ontologías. Fuente Barchini et. al. (2006)

Escenario 4. Diseño de programas e interfaces

Los programas de aplicación son una parte importante de la mayoría de los SI. Normalmente contienen mucho conocimiento del dominio que, por varias razones, no se guarda explícitamente en una BD. Algunas partes de este conocimiento se codifican en la parte estática del programa en la forma de tipo o declaraciones de clases, otras partes están implícitamente en la parte procedimental del programa. Ambas partes son susceptibles de transformarse en un SIBO con la ayuda de una ontología (Guarino, 1998).

En la siguiente figura se visualiza el uso de ontologías en la obtención de un SIBO y en el diseño de interfaces.

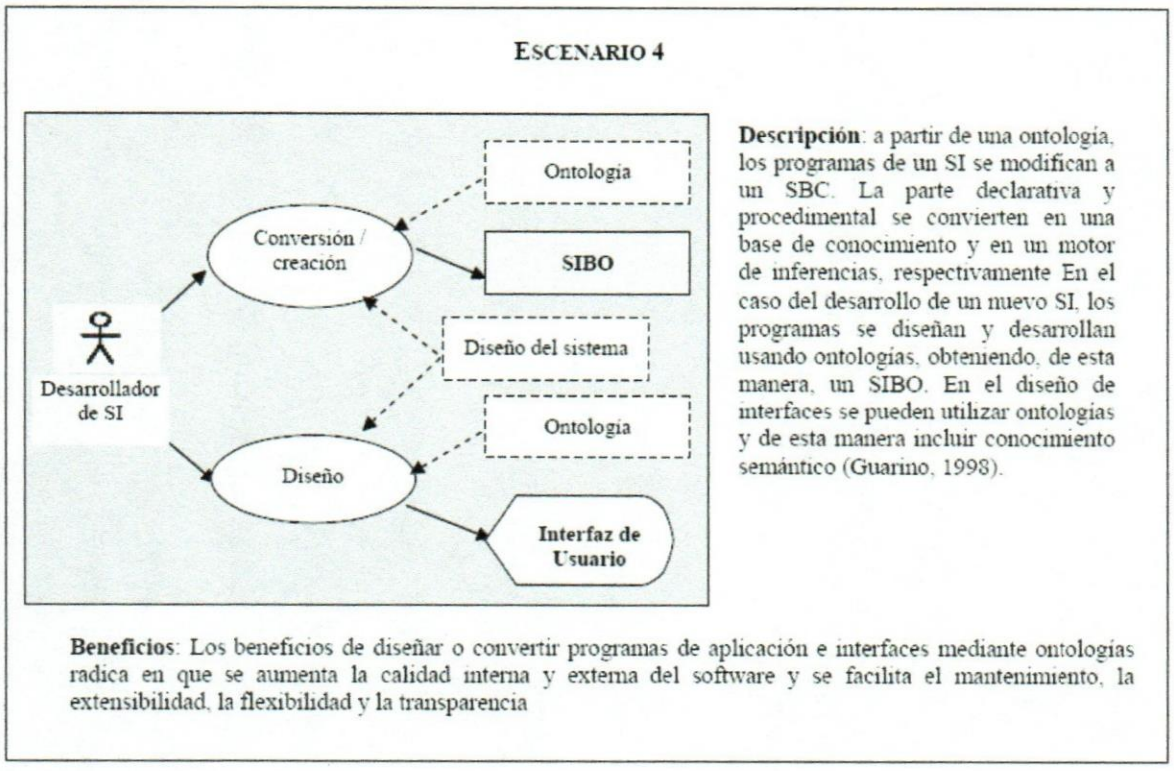
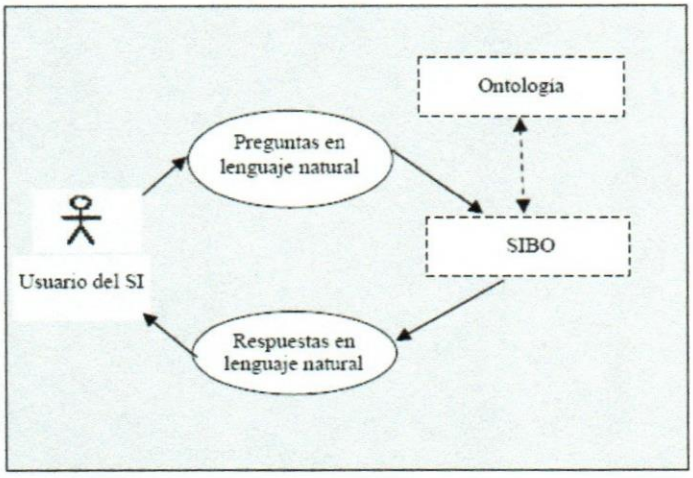


Figura 17. Diseño de programas e interfaces. Fuente Barchini et. al. (2006)

Escenario 5. Uso del SIBO

Durante el uso del SIBO (Sistema de Información Basado en Ontologías), el usuario interactúa con el sistema usando ontologías de forma explícita (el usuario es consciente, o sea conoce la existencia de la ontología y puede usar la misma como vocabulario) o implícita (el usuario no es consciente del uso de la ontología y la usa como parte normal de su interacción con el SIBO para hacer preguntas o para navegar). En la figura 10 se muestra el escenario respectivo.

ESCENARIO 5



Descripción: El usuario interactúa con el SIBO adoptando sus propios términos en el lenguaje natural los cuales son mapeados al vocabulario del SIBO mediante el uso de una ontología (Guarino, 1998).

Beneficios: El usuario accede y navega fácilmente en el SIBO y tiene la posibilidad de usar diferentes términos (sinónimos, hiperónimos, e hipónimos) del dominio de aplicación. Así, se consigue mayor amigabilidad y se alivian los problemas relacionados con la semántica de la información.

Figura 18. Uso del SIBO. Sistema de Información basado en ontologías. Fuente: Barchini et. al. (2006)

CAPITULO III. MARCO METODOLÓGICO

En este capítulo se describe la metodología que permitió desarrollar el presente proyecto, se muestran aspectos como el tipo y diseño de la investigación, las técnicas y procedimientos usados para su desarrollo, se detallan las variables en estudio y su tratamiento operativo para su evaluación a través de los instrumentos de recolección de datos con la finalidad de obtener la información necesaria para llevar el proyecto a su fase de desarrollo.

3.1. Tipo de Investigación

El desarrollo de la investigación se realizó bajo la modalidad de Proyecto factible, su elección está basada en la necesidad de plantear un modelo aplicable a solucionar un problema específico, en base a un estudio inicial, la elaboración del plan de trabajo y generación de una propuesta viable. Según el manual de trabajos de grado de especialización y maestría (UPEL, 2006) un proyecto factible:

> "consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El Proyecto debe tener apoyo en una investigación de tipo documental, de campo o un diseño que incluya ambas modalidades."(p.13).

El proyecto factible está apoyado en una investigación documental ya que se apoya en la revisión trabajos, revistas y publicaciones realizadas que aportan un marco conceptual para el problema que se plantea.

Este proyecto se apoya también en una investigación cualitativa del tipo investigación-acción siguiendo la clasificación indicada por Gómez (2004), entendiéndose esta como "un proceso de búsqueda de conocimientos sociales y prácticos, con participación de los grupos y de agentes externos, permite ligar la reflexión teórica con la práctica de una determinada realidad, con implicaciones ideológicas, teóricas y epistemológicas." (Hernández, 1991.p.115)."

3.2. Diseño de la investigación

La adaptación de este método de investigación acción se hace considerando las características del proyecto que se quiere realizar las cuales cumplen a cabalidad cuatro características destacadas en Baskerville (1999).

1. La investigación acción tiene como objetivo lograr un conocimiento creciente de una situación social, haciendo énfasis en la complejidad y naturaleza multivariada de sus especificaciones en el dominio de los sistemas de información.

2. La investigación-acción contribuye simultáneamente en la práctica a la resolución de problemas y expande el conocimiento científico. Este objetivo se enfoca en dos características importantes del proceso: En primer lugar, hay supuestos altamente interpretativos que surgen de la observación, en segundo lugar, el investigador interviene en la definición del problema.

3. La investigación-acción se lleva a cabo de manera colaborativa y mejora las competencias de los distintos actores. Un proceso de observación participativa está implícito.

4. La investigación-acción es principalmente aplicable para la comprensión del cambio de los procesos en los sistemas sociales.

La investigación es un proceso (generalmente cíclico) que combina la teoría con la práctica. La descripción más frecuente de Método Investigación-Acción se detalla en un proceso cíclico e iterativo de cinco fases descritas en Baskerville (1999), estas se pueden apreciar en la siguiente figura.

Figura 19. Ciclo de la Metodología Investigación-Acción. Fuente: Baskerville (1999)

En la siguiente tabla se describe en forma detallada las fases de la metodología de investigación acción de Baskerville (1999).

Tabla 5. Detalle de las fases de la metodología investigación-acción.

Fase	Descripción
Diagnosticar	Corresponde a la identificación de los principales problemas que son las causas que originan el deseo de cambio en la organización. El diagnóstico implica la libre interpretación del problema complejo de la organización, no a través de reducción y simplificación, sino más bien de una manera holística. Este diagnóstico se desarrollan supuestos teóricos (es decir, una hipótesis de trabajo) sobre el naturaleza de la organización y su dominio del problema.

Fase	Descripción
Planificar la acción	Los investigadores y los profesionales de entonces colaboran en esta actividad. Aquí se especifican las acciones organizativas que deberían aliviar o mejorar estos problemas primarios. El descubrimiento de las acciones previstas se guía por el marco teórico, lo que indica tanto un estado futuro deseado para la organización y los cambios que permitan lograr tal situación. El plan establece el objetivo para el cambio y el enfoque al cambio.
Tomar la acción	Los investigadores y los profesionales colaboran en la intervención activa en la organización en esta etapa, haciendo ciertos cambios a realizar. Varias formas de estrategia de intervención se pueden adoptar.
Evaluar	Después de finalizadas las actuaciones, los investigadores de y expertos evalúan los resultados. La evaluación incluye la determinación que indica si los cambios teóricos fueron realizados y si estos tuvieron efecto en la solución del problema.
Especificar el aprendizaje	Aunque la actividad de especificar el aprendizaje se lleva a cabo formalmente al final, es un proceso normalmente continuo. El conocimiento adquirido en la investigación-acción (si se tuvo éxito o no) puede ser dirigido a tres públicos: En primer lugar la reestructuración de las normas de organización para reflejar los nuevos conocimientos adquiridos por la organización durante la investigación. En segundo lugar, cuando el cambio no tuvo éxito, el conocimiento adicional puede proporcionar una base para el diagnóstico en la preparación de nuevas intervenciones de investigación-acción. • Por último, el éxito o el fracaso del marco teórico proporciona conocimientos importantes para la comunidad científica para hacer frente a situaciones de investigación futuras.

Fuente Baskerville (1999)

En base a lo anterior, se presentan las fases del método de investigación-acción para aplicarlo al modelo de ontologías y así determinar los pasos a realizar para el desarrollo del modelo ontológico para el uso de patrones en el desarrollo de software:

Tabla 6. Aplicación de las etapas de la metodología al proyecto.

Fase	Actividades
Diagnosticar	Diagnosticar el problema relacionado a la calidad del software desarrollado y la necesidad de recurrir a mejores prácticas para la reutilización de los componentes de software y optimización de los procesos de desarrollo de software.
Planificar la acción	Definir un marco de trabajo que proporcione una estrategia que describa los procesos para el uso, reutilización y difusión del conocimiento en el proceso de desarrollo de sistemas en una organización.
Tomar la acción	Implementar el uso de la metodología del proceso unificado para la representación de los objetos y modelos a desarrollar para patrones y componentes. Elaborar la taxonomía de los patrones de manera ontológica para agrupar el conocimiento de dominio según área de aplicación en el uso de patrones, componentes y "frameworks" en base a la metodología para elaboración de ontologías que indica McGuinness (2005).
Evaluar	Analizar resultados en base a la medición de los cambios realizados en función de los indicadores establecidos
Especificar el aprendizaje	Establecer Recomendaciones y conclusiones

Fuente adaptación de Baskerville (1999)

Por la naturaleza empírica y generalizada de las fases del método Investigación-Acción, es necesario complementar el desarrollo de esta investigación con el *framework* Metodológico del LISI Perez, Grimman, Mendoza y Rojas (2004) el cual permitió planificar, ejecutar y sobre todo evaluar de manera fiable y sistemática el proyecto a desarrollar. Este *framework*, se fundamenta en el método DESMET, Kitchenham (1996). El *framework* se basa en 11 pasos o actividades agrupadas en cinco grandes fases, que equivalen a las fases del método investigación acción, la unión con el *framework* indicado se visualiza en la figura 5.

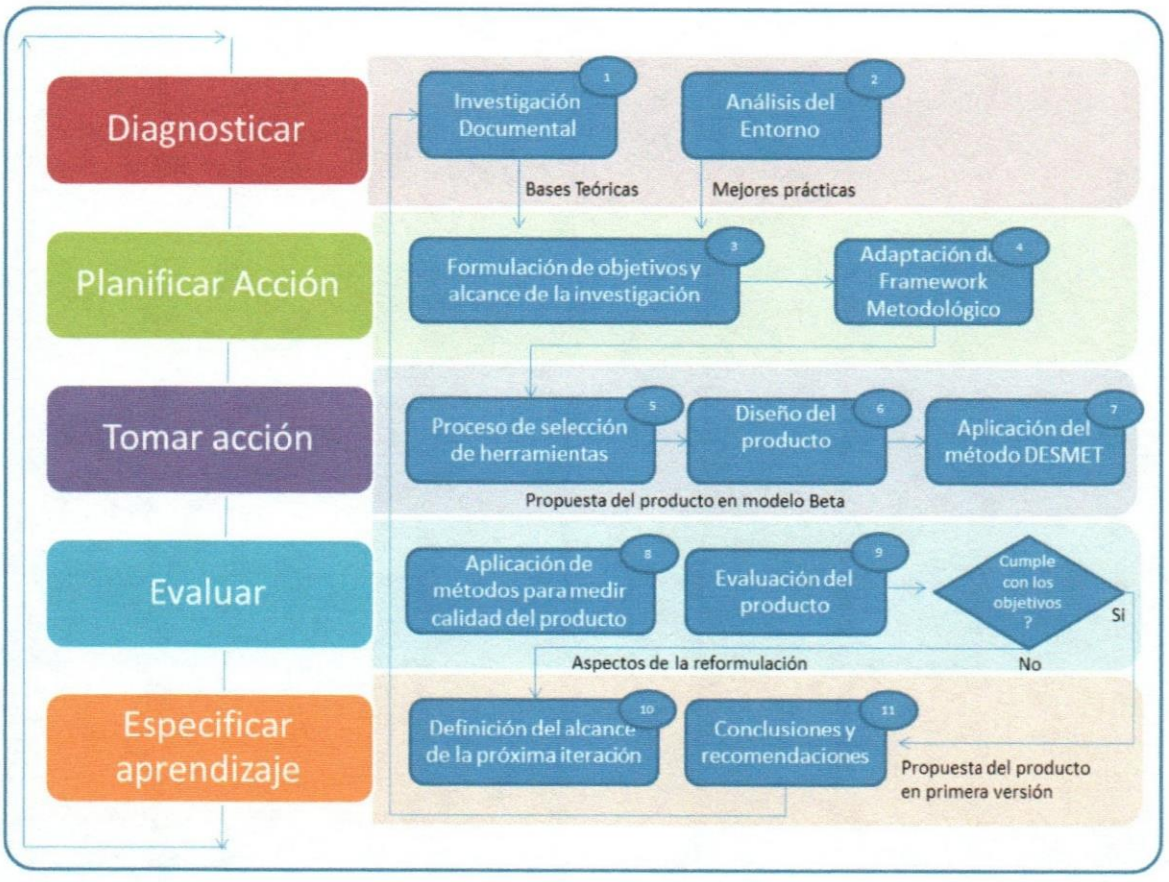


Figura 20. Framework metodológico para el proyecto. Fuente: Adaptado de Pérez et al (2004)

Considerando que el *framework* se basa en el método DESMET, es necesario describir los métodos y criterios que lo conforman con la finalidad de darle el enfoque adecuado al proyecto que es recomendado por el autor de la metodología.

En DESMET existen dos categorías principales propuestas para la selección: métodos de evaluación cuantitativos y cualitativos. Dentro de los cuantitativos se tienen los experimentos, casos de estudio y encuestas. Por otra parte, los cualitativos incluyen el análisis de características en modo de investigación, el análisis de características de caso de

estudio, el análisis de características de experimentos, y el análisis de características de encuestas en el cual se enfoca este proyecto.

Según Kitchenham (1996), DESMET identifica siete criterios para seleccionar el más apropiado: (1) contexto de evaluación; (2) naturaleza del impacto; (3) naturaleza del objeto de evaluación; (4) alcance del impacto; (5) madurez de los métodos/herramientas; (6) tiempo de aprendizaje asociada al método y la herramienta (7) habilidades de medición que posee la organización encargada de evaluar.

En la fase de tomar la acción se desarrolló la aplicación del método propuesto, específicamente se aplicó para la selección de un método adecuado que permitiera obtener los métodos de evaluación del producto propuesto.

3.3. Población y Muestra

a. Población

Tamayo (2003) Señala que la población es la totalidad del fenómeno a estudiar en donde las unidades de población poseen una característica común, la cual se estudia y da origen a los datos de la investigación. Por tal razón la población que representa la totalidad de individuos que conforman este estudio está formada por los departamentos de desarrollo de sistemas de las empresas del sector financiero.

b. Muestra

Luego de evaluar los métodos para seleccionar la muestra usada en metodologías cualitativas, se consideró el uso del muestreo teórico, esto considerando que este tipo de muestreo maximiza las ventajas de la selección de casos, la recolección de datos y el análisis de los mismos. (Krause, 1995). Según esta técnica, la muestra se selecciona mediante la utilización de una "estrategia sucesiva". Se eligen los primeros sujetos, documentos o situaciones de observación y se analizan los datos obtenidos. Mediante el análisis de estos primeros datos se desarrollan conceptos y categorías conceptuales que son utilizadas para generar criterios mediante los cuales se seleccionan los siguientes sujetos

que se integran a la muestra. Es por esto que se le ha denominado "muestreo teórico". Basándose en este enfoque, la investigación inicialmente consideró tomar la muestra de empresas del sector bancario en sus áreas de desarrollo de sistemas como una primera estrategia de la cual irán surgiendo otras muestras, toda vez que se vayan obteniendo y evaluando los datos recolectados.

3.4. Técnicas e instrumentos de recolección de datos

En cuanto a las técnicas e instrumentos para la recolección de información, se emplearon encuestas en modalidad escrita y entrevistas directas con expertos e investigadores del área, permitiendo obtener experiencias y observaciones que contribuyeron en la construcción del modelo. De igual forma se usaron técnicas que sustentan la investigación documental para obtener el basamento teórico que soporta las especificaciones técnicas de esta investigación y el análisis de contenido como técnica para interpretación de textos y resumen de características para agrupar patrones y componentes de software.

Para la medición de los instrumentos se utilizó el juicio de expertos, su desarrollo se lleva a cabo en todas las fases del proyecto. Esto con el objetivo de considerar en el modelo a desarrollar, un conjunto de instrumentos y herramientas que por medio de la consideración de distintos profesionales de la ingeniería de software permitirán obtener las métricas que nos develarán el marco de trabajo y todos los aspectos estructurales como de contenido. Se logra esto, a través de su evaluación pedagógica y lograr en base a sus observaciones, hacer las correcciones que sean requeridas y garantizar la calidad y efectividad del modelo a desarrollar.

3.5. Definición de variables.

A continuación se muestran las distintas variables a considerar en el presente proyecto:

Tabla 7. Operacionalización de variables consideradas en el proyecto.

Variables	Instrumentos	Definición conceptual	Definición operacional
Calidad de software	Cuestionarios	Lograr un alto nivel de calidad de un producto o servicio es el objetivo de toda organización, asegurando en cada una de las fases por las que transita el proceso de desarrollo. Basado en IEEE Computer Society (2004). Tiene seis dimensiones: flexibilidad, integración del sistema, tiempo de respuesta, recuperación en caso de fallas, conveniencia de uso y facilidad de uso	Uso de escalas de Likert de 5 puntos del instrumento de Chin, Diehl & Norman (1998) para medir niveles de calidad del software desarrollado
	Preguntas del cuestionario	Exactitud: El SI se debe demostrar precisión en los cálculos y en la información que genera	La valoración de ítems que califican la interacción con el SI se clasifican como: 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
	Preguntas del cuestionario	Facilidad de uso: El SI de información debe ser fácil, amigable e intuitivo para satisfacer sus requerimientos	La valoración de ítems que califican la interacción con el SI se clasifican como: 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto

Variables	Instrumentos	Definición conceptual	Definición operacional
	Preguntas del cuestionario	Flexibilidad: Los sistemas deben ser susceptibles de adaptarse y cambiar ante las necesidades del usuario. Nivel de esfuerzo requerido para actualizar o hacer mejoras a las aplicaciones	La valoración de los ítems que califican si el SI tiene la capacidad de ser: 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
	Preguntas del cuestionario	Integración del Sistema: Capacidad de los sistemas de comunicarse con otras aplicaciones.	La valoración de los ítems que califican si el SI tiene la capacidad de comunicarse con otros sistemas 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
	Preguntas del cuestionario	Recuperación en caso de fallas: El SI está preparado para reestablecerse al punto inicial en caso de presentarse una falla.	La valoración sumada de ítems que califican, la habilidad del SI para recuperarse en caso de fallas. La calificación es de 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
Reusabilidad del software	Cuestionarios	Determinar el uso de los patrones en el proceso de desarrollo de software	La calificación es de 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto Otras calificaciones para las preguntas son: Siempre, casi siempre, regularmente, pocas veces, nunca. Si 2. No

Variables	Instrumentos	Definición conceptual	Definición operacional
	Preguntas de cuestionario	Asertividad en la selección de patrones Es efectivo el proceso de selección de los patrones de desarrollo de software	Se califican en 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
	Preguntas de cuestionario	Reutilización de componentes: Efectividad en la reutilización de componentes desarrollados en distintas librerías	Se califican en 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
	Preguntas de cuestionario, Entrevistas	Comunicación e intercambio de información: Es el nivel de intercambio de información y realimentación de la información de los componentes desarrollados en las comunidades de software	Uso de escalas de Likert de 5 puntos del instrumento de Chin, Diehl & Norman (1998) permitirá determinar la disposición y las herramientas disponibles para propagar la información entre los grupos de desarrollo
Proceso de desarrollo	Preguntas de cuestionario, Entrevistas	Diseño de la arquitectura de los sistemas: Mide a nivel general la calidad del proceso de desarrollo en cuanto a evaluación de requerimientos de los sistemas, integración de componentes para armar la arquitectura del sistema, metodologías y procesos usados en la organización.	Se califican en 1. Siempre, 2. Casi siempre, 3. Regularmente, 4. Pocas veces, 5. Nunca. Se usa otra escala para determinar el uso de repositorios de componentes: 1: Si, 2: No.

