CONSTRUCCIÓN DE UN MECANISMO DE CONSULTA PARA ANÁLISIS DE INFORMACIÓN BASE OBTENIDA EN PROCESO DE INGENIERÍA INVERSA

INVESTIGADORES

Cindy Margarita Pacheco Álvarez

Alejandra Inés Ríos Rodríguez



UNIVERSIDAD DE CARTAGENA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS, 2013

CONSTRUCCIÓN DE UN MECANISMO DE CONSULTA PARA ANÁLISIS DE INFORMACIÓN BASE OBTENIDA EN PROCESO DE INGENIERÍA INVERSA

GIMATICA

Ingeniería De Software

INVESTIGADORES

Cindy Margarita Pacheco Álvarez

Alejandra Inés Ríos Rodríguez

Director: Martín Monroy Ríos, Msc. (Universidad de Cartagena)



UNIVERSIDAD DE CARTAGENA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS, 2013

Contenido

[INTRODUCCIÓN 9](#_Toc355045232)

[1. PLANTEAMIENTO DEL PROBLEMA 9](#_Toc355045233)

[1.1 DESCRIPCIÓN DEL PROBLEMA 9](#_Toc355045234)

[1.2 FORMULACIÓN DEL PROBLEMA 12](#_Toc355045235)

[2. JUSTIFICACIÓN 12](#_Toc355045236)

[3. MARCO TEÓRICO 15](#_Toc355045237)

[3.1 INGENIERÍA DE SOFTWARE 15](#_Toc355045238)

[3.2 INGENIERÍA INVERSA 16](#_Toc355045239)

[3.3 UML 18](#_Toc355045240)

[3.4 XMI 20](#_Toc355045245)

[3.5 XQuery 22](#_Toc355045246)

[4. ESTADO DEL ARTE 23](#_Toc355045247)

[5.2 OBJETIVOS ESPECÍFICOS 26](#_Toc355045248)

[6. METODOLOGÍA 26](#_Toc355045249)

[7. CRONOGRAMA 30](#_Toc355045250)

[8. RESULTADO ESPERADOS Y POTENCIALES BENEFICIOS 31](#_Toc355045251)

[11. BIBLIOGRAFÍA 37](#_Toc355045252)

**Índice de figuras**

[Figura 1. Diagrama de GANTT de las actividades 30](#_Toc355061218)

**Índice de tablas**

[Tabla 1. Cronograma de actividades 29](#_Toc355061166)

[Tabla 2. Generación de nuevo conocimiento. 31](#_Toc355061167)

[Tabla 3. Fortalecimiento de la comunidad científica 31](#_Toc355061168)

[Tabla 4. Apropiación social del conocimiento 32](#_Toc355061169)

[Tabla 5. Impactos esperados 33](#_Toc355061170)

[Tabla 6. Presupuesto global de la propuesta por fuentes de financiación (en miles de $). 34](#_Toc355061171)

[Tabla 7. Descripción de los gastos de personal (en miles de $). 34](#_Toc355061172)

[Tabla 8. Descripción y cuantificación de los equipos de uso propio (en miles de $). 34](#_Toc355061173)

[Tabla 9. Descripción detallada de Materiales y suministros (en miles de $). 35](#_Toc355061174)

[Tabla 10. Descripción detallada de la Bibliografía (en miles de $). 35](#_Toc355061175)

# INTRODUCCIÓN

La ingeniería inversa se define como “el proceso de análisis de un sistema para identificar sus componentes y las relaciones entre ellos, y para crear una representación del sistema en otra forma o en otro nivel de abstracción” (Chikovsfky & Cross, 1990). La ingeniería inversa, a pesar de ser una actividad antigua, es considerada una reciente disciplina, la cual se enfrenta a muchos retos (Monroy, Arciniegas , & Rodríguez, 2012). Uno de estos retos es mejorar la capacidad de consulta de la información base producto de un proceso de ingeniería inversa.

Todo proceso de ingeniería inversa está formado por tres aspectos fundamentales: un analizador, una información base y un visualizador. El analizador permite extraer la arquitectura del software; la información base es el resultado que se obtiene del analizador el cual puede ser compactado en un documento XMI y el visualizador es por medio del cual el resultado es representado al usuario para que se comprenda de mejor manera lo obtenido mediante diagramas, métricas, gráficas o reportes (Canfora & Di Penta, New Frontiers of Reverse Engineering, 2007).

Este proceso de ingeniería inversa le muestra al usuario toda la arquitectura que resulta de un software, lo cual facilita la recuperación de conocimiento de sistemas implementados con poca o nula documentación, disminuyendo la complejidad de las tareas de mantenimiento de software y con esto garantizar la evolución de productos de calidad. Pero, este proceso es limitado al momento de representar sistemas complejos que tienen miles de objetos como clases, relaciones, interfaces, entre otros (Canfora & Di Penta, 2007). Cuando se trata de sistemas con arquitecturas complejas, no se puede visualizar la arquitectura en forma completa en la pantalla de un computador, y al querer representar partes específicas de interés particular, el ingeniero de software debe desplazarse por toda la representación de la arquitectura en la pantalla, hasta encontrar la parte que le interesa, haciendo este trabajo dispendioso, ineficiente y inexacto, porque depende de las destrezas de la persona que está utilizando la herramienta.

Debido a lo anterior, surge la necesidad de un mecanismo de consulta a partir de la información obtenida en el proceso de ingeniería inversa, para facilitar la capacidad de análisis de los ingenieros de software en su búsqueda de la evolución y mantenimiento del producto

Se estima que el alcance del proyecto será aplicable en la comunidad de desarrollo de software y en la comunidad académica. En la primera será de mucha utilidad para mantener el control de calidad del software, facilitando la extracción de información de sistemas implementados que cuenten con documentación escasa o nula, reduciendo la complejidad de las tareas de mantenimiento y reutilización (Swebok, 2012), mientras que en el ámbito académico, apoyará a los procesos de aprendizaje de la comunidad del software y de desarrollo sobre la ingeniería inversa y la ingeniería del software, facilitando las tareas de mantenimiento de los productos realizados, lo cual garantiza la evolución de productos de calidad. Los resultados del trabajo de investigación serán directamente aplicados en el programa de Ingeniería de Sistemas de la Universidad de Cartagena, los cuales servirán como base para impartir conocimientos sobre los procesos de la ingeniería inversa y sus beneficios, y servirán como punto de partida para futuros trabajos de investigación.

La presente investigación se limita solo a uno de los tres aspectos fundamentales del proceso de ingeniería inversa: La visualización, debido a que asumir otros aspectos del proceso, se haría demasiado extenso.

Este proyecto será un avance importante en la ingeniería inversa, debido a que se dará respuesta a uno de los retos por los cuales esta atraviesa. Además, la construcción de un mecanismo de consulta, facilitaría la capacidad de análisis de los usuarios, y a vez facilitaría las tareas del mantenimiento del software y el control de evolución del mismo con el fin de tener productos de calidad.

# 

# 1. PLANTEAMIENTO DEL PROBLEMA

# 1.1 DESCRIPCIÓN DEL PROBLEMA

Un software es un programa de computador, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema informático (IEEE, 1990). La calidad se ha convertido en un elemento de vital importancia, de los procesos de desarrollo y de los productos software, la cual se garantiza por medio de la creación y aplicación de metodologías y modelos[[1]](#footnote-1). Como resultado de la correcta aplicación de estas metodologías y modelos se obtienen productos de calidad que buscan satisfacer las necesidades de los usuarios, pero estas necesidades son cambiantes y los sistemas informáticos deben estar preparados para esos cambios como lo son: la aplicación de nuevas tecnologías, crecimiento en número de usuarios, cambios de requerimientos, etcétera (Arciniegas Herrera, 2006). Para garantizar estos cambios lo que se busca es la evolución y actualización de los sistemas informáticos (Tonella, Torchiano, Du Bois, & Systä, 2007), que permitan asegurar la calidad de los mismos a través del tiempo. Esto se logra aplicando diferentes técnicas de la ingeniería de software.

