CONSTRUCCIÓN DE UN MECANISMO DE CONSULTA PARA ANÁLISIS DE LA BASE DE INFORMACIÓN OBTENIDA EN UN 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, 2014

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

E-SOLUCIONES

Ingeniería De Software

INVESTIGADORES

Cindy Margarita Pacheco Álvarez

Alejandra Inés Ríos Rodríguez

Director: Martín Monroy Ríos, Msc.

UNIVERSIDAD DE CARTAGENA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS, 2014

UNIVERSIDAD DE CARTAGENA

Tesis de Grado:	CONSTRUCCIÓN DE UN MECANISMO DE CONSULTA PARA ANÁLISIS DE BASE DE INFORMACIÓN OBTENIDA EN UN PROCESO DE INGENIERÍA INVERSA
Autores:	CINDY MARGARITA PACHECO ÁLVAREZ
	ALEJANDRA INÉS RÍOS RODRÍGUEZ
Director:	Msc. MARTÍN MONROY RÍOS

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Cartagena de Indias, ____ de _____ de 2014

RESUMEN

En las herramientas de ingeniería inversa, un mecanismo de consulta debe ser un componente esencial para el análisis de software debido a que la extracción de porciones de información facilita tareas de mantenimiento. Sin embargo, realizar consultas a la arquitectura es un reto, debido a que los avances realizados no han sido lo suficientemente potentes. La construcción de un mecanismo que permita realizar consultas representaría grandes beneficios a la ingeniería inversa y por lo tanto a la ingeniería del software.

El presente trabajo de investigación nace con el fin de abordar la necesidad descrita anteriormente, cuyo principal objetivo fue definir un mecanismo que permita realizar consultas a la arquitectura de software representada en una base de información, con el fin facilitar su análisis.

Para cumplir con los objetivos planteados se realizó una investigación de tipo exploratoria y aplicada, debido que persigue fines en los que no se había trabajado suficiente anteriormente. Inicialmente se construyó el estado del arte que permitió encontrar los antecedentes de las investigaciones realizadas, facilitando la identificación de las necesidades en las que se debían trabajar. Estas necesidades, permitieron definir los requisitos del mecanismo de consulta. Luego, estos facilitaron la realización del diseño del mecanismo y finalmente se construyó un prototipo software para validar las características del mecanismo implementado.

Gracias al trabajo que se realizó siguiendo el cumplimiento de los objetivos de la presente investigación, se obtuvo como resultado principal la definición de un mecanismo de consulta de arquitectura software representada en una base de información, XMI. Además, se consiguió un completo análisis y revisión de literatura plasmado en el estado del arte.

La investigación realizada permite concluir que gracias al mecanismo construido basado en algebra relacional y la generación de candidatos de búsqueda, se pueden realizar consultas a

la base de información obtenida en un proceso de ingeniería inversa, lo que facilita el análisis de esta base de información, de la cual no se tiene conocimiento previo, permitiendo generar conocimiento que puede ser utilizado para realizar mantenimiento al software. Además, el mecanismo permite el uso de un lenguaje de consulta robusto y bien estructurado.

Palabras clave: *Arquitectura de Software, Mecanismo de Consulta, Lenguaje de Consulta.*

ABSTRACT

In reverse engineering tools, a query mechanism should be an essential component analysis software because the extract important information facilitates maintenance tasks. However, the activity of architecture searching is a challenge, because the progress has not been powerful enough. Building a mechanism that allows you to do queries represent great benefits to reverse engineering and software engineering.

This research was created to address the need described above, whose main goal was to define a mechanism to do queries to software architecture represented in an information base in order to facilitate analysis.

In order to accomplish with research goals, the research was an applied and exploratory research, because it purposes had not been worked enough before. First of all the state of the art was built and it helped to identified the background of all previous researches and to identified the needs that this research had to work to. These needs allowed to define all query mechanism requirements. Then, these requirements allowed to do the design of the mechanism and finally a software prototype was built to validate all features of the created mechanism.

Due to the research carried out, the definition of a query mechanism for software architecture represented in an information base, represented in XMI, was obtained. In addition, a complete analysis and review of the literature in the art status of this research was achieved.