3.6. Aplicación de la metodología DESMET para seleccionar el método de evaluación del presente modelo

Para determinar el método de evaluación del modelo propuesto desde un punto de vista macro, se realizó un estudio simple de los criterios que influyen en la selección de los métodos de evaluación propuestos en la metodología, mencionados anteriormente.

Los elementos que conforman el contexto de la evaluación son:

- Contexto de la evaluación: Modelo ontológico para determinar el uso de Patrones, es el objeto a evaluar
- Investigador: el investigador es el creador del Modelo y es el encargado de recolectar y analizar los resultados de la evaluación
- La empresa: Organización que posee cumple con las características del objeto de estudio en cuanto a recursos técnicos y diversidad de áreas de desarrollo de software y dispone de personas con experiencia en el área de desarrollo del presente proyecto
- La naturaleza del impacto esperado de la utilización del método o la herramienta: Evaluación cualitativa
- La naturaleza del objeto de evaluación, es decir, si es un método, una herramienta o un método genérico: Se realizó un modelo aplicable al problema de selección de patrones de diseño adecuados
- El alcance del impacto del método o la herramienta: El modelo impacta directamente en el proceso de desarrollo y en la calidad del producto
- La madurez del método o herramienta. En el proceso de desarrollo de sistemas en las organizaciones financieras encuestadas no demuestra un nivel de madurez en el proceso de desarrollo y en la calidad del producto, esto se evidencia en el capítulo IV
- La curva de aprendizaje asociada al método o la herramienta: El tiempo estimado es a mediano plazo, está considerado dentro de los tiempos establecidos para el desarrollo del proyecto
- Las habilidades de medición que posee la organización encargada de evaluar

De acuerdo a los criterios evaluados y las condiciones favorables que se presentan en cada uno de los seis criterios para seleccionar el tipo de método, se determina que el **Análisis de Características por estudio de Caso**, permitió evaluar la aplicabilidad del modelo y obtener el análisis con el juicio de expertos en las áreas que intervienen en el proceso de desarrollo de software.

CAPÍTULO IV. ANÁLISIS DE LOS DATOS Y PRESENTACIÓN DE RESULTADOS

De acuerdo al tipo de investigación usado, y siguiendo con la fase de tomar la acción descrita en la metodología seleccionada, y haciendo uso de la herramienta que permitió hacer la evaluación en la empresa seleccionada para el caso de estudio, se realizó un análisis detallado de la información recabada en las encuestas realizadas con la finalidad de obtener resultados concluyentes sobre los datos codificados y ubicando tales datos en el contexto teórico (Palella y Martins, 2006). De acuerdo a Briones (1996, pp 49-50), la interpretación comprende comparaciones entre resultados, evaluación de magnitudes, consecuencias de ellas, relaciones con el marco problemático, probables consecuencias de los resultados, etc. Se realizó entonces una investigación explicativa, el análisis comprende la explicación de relaciones entre las variables entre las cuales se ha constituido una o más hipótesis.

Siguiendo las pautas del método de investigación acción descrito en Ruiz (2014), se llevó a cabo cada uno de los pasos para desplegar el proceso de investigación-acción, según se indica a continuación:

Obtención de datos: Se siguió un plan de elaboración de los cuestionarios y su aplicación en las instituciones seleccionadas para recoger la información.

Retroinformación sobre los datos: se procesaron los datos recogidos durante la encuesta, lo que conllevó a rediseñar algunas interrogantes del cuestionario elaborado.

Análisis de datos: Se elaboraron las distribuciones de frecuencias para las respuestas obtenidas en las encuestas.

Planificación de la acción: Se infieren resultados para obtener conclusiones que conllevan a plantear el modelo propuesto.

Ejecución: Desarrollo del modelo ontológico en función de las conclusiones obtenidas en el proceso de obtención de datos.

4.1. Resultados en el proceso de investigación de campo

En el anexo II se encuentran los detalles de la encuesta realizada donde se muestra la problemática presentada y se evidencia la necesidad existente en los departamentos de desarrollo de sistemas de aplicar un modelo que facilite el proceso de desarrollo de software e incremente la reutilización de software en las instituciones del sector, lo que redunda en la disminución de los tiempos en el desarrollo. Se debe incrementar la calidad del software en vista que se reutiliza software ya certificado y permite diseñar conjuntos de librerías de *frameworks* que serán incorporados en los proyectos nuevos y en el mantenimiento de aplicaciones existentes. También facilitó la implementación de otras tecnologías orientadas al desarrollo basado en componentes y tecnologías orientadas a servicios.

A continuación se explican los resultados obtenidos para cada uno de los aspectos evaluados:

Calidad de software.

Facilidad de uso: Los valores obtenidos indican que para los sistemas desarrollados, 14 personas indicaron que tienen un nivel alto en lo que refiere a la facilidad de uso, cumpliendo con requisitos como aprendizaje, comprensión, operatividad, etc.

Flexibilidad: De un total de 29 personas encuestadas, 17 personas indicaron que los niveles se ubican en altos o medios, en cuanto a la adaptación de los sistemas existentes para integrarse con otras aplicaciones o la integración con nuevos módulos desarrollados.

Mantenibilidad: La facilidad con la que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento o adaptarse a cambios en el entorno, indicaron que el 78% de los encuestados se inclinó por la opción entre alto y medio refiriéndose al nivel de esfuerzo para realizar cambios y adaptaciones a las aplicaciones existentes.

Integración del Sistema: En este aspecto se obtuvo un bajo nivel de integración de las aplicaciones, esto es la capacidad de interactuar o adaptarse a otras aplicaciones, procesos o componentes desarrollados de la organización. Esto puede denotar también un grado muy bajo de cohesión, es decir, la existencia de un módulo que realiza múltiples tareas no relacionadas directamente con sus funciones. Es posible también que exista un alto acoplamiento entre los componentes y la existencia de interdependencia entre los distintos módulos de un programa, disminuyendo la capacidad integración entre sistemas o procesos de negocio y haciendo más complejo su mantenimiento. Los términos cohesión y acoplamiento están asociados a la independencia funcional citada en Pressman (2010)

> "Un módulo cohesivo ejecuta una sola tarea, por lo que requiere interactuar poco con otros componentes en otras partes del programa. En pocas palabras, un módulo cohesivo debe (idealmente) hacer sólo una cosa. Aunque siempre debe tratarse de lograr mucha cohesión (por ejemplo, una sola tarea), con frecuencia es necesario y aconsejable hacer que un componente de software realice funciones múltiples... El acoplamiento es una indicación de la interconexión entre módulos en una estructura de software, y depende de la complejidad de la interfaz entre módulos, del grado en el que se entra o se hace referencia a un módulo y de qué datos pasan a través de la interfaz"

Recuperación en caso de fallas: La mitad de los encuestados indicó que existen niveles bajos e intermedios en cuanto a la capacidad de los sistemas de recuperarse en caso de fallas, considerando aspectos como generación de alertas, bitácoras, etc.

Reusabilidad del software e integración de componentes

Reutilización de componentes: Se evaluaron tres aspectos relevantes en este punto que indican que se desarrollan un gran número de componentes y conjuntos de librerías que son de uso particular, se demuestra que un 41% de los encuestados desconoce cuáles son esos desarrollos de componentes, librerías de uso común o patrones hechos en la empresa que se implementan para solucionar distintos problemas de la organización, lo que indica que no existe comunicación entre las áreas involucradas de manera que permita realizar el intercambio de información entre éstas, reutilizar y mejorar el código desarrollado y en muchos casos están implementados con el uso de malas prácticas. Otro aspecto importante que resalta, es que de los componentes y procesos desarrollados por los grupos, pocas

veces se logran integrar en un conjunto de soluciones que permitan ser aplicadas a distintos procesos de negocio.

Asertividad en la selección de patrones: Los tres puntos evaluados con respecto a la asertividad en la selección de patrones de software en el desarrollo y su aplicación, indicó dos aspectos importantes, en primer lugar existe un nivel intermedio de conocimientos en el uso de patrones y en segundo lugar, tales conocimientos no son aplicados en vista que existe del bajo nivel de integración de patrones y su uso en las empresas.

Proceso de desarrollo

Uso del ciclo de vida: Entre los aspectos evaluados, la mitad de los encuestados indicó que se encuentran identificados de manera clara y precisa los pasos de alguna metodología de desarrollo de sistemas en su organización, por otra parte el 66% de los encuestados indicó que pocas veces o nunca hacían uso de metodologías en el proceso de desarrollo de sistemas.

4.2. Caso de investigación

Aquí se presenta la aplicación de la herramienta que permitió medir la calidad de los procesos de desarrollo y la calidad del producto que se desarrolla en los departamentos de sistemas de las empresas financieras, así como también medir la calidad de la herramienta a propuesta a través del conocimiento de expertos del área de desarrollo y determinar así si el modelo es aplicable o no en este sector de empresas.

La empresa seleccionada para hacer la evaluación del modelo propuesto, es el Banco Venezolano de Crédito, la razón que conlleva su selección está en el acceso a la información y la posibilidad de interactuar con distintos grupos del área de desarrollo de sistemas que son objeto del estudio.

Entre otros aspectos se puede destacar sobre la institución que en la última década se ha llevado a cabo un importante proceso de expansión a través de la apertura de oficinas en

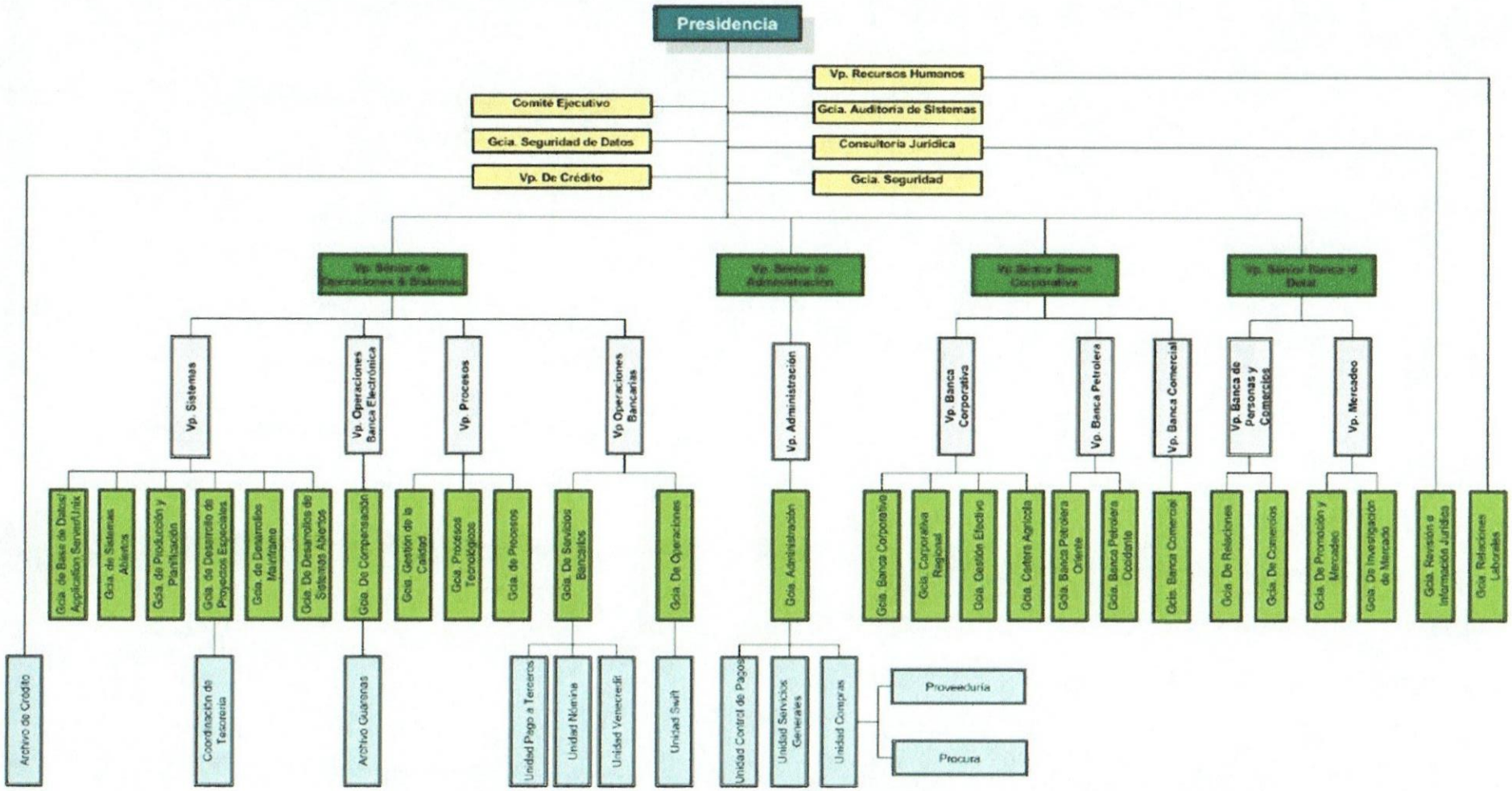


Figura 21. Organigrama estructural VCBU. Fuente: Venezolano de Crédito Banco Universal. (2012)

CAPÍTULO V. DISEÑO DE LA PROPUESTA

Siguiendo las fases de la metodología **Investigación-acción usada en el desarrollo del proyecto propuesto en Baskerville (1999)**, indicada en el capítulo III, se requiere implementar algunos pasos de las metodologías de trabajo que fueron aplicados a esta fase como es, la metodología indicada para la elaboración de ontologías descrita en el Capítulo III, (Corcho, 2005, McGuinness, 2005).

Elaborar la taxonomía de los patrones de manera ontológica que permitirá agrupar el conocimiento de dominio de patrones según área de aplicación, componentes y *frameworks* en base a la metodología para elaboración de ontologías indicada en Corcho, (2005). Estos dos aspectos dan cumplimiento a los objetivos específicos del proyecto y que aportan la base de conocimiento para la definición del marco de trabajo de la estrategia a seguir en el proceso de desarrollo de aplicaciones, considerando la reutilización y difusión del conocimiento en el área de desarrollo de sistemas en empresas financieras, que es el objetivo principal de esta investigación.

5.1. Elaboración de taxonomías de patrones usando ontologías.

a. Descripción del proceso de desarrollo de ontologías.

La creación de una ontología es un proceso y está compuesto de una serie de actividades que se realizan en un determinado orden, para un fin específico, sin embargo todo proceso de elaboración de ontologías atraviesa por 5 fases que conforman el ciclo de vida de construcción de una ontología, según indica Barrera et al (2012):

- Especificación: se identifica el propósito y el ámbito de la ontología.
- Conceptualización: se describe el modelo conceptual de la ontología.
- Formalización: se transforma el modelo conceptual en un modelo formal.
- Implementación: se implementa la ontología formalizada en un lenguaje de representación del conocimiento.

- Mantenimiento: se actualiza y corrige la ontología.

Además de las actividades que se deben realizar en las fases mencionadas, hay otras actividades que pueden llevarse a cabo durante todo el ciclo de vida como son:

• Adquisición de conocimiento: se adquiere el conocimiento del dominio

• Documentación: se documenta qué se ha hecho, cómo se ha hecho y por qué se ha hecho

• Evaluación: técnicamente se juzga la ontología.

b. Desarrollo de los pasos para la elaboración de la ontología usando la metodología Methontology

Especificación

Se detalla a continuación tabla de especificación de requerimientos que se indica en Corcho, (2005) con los parámetros formales del modelo propuesto.

Tabla 8. Documento de especificación de requerimientos para el modelo propuesto.

Documento de especificación de requerimientos	
Dominio	Patrones de desarrollo de software
Fecha	01/03/2014
Desarrollador(es)	Adriano Da Silva A.
Propósito	Desarrollar la taxonomía de patrones de software que permitirá definir el conocimiento del dominio que será usado en el proceso de desarrollo servicios y aplicaciones para la consulta, actualización y difusión del conocimiento patrones y contribuir en la mejora de la calidad en el producto de software.
Alcance	¿Cómo seleccionar de manera adecuada el marco trabajo en el proceso de desarrollo de software que mejor se adapta al problema planteado? ¿Es posible unificar conocimientos en un conjunto de procesos, técnicas y buenas prácticas para determinar el uso de patrones en el proceso de desarrollo de software en una organización? ¿Lograr que la especificación desarrollada no esté abierta a múltiples interpretaciones y restringir la libertad de criterios, puede incrementar las posibilidades de una mejor comprensión y la reducción de los llamados antipatrones?

Documento de especificación de requerimientos	
	¿Cuál es la problemática existente en relación a selección de los patrones y frameworks de desarrollo de software que mejor se adaptan al proyecto según sus características? ¿Es posible representar la estructura de los sistemas, su comportamiento y el modelo de datos a usar y además describir la lógica de negocio y restricciones de software en un mismo modelo ontológico?
Limitaciones	Se elabora la especificación, conceptualización, formalización e implementación.
Fuentes de conocimiento	Expertos en los departamentos de desarrollo de sistemas de instituciones financieras.

Siguiendo los pasos indicados en la metodología usada para el desarrollo de ontologías Methontology (Corcho, 2005), se detalla el desarrollo de los 11 pasos indicados en la metodología.

Conceptualización

Esta actividad tuvo especial cuidado, en ella se define lo fundamental para formalizar e implementar la Ontología. METHONTOLOGY indica una serie de tareas a realizar para completar esta actividad. A continuación se describe como se realizó cada una de las tareas:

Tarea 1: Glosario de Términos. Esta tarea se llevó a cabo realizando entrevistas a los expertos con la finalidad de elaborar un glosario especificando términos, sinónimos, acrónimos, descripción o significado del término y qué representa dentro de la ontología (concepto, atributo, relación, constante o instancia). En la Tabla que se muestra a continuación se puede ver un extracto del glosario definido.

Tabla 9. Desarrollo de la definición de conceptos indicado en la tarea 1

Nombre	Sinónimos	Descripción	Tipo
Abstract_Factory		Permite crear, mediante una interfaz, conjuntos o familias de objetos (denominados productos) que dependen mutuamente y todo esto sin especificar cuál es el objeto concreto	Concepto
Adapter		Permite crear, mediante una interfaz, conjuntos o familias de objetos (denominados productos) que dependen mutuamente y todo esto sin especificar cuál es el objeto concreto	Concepto
Antipatron		Es un patrón de diseño que invariablemente conduce a una mala solución para un problema	Concepto
Blackboard		Generaliza el patrón observador para permitir múltiples fuentes de datos y múltiples visualizadores. También realiza un completo desacoplamiento de los productores y consumidores de información.	Concepto
Blob	GodObject	Antipatrón es un objeto que conoce demasiado o hace demasiadas funciones. El objeto todopoderoso es un ejemplo de un anti-patrón	Concepto
Bridge	Handle/Body	Es una técnica usada en programación para desacoplar una abstracción de su implementación, de manera que ambas puedan ser modificadas independientemente sin necesidad de alterar por ello la otra.	Concepto
Broker		es un patrón de arquitectura que se utiliza para estructurar sistemas de software distribuidos con componentes desacoplados que interactúan por invocaciones de servicios remotos	Concepto
Builder		Es usado para permitir la creación de una variedad de objetos complejos desde un objeto fuente (Producto), el objeto fuente se compone de una variedad de partes que contribuyen individualmente a la creación de cada objeto complejo a través de un conjunto de llamadas a interfaces comunes de la clase Abstract Builder.	Concepto
Chain_of_responsibility		Es un patrón de comportamiento que evita acoplar el emisor de una petición a su receptor dando a más de un objeto la posibilidad de responder a una petición	Concepto

Nombre	Sinónimos	Descripción	Tipo
Channel_adapter		Patrón que especifica un único punto de entrada que luego puede conectarse de diferentes maneras e incluso con diferentes y varios adaptadores	Concepto
Command		Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto, con lo que además se facilita la parametrización de los métodos.	Concepto
Composite		El patrón Composite sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.	Concepto
Contenedor De Patrones		Un contenedor de patrones es un repositorio que permite agrupar un conjunto de patrones y donde se pueden acceder, modificar, con distintos métodos provistos por el contenedor y donde se puede compartir información a distintos grupos.	Concepto
Copy_and_paste		Es un antipatrón que consiste en aplicar soluciones copiando y modificando código existente en lugar de crear soluciones genéricas	Concepto
Decorator		Patrón que responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto evita tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera	Concepto
Facade		Es un patrón estructural que es requerido para estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre éstos	Concepto
Factory_Method	Template	Define una interfaz para crear objetos, pero deja que sean las subclases quienes decidan qué clases instanciar; permite que una clase delegue en sus subclases la creación de objetos.	Concepto
Flyweight		Es un patrón de diseño usado para eliminar o reducir la redundancia cuando	Concepto

Nombre	Sinónimos	Descripción	Tipo
		se tiene gran cantidad de objetos que contienen información idéntica, además de lograr un equilibrio entre flexibilidad y rendimiento (uso de recursos).	
Framework		Es un conjunto de clases que engloban un diseño abstracto de soluciones de diseño par una familia de problemas relacionados	Concepto
GodObject	Blob	Antipatrón, es un objeto que conoce demasiado o hace demasiadas funciones. El objeto todopoderoso es un ejemplo de un anti-patrón	Concepto
Interpreter		Es un patrón de diseño que, dado un lenguaje, define una representación para su gramática junto con un intérprete del lenguaje. Se usa para definir un lenguaje para representar expresiones regulares que representen cadenas a buscar dentro de otras cadenas. Además, en general, para definir un lenguaje que permita representar las distintas instancias de una familia de problemas	Concepto
Iterator		Patrón de diseño que define una interfaz que declara los métodos necesarios para acceder secuencialmente a un grupo de objetos de una colección.	Concepto
MVC		Es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones	Concepto
Patrón		Es una solución general para un problema que aparece con frecuencia en un dominio determinado	Concepto
motivacion		Escenario de ejemplo para la aplicación del patrón	Atributo
aplicabilidad		Usos comunes y criterios de aplicabilidad del patrón	Atributo
clasificacionDelPatron		Tipo de patrón	Atributo
codigoEjemplo		Código fuente ejemplo de implementación del patrón	Atributo
colaboracion		Explicación de las interrelaciones que se dan entre los participantes.	Atributo
conocidoComo		Otros nombres de uso común para el patrón.	Atributo
consecuencias		Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón.	Atributo
estructura		Diagramas de clases oportunos para describir las clases que intervienen en el	Atributo

Nombre	Sinónimos	Descripción	Tipo
		patrón.	
implementacion		Técnicas o comentarios oportunos de cara a la implementación del patrón.	Atributo
intencion		¿Qué problema pretende resolver el patrón?	Atributo
nombrePatron		Nombre estándar del patrón por el cual será reconocido en la comunidad (normalmente se expresan en inglés).	Atributo
participantes		Enumeración y descripción de las entidades abstractas (y sus roles) que participan en el patrón.	Atributo
usosConocidos		Ejemplos de sistemas reales que usan el patrón.	Atributo

Tarea 2: Construir taxonomía de conceptos.