La ingeniería de software es “la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, con el propósito de garantizar un software de calidad” (IEEE, 1990). El mantenimiento de software es una de las actividades más comunes en la [ingeniería de software](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) que permite llevar control sobre la evolución de un producto software, de tal manera que se pueda garantizar su calidad a través del tiempo (Monroy, Arciniegas , & Rodríguez, 2012). Existen diversas técnicas que permiten que se desarrolle el mantenimiento del software, como lo son: la re-ingeniería, la migración y la ingeniería inversa (Swebok, 2012), ésta última es la técnica por excelencia de mantenimiento de software (Canfora & Di Penta, 2007) (Canfora, Di Penta, & Cerulo, 2011) y se define como “el proceso de análisis de un sistema para identificar sus componentes y las relaciones entre ellos, y para crear una representación del sistema en otra forma o en otro nivel de abstracción” (Chikovsfky & Cross, 1990).

La ingeniería inversa permite la identificación de los componentes de un sistema y la comprensión de las relaciones existentes entre ellos, aspecto imprescindible de todo proceso de ingeniería del software porque garantiza la evolución del software. Además, ofrece métodos y herramientas relacionados con la comprensión y modificación de sistemas (Canfora & Di Penta, 2007). Una herramienta de ingeniería inversa está compuesta por tres aspectos importantes: un analizador, una información base y un visualizador (Panas, Löwe, & Aßmann, 2003). El analizador permite extraer la arquitectura del software; la información base es el resultado que se obtiene del analizador y el visualizador es por medio del cual el resultado es representado al usuario para que comprenda de mejor manera lo obtenido, por medio de diagramas, métricas, gráficas o reportes (Canfora & Di Penta, 2007).

Las herramientas de ingeniería inversa han permitido facilitar la recuperación de conocimiento de sistemas implementados con poca o nula documentación, permitiendo conocer su arquitectura y así, disminuir la complejidad de las tareas de mantenimiento (Monroy, Arciniegas , & Rodríguez, 2012), pero han generado dificultad al momento de analizar sistemas complejos que impliquen gran cantidad de información base generada por estas herramientas, la cual puede ser representada mediante diagramas, métricas o reportes. La representación de la información base es un aspecto indispensable en todo proceso de ingeniería inversa, pero al tratarse de sistemas complejos con gran cantidad de información generada, sería un proceso tedioso analizar toda esta información y por lo cual se hizo necesario contar con herramientas que permitieran realizar el análisis de una manera más específica y concreta, representado al usuario lo que de verdad le interesa visualizar, a partir de la realización de consultas a la información base (Canfora, Di Penta, & Cerulo, 2011).

Al notar ésta necesidad, investigadores como Canfora y Di Penta propusieron la creación de una herramienta de consulta que permitiera abstraer información a partir de la información base resultado del proceso de ingeniería inversa (Canfora & Di Penta, 2007), para lo cual se tenían soluciones que fueron reconocidas como GReQL, PROGRES, Grok y ESCAPE (Holt, Wu, & Winter, 2002), pero presentaban dificultades que hacían necesario mejorar la capacidad de consulta que permitiera facilitar la capacidad de análisis al usuario (Canfora, Di Penta, & Cerulo, 2011).

A pesar de los avances en el contexto de la ingeniería inversa enfocados al desarrollo de herramientas y la definición de técnicas, métodos y metodologías (Tonella, Torchiano, Du Bois, & Systä, 2007) (Monroy, Arciniegas , & Rodríguez, 2012), aun se siguen teniendo limitantes como mostrar el nivel de detalle adecuado a lo que el usuario necesita en particular y permitirle escoger entre artefactos que le representen un nivel más profundo o a un nivel más amplio de complejidad (Canfora & Di Penta, 2007), con el fin de facilitarle la capacidad de entendimiento y análisis al momento de enfrentarse con los modelos de la arquitectura que solicitó. Por tal razón, hace falta crear metodologías, por ejemplo, de cómo realizar y representar consultas a partir de la información base obtenida como resultado de un proceso de ingeniería inversa. Esta falencia, representa uno de los retos actuales de la ingeniería inversa (Canfora, Di Penta, & Cerulo, 2011).

En el ámbito local el Msc. Martín Monroy, profesor asociado de la Universidad de Cartagena y estudiante del programa de doctorado en Ingeniería Telemática de la Universidad del Cauca, como resultado de una exhaustiva revisión bibliográfica plantea en su tesis doctoral titulada: “Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento” la necesidad de contar con un mecanismo de consulta a partir de la información obtenida en un proceso de ingeniería inversa, que facilite los procesos de análisis, lo cual sustenta en su artículo “Framework for recovery and analysis of behavioral architectural views” al igual que Canfora, Di Penta y Cerulo, que se necesita mejorar la “Capacidad para consultar la información base obtenida en los artefactos generados por los analizadores.” (Monroy, Arciniegas , & Rodríguez, 2012) (Canfora, Di Penta, & Cerulo, 2011), lo cual constituye uno de los retos actuales de la ingeniería inversa.

En consecuencia nace el esfuerzo de buscar solución a esta necesidad, para lo cual se requiere la construcción de un mecanismo de consulta que permita presentar el nivel adecuado de detalle que solicita un usuario al momento de analizar la información que se obtiene como resultado de un proceso de ingeniería inversa, por medio de consultas que faciliten la selección de los elementos que el usuario considere pertinente de analizar. Todo esto con el fin de mejorar los procesos de ingeniería inversa, contribuir al control de calidad en el desarrollo de software y facilitar las tareas de mantenimiento que garantizan la evolución de un producto (Monroy, Arciniegas , & Rodríguez, 2012).

# 1.2 FORMULACIÓN DEL PROBLEMA

¿Cómo realizar consultas a partir de la información base obtenida en calidad de resultado de un proceso de ingeniería inversa?

# 2. JUSTIFICACIÓN

Las herramientas de ingeniería inversa han sido una solución importante para la ingeniería de software, porque han permitido facilitar la recuperación de conocimiento de sistemas implementados con poca o nula documentación, disminuyendo la complejidad de las tareas de mantenimiento de software. Pero, se quedan cortas al momento de representar sistemas complejos que poseen miles de objetos como clases, relaciones, interfaces, entre otros, sin tener en cuenta que representan un todo cuando existen usuarios que necesitan visualizar casos específicos (Canfora & Di Penta, 2007).

Todo software que pasa por una herramienta de ingeniería inversa se relaciona con tres aspectos importantes: un analizador, una información base y un visualizador (Panas, Löwe, & Aßmann, 2003). El analizador permite extraer la arquitectura del software; la información base es el resultado que se obtiene del analizador el cual puede ser compactado en un documento XMI y el visualizador es por medio del cual el resultado es representado al usuario para que se comprenda de mejor manera lo obtenido mediante diagramas, métricas, gráficas o reportes (Canfora & Di Penta, 2007).

Con este proceso se le muestra al usuario toda la arquitectura que resulta de un software, pero cuando se trata de problemas específicos y de arquitecturas complejas que no podrían visualizarse en una pantalla de computador, se hace necesario un mecanismo que permita el análisis de una manera más específica y concreta, que represente al usuario lo que de verdad quiere visualizar. La visualización de una arquitectura compleja con miles de clases y miles de relaciones, representa un gran reto al momento de analizarla, por lo que si la herramienta de ingeniería inversa no tiene mecanismos de consulta, en vez de representar una ayuda para el mantenimiento, se convierte en parte del problema.

Para permitir obtener la información que verdaderamente necesita el usuario al momento de examinar una arquitectura, ya sea simple o compleja, se propone un mecanismo de consulta que brindará beneficios tales como extraer de la arquitectura los aspectos específicos que solicite el usuario, logrando mayor simplicidad en el resultado y facilitando la capacidad de análisis del usuario (Canfora & Di Penta, 2007) (Canfora, Di Penta, & Cerulo, 2011). Esta última es de vital importancia ya que su objetivo es extraer conclusiones y previsiones a futuro del producto analizado, lo cual permitirá detectar problemas y con esto realizar nuevos productos mejorados, además de facilitar el manteamiento de los mismos.

Al mejorar la capacidad de análisis de los usuarios, se facilitarían las tareas de mantenimiento del software, lo cual contribuye al desarrollo de dos campos de acción de la ingeniería inversa: la producción de software y el ámbito académico. La producción de software se beneficiará porque se facilitaría la actualización continua de las vistas de alto nivel del sistema, lo que a su vez reduce la probabilidad de error (Canfora, Di Penta, & Cerulo, 2011), al mismo tiempo que será de mucha utilidad para mantener el control de calidad del software, facilitando la verificación de la coherencia del código implementado con los modelos propuestos por el arquitecto de software. Además, la codificación se facilitaría permitiendo identificar porciones de código que pueden ser reutilizadas. En términos generales al mejorar la producción de software se mejoraría la calidad de los productos garantizando la evolución de estos a través del tiempo (Monroy, Arciniegas , & Rodríguez, 2012).