The research leads to the conclusion that due to the design of the query mechanism based on relational algebra and lead generating candidate search, that can do queries to the information base obtained in a reverse engineering process, which facilitates the analysis of this information base. This information base does not have knowledge, allowing to extract knowledge of it, in order to perform software maintenance. Also, the mechanism allows the use of a query language robust and well structured.

Keywords: *Software Architecture, Consultation Mechanism, Query Language*

DEDICATORIA

A mis padres por su paciencia, dedicación y apoyo incondicional durante el tiempo en que escribíamos este trabajo de grado.

Al tutor Martin Monroy Ríos, por haber compartido su invaluable conocimiento.

Alejandra Inés Ríos Rodríguez

A mis padres por brindarme el ánimo de seguir adelante. A mi abuela por todo el apoyo.

Al tutor Martin Monroy Ríos, por haber compartido su invaluable conocimiento.

Cindy Margarita Pacheco Álvarez

AGRADECIMIENTOS

Queremos a agradecer en especial a nuestros padres por toda la dedicación, comprensión y apoyo incondicional que han tenido con nosotras. Son las personas que han estado siempre a nuestro lado, animándonos ante cualquier adversidad y brindándonos consejos que nos han hecho mejores personas cada día. Agradecemos a nuestros demás familiares que también han sido nuestra motivación de superación y ser cada día mejores seres humanos y profesionales.

Nos sentimos totalmente agradecidas con el programa de Ingeniería de Sistemas de la Universidad de Cartagena, por todos los conocimientos adquiridos bajo la orientación de grandes docentes y excelentes seres humanos. También, agradecer a todos nuestros compañeros y amigos que vivieron con nosotras la etapa de la universidad.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

La Ingeniería del Software brinda diferentes soluciones representados en modelos, técnicas, estándares, etc., los cuales permiten satisfacer necesidades de un determinado problema, siempre facilitando la producción de software y buscando que este sea de calidad. Por tal razón, se encuentra en un proceso de continuo mejoramiento, debido a que las necesidades de los usuarios finales son cada día cambiantes. La finalidad de la ingeniería de software es asegurar la calidad del software a través del tiempo, por lo cual se ha convertido una de las áreas donde más esfuerzo investigativo se ha invertido. A pesar de esto, cada día se proponen nuevos retos concernientes a la ingeniería del software (Canfora & Di Penta, 2007).

El mantenimiento del software es una de las actividades más comunes para asegurar la calidad del software (Arciniegas Herrera, 2006). Y una de las técnicas por excelencia para realizar el mantenimiento, es la ingeniería inversa (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 puede ser resumida en tres aspectos fundamentales: Análisis, identificación de componentes incluida sus relaciones y representación de estos en un nivel de abstracción más alto, con el fin de obtener mejoras o nuevas soluciones a partir del análisis realizado a la arquitectura (Chikovsfky & Cross, 1990). La ingeniería inversa permite analizar el funcionamiento de un software, a partir de su código fuente, permitiendo extraer su arquitectura con el fin de comprender su funcionamiento y composición interna, lo cual permite detectar fallas y realizar mejoras al mismo.

La identificación de los componentes de la arquitectura incluida sus relaciones, es representada en una base de información mediante gráficos, bases de datos, esquemas XMI, etc. A su vez, esta base de información es representada al usuario a un nivel de abstracción

más entendible para él. Sin embargo, en muchas ocasiones la base de información representa sistemas complejos, lo que dificulta el análisis al usuario, debido a que para el ser humano es una tarea tediosa el análisis de grandes volúmenes información.

Varias herramientas como Grok, GreQL, SemmleCode, JTransformer, OMEGA, Rigi (Alves, Hage, & Rademaker, 2011) (Holt, Winter, & Wu, 2002), entre otras, cuentan con mecanismos de consulta para la base de información, pero desgraciadamente estos mecanismos no son potentes al momento de manejar sistemas complejos, tampoco tienen en cuentan al momento de realizar las consultas la generalidad de los términos en la programación, donde un término puede referirse a varias términos. Además estas herramientas centran su análisis en el código fuente del software mas no en la arquitectura, limitando los tipos de consultas y el tipo de información que se puede obtener.