Se construye una taxonomía que defina la jerarquía entre conceptos. En primer lugar, se definieron los conceptos en el prototipo como se muestra en la tabla anterior. Luego de de esto, se definió la taxonomía indicando cuál clase es padre y cuál es hija, como indica la siguiente figura. Para finalizar esta fase se cargan en el software *Protégé* las clases identificadas, el resultado de este proceso es el conjunto de diagramas de taxonomías que se pueden observar a continuación:

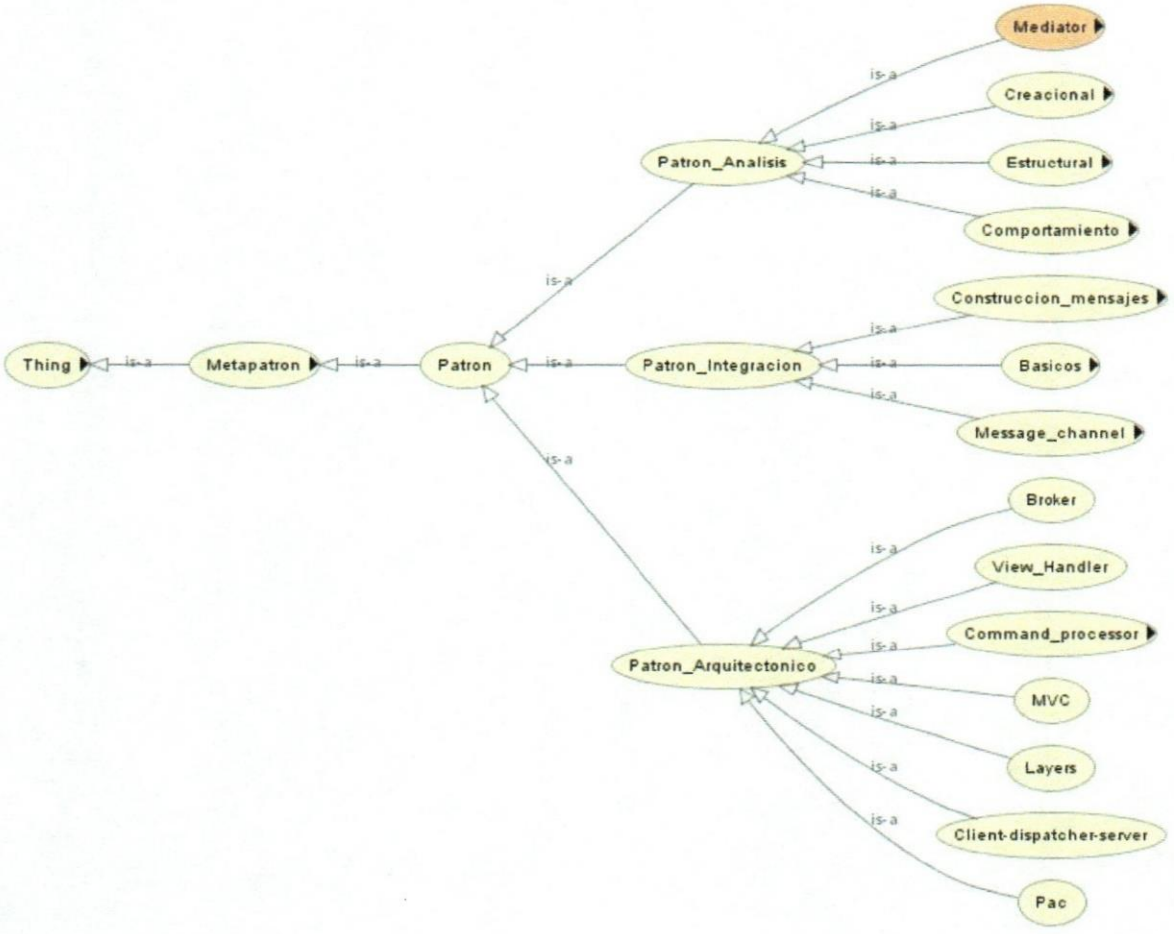


Figura 22. Fragmento de la taxonomía de conceptos de la ontología propuesta

Tarea 3. Construir un diagrama de relaciones binarias

Se trata de establecer las relaciones entre los conceptos, para crear dichas relaciones se define a través de la herramienta Protégé las propiedades (relaciones) asociadas a las clases (conceptos) y las instancias que conforman las clases. En este modelo se definieron distintas propiedades que pueden dar lugar a las relaciones de este modelo de patrones. Estas propiedades se pueden apreciar en la siguiente figura donde el lado izquierdo muestra las propiedades que definen distintos patrones y del lado derecho se muestra su uso para cada clase y sub clase.

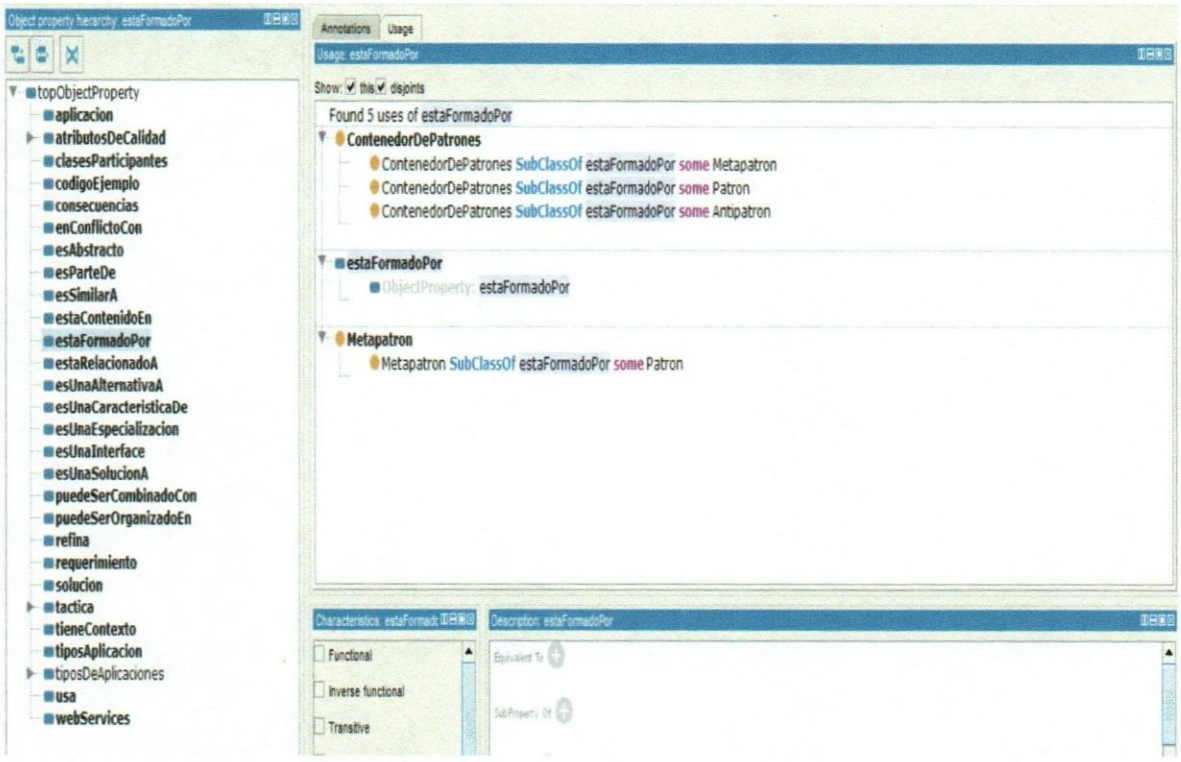


Figura 23. Fragmento de la definición de relaciones binarias entre clases a través de las propiedades

Tarea 4. Construir el diccionario de conceptos

Como una ampliación al contenido especificado en la metodología que debe ser desarrollado en esta tarea, se representaron todos los conceptos en la herramienta. Para poder visualizar esta fase, se incluye un fragmento de algunos conceptos que se han creado en la ontología y que describen relaciones, atributos, instancias, anotaciones, etc.

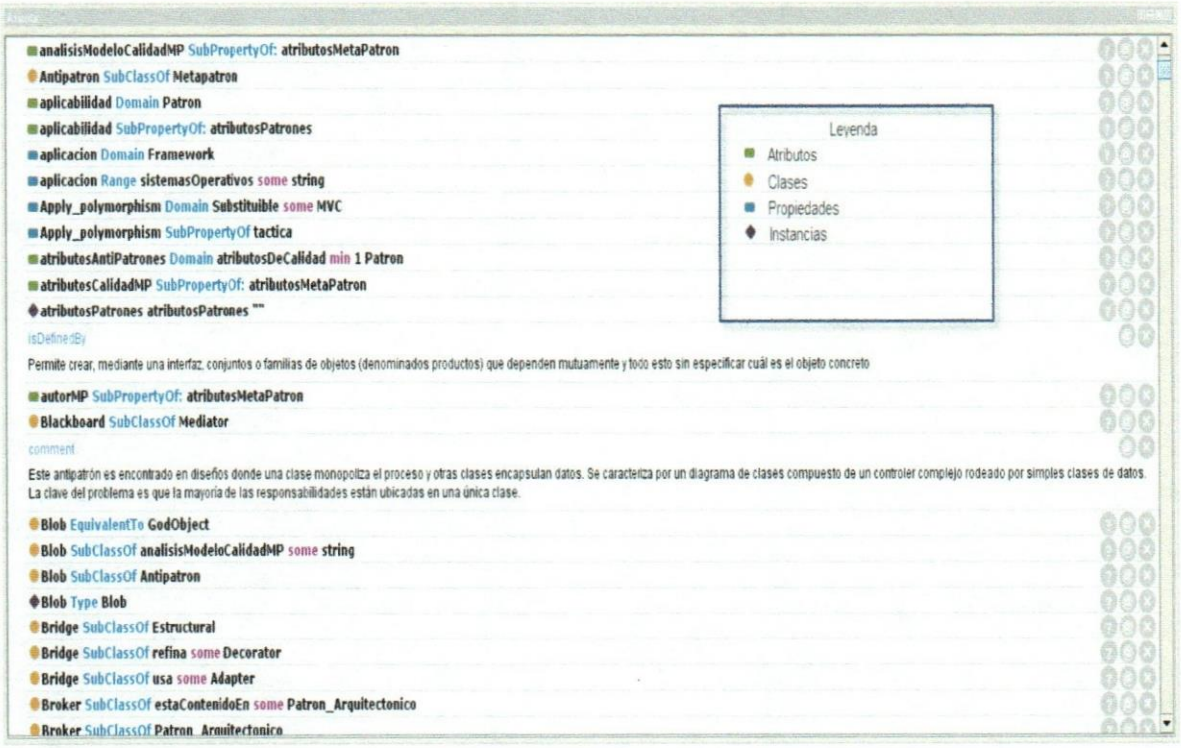


Figura 24. Fragmento de diccionario de conceptos de la ontología creada

Tarea 5. Describir relaciones binarias

En la ontología creada existen distintas relaciones binarias que describen relaciones mutuas entre clases como propiedades funcionales y funcionales inversas, para ejemplificar este caso, se tiene la clase patrón de comportamiento que tiene una relación con la clase patrón de análisis donde: la clase Patrón de comportamiento está contenido en el grupo de subclases del grupo patrón de análisis y este a su vez tiene como una de sus subclases a la clase Comportamiento, lo mismo ocurre con otras relaciones como es el caso de las clases Estructurar y Creacional con respecto a la clase patrón de análisis. Esto se aprecia en la imagen que se detalla a continuación.

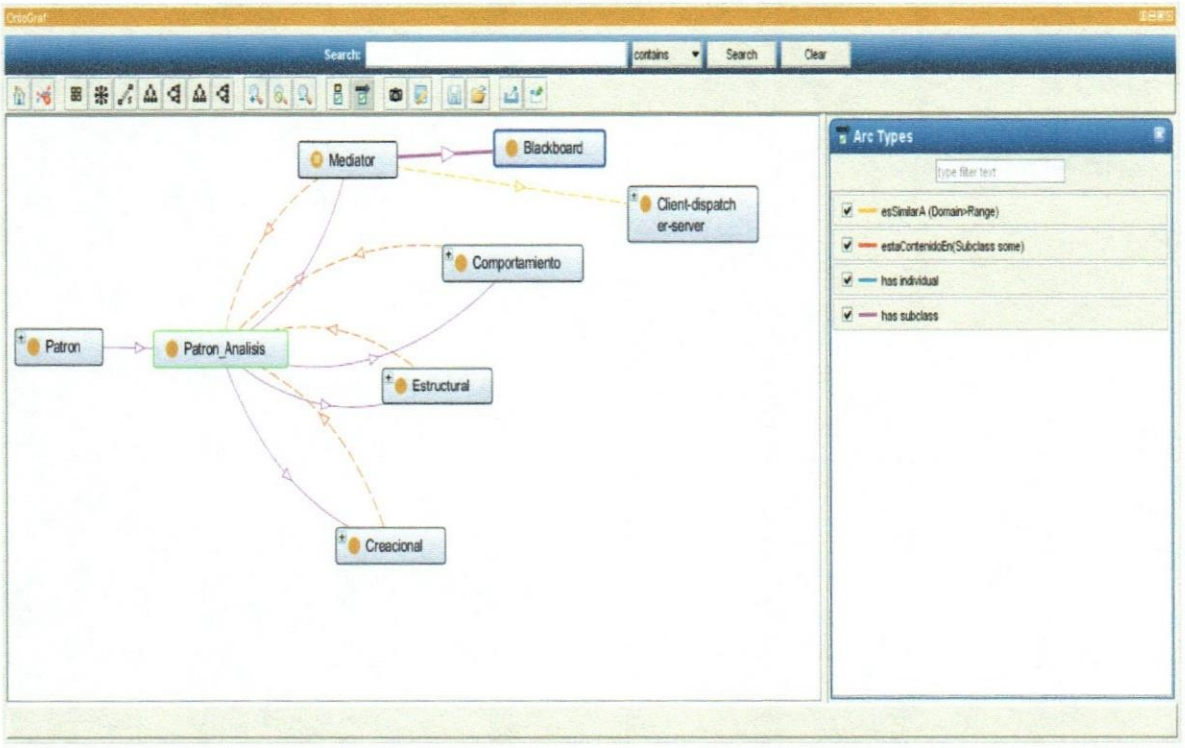


Figura 25. Fragmento de las relaciones de clases del modelo y sus relaciones binarias

Tarea 6. Describir atributos de instancia

Esta tarea busca crear los atributos de instancias de cada clase, en la versión 4.3. de Protégé los atributos de instancias son llamados *individuals*, estos no son otra cosa que la instancia de cada clase creada, es darle la definición a cada uno de los atributos que la conforman. En la siguiente figura se aprecia la carga de la instancia *command*, para ello se le asignaron los valores posibles a cada uno de los atributos asociados, en este caso serían: aplicabilidad, consecuencias, motivación, código ejemplo, intención, etc. estos últimos se aprecian en el lado derecho de la imagen siguiente.

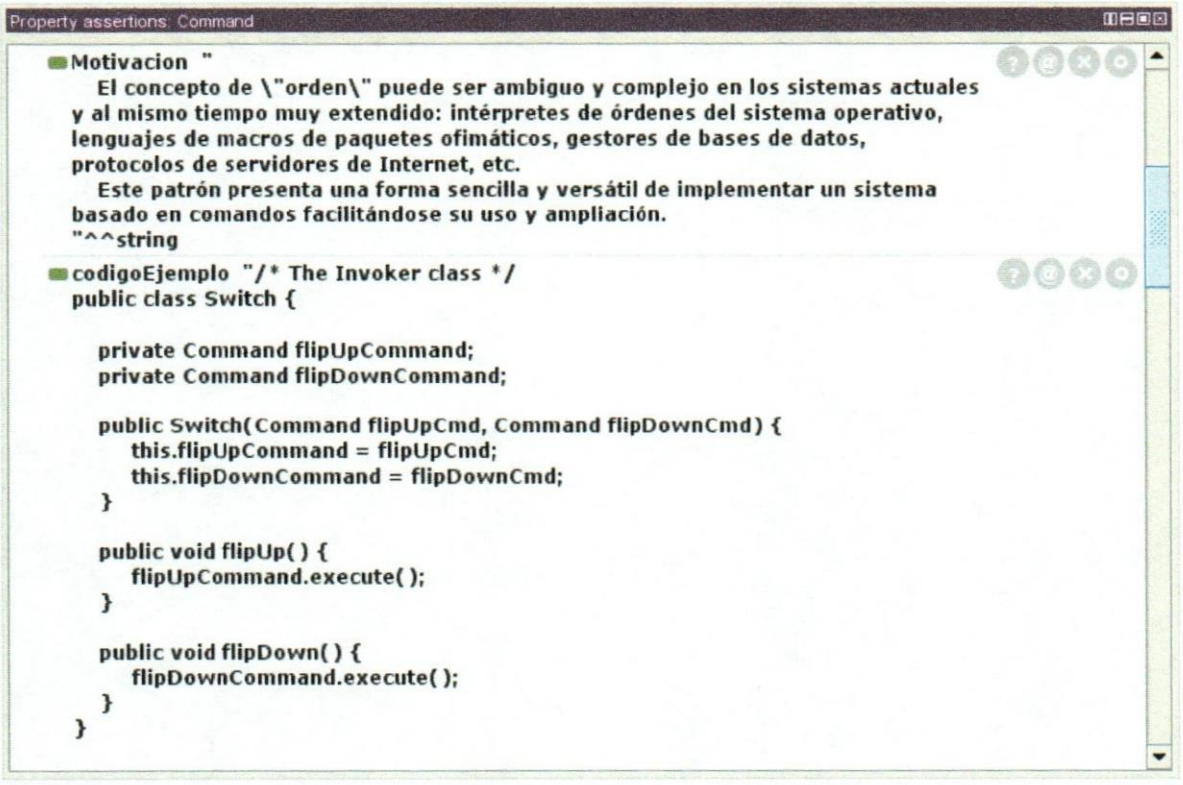


Figura 26. Creación de instancia de la clase Command para la ontología de patrones propuesta.

Tarea 7. Describir atributos de clase

Se definieron todos los tipos de datos de la ontología a través de la herramienta con la finalidad de indicar, nombres de atributos, tipo de valor, cardinalidad y valores que puede tomar el atributo, la siguiente imagen muestra parte del proceso de creación de los atributos de clase.

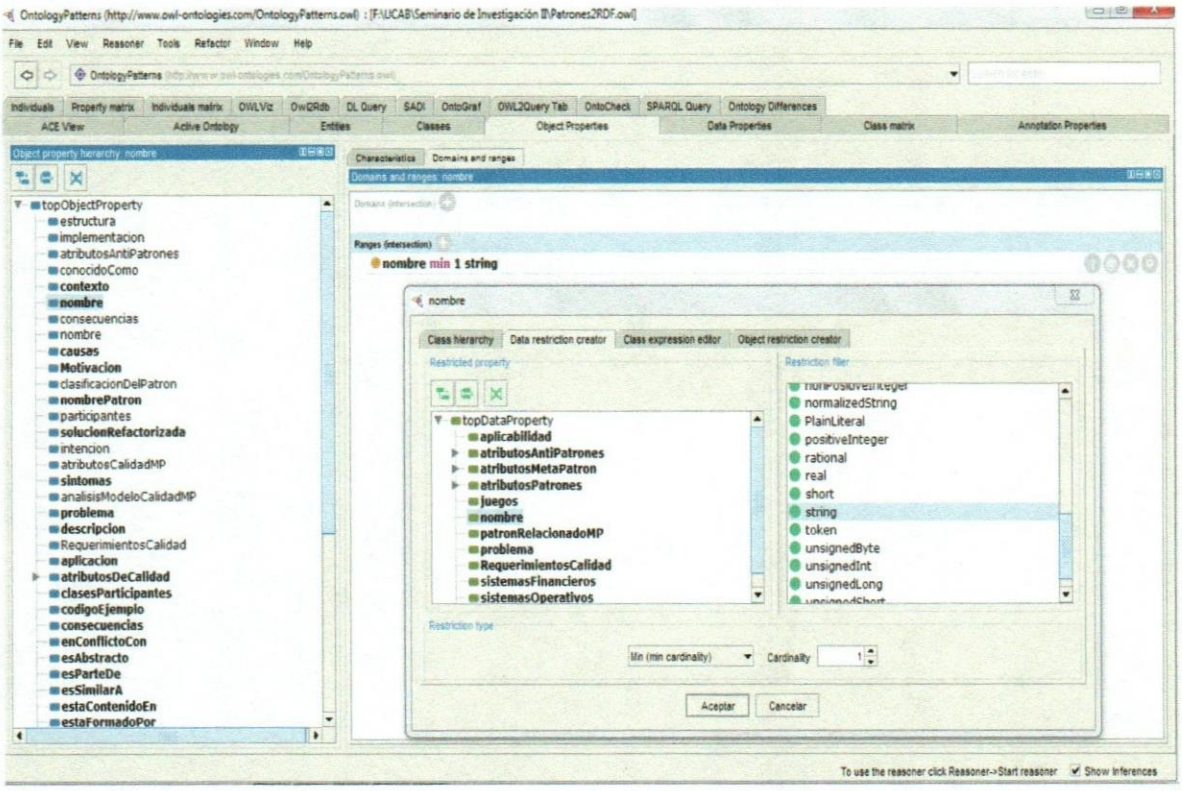


Figura 27. Creación de atributos de la ontología para asignar tipo de datos y cardinalidad.