Por otra parte, en un futuro también se tendrían beneficios económicos para la comunidad de desarrollo de software y la comunidad de ingeniería del software, debido a que se mejoraría la capacidad de análisis de una arquitectura por parte de los usuarios, disminuyendo el tiempo empleado para realizar esta actividad y por consiguiente los costos de la misma al momento de realizar actualizaciones y mantenimiento al software.

En el ámbito académico, serian de gran ayuda las herramientas de ingeniería inversa que permiten mejorar la capacidad de consulta y análisis de los usuarios, porque estas pueden ser usadas para el aprendizaje de los conceptos que hacen parte tanto del proceso de ingeniería directa como del proceso de ingeniería inversa.

La ingeniería inversa, a pesar de ser una actividad antigua, es considerada una reciente disciplina, la cual se enfrenta a muchos retos (Monroy, Arciniegas , & Rodríguez, 2012). Este trabajo de investigación es conveniente porque contribuye a la solución de uno de estos retos: la necesidad de mejorar la capacidad de consulta de la información base obtenida a partir de los analizadores (Canfora, Di Penta, & Cerulo, 2011) (Monroy, Arciniegas , & Rodríguez, 2012). Al permitir que el usuario realice las consultas en su arquitectura se podría garantizar la calidad del software a través del tiempo, respondiendo a las crecientes exigencias de calidad de los usuarios. Todo esto se consigue a partir del mantenimiento del software consecuencia de los procesos de recuperación de información que deben estar en constante actualización con el fin de contribuir y garantizar la evolución del producto (Swebok, 2012).

El programa de ingeniería de sistemas de la Universidad de Cartagena se verá directamente beneficiado con el desarrollo de este trabajo, debido a que la realización de un mecanismo de consulta serviría como base para impartir conocimiento sobre los procesos de ingeniería inversa y servirá como punto de partida, no sólo para estudiantes de ingeniería de sistemas de la Universidad de Cartagena, sino para investigadores de ingeniería sistemas de la región, que estén interesados en seguir abordando la temática del presente trabajo en futuras investigaciones, encaminados al fortalecimiento y desarrollo de la ingeniería del software.

Además, este trabajo de investigación contribuye a la realización de una tesis doctoral titulada: “Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento”, del profesor asociado Martin Monroy de la Universidad de Cartagena, quien plantea la necesidad de un mecanismo de consulta de la información base resultado de un proceso de ingeniería inversa.

La realización de esta investigación es viable desde el punto de vista económico porque los investigadores asumen con los costos que genera. También es viable desde el punto de vista tecnológico porque se dispone de tecnologías como XMI, UML, XQuery y Java que serán utilizadas para su desarrollo. De igual manera es viable desde los propósitos y nivel académico de los autores, porque contribuye a su formación profesional en el área de ingeniería de sistemas, a partir de los fundamentos conceptuales construidos en asignaturas como algoritmos, programación básica, programación orientada a objetos, estructuras de datos, ingeniería de software, bases de datos, entre otras.

Este proyecto se encuentra inscrito en línea de investigación Ingeniería de software, que forma parte de las líneas de investigación del grupo E-Soluciones, porque su objeto de estudio se encuentra inmerso en el campo de la ingeniería inversa, que a su vez hace parte de la ingeniería de software.

# 3. MARCO TEÓRICO

# 3.1 INGENIERÍA DE SOFTWARE

La ingeniería de software es definida por la IEEE cómo “la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software de calidad” (IEEE, 1990) que posea la capacidad de resolver problemas de cualquier tipo. (Pressman, 2002).

Los temas tratados por la ingeniería de software son:

* Inspección de software crítico.
* Software de Tecnologías de Procesos de Negocios.
* Arquitecturas de Software Distribuido.
* UML
* Control técnico de proyectos software.
* Estrategias de ingeniería inversa para migración de software.
* Modelado y análisis de arquitectura de software.
* Herramientas CASE.
* Análisis y Diseño Orientados a Objetos. (Pressman, 2002)

Dentro de las actividades que realiza la ingeniería de software en su ciclo de vida se encuentra la de mantenimiento, cuyo objetivo es modificar el software mientras se mantiene su integridad en búsqueda de un proceso de evolución y mejora. Dentro de las técnicas de mantenimiento se pueden encontrar (Swebok, 2012):

* Re-Ingeniería
* Ingeniería Inversa
* Migración
* Retiro

Siendo la ingeniería inversa el método por excelencia destinado al mantenimiento (Canfora & Di Penta, 2007).

# 3.2 INGENIERÍA INVERSA

La ingeniería inversa es el proceso de análisis de un sistema para (Chikovsfky & Cross, 1990):

* Identificar los componente de un sistema y su interrelación
* Crear representaciones del sistema de otra forma o en otro nivel de abstracción

Todo proceso de ingeniería inversa cuenta con un analizador, una información base y un visualizador. Gracias al analizador se obtiene la arquitectura a partir de un proceso de ingeniería inversa, el resultado de ese análisis es guardado dentro de una información base, ésta puede ser almacenada en formato XMI y se representada de diferentes maneras por medio del visualizador, fundamental en todos los procesos de análisis. (Tonella, Torchiano, Du Bois, & Systä, 2007)

Las actividades de la ingeniería inversa están destinadas a resolver algunos problemas específicos relacionados con el producto informático y con la consistencia entre los artefactos de software como por ejemplo el código fuente, ejecutables y documentación. El ingeniero de software es el encargado de realizar éstas actividades, las cuales les brindan beneficios cómo vistas y representaciones a favor de las actividades de mantenimiento y evolución (Canfora, Di Penta, & Cerulo, 2011).

Son múltiples los trabajos que se han hecho en torno a la ingeniería inversa, desde el desarrollo herramientas hasta la definición de métodos, técnicas y metodologías (Tonella, Torchiano, Du Bois, & Systä, 2007). Las herramientas de ingeniería inversa se clasifican en desambladores, analizadores, descompiladores y depuradores. Entre los analizadores tenemos de base de datos y código fuente, los de este último, son considerados como Herramientas CASE o Especializadas en Recuperación de Diseño.

Las herramientas CASE son “Herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento)” (Terry & Logee, 1990). En este tipo de herramientas, comercialmente se pueden resaltar productos, como Rational Rose, Enterprise Architecture, Visual Paradigm, Together, entre otras, las cuales brindan funcionalidades de ingeniería inversa (Monroy, Arciniegas , & Rodríguez, 2012). Además, existen propuestas menos comerciales y más que todo utilizadas en el ámbito académico, como Rigi (Storey & Wong, 1996), Bauhaus (Raza, Vogel, & Plodereder, 2006), CPP2XMI (Korshunova & Petkovic, 2006), XDRE (Li, Hu, Chu, & Chen, 2005), Jar2UML (Vrije Universiteit Brussel), SrcML (SDML, Software Development Laboratory), entre otras, las cuales son especializadas en recuperar diseño.

En términos generales, los trabajos de ingeniería inversa que se han realizado se han centrado en su mayoría en los aspectos estructurales del sistema, quedando muchos retos por responder, como la capacidad de consulta de la información base producto de un proceso de ingeniería inversa (Canfora, Di Penta, & Cerulo, 2011).

Canfora y Di Penta presentan brevemente una revisión general del campo de la ingeniería inversa, las principales revisiones, los logros que se han alcanzado y las áreas de aplicación, y resaltan los temas que se encuentran abiertos a futuras investigaciones (Canfora & Di Penta, 2007). Además, en su documento más reciente, Canfora y Di Penta establecen que aunque la ingeniería inversa es una actividad que lleva mucho tiempo, se ha convertido en una nueva disciplina y se hace indispensable que la industria y la academia unan esfuerzos para dar soporte a las herramientas, técnicas y metodologías de la ingeniería inversa, con el fin de facilitar el mantenimiento, evolución y calidad de los productos (Canfora & Di Penta, 2007).