Debido a lo descrito anteriormente, investigadores como Canfora, Di Penta y Cerulo (2011), proponen la creación de herramientas que permitan mejorar el proceso de análisis de la arquitectura, a partir de la realización de consultas a la arquitectura representada en una base de información obtenida en un proceso de ingeniería inversa. Estos aspectos permitieron la formulación del cuestionamiento por cual se motivó el presente trabajo de investigación: *¿Cómo realizar consultas a la arquitectura representada en una base de información obtenida en un proceso de ingeniería inversa?*

Como consecuencia de la pregunta anteriormente planteada, se tiene el objetivo principal de la investigación, basado en *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.*

Hacer frente a la problemática que representa mejorar la capacidad de consulta de la arquitectura con el fin de facilitar la capacidad de análisis, representa un avance significativo en materia de innovación e investigación, debido a que se facilitaran las tareas de

mantenimiento de software, lo cual contribuye al desarrollo de dos campos de acción de la ingeniería inversa: la producción de software y el ámbito académico.

El campo del desarrollo o producción de software se verá beneficiado debido a que gracias a las consultas realizadas a la arquitectura, se facilitará el análisis de la misma y por tanto, se verá simplificado el mantenimiento de una herramienta y al mismo tiempo se reducirán los tiempos y costos dedicados a realizar dicho manteamiento, aumentando las horas y recursos económicos disponibles para dedicar a otros proyectos. En el ámbito académico se tendrán grandes beneficios con el desarrollo de esta investigación debido que los resultados de esta, servirán para impartir conocimiento y además servirán como punto de partida para futuras investigaciones, lo que producirá un aumento en la producción científica acerca de la ingeniería inversa.

La calidad del software también se verá beneficiada, debido a que mejorará el análisis del software al momento de comparar la coherencia entre el código fuente y el diseño de este. A su vez, la seguridad e integridad de los productos se verá fortalecida puesto que el análisis permitirá encontrar posibles problemas y permitirá brindarles óptimas soluciones a estos.

Para llevar a cabo la presente investigación se clasificó como *Investigación Exploratoria* y *Aplicada* de acuerdo a sus características y fines que persigue. Se realizó un análisis minucioso de la literatura en fuentes académicas verificables y científicamente aceptadas dando buen soporte académico y científico al trabajo realizado. Inicialmente se construyó el estado del arte que permitió identificar los antecedentes de las investigaciones realizadas. El estado del arte fue analizado con el fin de identificar las necesidades en las que se debían trabajar y tratar de resolver mediante la realización de este trabajo de grado. A partir de las necesidades, se definieron los requisitos del mecanismo de consulta. Luego, estos requisitos permitieron realizar el diseño del mecanismo de consulta y finalmente se construyó un prototipo software para validar las características y funcionalidades del mecanismo implementado. La validación de este prototipo permitió identificar posibles fallas en el mecanismo y trabajos a futuro.

La presente investigación se realizó en la ciudad de Cartagena, Bolívar durante los años 2013 y 2014, teniendo en cuenta las pautas para presentar proyectos de investigación establecidas por el programa de Ingeniería de Sistemas de la Universidad de Cartagena. La investigación fue dirigida por el ingeniero Msc. Martin Emilio Monroy Ríos quien actualmente se desempeña como profesor e investigador del Programa De Ingeniería De Sistemas de la Universidad De Cartagena y adelanta estudios de doctorado en la Universidad Del Cauca.

2 OBJETIVOS Y ALCANCE

2.1 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

2.2 OBJETIVOS ESPECÍFICOS

- Construir y documentar el estado del arte sobre las herramientas de consulta a partir de la información obtenida en el proceso de ingeniería inversa

- 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 software que implemente las principales funcionalidades del mecanismo de consulta.