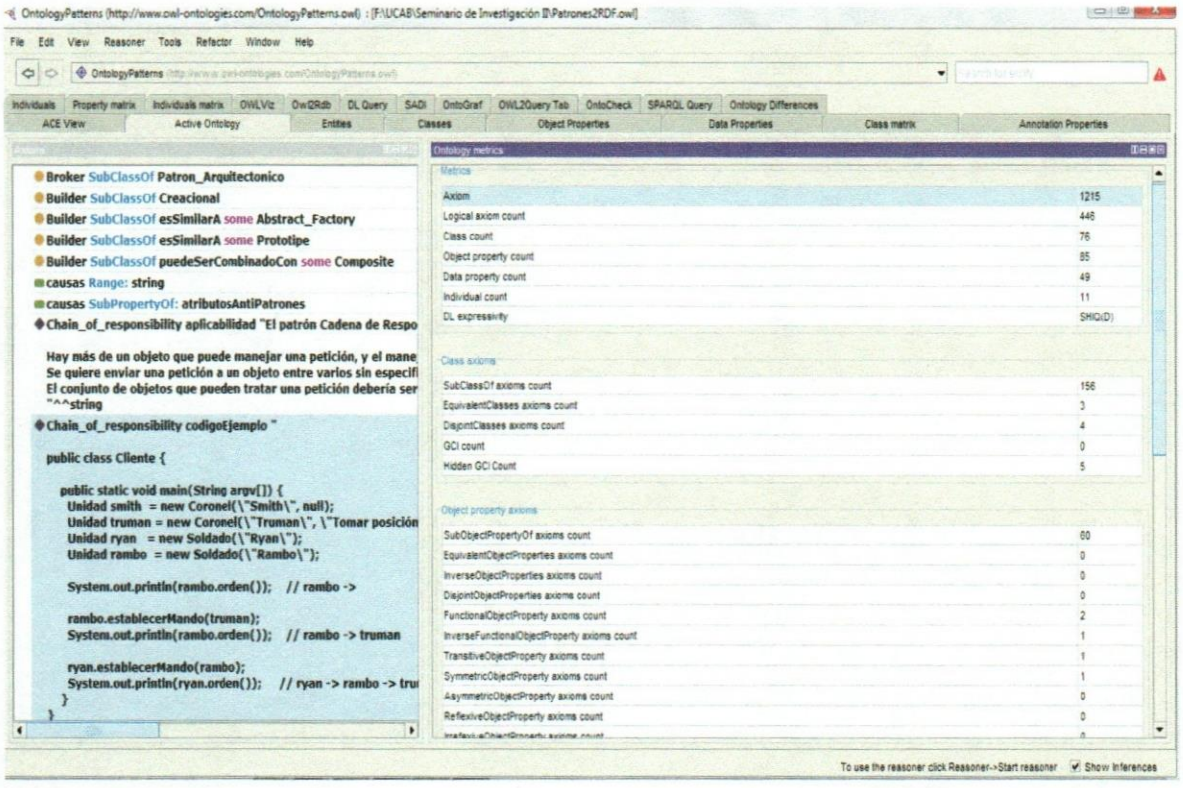
Tarea 8. Describir constantes

El ciclo de vida de las ontologías como modelos representativos del conocimiento de un dominio es necesariamente un modelo que debe ir perfeccionándose en el tiempo, para esta primera versión del modelo no se definieron constantes.

Tareas 9 y 10. Descripción de axiomas y descripción de reglas.

Estas tareas permitieron definir las reglas que se deben añadir a la ontología y permite describir el comportamiento de los conceptos o clases en el modelo. Para este modelo se identificaron en su fase inicial un conjunto de 1215 axiomas, los axiomas definidos van describen desde las relaciones directas, inversas o transitivas entres las clases. En la

siguiente imagen tomada de la ontología desarrollada en *Protégé*, se visualiza un fragmento de axiomas y las métricas obtenidas.



Una vez desarrolladas todas las actividades anteriores se obtiene una primera versión de la ontología que muestra un conjunto de clases, propiedades, axiomas, instancias que pueden ser visualizadas en las imágenes que se incluyen a continuación.

5.2. Desarrollo del modelo del repositorio de patrones donde se definirá la base de conocimientos obtenidos por los modelos a representar en ellos.

En los capítulos previos se presentaron aspectos relacionados con los patrones usados en el desarrollo de software y se detalló toda la teoría relacionada con las ontologías. En la sección anterior se realizó la ontología relacionada con los patrones de acuerdo a los conocimientos aportados por los expertos. Sin embargo, la ontología por sí sola no es la solución a los problemas planteados en esta investigación, para ello es necesario desarrollar un modelo de repositorio que permita el almacenamiento, y búsqueda de la información almacenada en él y a su vez permitir la divulgación de la información entre el grupo de expertos que la requieran. De esta manera, se cumple con varios de los requerimientos más importantes en el uso de patrones de software y que fueron planteados como interrogantes en esta investigación en el capítulo I, esto es: selección adecuada de patrones en el proceso de desarrollo, unificar conocimientos al permitir centralizar la información y definir un lenguaje común en el área de conocimiento de los expertos.

En primer lugar se propone la definición de la arquitectura de que soporta el modelo del repositorio para la creación de las distintas aplicaciones que pueden interactuar con la base de conocimientos de patrones de la ontología propuesta y validada por los expertos. Ver capítulo 6.

a. Arquitectura del modelo de repositorio para la ontología propuesta

La arquitectura que se propone debe soportar el uso de servicios e interfaces visuales que permitirán interactuar con la base de conocimientos. Es importante recordar que la información está codificada en formato RDF y debe proveer de herramientas que permitan, no solo la traducción de la información al experto en formato legible a los expertos, sino que permita explotar al máximo el uso de búsquedas semánticas en la ontología creada.

Para ello se dispone de una arquitectura que no está ligada a algún *framework* libre o propietario, sino que se propone una arquitectura abierta que puede ser implementada en

cualquier entorno de desarrollo empresarial que existe en el mercado como por ejemplo J2EE o .NET

El modelo planteado es representado en lenguaje de modelado unificado por sus siglas en inglés UML, su estructura en sí define un conjunto de casos de uso, diagramas de interacción y un diagrama de clase que modela en este caso el comportamiento de las clases su relación, sus métodos y cardinalidad.

Los casos de uso identificados en el modelo están relacionados con los procesos que se realizan para la incorporación, consulta y actualización de patrones. En la próxima figura se detalla una arquitectura usando el patrón arquitectural MVC y el *framework* que permitirá el acceso los repositorios y que permitirá realizar consultas a la base de conocimientos, incorporar y mantener el repositorio de patrones. Esto enmarcado en el patrón arquitectural MVC haciendo uso de servicios web que generan independencia de la plataforma ya que una vez que se traducen las búsquedas de la base de datos de conocimientos se mapean a objetos que serán empaquetados bajo el protocolo SOAP. Que pueden ser enviados en formato XML a los clientes.

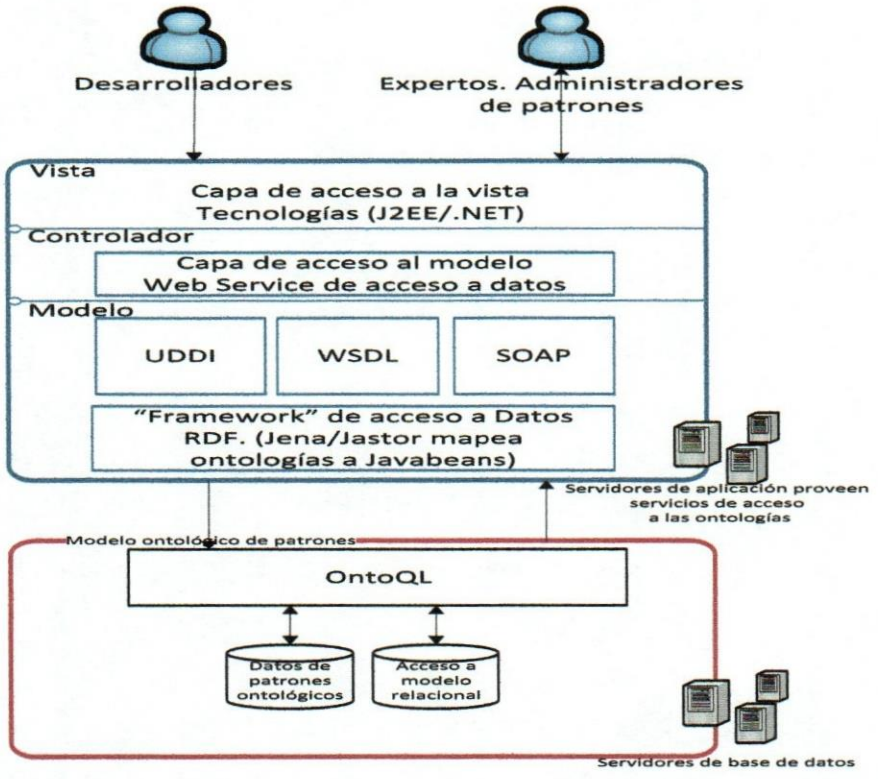


Figura 28. Arquitectura que soporta el modelo ontológico de patrones.

b. Modelo de repositorio que soporta el modelo ontológico propuesto

El diagrama de clase propuesto se presenta en un nivel de abstracción alto lo que permite una estructura genérica que representa un nivel de metadatos, esto se debe a que la descripción completa del modelo, sus relaciones, atributos, propiedades, entidades, está detallado en la ontología. De esta manera, las clases que soportan toda la estructura de la cual heredan las clases específicas, contiene entre sus clases es un modelo de metapatrón adaptado que permite describir conjuntos de patrones que interactúan en una determinada solución. En el diagrama de clase que se muestra a continuación se puede observar las siguientes relaciones entre las clases del repositorio que definen metadatos del repositorio y que sirven de información complementaria al proceso de creación y alimentación de la base de conocimiento. Esto es posible haciendo uso de la arquitectura representada donde haciendo uso de tecnologías orientadas a servicios se pueden separar en capas los distintos procesos necesarios para la creación y mantenimiento del modelo ontológico.

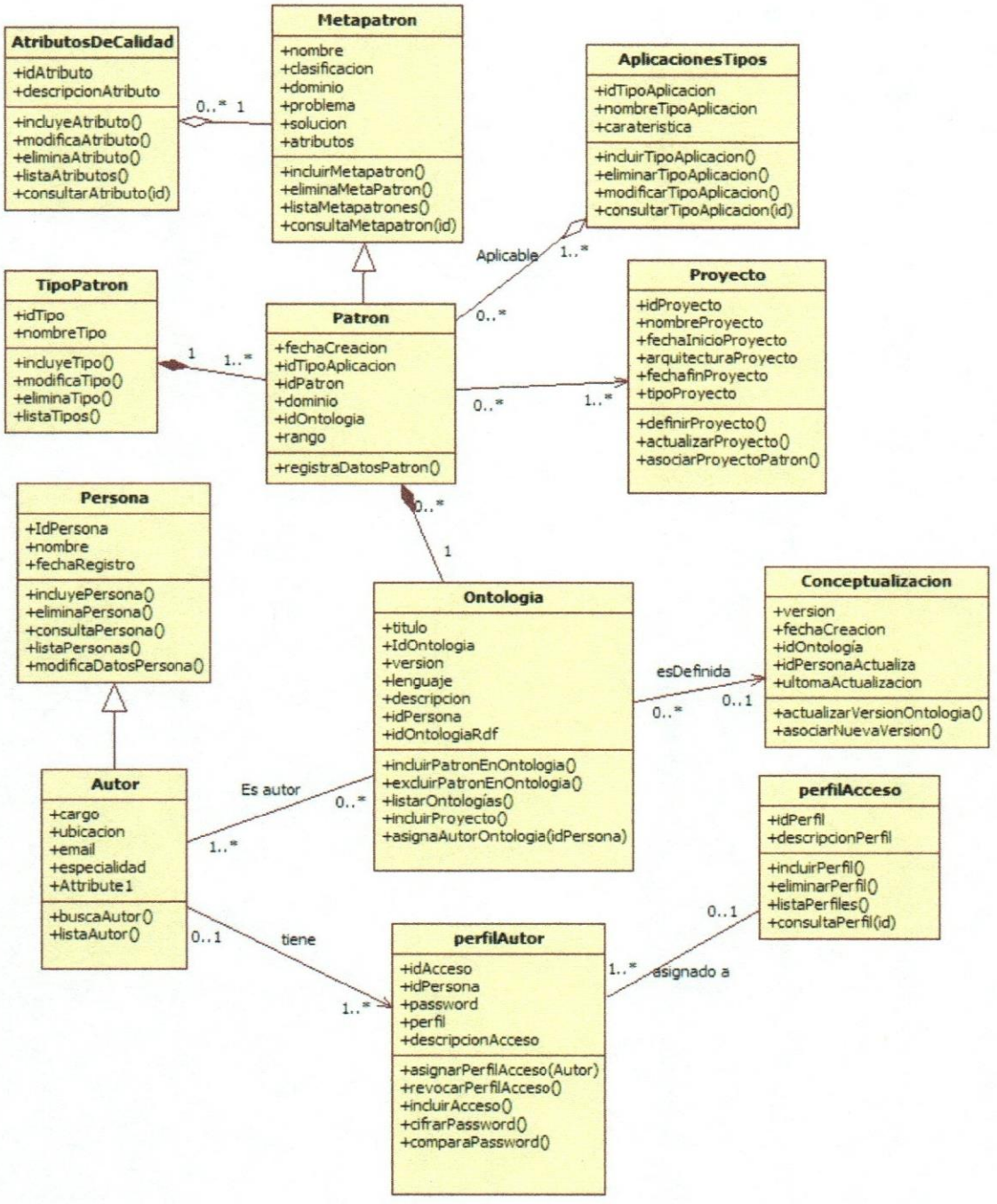


Figura 29. Modelo de repositorio propuesto para soportar la estructura de metadatos de la ontología

5.3. Desarrollo de un marco de trabajo para proporcionar una estrategia en el uso, reutilización y difusión del conocimiento en el proceso de desarrollo de sistemas.

En base a resultados obtenidos en la encuesta de reutilización de componentes, se determinó que no existe o es poca la comunicación entre los grupos de desarrollo en la difusión de componentes desarrollados, esto conlleva no solo a que no se logre la creación de soluciones comunes o conjuntos de librerías reutilizables que permitan disminuir el tiempo de desarrollo y disminuir las fallas en el código, sino que incide directamente en la calidad del software desarrollado.

Es por esto que se considera imprescindible para la implementación del modelo ontológico, que las organizaciones adopten modelos que fomentan la centralización de las funciones de administración de los repositorios de componentes y que se disponga de las herramientas y el conocimiento relacionado con cada dominio de componente para que este pueda ser difundido y realimentado con el conocimiento acumulado de los grupos de trabajo, además de definir estrategias y conjuntos de buenas prácticas en la organización para maximizar el uso de los recursos destinados en los proyectos en el área TIC.

En este punto es importante aclarar que este marco de trabajo va orientado a ofrecer un perfil de acceso basado en perfiles de grupo que permite segregar funciones en el uso del modelo ontológico propuesto, en base al modelo propuesto por Sametinger (1997). A continuación se visualizan los grupos de acceso propuestos y las funciones que tienen asociadas.

Figura 30. Grupo de acceso para desarrolladores

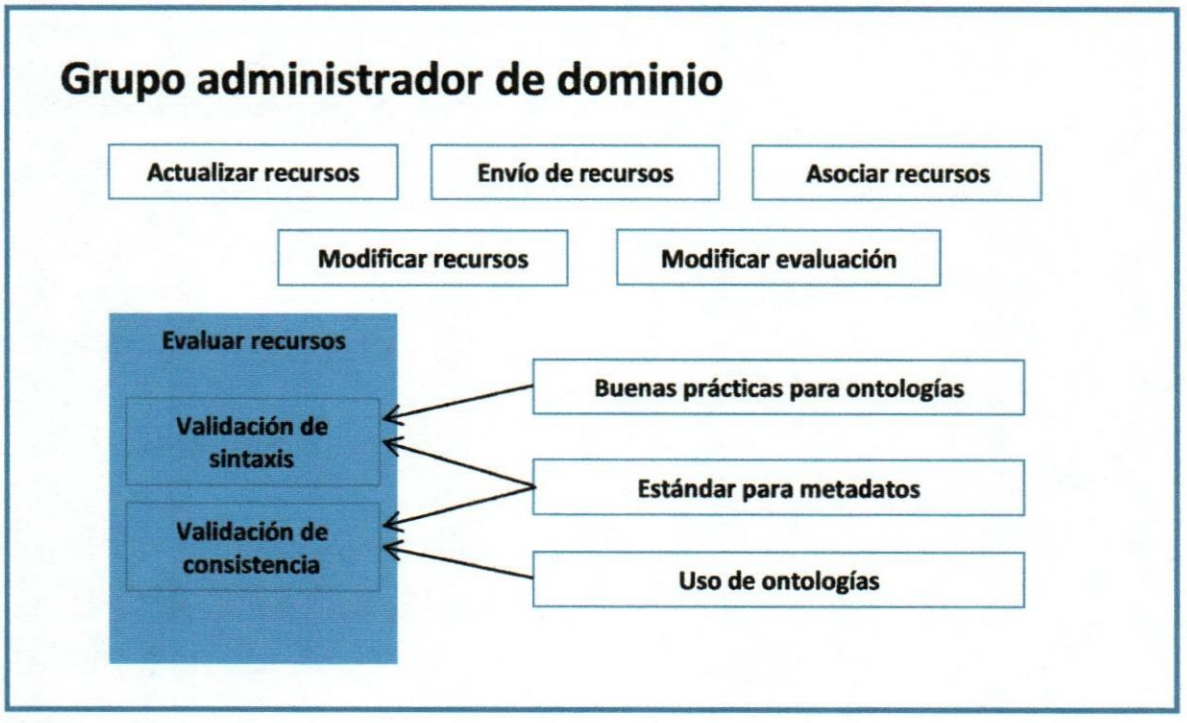


Figura 31. Grupo de acceso para administrador de dominio

En base a lo anterior el esquema de acceso a la ontología es recomendable hacer uso de un modelo de reutilización basada en dominios, esto es adaptable empresas medianas o grandes. En el caso de empresas de menor tamaño se recomendaría hacer uso como mínimo del esquema según el modelo de reutilización de software basado en componentes.

CAPITULO VI. VALIDACIÓN DEL MODELO ONTOLÓGICO PROPUESTO

6.1. Validación del modelo propuesto

Como toda aplicación de software, el contenido de las ontologías (conceptos, taxonomía y axiomas) debe ser evaluado antes de ser usado o reutilizado en otras ontologías o aplicaciones. En Gómez-Pérez, (2004) se señala, que la evaluación de la ontología es un juicio técnico de su contenido con respecto a un marco de referencia (requerimientos, preguntas de competencia, entre otros) durante todas las fases de su ciclo de vida.

Este proceso incluye la verificación y la validación de la ontología y debe llevarse a cabo en las definiciones y axiomas establecidos explícitamente en ella y en las que pueden ser inferidas desde otras aplicaciones. La verificación se refiere a construir la ontología correctamente, es decir, asegurar que sus definiciones implementan correctamente los requerimientos de la ontología.

Por otra parte, la validación se refiere a garantizar que las definiciones ontológicas modelan realmente el mundo real para el que la ontología fue creada. En el proceso de evaluación se establecen y ejecutan un conjunto de pruebas y se analizan los resultados de éstas (Hartman, 2005), partiendo de tres posibles estados de las ontologías:

a) Premodelado, donde se consideran la revisión y evaluación de los materiales disponibles para su construcción

b) Modelado, donde se comprueba la calidad de los significados y la consistencia y redundancia de los conceptos, utilizando otras ontologías y preguntas de competencia, así mismo, se evalúan los posibles errores sintácticos cometidos durante la codificación

c) Entregas ("delivery"), donde se determina su calidad, comparando con otras ontologías diferentes pero equivalentes

Según Ramos (2012), éstas propuestas coinciden en la evaluación de los siguientes criterios: a) la rigurosidad taxonómica, b) el lenguaje utilizado para la codificación, c) el rendimiento de las aplicaciones o tareas que utilizan las ontologías y d) el vocabulario utilizado para representar los conceptos y relaciones del dominio modelado.

En Ramos (2012), destacan que muchos de estos criterios se evalúan sobre la base de la comparación con otras ontologías disponibles las cuales se usan como referencia. Sin embargo, cuando ocurre que la ontología que se está desarrollando es la única descripción que se conoce del dominio, los desarrolladores se ven en la necesidad de utilizar de manera parcial los métodos disponibles no garantizando una evaluación suficientemente confiable y completa. En este sentido, Ramos (2012) indica la posibilidad de valorar la calidad de una ontología sin la necesidad de recurrir a referencias previas, examinando un conjunto mínimo de criterios como son: que el vocabulario utilizado para representar el conocimiento tenga cobertura suficiente del corpus, que la ontología esté escrita de manera correcta, sin errores y conforme a las reglas del lenguaje utilizado, que la estructura taxonómica que organiza los conceptos y términos del dominio sea completa, sin redundancias y consistente y que satisfaga los requerimientos para los cuales fue creada y, de manera particular, que las preguntas de competencia sean respondidas adecuadamente. A continuación, se presenta el esquema de evaluación propuesto en Ramos (2012) el cual está constituido por cuatro fases, una para cada criterio a evaluar: (1) Uso correcto del lenguaje; (2) Exactitud de la estructura taxonómica; (3) Validez del vocabulario y (4) Adecuación a requerimientos. La Tabla 1 describe para cada fase: las actividades, los insumos y el producto obtenido.

Tabla 10. Esquema para evaluación de Ontologías de un dominio.

DEL ESQUEMA PARA LA EVALUACION DE ONTOLOGIAS UNICAS EN UN DOMINIO			
NOMBRE/DESCRIPCION	**ACTIVIDADES**	**INSUMOS**	**PRODUCTO**
Uso correcto del lenguaje: Se evalúa la codificación, basado en las características y reglas de construcción del lenguaje usado.	✓ Seleccionar un lenguaje de codificación sólido y completo, que cumpla con estándares de desarrollo ontológico ✓ Evaluar sintácticamente la ontología en cada fase, para corregir inconsistencias sintácticas	✓ Ontología ✓ Herramientas para la evaluación (Editor Protegé-OWL, Analizador sintáctico DAML, entre otros)	Código libre de errores
Exactitud de la estructura taxonómica: Se examina la taxonomía. Para ello se considera la consistencia, completitud y no redundancia de los conceptos y términos.	✓ Identificar inconsistencias (conceptos que no pertenecen a una clase en particular, clases definidas como generalizaciones o especializaciones de sí mismas, entre otros) ✓ Evaluar la completitud de los conceptos codificados ✓ Evaluar la existencia de redundancias en clases, instancias y relaciones ✓ Identificar la existencia de clases e instancias con diferentes nombres y definiciones similares, clases que tienen más de una relación de subclase. ✓ Identificar la omisión de conocimiento disjunto entre clases de la estructura.	✓ Taxonomía	Taxonomía completa y libre de inconsistencias
Validez del vocabulario: Se evalúa el significado de los términos y conceptos a partir del conocimiento de expertos, recopilaciones de textos u otra fuente disponible del dominio.	✓ Identificar, extraer y organizar los términos significativos del dominio. ✓ Evaluar el vocabulario considerando medidas de calidad de resultados usadas en escenarios de recuperación de información.	✓ Conocimiento de expertos, textos y otras fuentes.	Tabla de términos válidos en el dominio
Adecuación a requerimientos: Se valida si la ontología implanta los requerimientos preestablecidos y si responde a las preguntas de competencia.	✓ Verificar que las especificaciones del documento de requerimientos se cumplan. ✓ Verificar que las respuestas proporcionadas por la ontología a las preguntas de competencia sean correctas y pertinentes.	✓ Documento de requerimientos de la ontología	Una Ontología que satisface los requerimientos identificados inicialmente [24]

Fuente Ramos (2012)

6.2. Evaluación de la ontología basada en métricas.

La creación de una ontología es un proceso iterativo e incremental por lo tanto, es una actividad que se realiza de manera paralela a cada una de las fases de la metodología utilizada; esto permite detectar errores y documentar cada fase de desarrollo. Durante el proceso de desarrollo del modelo se realizó un refinamiento de tres versiones del modelo.