# 3.3 UML

UML o Lenguaje de Modelado Unificado define los componentes que se utilizarán para construir el sistema y las interfaces que conectarán los componentes, utilizando una combinación del desarrollo incremental e iterativo (Pressman, 2002). Éste provee un lenguaje grafico común y simple entre desarrolladores, arquitectos de software y usuarios experimentados para representar la implementación y el diseño de software. Con UML se puede expresar un diseño detallado en el nivel de las clases, ver donde hay concurrencia y paralelismo, además de poder aumentar características como la robustez (Pilone, 2006).

Desde sus inicios en 1997, el Lenguaje de Modelado Unificado ha atraído a muchas organizaciones y profesionales, debido a que UML es un idioma de modelado para el desarrollo de software (Lange, Chaudron, & Muskens, 2006).  La versión más actual de UML es la 2.0, la cual define trece tipos de diagramas, divididas en tres categorías: seis tipos de diagramas representan la estructura de aplicación estática, tres representan tipos generales de comportamiento, y cuatro representan diferentes aspectos de las interacciones (Object Management Group (OMG)):

* Diagramas de estructura incluyen el diagrama de clases, diagrama de objetos, diagrama de componentes, diagrama de la estructura compuesta, Diagrama de paquetes, y diagrama de implementación.
* Diagramas de comportamiento incluyen el diagrama de casos de uso (utilizado por algunas de las metodologías en la recogida de requisitos); diagrama de actividades y diagrama de estados.
* Diagramas de interacción, los derivados de los diagrama de comportamiento, incluyen el diagrama de secuencia, diagrama de Comunicación, Cronograma, y diagrama de interacción general.

En la mayoría de los proyectos, los modelos UML son los primeros artefactos que se utilizan para representar sistemáticamente una arquitectura de software. A continuación, estos modelos sufren modificación con el fin de perfeccionarlos durante el proceso de desarrollo del software (Lange, Chaudron, & Muskens, 2006). La importancia de estos modelos ha incrementado con la llegada de la metodología de la arquitectura basada en modelos (Object Management Group) .

UML 2.0 refleja mejores prácticas de modelado. Entre las mejoras principales de esta versión, tenemos (Francia, Ghosh, Dinh-Trong, & Solberg, 2006):

* Mejor soporte para el concepto de UML como una *familia de lenguas* a través de la utilización de perfiles y puntos de variación semántica que marcan las partes de UML, dejando intencionadamente sin semántica para dar cabida a los definidos por el usuario.
* Mejora de la expresividad de modelado, incluyendo un mejor modelado de procesos de negocio, soporte para el modelado de clasificadores reutilizables, y el apoyo para el modelado de las arquitecturas de sistemas heterogéneos, distribuidos.
* Integración de la semántica de acción que los desarrolladores pueden utilizar para definir la semántica de ejecución de los modelos y proporcionar la precisión semántica necesaria para analizar los modelos y traducirlos en las implementaciones.

El estándar UML 2.0 contiene un amplio conjunto de conceptos de modelado que se relacionan de manera compleja. En la defensa de la magnitud y complejidad de UML 2.0, los arquitectos señalan que el lenguaje tiene como objetivo apoyar el modelado en una variedad de dominios. Para hacer frente a esta complejidad, los diseñadores organizaron la norma en cuatro partes (Francia, Ghosh, Dinh-Trong, & Solberg, 2006):

* Infrastructure: define las clases base que proporcionan la base para construcciones de modelado UML;
* Superstructure: define los conceptos que los desarrolladores utilizan para crear modelos UML;
* Object constraint language: define el idioma utilizado para especificar consultas, invariantes y especificaciones de operación en los modelos UML, y
* Diagram interchange: define una extensión del metamodelo de UML que soporta el almacenamiento y el intercambio de información relacionada con el diseño de los modelos UML.

# 3.4 XMI

El XMI o XML Metadata Interchange Specification es un estándar que permite representar los objetos de programación orientada a objetos utilizando XML (Extensible Markup Language), el cual es un estándar universal para la representación de datos en internet. Éste está estrechamente relacionado con las normas de modelado, lo que le permite emplear el modelado de manera efectiva en sus esfuerzos XML (Grose, Doney, & Brodsky, 2002).

El XMI tiene como principal propósito brindar una manera sencilla de permitir el intercambio de datos entre diagramas de modelado y está basado en tres estándares muy importantes para la industria del software como lo son:

* XML – (eXtensible Markup Language) (OMG , 2009)
* UML (Unified Modeling Language) (OMG , 2009)
* MOF – (Meta Object Facility) (ISO/IEC 19503:2005, 2005)

Para comprender cómo los modelos se pueden intercambiar utilizando el Estándar XMI se tiene que entender cómo se organizan los modelos de acuerdo a un lenguaje de modelado. De acuerdo a los estándares de modelado de OMG (Object Management Group (OMG)), un lenguaje de modelado, como UML se define como un modelo en el lenguaje MOF (Alanen & Porres, 2005).

XMI incluye dos reglas de generación. Una son las reglas de generación de esquemas de documentos XMI y las otras son las reglas de generación de documentos XMI. Se utiliza el esquema de documentos XMI para verificar el contenido del documento XMI. Estas dos reglas de XMI sirven para apoyar UML y MOF. Por lo tanto, usando reglas para la generación de XMI, podemos producir esquemas de documentos XMI y documentos XMI del MOF y UML. Debido a que el metamodelo UML es una instancia del modelo de MOF y UML es definido por el MOF (Pagano & Anne, 2009.), los esquemas de XMI generados por el metamodelo de UML se pueden utilizar para verificar el documento XMI generado por los modelos UML. XMI también proporciona un mecanismo para el intercambio de los cambios de modelo de UML (Yan & Du, 2010).

Cada esquema XMI se compone de las siguientes declaraciones (Object Management Group (OMG)):

* Una instrucción de procesamiento de la versión XML. Ejemplo: <? Versión XML = "1.0">
* Una declaración de codificación opcional que especifica el conjunto de caracteres, que sigue la norma ISO-10646 (también llamado extendido Unicode). Ejemplo: <? Versión XML = "1.0" encoding = "UCS-2">
* Cualquier otra instrucción valida de procesamiento XML.
* Un elemento de esquema XML.
* Un elemento XML de importación para el espacio de nombres XMI.
* Declaraciones de un modelo específico.

Los documentos XMI están obligados a ser válidos y estar bien formados de acuerdo a la recomendación definida por XMI. Cada documento XMI consta de las siguientes declaraciones, a menos que el XMI está incrustado en otro documento XML (Object Management Group (OMG)):

* Una instrucción de procesamiento XML versión.
* Una declaración de codificación opcional que especifica el conjunto de caracteres.
* Cualquier otra instrucción valida de procesamiento XML.

Cuando se trabaja con XMI se tienen los siguientes beneficios:

* XMI proporciona una representación estándar de los objetos en XML, lo que permite el intercambio efectivo de los objetos utilizando XML.
* XMI especifica cómo crear esquemas XML a partir de modelos.
* XMI permite crear documentos XML sencillos y hacerlos más avanzada como sus aplicaciones evolucionan (Grose, Doney, & Brodsky, 2002).
* Al ser basado en XML permite realización de consultas internamente por medio de XQuery (Gutiérrez, Escalona, Mejías, Torrres, & Villadiego, 2005).

# 3.5 XQuery

Un grupo de trabajo diseñó y desarrolló el lenguaje XQuery, el cual fue impulsado por la estructura de XML y por un conjunto de casos de uso para realizar consultas en una variedad de ajustes (W3C, 2007). XQuery es un lenguaje estandarizado de consulta diseñado para escribir consultas sobre colecciones de datos expresadas en XML cuya función es la de extracción de información de un ese conjunto de datos organizados (W3C) (Gutiérrez, Escalona, Mejías, Torrres, & Villadiego, 2005).

XQuery es un lenguaje funcional, lo que significa que, en vez de ejecutar una lista de comandos como un lenguaje procedimental clásico, cada consulta es una expresión que es evaluada y devuelve un resultado, al igual que en SQL (W3C), pero XQuery funciona en una amplia variedad de entornos, con y sin bases de datos o servidores de aplicaciones (Robie, XML Processing and Data Integration with XQuery, 2007) . XQuery no sólo es capaz de consultar una variedad de datos XML y los documentos, sino también datos de múltiples bases de datos. En la actualidad, muchos de base de datos proporcionan apoyo para la especificación XQuery (Zhang, 2012).