2.3 ALCANCE

En la actualidad, existe un amplio grupo de técnicas, metodologías, herramientas y estándares que facilitan el enfrentamiento a cada una de estas actividades de la ingeniería del software. Sin embargo, hay situaciones que se han convertido en escenarios de investigación, en las cuales es necesario realizar grandes esfuerzos. Por ejemplo: facilitar la recuperación de conocimiento de sistemas implementados con poca o nula documentación, disminuyendo la complejidad de las tareas de mantenimiento.

La presente investigación, representa un alcance de carácter exploratorio, con la posibilidad de llevar a cabo una investigación más completa y establecer prioridades para investigaciones futuras respecto al diseño de mecanismos de consultas, ya que representa la creación del modelo conceptual de un mecanismo de consulta aplicado a la base de información obtenida a partir de un proceso de ingeniería inversa, determinando inicialmente, las falencias y necesidades en los mecanismos de consultas existentes, a las cuales se les dio solución mediante la creación del modelo conceptual de un nuevo mecanismo de consulta. En este modelo conceptual, la base de información representa la arquitectura del software que es analizada; el mecanismo definido permite al ingeniero de software realizar consultas para facilitar el análisis de tipo estructural, mediante el cual se pueden identificar los elementos presentes en la base de información, como paquetes, clases y métodos; y la funcionalidad del mecanismo se limita a la obtención de información y no a la modificación de la misma, debido a que existen herramientas que ya se centran en esa funcionalidad cómo Grok y GreQl.

Las consultas que el mecanismo reconoce son las permitidas por el lenguaje de consulta estructurado XQuery, además es el ingeniero de software quien definirá las consultas que serán procesadas por el mecanismo, porque es él quien sabe que preguntas desea hacer a la base de información. La base de información está representada en el estándar XMI. Fue de mucha importancia conocer que tipos de consulta soportaba el mecanismo debido a que de acuerdo a esto, el mecanismo realizó la ejecución de las consultas. Algunas consultas se generaron de manera directa y otras se generaron por candidatos de búsqueda, lo cual permitió obtener resultados más óptimos.

El marco conceptual de la presente investigación pertenece a las ramas de ingeniería inversa y mantenimiento de software, que a su vez hacen parte del campo de la ingeniería de software. Estas temáticas, incluidos sus contenidos y herramientas, permitieron la construcción del estado de arte y por consiguiente el desarrollo del resto del trabajo de investigación. El estado del arte permitió determinar falencias y necesidades en los mecanismos de consultas existentes, y permitió definir los requisitos de cumplimiento del nuevo mecanismo.

La investigación se llevó a cabo por medio de un mapeo sistemático de literatura, realizando consultas con base a unas preguntas de investigación. Para el mecanismo creado, se utilizó el lenguaje de consulta estructurado XQuery, que permite realizar consultas basándose en el álgebra relacional. El mecanismo solo reconocerá consultas realizadas en este lenguaje.

La propuesta del modelo conceptual del mecanismo fue expresado en modelos UML. El mecanismo tiene como entrada una consulta XQuery, la cual es procesada y luego arroja como salida la respuesta dicha consulta sobre la arquitectura del sistema analizado. Gracias a la naturaleza de Xquery, la salida puede ser representada mediante XML o cadenas de texto, dependiente de cómo lo solicite el ingeniero.

El mecanismo fue validado mediante la creación de un prototipo software que permitió utilizar las funcionalidades básicas del mecanismo propuesto. El prototipo se limita a realizar las funcionalidades de simulación de las entradas, proceso y salidas del mecanismo, no realiza otras funcionalidades como generación de base de información o validación del formato de la base de información. El prototipo solo reconoce sentencias Xquery 1.0 y base de información almacenada en XMI 2.4.1. El escenario de prueba para validar el prototipo consistió en realizar el análisis a un software del cual no se contaba con la documentación. El software analizado fue desarrollado de C#, no produce impacto el hecho de utilizar bases de información resultado de procesos ingeniería inversa de software con cualquier lenguaje de programación. Lo único necesario es la versión ya mencionada pues es una de las entradas del mecanismo.

3 MARCO DE REFERENCIA

3.1 ANTECEDENTES

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).

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) (Chikovsfky & Cross, 1990).