Existe distintas herramientas técnicas para evaluar Ontologías basadas en distintos enfoques como: 1. cambios en la evolución del modelo, 2 Cambios en la lógica (Basado en reglas), 3 basado en métricas (Tartir et.al, 2010). En vista de ello se elabora un cuadro comparativo con las ventajas de cada uno a fin de evaluar los métodos aplicables al modelo propuesto:

Tabla 11. Tabla comparativa de algunas técnicas de evaluación de ontologías. x

Técnica	Enfoque	Tipo de usuario	Automático/Manual	Modelo
Arpinar et al. (2006)	Lógico	Developers	Manual	Schema +KB
Qood	Lógico	D+E	Manual	Schema +KB
Swoop	Logical	Developers	Automático	Schema
OntoMetric	Metric	Developers	Manual	Schema
Supekar etal. (2004)	Metric	D+E	Automático	Schema
AKTiveRank	Metric	D+E	Automático	Schema
Mostowfi and Fatouhi (2006)	Metric	Developers	Automático	Schema
oQual	Metric	D+E	Manual	Schema
OntoClean	Metric	Developers	Manual	Schema
OntoQA	Metric	D+E	Automático	Schema +KB

Fuente Ramos (2012)

En base a la información reflejada en la tabla anterior, se consideró hacer uso de aquellas técnicas que cubren el aspecto relacionado con el criterio de expertos de aplicación final y una técnica basada en métricas y de ejecución automática que será considerada en primer término para realizar las revisiones sucesivas del modelo. Esto considerando que el proceso de construcción de ontologías es un proceso de diseño necesariamente iterativo que ocurre durante todo el ciclo de vida de la ontología. (McGuinness, 2005).

Para la evaluación basada en métricas se seleccionó la aplicación OntoQA. Esta herramienta se basa en el análisis de las fuentes de información de la ontología como son las instrucciones en el lenguaje OWL por sus siglas en inglés Ontology Web Language, que contiene las declaraciones en lenguaje de marcado construido sobre las especificaciones de RDF y la base de conocimientos creada en el modelo. El motor de procesamientos de OntoQA procesa la información y genera las métricas basadas en métricas sobre la base de conocimientos KB metrics y basadas en el esquema de la Ontología Schema metrics, las cuales son requeridas para evaluar el modelo propuesto y tomar las decisiones relacionadas con el caso.

Definición 1: La riqueza en Las relaciones (RD) de un esquema están definidas como la tasa del número de relaciones (no por herencia) (P), dividido por el número total de relaciones definidas en el esquema (la sumatoria del número de relaciones por herencia (H) más el número de relaciones no hereditarias (P)). Esto es, si una ontología tiene un valor RD cercano a 0 indica que la mayoría de las relaciones son hereditarias, lo contrario si el valor está cercano a 1 indica que las relaciones son en mayoría no heredadas.

$$RD = \frac{|P|}{|P| + |H|}$$

Los valores obtenidos con la herramienta OntoQA aplicados a los valores calculados para la ontología se tiene que la riqueza en las relaciones es de un 0,6296, dado que P = 588 y H = 354. Indicando un porcentaje alto en cuanto a la variedad de relaciones de la ontología propuesta.

Profundidad del esquema (SD): Esta medida describe la distribución de la información a través de diferentes niveles de del árbol de la ontología. Este es un indicador que el conocimiento está agrupado en diferentes categorías y subcategorías en la ontología. Esta medida permite distinguir una ontología horizontal (donde las clases tienen un alto número de subclases directas) de una ontología vertical (donde las clases tienen un bajo número de subclases directas).

Una ontología con bajo número en la diversidad de relación en la herencia será una ontología con profundidad, lo que indica que la ontología cubre un dominio específico de manera detallada, mientras que una ontología con un valor alto en IR será una ontología de poca profundidad lo que indica que la ontología representa un amplio rango de conocimiento general sin un alto nivel de detalle:

Definición 2: La riqueza de la herencia del esquema (IR) está definida como el promedio del número de subclases (H) por cada clase (C).

$$SD = \frac{|H|}{|C|}$$

Aplicándo los valores de los términos dados H=354, C=76

SD=4,66

Profundidad de las propiedades (AR): Es el número de atributos (Slots) que son definidos para cada clase, pueden indicar tanto la calidad del diseño de la ontología como la cantidad de información pertinente a los datos de la instancia. En general esta regla asume que mientras más slots tiene el esquema, mas conocimiento provee la ontología.

Definición 3: La profundidad de los atributos está definida como el promedio del número de atributos o slots por clase. Se calcula como el número de atributos para todas las clases (att), dividido entre el total de clases (C).

Se obtiene al dividir el total de atributos o propiedades del esquema att=133 entre el total de clases creadas C=76

$$AR = \frac{|att|}{|C|}$$

AR=133/76 = 1,75 Lo que indica que existe un promedio de casi dos atributos por cada clase del esquema.

b. Métricas aplicables a la base de conocimientos

Este grupo de métricas ofrecen una visión de cómo las instancias son representadas en la base de conocimientos.

Uso de las clases (CU): Esta métrica refleja como las clases definidas en el esquema son usadas por la base de conocimientos indicando una base de conocimientos más amplia a medida que el valor es más alto.

Definición 4: El uso de las clases (CU) de una ontología esta definida como la tasa que se obtiene del número de clases instanciadas (C') entre el número total de clases (C). El resultado es un porcentaje indicando como la base de conocimientos utiliza las clases definidas en el esquema. Si la base de conocimiento tiene un valor muy bajo de CU,

entonces indica que no tiene los datos suficientes que ejemplifiquen todo el conocimiento que refleja el esquema. Viene dado por:

$$CU = \frac{|C'|}{|C|}$$

Total de clases con instancias (C'): 13 (Directas e indirectas)

Total de clases del esquema: C: 76

CU: 13/76= 17%

Conectividad de las clases Conn(Ci) : Este valor aporta un indicativo de la centralización de las clases, basado en las relaciones de instancias. Esta medida funciona a la par con la métrica de importancia de la clase para lograr un mejor entendimiento del uso de las clases en el esquema y determinar la naturaleza de la ontología, indicando cuales clases juegan un rol central comparado con otras clases.

Definición 5: La conectividad de una clase (*Conn*(C_i)) está definida como el número total de relaciones de las instancias de las clases que tienen instancias relacionadas con otras clases (NIREL).

El resultado de la fórmula es un valor entero que representa la popularidad de las instancias de la clase, una clase con un al valor de Conectividad indica que juega un papel centralizador en distintos conceptos del dominio, ya que se relaciona con instancias de otras clases.

A continuación se presenta el resumen de relaciones de clases obtenido a partir del análisis realizado por OntoQA.

$$Conn(C_i) = |NIREL(C_i)|$$

Total conecciones de instancias directas: 60

Conectividad total de las clases: 60

Con esta métrica se demuestra que se requiere incrementar

la cantidad de instancias para cada clase a fin de ir representando el dominio total de la ontología.

Importancia de las clases Imp(C_i): Al igual que el indicador anterior, la importancia de clases se aplica a cada una de las clases. Su importancia se debe a que ayuda a identificar cuales áreas del esquema están enfocadas en ciertos aspectos de la ontología, cuando las instancias de la información que contienen sean extraídas para su consulta o actualización.

Definición 7: La importancia de la clase está definida como el número de instancias que pertenecen a cada una de las herencias por cada clase en la base de conocimientos, comparado con el número total de instancias de clases en la ontología.

$$\mathrm{Imp}(C_i) = \frac{|\mathrm{Inst}(C_i)|}{|\mathrm{KB}(CI)|}$$

Al aplicar la fórmula anterior a cada clase del modelo, se obtiene un índice de Imp(Ci) bajo, esto en vista que las instancias de clase no se han generado en su totalidad. Es importante destacar que las instancias representan la información del modelo y que las ontologías se va ampliando el conocimiento a medida que se intercambia información con los actores que la usan. Este indicador se irá incrementando a medida que el modelo va madurando en conocimiento y relaciones. Se puede ver el detalle de los valores de las instancias de cada clase con sus distintos valores calculados en el anexo IV.

Cohesión (Coh): Esta métrica representa el número de componentes conectados en la base de conocimiento. Su aporte puede ayudar particularmente a determinar las llamadas "islas" de información donde no existe conectividad entre componentes lo que permitiría al desarrollador ampliar la relación de conceptos y su semántica en la ontología.

Definición 7 La medida de cohesión de las clases esta dada por el número de componentes conectados (CC) en la representación de la ontología

$Coh = |CC|$ La conectividad de la ontología se ubica en 60 conexiones entre clases.

Ver Anexo IV. Coh = 60

6.3. Cálculo de la puntuación general de la ontología. (Puntuación)

Puntuación: Los datos aportados por OntoQA permiten evaluar la ontología usando los valores relacionados y determinados por la sumatoria de los pesos de las métricas indicadas en los pasos anteriores. Esto es:

$$Score = \Sigma W_i * Métrica_i$$

Definición 8: La puntuación a obtener en una ontología puede ser medida como el promedio de los pesos de las métricas del esquema por el valor general para la métrica específica.

Donde:

Métrica {} = {RD,SD,CU,Coh,#Clases,#Relaciones,#Instancias,Promedio(Imp(ci)),Promedio(RU(Ci)),Promedio(Imp(Ri))} => Es el conjunto de métricas usadas en el cálculo global del la puntuación de una ontología. (Los promedios son para clases y relaciones asociadas a los términos)

W{} => Es el conjunto de pesos para cada métrica.

Cabe destacar que el método de cálculo indicado por OntoQA, no especifica los valores para los pesos que se pueden asignar a cada métrica para el cálculo. En este caso, recomiendan ir ajustando los pesos a cada una de las métricas calculadas a medida que el usuario pueda darle mayor importancia a alguno de los aspectos evaluados para cada una, incluso en cada fase del proceso de desarrollo, permitiendo a los desarrolladores considerar diferentes enfoques para reforzar indicadores que tengan mayor deficiencia.

Para efectos de este modelo, se asignó un mismo peso a cada métrica excepto para la cantidad de clases y la profundidad en la herencia con un valor de 20 puntos y valores por debajo de 20 para el resto de las variables del cálculo de la puntuación general.

Por otra parte, el método tampoco considera rígido el uso de las distintas métricas para el cálculo de la puntuación, por lo que se seleccionaron variables que aportan un valor determinante para la fase de desarrollo de la ontología propuesta. Adicionalmente se considera hacer uso de pesos valorados entre 1 y 100 de manera que superen los 100 puntos en la sumatoria total de pesos por ítem.

Por ello se consideraron las siguientes métricas para el cálculo:

Métrica {} = {RD,SD,CU,Coh,#Clases,#Relaciones,#Instancias,Promedio(Imp(ci)),RR, IR} => Es el conjunto de métricas usadas en el cálculo global de la puntuación de una ontología. (Los promedios son para clases y relaciones asociadas a los términos)

RR = Profundidad de las relaciones (Relationship Richnness)

IR = Profundidad de la herencia (Inheritance Relationship)

Quedando la fórmula final como:

$$Score = (\Sigma W_i * Métrica_i)/(\Sigma W)$$

A continuación se indican resultados:

Tabla 12. Tabla de resultados de las métricas obtenidas.

Peso	Métrica	Descripción	Origen	Valor de la Métrica	Resultado (Valor * peso)
10	RD	Diversidad de relaciones	Cálculos de métricas	62,96	629,6
5	SD	Profundidad del esquema (SD)	Cálculos de métricas	4,66	23,3
5	CU	Uso de las clases (CU):	Cálculos de métricas	17	85
10	Coh	Cohesión	Cálculos de métricas	60	600
20	#Clases	Cantidad de clases	OntoQA	76	1520
5	#Relaciones	Número de relaciones	OntoQA	45	225
10	#Instancias	Instancias	OntoQA	13	130
10	RR	Profundidad de relaciones	OntoQA	21	210
20	IR	Profundidad de la herencia	OntoQA	64,17	128,3
5	Promedio(Con(C_i))	Promedio de Conexiones para las clases	Cálculos de métricas	80	400
100			Puntuación:		51,06

6.4. Evaluación de la Ontología basada en opinión de expertos

Según se indica en el punto anterior se selecciona como segunda técnica la opinión de expertos y se consideraron un grupo de variables indicadas en la metaontología, Qood (Quality oriented ontology Descriptions) planteado por Gangemi et al.(2006) que consiste en evaluar 9 principios que definen niveles de calidad en ontologías y que serán

considerados en este proceso de evaluación de ontologías como las variables a medir por los expertos del área de la empresa seleccionada como estudio de caso para evaluar la efectividad del modelo. A continuación se detallan estos 9 principios:

Tabla 13. Framework basado en metaontologías para evaluación y selección de ontologías. Fuente: Gangemi, et al,(2006)

Variable evaluada	Instrumento	Definición conceptual	Definición Operacional
Ergonomía cognitiva: Cognitive ergonomics	Cuestionario, Opinión de expertos	Perspectiva de una ontología que pueda ser fácilmente comprendida, manipulada y explotada por los usuarios finales	Valoración de Items presentes. Valores posibles 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
Transparencia: (Transparency)	Cuestionario, Opinión de expertos	Perspectiva de que la ontología puede ser analizada en detalle, con profundidad en los conceptos y motivaciones	Valoración de Items presentes. Valores 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
Integridad computacional y eficiencia	Cuestionario, Opinión de expertos	Perspectiva de que la ontología puede ser analizada fácilmente por el razonador.	Valoración de Items presentes. Valores posibles 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto

Variable evaluada	Instrumento	Definición conceptual	Definición Operacional
Integridad(Meta-level integrity)	Cuestionario, Opinión de expertos	Perspectiva de que la ontología considera un criterios que se asumen como indicadores de calidad	Valoración de Items presentes. Si está presente agrega un punto a la calificación final. Valores posibles 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
Flexibilidad	Cuestionario, Opinión de expertos	Perspectiva de que la ontología puede ser adaptable a múltiples vistas de la información	Valoración de Ítems presentes. Si está presente agrega un punto a la calificación final. Valores posibles 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto
Cumple con el proceso de experticia general	Cuestionario, Opinión de expertos	Perspectiva de que la ontología puede ser aplicable por un usuario experto a través de la herramienta proporcionada	Valoración de Ítems presentes. Si está presente agrega un punto a la calificación final. Valores posibles 1. Muy bajo, 2. bajo, 3. Regular, 4. Alto, 5. Muy Alto

6.5. Resultados obtenidos de la encuesta realizada a expertos en el dominio de los patrones de diseño.

A continuación se muestra un gráfico donde se aprecia el resultado de la evaluación de expertos del Modelo propuesto:

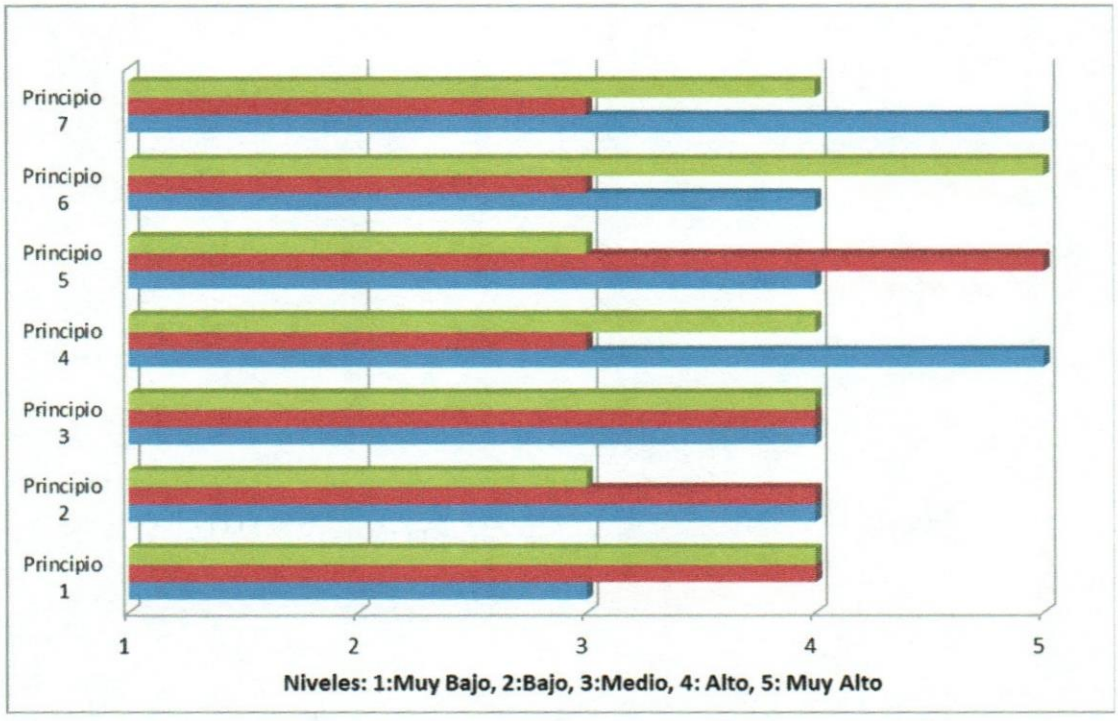


Figura 33. Nivel de cumplimiento de los principios para la evaluación de ontologías indicados en Gangemi et al., (2006) del modelo propuesto.

En la siguiente tabla se aprecian los resultados obtenidos luego de promediar el comportamiento general de la ontología en cuanto a los principios para la evaluación de ontologías, los detalles de los valores obtenidos en el cuestionario se pueden visualizar en el anexo III.

De estos resultados se puede inferir que existen competencias a reforzar sobre todo en los dos primeros principios por encontrarse en niveles medios a altos.

Tabla 14. Promedios obtenidos en la evaluación de expertos en relación al uso de los principios para la evaluación de ontologías.

Principio	Promedio
Sobre las características relacionadas con la Ergonomía cognitiva.	3,67
Sobre las características relacionadas con la Transparencia	3,67
Sobre las características relacionadas con la Integridad computacional y eficiencia	4,00
Sobre las características relacionadas con la Integridad (Meta-level integrity).	4,00
Sobre las características relacionadas con la Flexibilidad	4,00
Sobre las características relacionadas con cumplimiento del proceso de experticia general	4,00
Sobre las características relacionadas a la adaptabilidad al negocio	4,00
Promedio General	3,90

6.6. Cálculo global del porcentaje para determinar la factibilidad del modelo

(1) Por otra parte el resultado obtenido del cálculo a través de métricas fue de un 51,06 de la puntuación máxima de 100, lo que se traduce a un 51,06%.

(2) En la medición de expertos se obtuvo un valor de 3,90 de la puntuación máxima de 5 lo que indica que se ubica en un porcentaje del 78% de la calificación máxima para la evaluación de expertos.

Se procedió a unificar los porcentajes para lograr la puntuación global que permitió determinar la aplicabilidad del modelo en la empresa objeto del caso de estudio. Para realizar esta actividad se procedió a la aplicación del modelo de decisión propuesto en Bonillo (2004). En el modelo propuesto por el autor, la decisión final parte de una asignación porcentual asociada a los resultados obtenidos en cada técnica usada para evaluar resultados, esto es, asignándole un peso específico a cada uno.

Para efectos de este trabajo se asignó un valor del 25% al resultado obtenido en el método de cálculo a través de métricas del sistema, tomando en cuenta que en esta fase de la ontología, está más desarrollada la conceptualización del modelo de conocimiento que la definición de su contenido e instancias que incrementan su valor en las métricas como se puede apreciar en el procedimiento de cálculo de las métricas, y un 75% a las preguntas clave aplicadas a los expertos que aportaron una revisión completa de la taxonomía para determinar que conceptualmente es aplicable el modelo. En la siguiente figura se aplica el modelo de decisión de MOSEFPIV (Bonillo, 2004)

Aplicando los pesos a los valores calculados en **(1)** y **(2)** se obtiene que:

Tabla 15. Cálculo global del porcentaje para determinar la factibilidad del modelo

Método de cálculo	Resultado del método	Peso aplicable	Resultado del cálculo global
(1) Método de evaluación a través del cálculo de métricas	51,06%	25%	12,77%
(2) Técnica de medición Ontológica a través evaluación de expertos.	78%	75%	59%
Resultado			71,27%

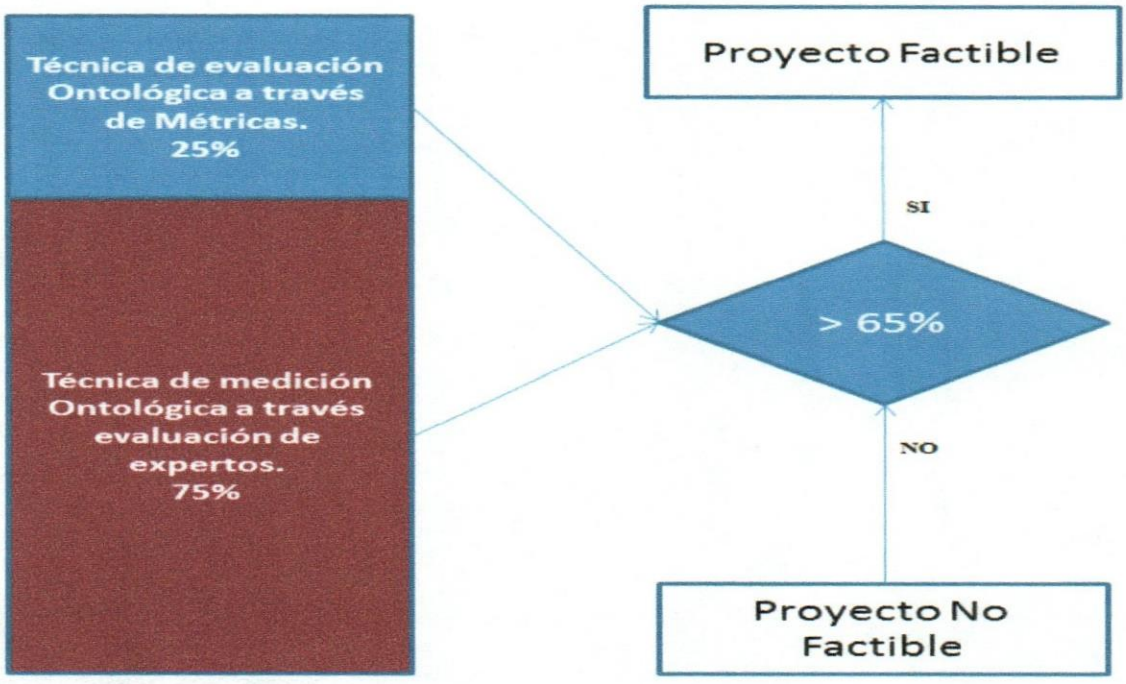


Figura 34. Modelo de decisión de MOSEFPIV para determinar la aplicabilidad del proyecto en empresas del sector financiero. Fuente: adaptado de Bonillo (2004)

Según el resultado obtenido que se indica en la tabla 15, se puede concluir que el modelo ontológico desarrollado es factible su implementación toda vez que supere siempre el 65% que se indica en la figura anterior. Cabe destacar que los valores obtenidos en las métricas arrojan un resultado del 51,06% que se encuentra aún en la mitad del valor óptimo que para el cálculo global representa un 12,77%. Esto se debe a que la ontología desarrollada debe ir completándose con los valores de las instancias correspondientes a los patrones, antipatrones y *frameworks* identificados lo que incrementará aún más los resultados.