XQuery simplifica el procesamiento de XML, ya que es un lenguaje nativo XML que trabaja con XML con tanta naturalidad como un lenguaje orientado a objetos (Robie, Procesamiento de XML y la integración de datos con XQuery, 2007). Diversas expresiones pueden combinarse de una manera muy flexible con otras expresiones para crear nuevas expresiones más complejas y de mayor potencia semántica (Gutiérrez, Escalona, Mejías, Torrres, & Villadiego, 2005).

XQuery es un nuevo tipo de lenguaje de consulta, capaz de accesar a los datos XML de una consulta de manera eficiente. XQuery absorbe las ventajas de una variedad de otros lenguajes de consulta y puede consultar todo tipo de origen de datos XML. Por lo tanto, tiene una potente capacidades de consulta, simple, flexible y fácil de implementar.

El modelo de datos XQuery (W3C, 2010), es el mismo modelo de datos que utiliza XPath 2.0, el cual es un lenguaje de expresión que permite el procesamiento de valores (Robie, XML Processing and Data Integration with XQuery, 2007) y representa las estructuras en documentos XML. Además, para buscar estructuras XML, XQuery utiliza expresiones de ruta XPath 2.0 's, que permiten especificar las consultas de las relaciones estructurales entre los nodos (ascendiente, descendiente, ascendiente, descendiente, anterior y siguiente) y comparar los valores (Robie, 2007).

# 4. ESTADO DEL ARTE

La ingeniería inversa a pesar de que es una actividad antigua, sigue siendo una disciplina joven (Canfora, Di Penta, & Cerulo, 2011) , cuyo concepto ha sabido evolucionar de una primera definición que la detalla como “el proceso de análisis de un sistema para identificar sus componentes y las relaciones entre ellos, y para crear una representación del sistema en otra forma o en otro nivel de abstracción” (Chikovsfky & Cross, 1990), a una que la incluye como “todo método destinado a la recuperación del conocimiento de un sistema software, en apoyo a la realización de actividades de ingeniería de software” (Tonella, Torchiano, Du Bois, & Systä, 2007) (Monroy, Arciniegas , & Rodríguez, 2012). A partir de la recuperación del conocimiento se convierte indispensable la visualización en el proceso de análisis, en las cuales estas vistas de software se construyen a partir de la abstracción realizada a la información base en la cual se almacena el resultado de la extracción de los artefactos del software realizada por el analizador (Canfora, Di Penta, & Cerulo, 2011).

Actualmente el ingeniero de software puede dar retroalimentación a las herramientas de ingeniería inversa con el fin de le provean un resultado más refinado y preciso de las vistas que se pueden obtener. Por otro lado existe un tipo particular de vista que cada día está siendo más utilizada: los sistemas de recomendación, que brindan sugerencias al ingeniero de software y propone cambios que podrían ser necesarios (Canfora, Di Penta, & Cerulo, 2011). La ingeniería inversa se ​​ha centrado tradicionalmente en la recuperación de vistas funcionales y arquitectónicas de los artefactos de un software existentes (Canfora, Di Penta, & Cerulo, 2011) y es por eso a partir de lo anterior uno de sus objetivos principales su evolución es producir vistas de software generadas a partir de las abstracciones almacenadas en una base de conocimiento, pero mientras se da ese proceso, hay una serie de retos que debe superar como lo son: la capacidad para consultar la información base obtenida en los artefactos generados por los analizadores. La primera vez que fue planteado como reto de la ingeniería inversa fue en el 2007 (Canfora & Di Penta, 2007) a pesar de que ya existían lenguajes de consulta desde años anteriores (Holt, Wu, & Winter, 2002) como Grok[[2]](#footnote-2) y GReQL[[3]](#footnote-3).

El término Grok fue acuñado por Heinlein en 1961 en su novela de ciencia ficción “Stranger in a Strange Land”. Esta palabra significa "compartir la misma realidad o línea de pensamiento con otra entidad física o conceptual". Para el área de ingeniería de software, Grok es una notación para la manipulación de relaciones binarias utilizando el álgebra relacional Tarski, que comprende operadores para manipular conjuntos o relaciones, aplicados a la ingeniería del software. El lenguaje Grok se llama así porque uno de sus objetivos es ayudar a las personas a pensar y compartir conocimiento acerca de arquitectura de software (Holt R. , 2008).

El triunfo principal en el uso de Grok es la eficiencia en el procesamiento de grandes gráficos que representan las relaciones entre los componentes de software, producto de la extracción de la arquitectura del software en un proceso de ingeniería inversa. Los gráficos que representan la arquitectura software que ha sido extraída puede ser manipulada o transformada con el fin de obtener más información o hacer que la arquitectura sea más fácil de entender, aquí encontramos una desventaja del uso de Grok, debido a que hay algunas transformaciones que son difíciles de expresar utilizando el álgebra relacional. Grok tiene algunas construcciones algebraicas no relacionales, que se pueden utilizar para expresar estas transformaciones, sin embargo, la ejecución de dichas transformaciones son lentas, incluso para pequeños gráficos (Fahmy, Holt, & Cordy, 2001). Grok es un lenguaje prometedor de la ingeniería inversa y por lo tanto debe ser sometido a un mayor número de investigaciones, con el fin de evaluar su funcionamiento y facilitar la evolución del mismo.

Paralelo a Grok, se tiene GReQL (Graph REpository Query Language), es una expresión que ofrece lenguaje especializado en realizar consultas en estructuras TGraph (Kullbach, Winter, Dahm, & Ebert, 1998). GReQL se parece mucho a SQL, pero se utiliza para realizar consultas gráficas escritas. Puede extraer n-patrones. A diferencia de Grok, GReQL no actualiza los gráficos, por lo que fue más diseñado para reconocer patrones en lugar de a transformar la estructura arquitectónica (Holt R. , 2008). GReQL sufre limitaciones, como que no puede expresar manipulaciones de datos que se deben realizar después de realizar una consulta (Cox & Clarke, 2001).

Actualmente, el reto es de brindar un lenguaje de consulta poderoso que permita la creación de vistas que brinden información significativa proveniente de la información base (Canfora, Di Penta, & Cerulo, 2011) (Monroy, Arciniegas , & Rodríguez, 2012), teniendo en cuenta las limitantes generadas por las herramientas de consultas existentes, como GReQl y Grok.

Se han realizado esfuerzos en la visualización de la información base**,**  como Korshunova (Korshunova & Petkovic, 2006) y FengQin (FengQin, HengJin, & JianBin, 2009) que proponen reconstruir el comportamiento del sistema a través de diagramas de secuencia, mientras que Quingshan, propone que se haga por medio de diagramas de casos de uso (Qingshan, Shengming, Ping, Wu, & Wei, 2007). Pero, no se trabajado en la preparación de un mecanismo poderoso que permita la realización de consultas que no presente limitaciones, teniendo en cuenta que al trabajar con los modelos UML resultantes del trabajo del analizador – representados en XMI- y almacenados en la información base, se posee disponible un lenguaje universal de consulta llamado XQuery y por tratarse de XMI se puede manejar la información de manera jerarquizada al igual que XML (Gutiérrez, Escalona, Mejías, Torrres, & Villadiego, 2005) (Grose, Doney, & Brodsky, 2002).

En el ámbito local el Mag. Martín Monroy, profesor Asociado de la Universidad de Cartagena y estudiante del programa de doctorado en Ingeniería Telemática de la Universidad del Cauca, plantea su tesis doctoral “Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento” la necesidad de poseer un mecanismo de consulta a partir de la información obtenida en un proceso de ingeniería inversa además de afirmar en su artículo “Framework for recovery and analysis of behavioral architectural views” al igual que Canfora, Di Penta y Cerulo que se necesitaba mejorar la “Capacidad para consultar la información base obtenida en los artefactos generados por los analizadores.” (Monroy, Arciniegas , & Rodríguez, 2012) (Canfora, Di Penta, & Cerulo, 2011)

5. OBJETIVOS

5.1 OBJETIVO GENERAL

Construir un mecanismo de consulta a partir de la información obtenida en el proceso de ingeniería inversa, para facilitar la capacidad de análisis de los ingenieros de software en su búsqueda de la evolución y mantenimiento del producto, utilizando el lenguaje de consulta XQuery.

# 5.2 OBJETIVOS ESPECÍFICOS

* Identificar los requisitos de consulta que debe atender el mecanismo que se construirá para realizar la búsqueda de la información solicitada
* Diseñar el mecanismo de consulta que provea los aspectos necesarios para que la información resultado sea representada de manera entendible para el Ingeniero de Software.
* Validar el mecanismo a través de la construcción de un prototipo

# 6. METODOLOGÍA

Este trabajo de grado será considerado como una investigación aplicada de tipo exploratorio, debido a que persigue fines en los cuales no se ha trabajado lo suficiente anteriormente. Dado que la necesidad de mejorar la “Capacidad para consultar la información base obtenida en los artefactos generados por los analizadores” (Monroy, Arciniegas, & Rodriguez, 2012) (Canfora, Di Penta, & Cerulo, 2011) es un tema relativamente reciente, planteado por primera vez como reto de la ingeniería inversa en el año 2007(Canfora, & Di Penta, 2007).

Lo anterior, constituye uno de los retos que se deben superar durante el proceso de producción de vistas de software generadas a partir de las abstracciones almacenadas en una base de conocimiento. Se han realizado esfuerzos en la creación de lenguajes de consulta mas no en la preparación de un mecanismo que permita la realización de consultas, y los lenguajes de consultas existentes no son los suficientemente poderosos para permitir la creación de vistas que brinden información significativa proveniente de la información base. (Monroy, Arciniegas, & Rodriguez, 2012)(Canfora, Di Penta, & Cerulo, 2011).

Por tal razón, es de vital importancia para el desarrollo de este trabajo de grado conocer y entender los temas mencionados, en búsqueda de aumentar el grado de familiaridad con los mismos, haciendo buen uso del tiempo y los recursos bibliográficos con los que se va a contar. De igual forma, es indispensable realizar una exhaustiva revisión de la literatura que permita alcanzar los objetivos definidos. Además, esta investigación debe servir como punto de partida para investigaciones que se realicen en un futuro.

Para dar cumplimiento a cada uno de los objetivos específicos orientados hacia el logro del objetivo general del proyecto y teniendo en cuenta que es un trabajo exploratorio se propone la siguiente metodología para cubrir las necesidades de este trabajo de investigación.

*Construir el estado del arte sobre las herramientas de consulta a partir de la información obtenida en el proceso de ingeniería inversa, que faciliten la capacidad de análisis.*

Se realizará una revisión minuciosa de la literatura concerniente al proceso de ingeniería inversa y uno de los retos que este proceso presenta relacionado con la “Capacidad para consultar la información base obtenida en los artefactos generados por los analizadores”, en las bibliotecas digitales disponibles en la Universidad Cartagena, como la IEEE Computer Society Digital Library (CSDL)[[4]](#footnote-4). Lo anterior, posibilitará la identificación de trabajos afines que se hayan realizado y los que faltan por realizar respecto al tema.

*Especificar los requisitos que involucra la construcción de un mecanismo de consulta a partir de la información base obtenida del proceso de ingeniería inversa, con el fin de realizar búsquedas de la información solicitada.*

Se analizarán y estudiarán los requerimientos más relevantes relacionados a la forma en que se encuentran diseñados los lenguajes de consulta existentes en la actualidad, teniendo en cuenta cuál de estos cuentan con mecanismos de consulta y en tal caso de que no lo hagan, identificar las necesidades de esta falencia. Lo anterior será posible gracias a la construcción de estado del arte mencionado anteriormente. Ésta información recolectada será el punto de partida para identificar y describir los requerimientos involucrados en la construcción de un mecanismo de consulta, porque permitirá identificar que se ha realizado y que falta por realizar.

*Diseñar un mecanismo de consulta que provea los aspectos necesarios para que la información resultado sea representada de manera entendible para el Ingeniero de software.*

De acuerdo a los requerimientos especificados en el anterior ítem, se propondrá un diseño que se ajusta a esas necesidades.

*Diseñar un prototipo[[5]](#footnote-5) para validar el mecanismo de consulta.*

En base al mecanismo de consulta diseñado anteriormente, se requiere el diseño de un prototipo que permita validarlo. Para realizar el diseño del prototipo, es necesario identificar y seleccionar los patrones arquitectónicos y de diseño que mejor se adapten a las necesidades. Para la realización del prototipo tomaremos como base la metodología RUP (Rational Unified Proccess) (IMB), la cual se divide en cuatro fases:

* Inicio y Elaboración: Comprensión del problema y de las tecnologías a utilizar, se delimitará el ámbito del proyecto, se eliminarán los riesgos críticos y se establecerá una arquitectura a seguir. Durante la fase inicio se hará énfasis en las actividades de modelado de negocio y de requisitos funcionales y no funcionales. En la fase de elaboración, el trabajo se orientará de acuerdo a la arquitectura seleccionada, realizando el refinamiento del modelo de negocios, análisis, diseño y una parte de la implementación orientada a la arquitectura.
* Construcción: En esta fase se llevará a cabo la construcción del prototipo de acuerdo al análisis y diseño realizado en la fase de elaboración.
* Transición: En esta fase se pretende garantizar que se tiene el prototipo realizado.

Se realizarán iteraciones, con el fin de refinar el análisis y diseño realizado, procediendo a la implementación del prototipo y realización de pruebas. La implementación del prototipo se hará con el fin de validar el mecanismo de consulta diseñado, definiendo escenarios de pruebas. Para esta implementación se deben identificar y seleccionar las herramientas con las cuales se a trabajar, tales como, lenguaje de programación y mecanismos de persistencia. La implementación de este prototipo permitirá validar la solución conceptual del mecanismo de consulta.

*Elaborar un informe final sobre el mecanismo de consulta.*

De acuerdo a la validación del mecanismo de consulta a partir de la implementación del prototipo, se deben realizar las modificaciones y ajustes correspondientes al diseño del mecanismo. A partir de esto, se debe elaborar un informe final y entregar los resultados de este informe.

# 7. CRONOGRAMA

Tabla 1. Cronograma de actividades

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre** | **Duración** | **Comienzo** | **Fin** |
| **Actividad 1** | **Recolección de información** | **20 días** | 01-abr-13 | 20-abr-13 |
| **Actividad 2** | **Análisis de la información recolectada** | **20 días** | 06-abr-13 | 25-abr-13 |
| **Actividad 3** | **Construcción del estado del arte** | **45 días** | 8-abr-13 | 22-may-13 |
| **Actividad 4** | **Especificar los requerimientos que involucra la construcción de un mecanismo de consulta a partir de la información base obtenida del proceso de ingeniería inversa, con el fin de realizar búsquedas de la información solicitada.** | **10 días** | 20-may-13 | 29-may-13 |
| Subactividad 1 | *Identificar de características y funcionalidades que debe tener el mecanismo* | *8 días* | *20-may-13* | *27-may-13* |
| Subactividad 2 | *Especificar requerimientos del mecanismo de consulta a proponer* | *6 días* | *24-may-13* | *29-may-13* |
| **Actividad 5** | **Diseñar un mecanismo de consulta que provea los aspectos necesarios para que la información resultado sea representada de manera entendible para el Ingeniero de software.** | **45 días** | 26-may-13 | 9-jul-13 |
| **Actividad 6** | **Diseñar un prototipo que permita validar el mecanismo de consulta diseñado** | **60 días** | 9-jul-13 | 6-sep-13 |
| Subactividad 1 | *Identificar y seleccionar patrones arquitectónicos y de diseño que mejor se adapten a las necesidades e utilizarlos en el diseño del prototipo. Y de acuerdo a estos, realizar el diseño del prototipo.* | *3 días* | *9-jul-13* | *11-jul-13* |
| Subactividad 2 | *Seleccionar las herramientas con las cuales se va a trabajar, tales como, lenguaje de programación y mecanismos de persistencia* | *3 días* | *10-jul-13* | *12-jul-13* |
| Subactividad 3 | *Definición de escenarios de prueba para el prototipo* | *10 días* | *10-jul-13* | *19-jul-13* |
| Subactividad 4 | *Implementar un prototipo para validar la solución conceptual del mecanismo de consulta y utilizarla en distintos escenarios.* | *51 días* | *18-jul-13* | *6-sep-13* |
| **Actividad 7** | **Elaborar un informe final sobre el mecanismo de consulta.** | **25 días** | 06-sep-13 | 30-oct-13 |
| Subactividad 1 | *Realizar modificaciones y ajustes al mecanismo de consulta* | *15 días* | *06-sep-13* | *20-sep-13* |
| Subactividad 2 | *Elaboración del informe final y entrega de resultados* | *15 días* | *16-sep-13* | *30-oct-13* |