CAPITULO VII. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

En este proyecto, se realizó una propuesta que permite demostrar la necesidad de aplicar buenas prácticas y uso estándares de calidad en el proceso de desarrollo de software y en el producto final, considerando siempre la importancia del proceso de desarrollo de software basado en la reutilización de componentes. El objeto de este estudio fue lograr definir un modelo basado en ontologías que permita definir la base de conocimientos de los distintos patrones y estándares en la industria de software, y los patrones de software especializados implementados en las áreas de desarrollo de los sistemas bancarios, definir un esquema de trabajo amplio que abarca la definición de un repositorio ontológico de patrones y el esquema de organización de las áreas de administración de las ontologías para hacer el mantenimiento y control del repositorio de patrones.

Su desarrollo representó además, un avance significativo en el proceso de desarrollo de software en el sector financiero en que se plante la presente investigación, esto se debe a que se demostró que las prácticas de uso de patrones bajo criterios adecuados en las empresas no está siendo usado en la actualidad en las organizaciones y puede ser optimizado implementando mejores prácticas. Además la optimización en el uso de los recursos en función del tiempo y recursos y la definición de un conjunto de herramientas reutilizables que van a ir perfeccionándose de los conocimientos de los grupos haciendo énfasis en la reutilización de los componentes y gestión del conocimiento como activo de la información para la empresa.

Se realizó un diagnóstico adecuado del problema relacionado a la calidad del software desarrollado y la necesidad de recurrir a mejores prácticas para la reutilización de los componentes de software y optimización de los procesos de desarrollo de software, esto se logró a través de los distintos mecanismos para recabar información como encuestas y entrevistas directas con los expertos del área de estudio. Se identificó claramente que existen deficiencias notorias en los procesos de desarrollo y en el uso de buenas prácticas

en la construcción de software, un caso particular destacable que se presenta en distintas organizaciones es el bajo índice de reutilización de software que existe y el desconocimiento del trabajo realizado por otras áreas de desarrollo de la misma empresa.

La elaboración de una taxonomía de patrones de manera ontológica permitió agrupar el conocimiento más básico del dominio estudiado según su área de aplicación en el uso de patrones y componentes en base a la metodología seleccionada para elaboración de ontologías. El resultado fue satisfactorio y se determina que es factible su implementación, es importante que esta primera versión de la ontología abarca los conceptos básicos en el conocimiento de los patrones de diseño de software conocidos en el mercado y que necesariamente debe ser enriquecida con las nuevas prácticas, nuevos modelos de patrones y nuevos conocimientos de los expertos en esta materia. Cabe destacar que los valores obtenidos de las métricas proporcionadas por OntoQA fueron valores intermedios que deben ir ampliándose en base a la realimentación del conocimiento y creación de las instancias de clases propuestas, lo que permitirá optimizar los valores obtenidos y optimizar la calificación del modelo.

Conjuntamente con el modelo ontológico, se elaboró una propuesta para desarrollar a futuro el repositorio de patrones donde se alojará la base de conocimientos obtenidos por el modelo desarrollado, su integración al esquema ontológico propuesto y uso a lo largo del tiempo representa un activo importante de información para la organización.

El marco de trabajo aplicado es una propuesta importante para reorganizar la estructura del área de sistemas, toda vez que se quiera implantar el modelo en la organización, ya que es imprescindible que exista un área de control y administración del recurso de la información que se genera a través de la ontología propuesta, que permite centralizar el conocimiento y ente difusor de la información que aplica al proceso de desarrollo de sistemas de la organización.

Los resultados de la evaluación del modelo propuesto se realizaron de manera satisfactoria validándose éste con dos técnicas reconocidas para su certificación en su fase inicial. Se

destaca esta primera fase porque dispone de los conceptos, relaciones, propiedades y axiomas básicos para el uso de patrones de software. En esta etapa de la investigación se logra calificar el modelo según el uso de conocimientos de expertos que se desempeñan en las empresas objeto del estudio, específicamente en el área de desarrollo de sistemas. Se aplicó un segundo método de evaluación del tipo cuantitativo para el cálculo de la efectividad del conocimiento contemplado en el esquema de la ontología donde se obtuvo una calificación que demuestra que contiene los elementos necesarios desde un punto de vista técnico. Se aplica un método para unificar los resultados de ambas evaluaciones obteniéndose un valor superior al esperado que es determinante en la decisión de aplicabilidad del modelo.

7.2. Recomendaciones

El nivel de madurez del modelo ontológico presentado es comparable con el modelo de madurez CMMI de grado 2, es decir, es un modelo conceptual repetible que debe ser implementado siguiendo distintas pautas que van desde la realimentación del modelo ontológico a través del conocimiento de los grupos de trabajo y creando instancias y nuevas reglas que alimentan el modelo hasta la medición de la calidad usando distintas métricas aquí explicadas que corresponden al modelo de evaluación de ontologías OntoQA. Como se ha comentado a lo largo de esta investigación, esta ontología está desarrollada considerando los conocimientos actuales y estándares del área de la investigación y es imprescindible que este conocimiento se amplíe siguiendo las pautas del ciclo de vida de las ontologías.

La arquitectura del modelo de repositorio propuesto puede ser desarrollada bajo cualquier framework del mercado, siguiendo las mejores prácticas definidas en la organización, la propuesta indica que deben existir como mínimo un número de tres capas que permitan separar la lógica de negocio, la lógica del modelo y el componente visual. La capa de acceso a datos debe considerar módulos de acceso al conocimiento de las ontologías a través del lenguaje de consultas OntoQL y módulos de acceso al modelo relacional propuesto con el uso de lenguaje de consultas estándar.

A nivel organizacional es imprescindible definir un esquema que permita como mínimo tener un grupo que administre el conocimiento del dominio de los patrones de software, esto es, al menos un área que permita actualizar los recursos, diseñar y normar el uso de buenas prácticas en el uso de las ontologías existentes en el repositorio de la organización y que permita la abstracción de tales funciones a los grupos de desarrollo de software.

Cabe destacar que el modelo en sí mismo no es la solución definitiva, es realmente la base de conocimiento que debe ser accedida a través de herramientas de apoyo para el uso de la ontología y que permitan por ejemplo, automatizar el proceso de autoevaluación y búsqueda de la información contenida en el esquema ontológico y facilitar el proceso de

realimentación y validación del modelo. Sin embargo, se debe mantener el concepto de colaboración entre todas las demás áreas involucradas para garantizar la realimentación del modelo planteado y proveer una infraestructura estable y al mismo tiempo adaptable a los cambios del modelo de la empresa para garantizar la interoperabilidad de las áreas de desarrollo de software e interactuar con el modelo para intercambio de información entre las áreas.

REFERENCIAS BIBLIOGRÁFICAS

Acosta, A., & Zambrano, N. (2004). ***Patterns and Objects for User Interface.*** **Caracas, Venezuela: Journal of Technology Published by ETH Zurich, Chair of Software Engineering.**

Bachmann, F., Bass, L., Buhman, C., Comela-Dorda, S., & Long, F. (2000). Volume II: Technical Concepts of Component-Based Software Engineering, 2nd Edition (CMU/SEI-2000-TR-008). Retrieved November 01, 2013, from the Software Engineering Institute, Carnegie Mellon University. Obtenido de http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5203

Barchini, G., Álvarez, M., & Herrera, S. (2006). ***Sistemas de Información: Nuevos Escenarios basados en ontologías.*** **Revista de Gestão da Tecnologia e Sistemas de Informação. Vol. 3, No. 1, 2006, p.2-18.**

Barrera, M., Núñez, H., & Ramos, E. (2012). ***Ingeniería Ontológica. Lecturas en Ciencias de la Computación.*** **Caracas: Centro de Ingeniería de Software y Sistemas (ISYS).**

Baskerville, R. (1999). Investigating information Systems with action research. Comunications of the Association for Information Systems. Georgia, USA: Computer Information Systems Department. Georgia State University.

Bonillo, P. (2004). Modelo sistémico de evaluación financiera de proyectos de inversión de tecnología de la información en Venezuela. Universidad de Yacambu.

Bonillo, P. (2006). ***Metodología para la gerencia de los procesos del negocio sustenda en el uso de patrones.*** **Revista de Gestão da Tecnologia e Sistemas de Informação Journal of Information Systems and Technology Management Vol. 3, No. 2.: Centro ISYS, Facultad de Ciencias, UCV.**

Briones, G. (1996). ***Metodología de la investigación cuantitativa en las ciencias sociales.*** **Bogotá, Colombia: Instituto colombiano para el fomento de la educación superior, ICFES.**

Budgen, D. (2003). ***Software design.*** **Harlow, England: Addison-Wesley Publishers Limited.**

Corcho, O., Fernández, M., Gomez, A., & Lopez, A. (2005). ***Construcción de ontologías legales con la metodología METHONTOLOGY y la herramienta WebODE.*** **Madrid: Facultad de Informática. Universidad Politécnica de Madrid.**

Ding, P., & Yan, P. (2006). ***Using Ontology Repository to Support Data Mining.*** **Dalian, China: Proceedings of the 6th World Congress on Intelligent Control. IEEE.**

Flores, N. (2005). ***Engenharia Reversa de Padrões em Arquitecturas Reutilizáveis.*** **Porto, Portugal: Faculdade de Engenharia da Universidade do Porto.**

Gamma, E., Helm, R., Jhonson, R., & Vlissides, J. (1997). ***Design Patterns Elements of reusable Object-oriented software.*** **California: Adison-Wesley.**

Gangemi, A., Ciaramita, C., & Lehmann, J. (2006). ***Qood Grid A Metaontology-based framework for ontology evaluation and selection.*** **Proceedings of the EON'2006 Workshop.**

Giraldo, G. L., Acevedo, J., & Moreno, D. (2011). ***Una ontología para la representación de conceptos de diseño de software.*** **Medellín, Colombia: Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia.**

Gomez, A., Suarez, M., & Villazón, B. (2004). ***Ontological Engineering.(Vol. 139).*** **London: Heidelberg: Springer.**

Gomez, D., & Valdeoriolla, J. (2009). Metodología de la Investigación. Universidad Oberta de Catalunya. Catalunya, España: FUOC.

Gonzalez, R., & Torres, M. (2005). Ontología de Componentes de Software para PyMEs: Informe Final.Pontificai Universidad Javeriana. ESpaña: Departamento de Ingeniería de Sistemas.

Greenfield, J., & Short, K. (2004). Software Factories Assembling Applications with Patterns, Models, Frameworks and Tools. Primera Edición. California, USA.: Wiley.

Grimán, A., Domínguez, K., Pérez, M., & Mendoza, L. (2006). ***Quality in Development Process for Software Factories According to ISO 15504.*** **Omaha, USA: roceedings of the Eleventh Americas Conference on Information Systems.**

Grimán, A., Pérez, M., Garrido, J., & Rodriguez, M. (2006). ***An ontology for architectural evaluation case study: collaboration systems.*** **International Conference on Enterprise Information Systems - ICEIS.**

Grimán, A., Pérez, M., Mendoza, L., y Dominguez, K. (2005). ***Hacia una ontología para fábricas de software.*** **Valdivia, Chile: V Workshop de Ingeniería del Software.**

Guarino, N. (2013 de Septeiembre de 1998). ***Formal Ontology and Information Systems. Proceedings of FOIS '98.*** **Obtenido de http://www.loa.istc.cnr.it/Papers/FOIS98.pdf**

Hamza, H. (2004). ***Improving Analysis Patterns Reuse:An Ontological Approach.*** **Nebraska, USA: Computer Science & Engineering Dept. University of Nebraska-Lincoln.**

Hartmann, J., Spins, P., Giboin, A., & Maynard, D. (2005). ***Methods for Ontology Evaluation. Knowledge Web.*** **Project Number IST-2004-507-507-482.**

Henninger, S., & Corrêa, V. (2007). Software Pattern Communities: Current Practices an challenges. CSE Technical reports Computer Science and Engineering. Nebraska, USA.

Hernández, A. (1991). ***La investigación-Acción Participativa y la Producción de Conocimientos.Revista de Faces.*** **Valencia, Carabobo: Facultad de Ciencias Económicas y Sociales. Universidad de Carabobo.**

IEEE Computer Society. (2004). SWEBOK - Guide to the Software Engineering Body of Knowledge. IEEE.

IEEE.Institute of Electrical and Electronic Engineers. (30 de 10 de 2013). ***IEEE - Code of Ethics*** **. Obtenido de www.ieee.org/about/corporate/governance/p7-8.htm**

Kitchenham, B. (1996). DESMET: A method for evaluating Software Engineering methods and tools. Staffordshire, UK: University of Keele .

Krause, M. (1995). La investigación cualitativa: Un campo de posibilidades y desafíos. Revista de temas de educación. N° 7.

Laudon, K., & Laudon, J. (2012). ***Sistemas de Información Gerencial. Décimosegunda edición.*** **México: Pearson Educación.**

Mattsson, M. (1996). ***Object-Oriented Frameworks. A Survey of methodolical issues.*** **Lund, Sweden: Department of Computer Science. Lund University.**

McGuinness , D., & Noy, N. (2005). ***Ontología, Desarrollo de Ontologías-101: Guía Para Crear Tu Primera.*** **Standford, California: Stanford University, Stanford, CA, 94305.**

Medina, F. (2010). Marco Metodológico para la Mejora de la Eficiencia de Uso de los Procesos de software. Leganés. España: Universidad carlos III de madrid.

Montilva, J., Arapé, N., & Colmenares, J. (2003). Desarrollo de Software Basado en Componentes.IV Congreso de Automatización y Control. Mérida, Venezuela.

Palella, S., & Martins, P. (2006). ***Metodología de la investigación cuantitativa.*** **Venezuela: Fedeupel.**

Perez, M., Griman, A., Mendoza, L., & Rojas, T. (2004). A Systemic Methodological Framework for IS Research.Proceedings of the Tenth Americas Conference on Information Systems. New York, New York: Processes and Systems Department – LISI.

PMI. (2008). PMBOK Guía de los Fundamentos de la Dirección de Proyectos. Pennsylvania.

Pressman, R. (2010). Ingenieria del Software. In enfoque práctico. Qunta Edición. Madrid: Mc Graw Hill.

Ramos, E., Nuñez, H., & Casañas, R. (2012). ***Esquema para evaluar ontologías únicas para un dominio de conocimiento.*** **Obtenido de Enl@ce: Revista Venezolana de Información,Tecnología y Conocimiento, 6 (1), pp: 57-71. ISSN: 1690-7515.: http://dialnet.unirioja.es/servlet/articulo?codigo=2932226**

Rojas, E., González, D., Roche, F., & Toledo, E. (2012). Aplicación informática para gestionar repositorios, establecer semejanzas y caracterizar ontologías.Revista Cubana de Ciencias Informáticas. La Habana Cuba: Grupo Editorial "Ediciones Futuro".

Ruiz, J. (06 de 01 de 2014). ***Impulsos. Emprender, innovar y liderar desde el cambio.*** **Obtenido de http://www.innpulsos.com/2012/11/18/pasos-para-desplegar-la-investigacion-accion-i-de-ii/**

Sametinger, J. (1997). ***Software Engineering with Reusable Components.*** **Linz, Austria: Springer-Verlag.**

Scalone, F. (2006). Estudio compartivo de los modelos y estándares de calidad de software.Universidad Tecnológica de Buenos Aires. Buenos Aires.

Tamayo, T. &. (2003). ***El proceso de la investigación científica*** **. Mexico: Limusa Noriega Editores .**

Tartir, S., Arpinar, B., & Sheth, A. (2010). ***Theory and Applications of Ontology: Computer Applications,.*** **Springer Science+Business Media B.V. pp 115-130.**

Tonietto, C. (2007). ***Rpao : repositório de padrões de análise mapeados em ontologias.*** **São Leopoldo, Brasil: Universidade do vale do rio dos sinos. Centro de ciências exatas e tecnológicas.**

Toward, G. (2011). Principles for the Design of Ontologies Used for Knowledge. California: Stanford University.

UPEL Universidad Pedagógica Experimental Libertador. (2006). ***Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales. 3a reimpresión, Mayo 2006.*** **Caracas, Venezuela: Fondo Editorial de la Universidad Pedagógica Experimental Libertador.**

Uschold, M. (1996). ***Building ontologies: Towards a unified methodology.*** **Obtenido de http://www.aiai.ed.ac.uk/project/pub/documents/1996/96-es96-unified-method.pdf**

Uschold, M., & Jasper, R. (1999). ***A Framework for Understanding and Classifying Ontology Applications.*** **Seattle, USA: Boeing Math and Computing Technology.**

Venezolano de Crédito Banco Universal. (2012). Manual de Normas y Procedimientos Generales. Caracas, D.C., Venezuela.

Wand, Y., & Weber, R. (1990). ***An ontological model of an information system. IEEE Transactions on Software Engineering. November, pp. 1282-92.*** **Obtenido de www.public.it-sudparis.eu/~gibson/Teaching/CSC7302/ReadingMaterial/WandWeber90.pdf**

Wohed, P., & Johannesson , P. (1998). ***Deontic Analysis Patterns.*** **Stockholm: Dept. of Computer and Systems Sciences.**

ANEXOS

ANEXO I

Tabla 16. Colecciones de patrones por el número de patrones que contienen. Fuente: Henninger (2007)

Title	Source	# of Patt.	Year
Patterns in Interaction Design	http://www.welie.com/	146	2005
"Analysis Patterns: Reusable Object Models"	"Analysis Patterns: Reusable Object Models"	95	1996
"Designing Interfaces: Patterns for Effective Interaction Design"	http://www.mit.edu/~jtidwell/common_ground_onefile.html	94	2005
Ajax Design Patterns	http://ajaxpatterns.org	70	2006
"Requirements Patterns and Antipatterns: Best (and Worst) Practices for Defining Your Requirements"	http://www.tabletuml.com/RPandAP/default.aspx	69	2007
"Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions"	http://www.eaipatterns.com/toc.html	65	2003
Yahoo! Design Pattern Library	http://developer.yahoo.com/ypatterns/	63	2005
"Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects"	"Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects"	55	2004
"J2EE Antipatterns"	"J2EE Antipatterns"	52	2003
"Patterns of Enterprise Application Architecture"	http://www.martinfowler.com/eaaCatalog/	51	2002
"Object Oriented Reengineering Patterns"	http://www.iam.unibe.ch/~scg/OORP/book.html	49	2002
A Generative Development-Process Pattern Language	http://users.rcn.com/jcoplien/Patterns/Process/index.html	48	1995
UML Pattern Language	http://www.ncc.up.pt/~zp/aulas/0607/es/geral/bibliografia/UML%20Pattern%20Language.pdf	46	2000
"Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems"	Addison Wesley Professional	44	2002
"AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis"	John Wiley & Sons	42	1998
WikiPatterns	http://www.wikipatterns.com/	42	2007
"Patterns for Effective Use Cases"	Addison Wesley Professional	32	2002
"Enterprise Solution Patterns Using Microsoft .NET Version 2.0: Patterns & Practices"	Microsoft Press	32	2004
"Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware"	John Wiley & Sons	32	2004
XML Design Patterns	http://www.xmlpatterns.com/	28	2000
Hypermedia Design Patterns Repository	http://www.designpattern.lu.unisi.ch/index.htm	28	1997
Embedded Design Patterns	http://www.eventhelix.com/RealtimeMantra/Patterns/	28	2004
"Small Memory Software: Patterns for Systems with Limited Memory"	http://hillside.net/patterns/books/Details/056.htm	27	2001
A Pattern Language for Pattern Writing	http://hillside.net/patterns/writing/patternwritingpaper.htm	26	1997
Experiences -- A Pattern Language for User Interface Design	http://www.maplefish.com/todd/papers/Experiences.html	26	2003
Data Access Patterns: Database Interactions in Object-Oriented Applications"	http://helloworld.siteburg.com/content/databases/db2/0131401572_toc.html	25	2003
GoF Patterns	http://www.vico.org/pages/PatronsDisseny.html	23	1995
Caterpillar's Fate: A Pattern Language for the Transformation from Analysis to Design	http://c2.com/ppr/catsfate.html	21	1995
User Interface Design Patterns	http://www.cs.helsinki.fi/u/salaakso/patterns/index.html	21	2003
Workflow Patterns	http://www.workflowpatterns.com/patterns/index.php	21	2000
Patterns for System Testing	"Pattern Languages of Program Design 3"	20	1997
Web Design Patterns Library	http://harbinger.sims.berkeley.edu/ui_designpatterns/webpatterns2/webpatterns/home.php	20	2006
A Pattern Language for Writers' Workshops	users.rcn.com/jcoplien/Patterns/WritersWorkshop/	19	1999
"Patterns for Parallel Programming"		19	2004
"Microsoft Integration Patterns"	http://download.microsoft.com/download/a/c/f/acf079ca-670e-4942-8a53-e587a0959d75/IntPatt.pdf	18	2004
Patterns Systems for Hypermedia	http://www-di.inf.puc-rio.br/schwabe//papers/PloP97.pdf	18	1997
POSA 1 Patterns	http://www.vico.org/pages/PatronsDisseny.html	17	1996
POSA 2 Patterns	"Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects "	17	2000
RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development	http://www2.umassd.edu/SWPI/ATT/pattern/rapel.html	17	1995
Understanding and Using the ValueModel Framework in VisualWorks Smalltalk	http://c2.com/ppr/vmodels.html	17	1994
An Input and Output Pattern Language: Lessons from Telecommunications	http://hillside.net/plop/plop98/final_submissions/P31.pdf	17	1999
New Clients with Old Servers: A Pattern Language for Client/Server Frameworks	http://citeseer.ist.psu.edu/156837.html	16	1995
Lazy Optimization: Patterns for Efficient Smalltalk Programming	"Pattern Languages of Program Design 2"	16	1996
EPISODES: A Pattern Language of Competitive	http://c2.com/ppr/episodes.html	16	1996