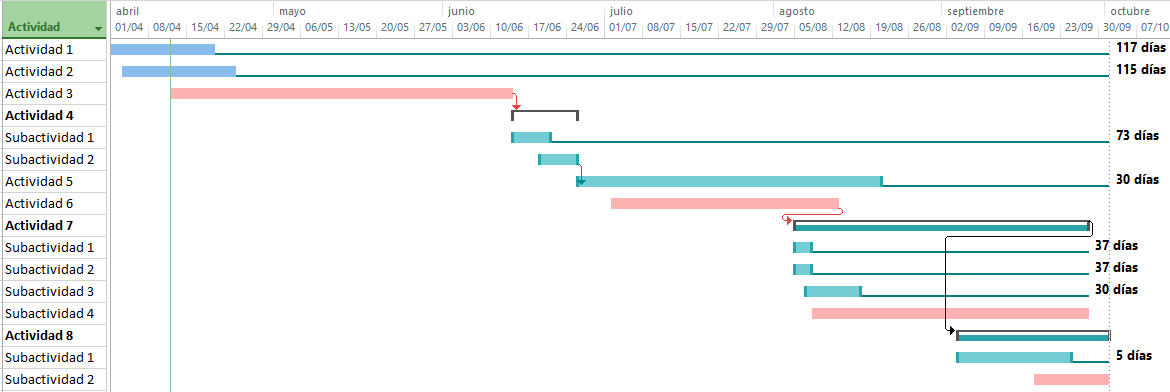


Figura 1. Diagrama de GANTT de las actividades

# 8. RESULTADO ESPERADOS Y POTENCIALES BENEFICIOS

Al término de este trabajo de investigación se obtendrá informe detallado de los requerimientos de consulta que debe atender el mecanismo que se construirá para realizar las búsquedas solicitadas. A partir de este informe, se podrá presentar el diseño del mecanismo de consulta que proveerá los aspectos necesarios para que la información resultado sea representada de manera entendible para el Ingeniero de Software.

Con base en el diseño del mecanismo presentado, inicialmente se diseñará un prototipo que permita validarlo y luego se implementará dicho prototipo haciendo usos de escenarios de prueba. Luego de validar la solución conceptual del mecanismo de consulta se obtendrá un informe detallado con los resultados obtenidos. Este informe permitirá realizar modificaciones y ajustes correspondientes al diseño del mecanismo de consulta.

Además, se pretende que este trabajo se convierta en un referente para futuras investigaciones que tengan como propósito general la construcción de un mecanismo de consulta a partir de la información obtenida en el proceso de ingeniería inversa, para facilitar la capacidad de análisis de los ingenieros de software en su búsqueda de la evolución y mantenimiento del producto.

En las siguientes tablas se presentarán los resultados esperados y potenciales beneficios del proyecto en relación Generación del Nuevo Conocimiento, Fortalecimiento de la Comunidad Científica y Apropiación Social del Proyecto.

* Generación de nuevo conocimiento.

Tabla 2. Generación de nuevo conocimiento.

|  |  |  |
| --- | --- | --- |
| Resultado Esperado | Indicador | Beneficiario |
| **Mecanismo de consulta propuesta** | Documentación técnica del mecanismo de consulta | -Universidad de Cartagena  -Comunidad académica en general  -Comunidad de Ingeniería del Software |
| **Prototipo mecanismo de consulta** | Producto software | -Universidad de Cartagena  -Comunidad académica en general  -Comunidad de Ingeniería del Software |

* Fortalecimiento de la comunidad científica.

Tabla 3. Fortalecimiento de la comunidad científica

|  |  |  |
| --- | --- | --- |
| Resultado Esperado | Indicador | Beneficiario |
| **Modelo de referencia para futuras investigaciones que tengan por objeto la construcción de mecanismos de consulta a partir de la información obtenida en el proceso de ingeniería inversa** | Informe Final de la investigación | -Universidad de Cartagena  -Instituciones Educativas  -Comunidad Ingeniería de Software |

* Apropiación social del conocimiento.

Tabla 4. Apropiación social del conocimiento

|  |  |  |
| --- | --- | --- |
| Resultado Esperado | Indicador | Beneficiario |
| **Socialización del proyecto realizado** | Una ponencia | -Universidad de Cartagena  -Comunidad académica en general  -Comunidad de Ingeniería del Software |
| Artículo sobre el mecanismo de consulta diseñado | Documento de aceptación de la publicación | -Universidad de Cartagena  -Comunidad académica en general  -Comunidad de Ingeniería del Software |

9. IMPACTOS ESPERADOS A PARTIR DEL USO DE LOS RESULTADOS

Al identificar los requerimientos de consulta y construir un mecanismo de consulta a partir de estos requerimientos, se habrá dado un gran paso en la solución de uno de los retos que enfrenta la ingeniería inversa: la necesidad de mejorar la capacidad de consulta de la información base obtenida a partir de los analizadores. Lo anterior, facilitará la capacidad de análisis de los ingenieros de software en su búsqueda de la evolución y mantenimiento del producto.

A continuación, se presenta de manera sintetizada el impacto esperado a partir del uso de los resultados de la investigación.

Tabla 5. Impactos esperados

|  |  |  |  |
| --- | --- | --- | --- |
| Impacto Esperado | Plazo | Indicador Verificable | Supuestos |
| Mejoramiento de los procesos de análisis a los ingenieros software al momento de actualizar y hacer mantenimiento a los sistemas. | Mediano plazo | Disminución del tiempo dedicado al análisis de la información base obtenida en los procesos de ingeniería inversa | El mecanismo de consulta contribuirá disminuirá en un 50% el tiempo requerido para el análisis de la información base obtenida en los procesos de ingeniería inversa |
| Generación de nuevos proyectos de  investigación | Mediano plazo | Cantidad de nuevos proyectos de  Investigaciones en el área de la ingeniería inversa | Se espera que se genere por lo menos un proyecto semestral sobre el área de la ingeniería inversa en el programa de ingeniería de sistemas |
| Hacer más eficiente el uso de los resultados obtenidos a partir de una herramienta de ingeniera inversa | Mediano plazo | Cantidad de personas que facilitan las actividades de mantenimiento y actualización del software a partir del mecanismo de consulta planteado. | La mayoría de ingenieros software y personas interesadas harán uso del mecanismo de consulta diseñado |

10. PRESUPUESTO

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RUBROS | | UNIVERSIDAD DE CARTAGENA | | INVESTIGADORES | | Total |
| **Efectivo** | **Especie** | **Efectivo** | **Especie** |
| Personal | | - | 8.000 | - | 10.000 | 18.000 |
| Equipos | Compra | - | - | - | - |  |
| Uso | - | - | - | 3.160 | 3.160 |
| Materiales e insumos | | - | - | 220,5 | - | 220,5 |
| Divulgación | | - | - | 4.000 | - | 4.000 |
| Material bibliográfico | |  | 650 | - | - | 650 |
| Administración | | - | - | - | 800 | 800 |
| TOTAL | | - | 8.650 | 4.220,5 | 13.960 | 26.830,5 |

Tabla 6. Presupuesto global de la propuesta por fuentes de financiación (en miles de $).

Tabla 7. Descripción de los gastos de personal (en miles de $).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nombre del  Investigador /  Experto/ Auxiliar | Formación  Académica | Función  dentro en el  proyecto | DEDICACIÓN  Horas/semana | RECURSOS | | | | TOTAL |
| Universidad de Cartagena | | Investigadores | |
| Efectivo | Especie | Efectivo | Especie |
| Martin Monroy | Ing. De Sistemas. | Director | 4 | - | 8.000 |  |  | 8.000 |
| Cindy Pacheco Álvarez | Estudiante de Ingeniería de Sistemas | Investigador | 16 | - |  |  | 5.000 | 5.000 |
| Alejandra Ríos Rodríguez | Estudiante de Ingeniería de Sistemas | Investigador | 16 | - |  |  | 5.000 | 5.000 |

Tabla 8. Descripción y cuantificación de los equipos de uso propio (en miles de $).

|  |  |
| --- | --- |
| EQUIPO | VALOR |
| Computador portátil | 3.100 |
| Memorias USB | 60 |
| TOTAL | 3.160 |

Tabla 9. Descripción detallada de Materiales y suministros (en miles de $).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Descripción de los materiales | Justificación | Recursos | | | | Total |
| Universidad de Cartagena | | Investigadores | |
| Efectivo | Especie | Efectivo | Especie |
| Copias | Material bibliográfico de temas especializados. | - | - | 60,5 | - | 60,5 |
| Impresiones | Presentación de documentos ante Comité de Evaluación. | - | - | 160 | - | 160 |
| TOTAL | |  | | 220,5 |  | 220,5 |

Tabla 10. Descripción detallada de la Bibliografía (en miles de $).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Descripción | Justificación | Recursos | | | | Total |
| Universidad de Cartagena | | Investigadores | |
| Efectivo | Especie | Efectivo | Especie |
| Artículos | Material de apoyo | - | 300 | - | - | 300 |
| Libros | Material de apoyo | - | 350 | - | - | 350 |
| TOTAL | |  | |  |  | 650 |

# 11. BIBLIOGRAFÍA

*W3C*. (2007). Obtenido de XML Query Use Cases: http://www.w3.org/TR/xquery-use-cases/

*OMG* . (2009). Obtenido de Uniﬁed Modeling Language(UML) : http://www.omg.org/technology

*OMG* . (2009). Obtenido de Extensible Markup Language(XML) : http://www.w3.org

*W3C*. (2010). Obtenido de XQuery 1.0 and XPath 2.0 Data Model (XDM) (Second Edition): http://www.w3.org/TR/xpath-datamodel/

Alanen, M., & Porres, I. (2005). Model Interchange Using OMG Standards.

Arciniegas Herrera, J. L. (2006). Contribution to Quality-driven Evolutionary Software Development Process for Service-Oriented Architecture. Madrid, España.

Beck, F., & Diehl, S. (2010). *Visual Comparison of Software Architectures.*

Canfora, G., & Di Penta, M. (2007). New Frontiers of Reverse Engineering. *IEEE Computer Society*, 1.

Canfora, G., Di Penta, M., & Cerulo, L. (2011). Achievements and Challenges in Software Reverse Engineering. *Communications of the acm*, 147.

Chikovsfky, E., & Cross, J. (1990). Reverse Engineering and Design Recovery: a Taxonomy. *IEEE Software*, 5.

Cox, A., & Clarke, C. (2001). Representing and Accessing Extracted Information.

Dajsuren, Y., Goldstein, M., & Moshkovich, D. (2010). Modernizing Legacy Software Using a System Grokking technology.

Fahmy, H., Holt, R., & Cordy, J. (2001). Wins and Losses of Algebraic Transformations of Software Architectures.

FengQin, W., HengJin, K., & JianBin, L. (2009). Towards the Reverse Engineer of UML 2.0 Sequece Diagram for Procedure Blueprint.

Francia, R. B., Ghosh, S., Dinh-Trong, T., & Solberg, A. (2006). Model-Driven Development Using UML 2.0: Promises and Pitfalls.

Grose, T., Doney, G., & Brodsky, S. (2002). *Mastering XMI Java Programming with XMI, XML, and UML.* Wiley Computer Publishing.

Gutiérrez, J., Escalona, M., Mejías, M., Torrres, J., & Villadiego, D. (2005). XQuery. *Lenguajes y Sistemas Informáticos, Universidad de Sevilla*.

Holt, R. (2008). Grokking Software Architecture.

Holt, R., Wu, J., & Winter, A. (2002). Towards a Common Query Language for Reverse Engineering.

IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology S t .610.121990.*

*IMB*. (s.f.). Obtenido de IBM Rational Unified Process (RUP): http://www-01.ibm.com/software/rational/rup/

ISO/IEC 19503:2005. (2005). *XML Metadata Interchange Specification.*

*Object Management Group (OMG)*. (2005 de Julio). Obtenido de Introduction To OMG's. Unified Modeling Language™ (UML®): http://www.omg.org/gettingstarted/what\_is\_uml.htm

Korshunova, E., & Petkovic, M. (2006). CPP2XMI Reverse Engineering of UML Class, Sequence, and Activity Diagrams from C++ Source Code. 13th Working Conference on Reverse Engineering.

Kullbach, B., Winter, A., Dahm, P., & Ebert, J. (1998). Program Comprehension in Multi-Language Systems.

Lange, C., Chaudron, M., & Muskens, J. (2006). In Practice: UML Software Architecture and Design Description.

Li, Q., Hu, S., Chu, H., & Chen, P. (2005). XDRE: A Reverse Engineering Tool of Object-Oriented Software.

Monroy, M., Arciniegas , J., & Rodríguez, J. (2012). Framework for recovery and analysis of behavioral architectural views. *EATIS'12 Conference Proceedings*, 430.

*Object Management Group (OMG).* (s.f.). Obtenido de Documents Associated With MOF/XMI Mapping, Version 2.4.1: http://www.omg.org/spec/XMI/2.4.1/PDF

*Object Management Group (OMG)*. (s.f.). Obtenido de About OMG: http://www.omg.org/gettingstarted/gettingstartedindex.htm

Object Management Group, O. (s.f.). Obtenido de MDA - The Architecture Of Choice For A Changing World. OMG Model Driven Architecture: http://www.omg.org/mda/

Pagano, D., & Anne, B. (2009.). Engineering Document Applications - From UML Models to XML Schemas.

Panas, T., Löwe, W., & Aßmann, U. (2003). Towards the Unified Recovery Architecture for Reverse Engineering.

Pilone, D. (2006). *UML 2.0 Pocket Reference.* O'Reilly.

Pressman, R. (2002). *Ingeniería de Software: Un enfoque practico .* McGRAW-HILL.

Qingshan, L., Shengming, H., Ping, C., Wu, L., & Wei, C. (2007). Discovering and Mining Use Case Model in Reverse Engineering. *IEEE*.

Raza, A., Vogel, G., & Plodereder, E. (2006). Bauhaus- a Tool Suite for Program Analysis and Reverse Engineering.

Robie, J. (2007). Procesamiento de XML y la integración de datos con XQuery.

Robie, J. (2007). XML Processing and Data Integration with XQuery.

*SDML, Software Development Laboratory*. (s.f.). Obtenido de srcML: A document-oriented XML representation of source code: http://www.sdml.info/projects/srcml/

Storey, M., & Wong, P. (1996). On Designing an Experiment to Evaluate a Reverse Engineering Tool. Third Working Conference on Reverse Engineering.

Swebok. (2012). SWEBOK Guide V3 – Alpha Version. *IEEE*.

Terry, B., & Logee, D. (1990). Terminology for Software Engineering and Computer-aided Software Engineering.

Tonella, P., Torchiano, M., Du Bois, B., & Systä, T. (2007). Empirical studies in reverse engineering: state of the art and future trends. *Springer Science + Business Media*.

*Vrije Universiteit Brussel*. (s.f.). Obtenido de Sofware Languages Lab, Jar2UML: http://soft.vub.ac.be/soft/research/mdd/jar2uml

*W3C*. (s.f.). Obtenido de http://www.w3.org/TR/xquery/

Yan, W., & Du, Y. (2010). Research on reverse engineering from formal models to UML models.

Zhang, L. (2012). The Query and Application of XML Data Based on XQuery .

1. Modelos como CMMI, ISO 9126, McCall son un ejemplo para ésta afirmación. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. GReQL es un lenguajes de consulta completas de TGraphen y ofrece como tal, además de acceder a todas las propiedades básicas de los gráficos, los nodos y los bordes de un número de otros elementos de lenguaje para un acceso fácil y eficiente a los gráficos [↑](#footnote-ref-3)
4. Es la colección más completa de las publicaciones de esta sociedad profesional,  líder en proveer información técnica y servicios en las áreas de la computación, informática y sistemas. [↑](#footnote-ref-4)
5. “Ejemplar original o primer molde en que se fabrica una figura u otra cosa”. [↑](#footnote-ref-5)