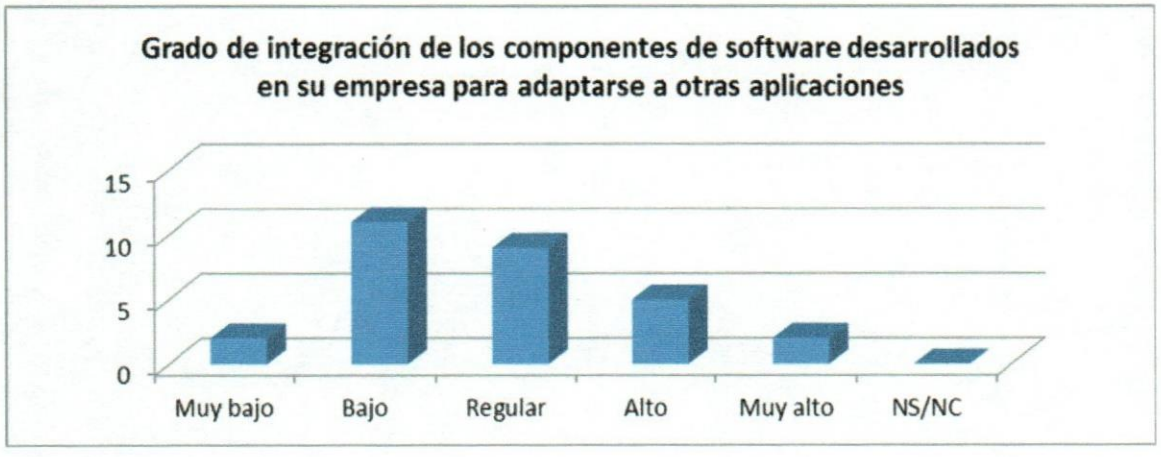
Title	Source	# of Patt.	Year
"Data Model Patterns: Conventions of Thought"	http://www.tdan.com/i005fe03.htm	15	1995
"Core J2EE Patterns: Best Practices and Design Strategies"	http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html	15	2003
Prioritizing Forces in Software Design	"Pattern Languages of Program Design 2"	13	1996
C++ Idioms	www.laputan.org/pub/sag/coplien-idioms.pdf	13	1999
Capable, Productive, and Satisfied: Some Organizational Patterns for Protecting Productive People	http://hillside.net/plop/plop98/final_submissions/P54.pdf	11	1999
SCRUM: A Pattern Language for Hyperproductive Software Development	http://citeseer.ist.psu.edu/397129.html	11	1999
"Use Cases: Patterns and Blueprints"	http://www.awprofessional.com/articles/article.asp?p=353171&seqNum=2&rl=1	11	2004
POSA 3 Patterns	"Pattern-Oriented Software Architecture: Patterns for Resource Management"	10	2004
G++: A Pattern Language for Computer-Integrated Manufacturing	http://citeseer.ist.psu.edu/134161.html	10	1995
The CHECKS Pattern Language for Information Integrity	http://c2.com/ppr/checks.html	10	1994
Selecting Locking Designs for Parallel Programming	http://citeseer.ist.psu.edu/493802.html	10	1996
A Pattern Language for Improving the Capacity of Reactive Systems	"Pattern Languages of Program Design 2"	10	1996
Customer Interaction Patterns	http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/P11/P11.htm	10	1999
"Java Testing Patterns"		10	2004
Patterns of Cooperative Interaction	http://www.comp.lancs.ac.uk/computing/research/cseg/projects/pointer/patterns.html	10	2001
Process Patterns	"Process Patterns"	10	1998
A Generative Pattern Language for Distributed Processing	"Pattern Languages of Program Design 1"	9	1995
Patterns for Evolving Frameworks	http://st-www.cs.uiuc.edu/~droberts/evolve.html	9	1997
Tropyc: A Pattern Language for Cryptographic Object-Oriented Software	http://citeseer.ist.psu.edu/62190.html	9	1999
Finite State Machine Patterns	"Pattern Languages of Program Design 4"	9	1999
"Analysis Patterns 2"	http://www.martinfowler.com/ap2/index.html	9	
Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks	http://st-www.cs.uiuc.edu/users/droberts/evolve.html	9	1996
Patterns for Software Architectures	http://citeseer.ist.psu.edu/shaw96some.html	8	1995
MOODS: Models for Object-Oriented Design of State	http://www.soberit.hut.fi/tik-76.278/alex/plop95.htm	8	1996
Crossing Chasms: A Pattern Language for Object-RDBMS	"Pattern Languages of Program Design 2"	8	1996
Transactions and Accounts	http://c2.com/cgi-bin/wiki?TransactionsAndAccounts	8	1996
Some Patterns for Software Architecture	http://www.cs.cmu.edu/afs/cs.cmu.edu/project/vit/ftp/pdf/PLoP95.pdf	8	1996
Fault-Tolerant Telecommunications System Patterns	http://users.rcn.com/jcoplien/Patterns/PLoP95_telecom.html	8	1996
Accessing Relational Databases	http://citeseer.ist.psu.edu/90550.html	8	1997
High-Level and Process Patterns from the Memory Preservation Society: Patterns for Managing Limited Memory	http://jerry.cs.uiuc.edu/plop/plopd4-submissions/P54.doc	8	1999
A Collection of History Patterns	hillside.net/plop/plop98/final_submissions/P63.pdf	8	1999
Display Maintenance: A Pattern Language	hillside.net/plop/plop98/final_submissions/P15.pdf	8	1999
More Process Patterns	"More Process Patterns"	8	1999
A Pattern Language for Tool Construction and Integration Based on the Tools and Materials Metaphor	http://www.riehle.org/computer-science/research/1994/plop-1994-tools.pdf	7	1995
Stars: A Pattern Language for Query-Optimized Schemas	http://c2.com/ppr/stars.html	7	1994
Reusability Through Self-Encapsulation	"Pattern Languages of Program Design 1"	7	1995
Partitioning Smalltalk Code into ENVY/Developer Components	http://c2.com/ppr/envy/	7	1996
State Patterns	http://citeseer.ist.psu.edu/396622.html	7	1997
The Selfish Class	http://www.joeyoder.com/papers/patterns/Selfish/selfish.html	7	1997
Architectural Patterns for Enabling Application Security	st-www.cs.uiuc.edu/~hanmer/PLoP-97/Proceedings/yoder.pdf	7	1999
Big Ball of Mud	http://www.laputan.org/mud/	7	1999
The Diemen Repository of Interaction Design Patterns	http://www.visiblearea.com/cgi-bin/twiki/view/Patterns/Patterns_repository	7	2003
Implementation Patterns for the Observer Pattern	"Pattern Languages of Program Design 2"	6	1996
Accountability and Organizational Structures	"Pattern Languages of Program Design 2"	6	1996
Smalltalk Scaffolding Patterns	"Pattern Languages of Program Design 4"	6	1999
Parallel Patterns for Synchronization on Shared-Memory Multiprocessors	http://c2.com/ppr/mutex/mutexpat.html	6	1995
Lifecycle and Refactoring Patterns That Support Evolution and Reuse	http://www.laputan.org/Lifecycle.html	5	1995
Discovering Patterns in Existing Applications	"Pattern Languages of Program Design 1"	5	1995

ANEXO II

Resultados del Cuestionario para determinar la necesidad de aplicar conocimientos en la reutilización de software, uso de patrones, repositorios e intercambio de información entre áreas de desarrollo de software.

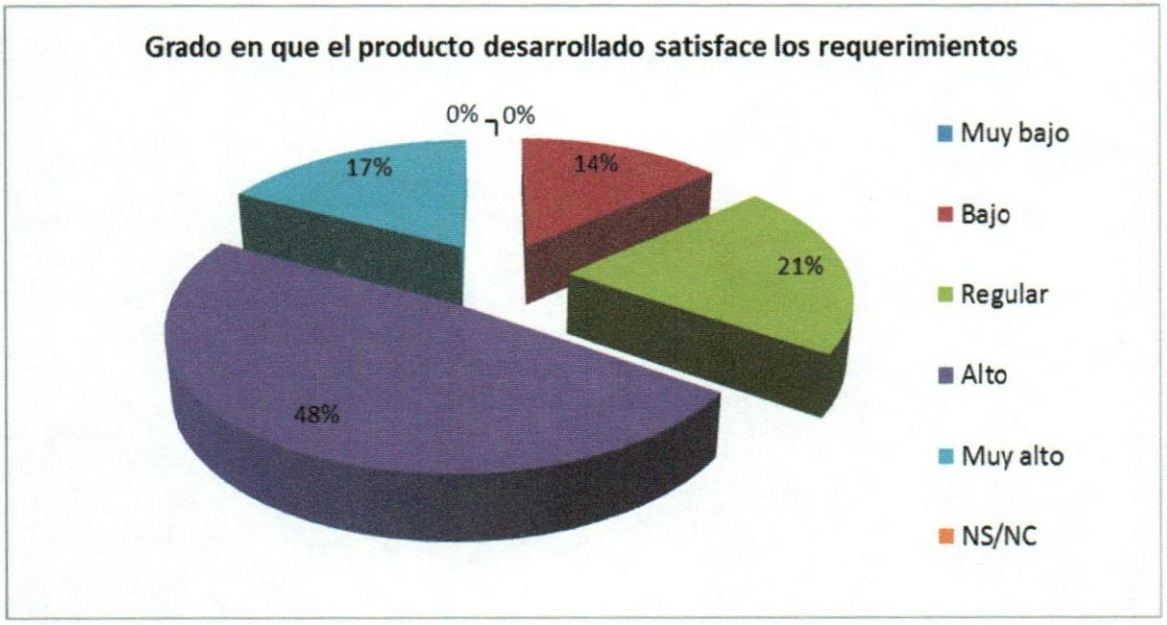
Flexibilidad

Grado de integración de los componentes de software desarrollados en su empresa para adaptarse a otras aplicaciones o procesos	Respuestas
Muy bajo	2
Bajo	11
Regular	9
Alto	5
Muy alto	2
NS/NC	0



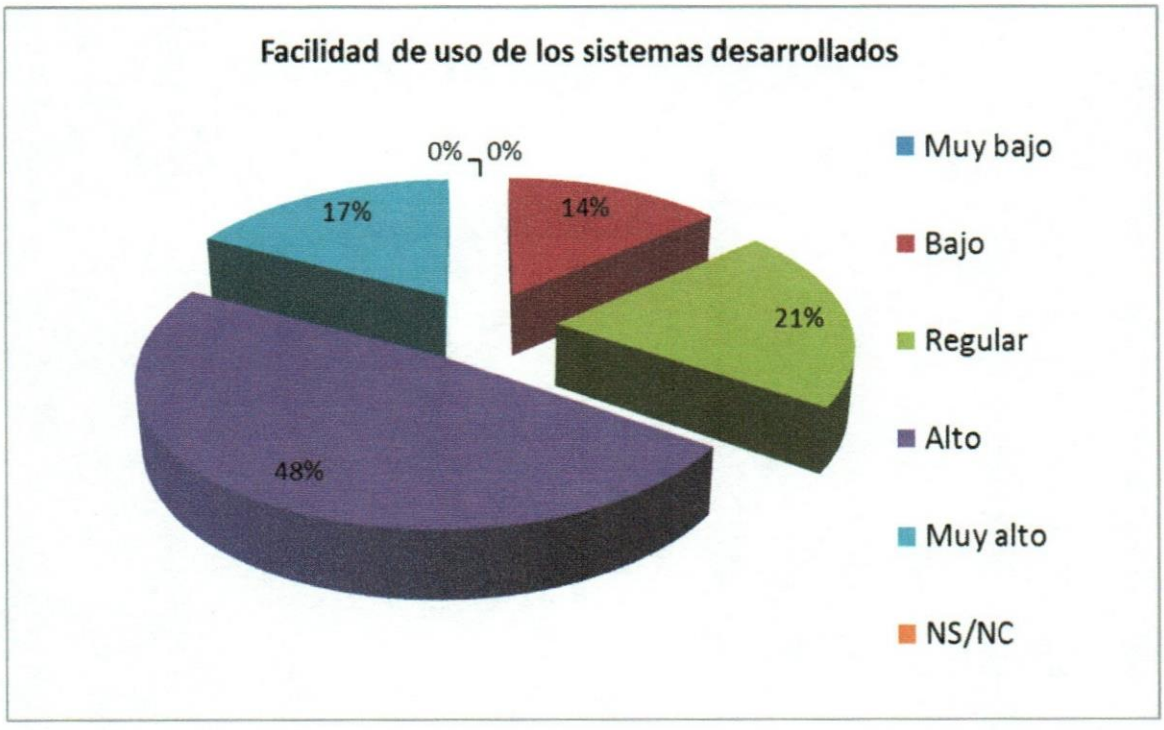
Calidad en el producto final

Según su critero, indique el grado en que el producto desarrollado en la empresa satisface las especificaciones indicadas en función de los requerimientos de los clientes	Respuestas
Muy bajo	0
Bajo	4
Regular	6
Alto	14
Muy alto	5
NS/NC	0



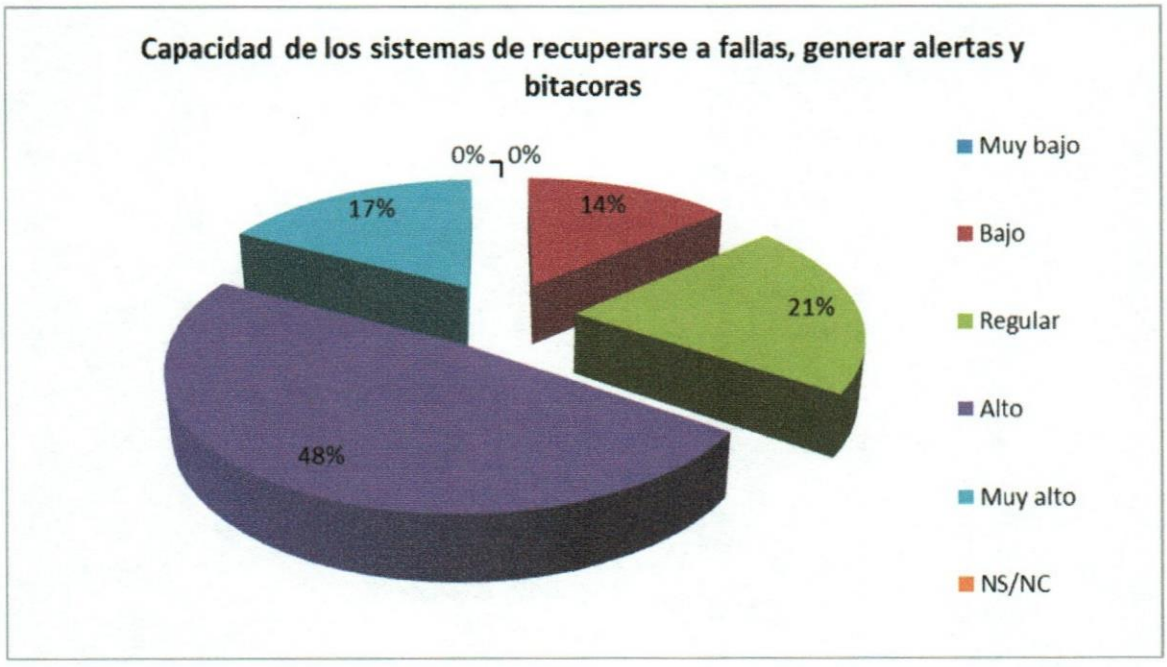
Facilidad de uso

Facilidad de uso de los sistemas desarrollados en su organización en el último año?	Total de respuestas
Muy bajo	1
Bajo	2
Regular	6
Alto	14
Muy Alto	6
NS/NC	0



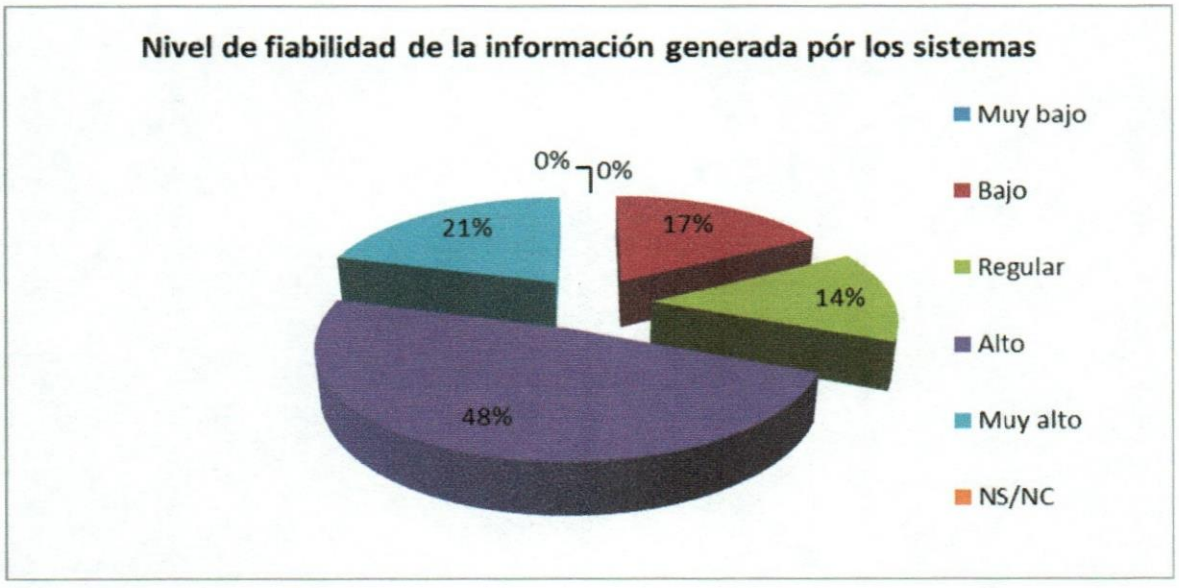
Recuperación

Capacidad de los sistemas de recuperarse a las fallas presentadas y generar alertas, bitácoras de fallas, etc	Total de respuestas
Muy bajo	1
Bajo	6
Regular	8
Alto	10
Muy alto	4
NS/NC	0



Exactitud

Nivel de fiabilidad de la información generada por los sistemas y procesos desarrollados en la empresa	Total de respuestas
Muy bajo	0
Bajo	5
Regular	4
Alto	14
Muy alto	6
NS/NC	0

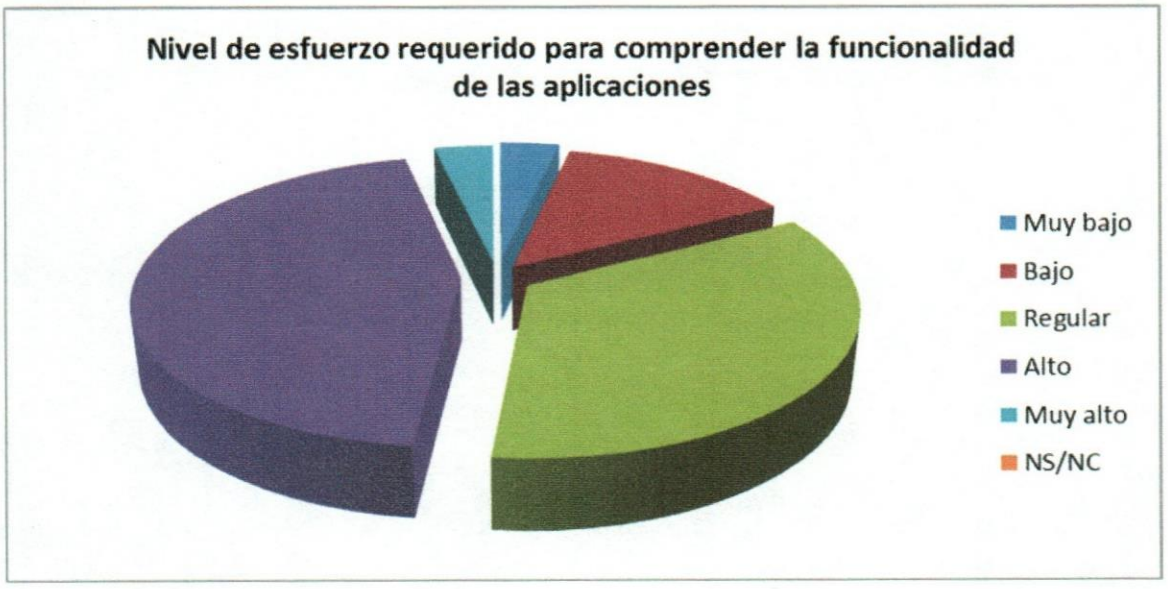


Flexibilidad

Indique el nivel de adaptación de los sistemas desarrollados para integrarse con otras aplicaciones, interfaces, servicios, etc.	Total de respuestas
Muy bajo	2
Bajo	7
Regular	9
Alto	8
Muy alto	3
NS/NC	0

Mantenibilidad

Indique el nivel de esfuerzo requerido para comprender la funcionalidad de una aplicación desarrollada, hacerle cambios, realizar pruebas y aplicar los cambios en producción	Total de respuestas
Muy bajo	1
Bajo	4
Regular	10
Alto	13
Muy alto	1
NS/NC	0

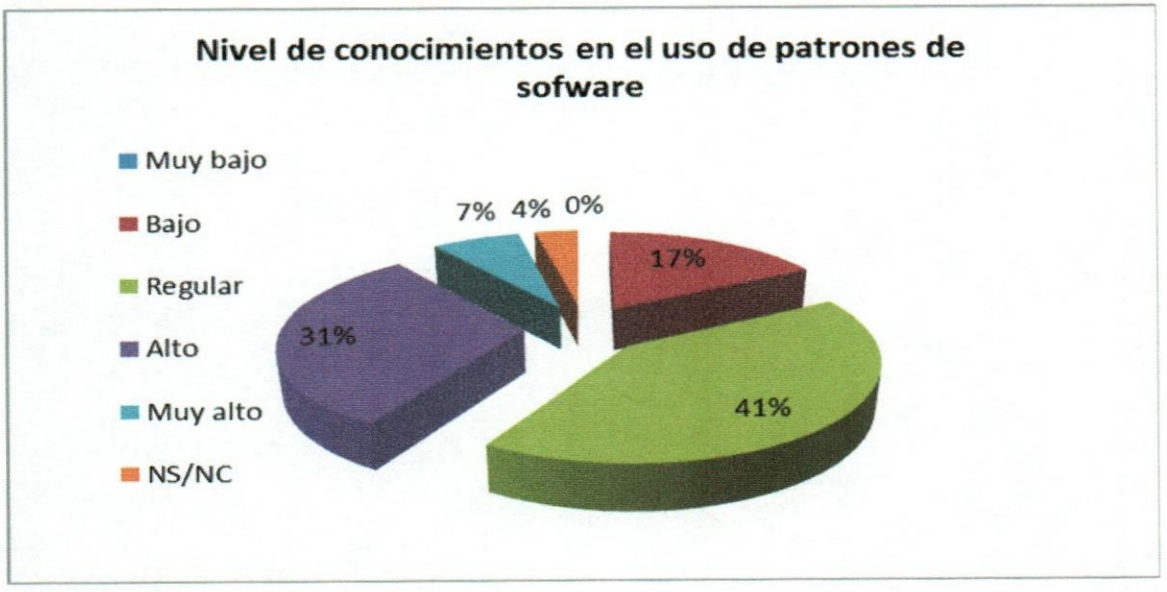


Reutilización de componentes

Según su experiencia, ¿Ha requerido aplicar sus conocimientos en el uso de patrones de desarrollo de software?	Total de respuestas
Siempre	2
Casi siempre	5
Regularmetne	13
Pocas veces	7
Nunca	1
NS/NC	1

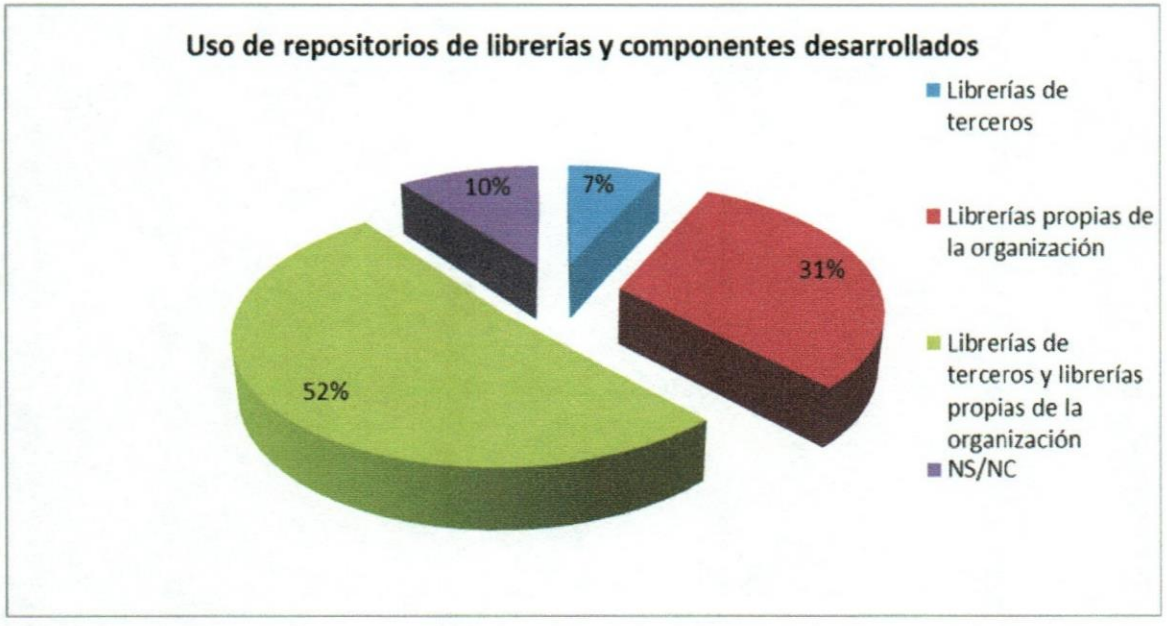
Reutilización de componentes

Su nivel de conocimiento en el uso e integración de patrones de software	Total de respuestas
Muy bajo	0
Bajo	5
Regular	12
Alto	9
Muy alto	2
NS/NC	1



Reutilización de componentes

¿Su organización dispone de repositorios de librerías y componentes desarrollados para la construcción de sistemas?	Total de respuestas
Librerías de terceros	2
Librerías propias de la organización	9
Librerías de terceros y librerías propias de la organización	15
NS/NC	3



Reutilización de componentes

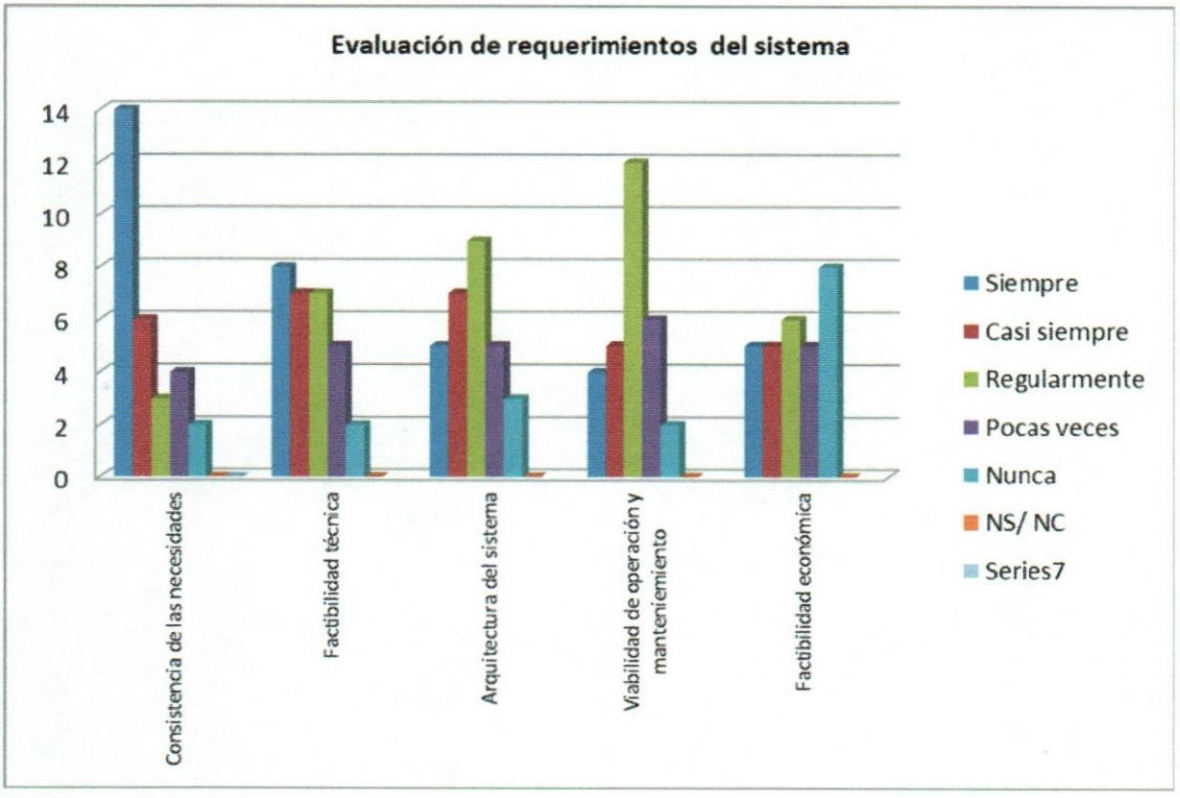
Durante el proceso de desarrollo, ¿logra integrar distintos componentes o patrones de software diseñados por la organización para solventar situaciones comunes?	Total de respuestas
Siempre	0
Casi siempre	3
Regularmente	9
Pocas veces	14
Nunca	2
NS/NC	1

Reutilización de componentes

¿Tiene usted conocimiento de las distintas herramientas y librerías desarrolladas por otros miembros del grupo de trabajo?	Total de respuestas
Siempre	1
Casi siempre	5
Regularmente	5
Pocas veces	12
Nunca	6
NS/NC	0

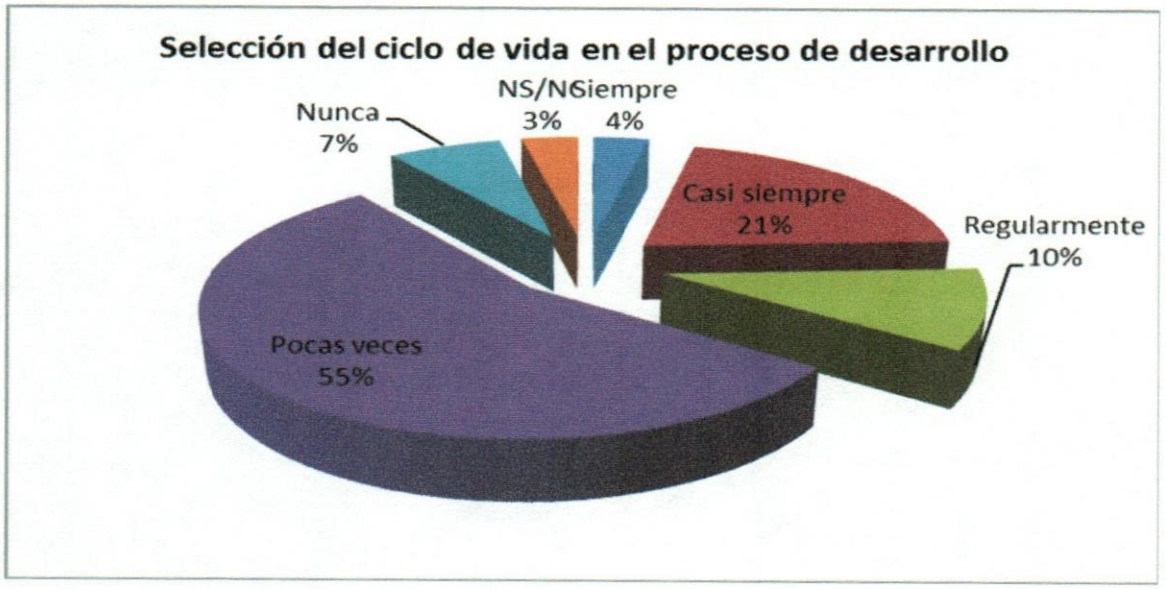
Proceso de desarrollo

En su organización se evalúan los requerimientos del sistema teniendo en cuenta los criterios enumerados a continuación:	Siempre	Casi siempre	Regularmente	Pocas veces	Nunca	NS/ NC
Consistencia de las necesidades	14	6	3	4	2	0
Factibilidad técnica	8	7	7	5	2	0
Arquitectura del sistema	5	7	9	5	3	0
Viabilidad de operación y manteniemiento	4	5	12	6	2	0
Factibilidad económica	5	5	6	5	8	0



Para el desarrollo de software se establece o selecciona un modelo de ciclo de vida adecuado a las características del proyecto	Total de respuestas
Siempre	1
Casi siempre	6
Regularmente	3
Pocas veces	16
Nunca	2
NS/NC	1

Proceso de desarrollo de software



Proceso de desarrollo de software

¿En su organización se encuentran claramente identificados las fases del proceso de desarrollo de sistemas a través de alguna metodología de desarrollo?	Total de respuestas
SI	15
NO	11
NS/NC	3

ANEXO III

Resultados obtenidos de la aplicación de la encuesta de evaluación de la ontología según los criterios basados en juicios de expertos.

Tabla 17. Tabla de resultados obtenidos de la aplicación de la encuesta de evaluación de la ontología propuesta

Principios	1	2	3	4	5	6	7
	3	4	4	5	4	4	5
	4	4	4	3	5	3	3
	4	3	4	4	3	5	4
Sumatoria	11	11	12	12	12	12	12
Promedio	3,67	3,67	4,00	4,00	4,00	4,00	4,00

Principio
Sobre las características relacionadas con la Ergonomía cognitiva.
Sobre las características relacionadas con la Transparencia
Sobre las características relacionadas con la Integridad computacional y eficiencia
Sobre las características relacionadas con la Integridad(Meta-level integrity).
Sobre las características relacionadas con la Flexibilidad
Sobre las características relacionadas con cumplimiento del proceso de experticia general
Sobre las características relacionadas a la adaptabilidad al negocio

ANEXO IV

Tabla 18. Tabla de valores obtenidos por OntoQA sobre la base de la ontología propuesta.

Clases	Instancias Directas	Instancias Indirectas	Subclases Directas	Rel.	Importancia de la Clase (Imp(Ci))	Conect.Con)	Prof.de las relaciones (RR)	Prof. en la herencia (IR)
Abstract_Factory	0,00	0,00	2,00	3,00	0,00	0,00	0,00	2,00
AbstractProduct	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Adapter	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Antipatron	0,00	1,00	11,00	2,00	11,11	0,00	0,00	11,00
Basicos	0,00	0,00	5,00	0,00	0,00	0,00	0,00	5,00
Blackboard	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Blob	1,00	0,00	0,00	0,00	11,11	1,00	0,00	0,00
Bridge	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Broker	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Builder	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Chain_of_responsibility	1,00	0,00	0,00	1,00	11,11	16,00	9,00	0,00
Channel_adapter	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Client-dispatcher-server	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Command	1,00	0,00	0,00	1,00	11,11	15,00	8,00	0,00
Command_message	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Command_processor	0,00	1,00	1,00	0,00	11,11	0,00	0,00	1,00
Comportamiento	0,00	1,00	11,00	0,00	11,11	0,00	0,00	11,00

Clases	Instancias Directas	Instancias Indirectas	Subclases Directas	Rel.	Importancia de la Clase (Imp(Ci))	Conect.Con)	Prof.de las relaciones (RR)	Prof. en la herencia (IR)
Composite	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
ConcreteFactory	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Construccion_mensajes	0,00	0,00	3,00	0,00	0,00	0,00	0,00	3,00
ContenedorDePatrones	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Copy_and_paste	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Creacional	0,00	0,00	5,00	0,00	0,00	0,00	0,00	3,50
Decorator	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Document_message	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Duplicate_Code	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Estructural	0,00	1,00	9,00	0,00	11,11	0,00	0,00	5,00
Event_message	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Facade	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Factory_Method	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Flyweight	1,00	0,00	0,00	2,00	11,11	11,00	3,50	0,00
Framework	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Functional_Decomposition	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
GodObject	1,00	0,00	0,00	0,00	11,11	7,00	0,00	0,00
Indecent_Exposure	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Interpreter	1,00	0,00	0,00	1,00	11,11	1,00	1,00	0,00
Iterator	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Large_Class	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Clases	Instancias Directas	Instancias Indirectas	Subclases Directas	Rel.	Importancia de la Clase (Imp(Ci))	Conect.Con)	Prof.de las relaciones (RR)	Prof. en la herencia (IR)
Layers	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Layers	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Lazy_Class	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Long_Method	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Mediator	0,00	0,00	1,00	2,00	0,00	0,00	0,00	1,00
Memento	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Message	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Message_bus	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Message_channel	0,00	0,00	4,00	0,00	0,00	0,00	0,00	4,00
Message_endpoint	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Message_router	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Message_translator	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Metapatron	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Microkernel	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
MVC	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Observer	0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Pac	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Patron	0,00	0,00	3,00	9,00	0,00	0,00	0,00	4,21
Patron_Analisis	0,00	0,00	4,00	0,00	0,00	0,00	0,00	4,71
Patron_Arquitectonico	0,00	0,00	7,00	0,00	0,00	0,00	0,00	4,00
Patron_Integracion	0,00	0,00	3,00	0,00	0,00	0,00	0,00	3,75

Clases		Instancias Directas	Instancias Indirectas	Subclases Directas	Rel.	Importancia de la Clase (Imp(Ci))	Conect.Con)	Prof.de las relaciones (RR)	Prof. en la herencia (IR)
Pipes_and_Filters		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Point_to_Point_channel		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Poltergeist		1,00	0,00	0,00	0,00	11,11	3,00	0,00	0,00
Protection_Proxy		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Prototipe		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Proxy		0,00	0,00	1,00	0,00	0,00	0,00	0,00	1,00
Publish_subscriber_channel		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Singleton		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Singleton_Overuse		1,00	0,00	0,00	0,00	11,11	3,00	0,00	0,00
Spaguetti		1,00	0,00	0,00	0,00	11,11	3,00	0,00	0,00
State		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Strategy		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
Template		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
The_Publisher-Subscriber		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
View_Handler		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Visitor		0,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00
TOTALES		9,00	4,00	70,00	45,00	144,43	60,00	21,50	64,17
Promedios	(*)	17,333333	(*)	93,33333	60	192,5733333	80	28,66666667	85,56

(*) Se suman ambas instancias en un solo promedio

ANEXO V

Proceso de generación de métricas de la Ontología creada usando OntoQA.

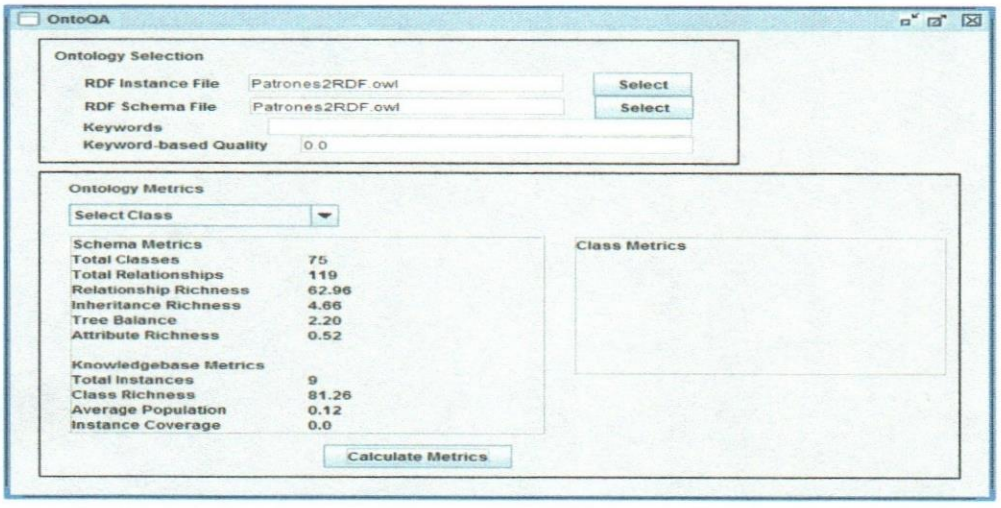


Figura 35. Generación de métricas usando OntoQA para la ontología propuesta

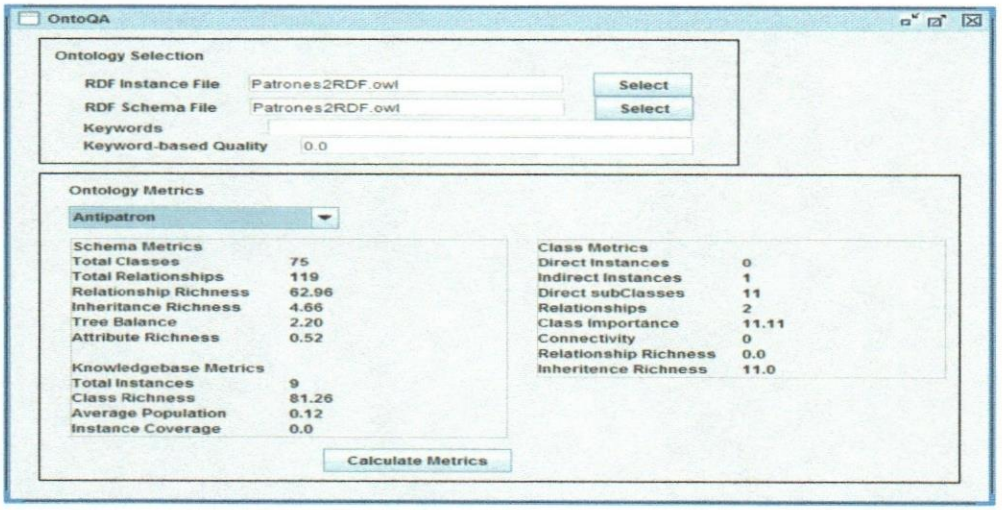


Figura 36. Métricas de la clase Antipatrón usando OntoQA

ANEXO VI

Documento de especificación de requerimientos

Documento de especificación de requerimientos
Dominio
Fecha
Desarrollador(es)
Propósito
Alcance
Fuentes de conocimiento

ANEXO VII

CONSTANCIAS DE VALIDACIÓN

APROBACIÓN DE LA EMPRESA

Caracas, a los 02 días del mes de abril de 2014

Sres.
UNIVERSIDAD CATÓLICA ANDRÉS BELLO
Postgrado en Sistemas de Información
Caracas

Nos dirigimos a ustedes para informarles que hemos autorizado al Lic. Adriano Da Silva Abreu C.I.: V-11.677.444, quién labora en esta organización, a hacer uso de la información proveniente de esta institución, para documentar y soportar los elementos de los distintos análisis estrictamente académicos que conllevarán a la realización del Trabajo Especial de Grado titulado "Modelo Ontológico Para Determinar El Uso De Patrones De Software Orientado A Departamentos De Desarrollo De Sistemas En Empresas Del Sector Bancario" como requisito para optar al título de Especialista/Magíster en Sistemas de Información, exigidos por la Dirección General de los Estudios de Postgrado de la Universidad Católica Andrés Bello.

Sin más a que hacer referencia, atentamente,

Nombre: Isabel Sira
Cargo: Vp. Sistemas

CONSTANCIA DE VALIDACIÓN

Yo, Pedro Nolasco Bonillo Ramos, titular de la cédula de Identidad N° V-10.868.537, de profesión Dr. en ciencias de la computación ejerciendo actualmente como Director, en la institución PHO Consultores.

Por medio de la presente hago constar que he revisado y validado a los fines de su aplicación los instrumentos (cuestionarios) para le trabajo de grado que lleva por título: ***Modelo Ontológico Para Determinar El Uso De Patrones De Software Orientado A Departamentos De Desarrollo De Sistemas En Empresas Del Sector Bancario.***

En Caracas, a los 2 *días del mes de* Abril *de* 2014

Bonillo Ramos

CONSTANCIA DE VALIDACIÓN

Yo, ALVARO JESUS MENDEZ RODRIGUEZ, titular de la cédula de Identidad N° 9947659, de profesión ING. EN COMPUTACIÓN ejerciendo actualmente como COORDINADOR, en la institución BANCO VENEZOLANO DE CRÉDITO

Por medio de la presente hago constar que he revisado y validado a los fines de su aplicación los instrumentos (cuestionarios) para le trabajo de grado que lleva por título: ***Modelo Ontológico Para Determinar El Uso De Patrones De Software Orientado A Departamentos De Desarrollo De Sistemas En Empresas Del Sector Bancario.***

En Caracas, a los 15 *días del mes de* JULIO *de* 2014

Alvaro Mendez

Firma