Actualmente el ingeniero de software puede dar retroalimentación a las herramientas de ingeniería inversa con el fin de que le provean un resultado más refinado y preciso de las vistas que se pueden obtener (Tonella, Torchiano, Du Bois & Systä, 2007). 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 (Resnick & Varian, 1997).

La ingeniería inversa se ha centrado tradicionalmente en la recuperación de vistas funcionales y arquitectónicas de los artefactos de un software existentes (Rasool & Asif, 2007) y es por eso a partir de lo anterior que uno de sus objetivos principales de 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 (Canfora, Di Penta & Cerulo, 2011).

Uno de estos restos es: 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 y GReQL.

Actualmente, el reto es 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), teniendo en cuenta las limitantes generadas por las herramientas de consultas existentes.

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).

A pesar de los esfuerzos realizados, aun no se ha 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 el cual por contener XMI en su definición, 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 Msc. 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, el Msc. Martin Monroy afirma 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).

Del mismo modo se encuentra el trabajo realizado por Arciniegas en calidad de tesis doctoral, quien propone “Que- ES”, un modelo de proceso de desarrollo software evolutivo para arquitecturas, que comprende entre otros un modelo de proceso Que-ES (QPM), conformado a su vez por varios métodos, entre los cuales se resalta el de recuperación de la arquitectura (QAR), cuyo propósito es analizar sistemas ya implementados con el fin de recuperar su arquitectura. QAR sigue una dirección opuesta al flujo normal de desarrollo del software, con el fin de promover la reutilización de sistemas y activos ya existentes orientadas a servicios (Arciniegas Herrera, 2006).

En la universidad de Cartagena, también se tiene el trabajo de grado realizado por los ingenieros de sistemas Christina Issabel Arrieta Pacheco y Kenny Martínez Herrera, titulado “Evaluación de métodos de ingeniería inversa para la recuperación de vistas estáticas”, en el cual describen los métodos de ingeniería inversa para la recuperación de vistas estáticas, dichos métodos han sido detallados, con el fin de aplicar una evaluación sobre cada uno de ellos, mediante criterios de evaluación definidos con base en las características encontradas entre dichos métodos.

Asimismo, se tiene en desarrollo el trabajo de grado titulado “Marco de referencia para la selección de herramientas de ingeniería inversa”, de los estudiantes de Ingeniería de Sistemas de Universidad de Cartagena, Ricardo Andrés Cardona Alarcón y Jason David Ovalle, que pretenden elaborar un marco de referencia para la selección de herramientas de ingeniería inversa dependiendo del contexto de uso.

Necesidad de un mecanismo de consulta para el análisis de software

La posibilidad de realizar consultas es una tarea importante al momento de analizar la arquitectura de un software. Este tipo de análisis es necesarios para facilitar tareas de refactorización, identificación de bugs, identificación de patrones de diseño y para proveer una alternativa de comprender mejor un sistema. El análisis y comprensión de software son

procesos netamente humanos, donde el ingeniero de software debe adquirir los conocimientos necesarios para poseer la capacidad de cumplir exitosamente con alguna tarea, ya sea de mantenimiento, mejora, entre otros, por medio del análisis, la suposición y formulación de hipótesis. El ingeniero de software en el proceso de análisis de diseño de un sistema tiene, adicionalmente, dos objetivos principales: el primero, de comprender la arquitectura y el diseño del mismo y el segundo, de evaluar su calidad (Canfora, Di Penta, & Cerulo, 2011).

En el proceso de ingeniería inversa, uno de los objetivos importantes es el de producir vistas de software para extraer información de los artefacto disponibles y el de realizar abstracciones de este. Existen diferentes tipos de vista como las de arquitectura, las de código, las de métricas y las históricas (Canfora & Penta, 2008). En el caso del proyecto se centró simplemente en las vistas de arquitectura, en las cuales se vio necesaria la capacidad de representar el sistema en un nivel de detalle más refinado, en vez de proveer la vista completa. Estos filtros de detalles de vieron posibles por medio de un mecanismo de consulta por medio de la cual se consulta a la base de información del modelo.

El siguiente diagrama muestra el proceso de compresión de software por parte del ingeniero de software (Canfora & Penta, 2008):

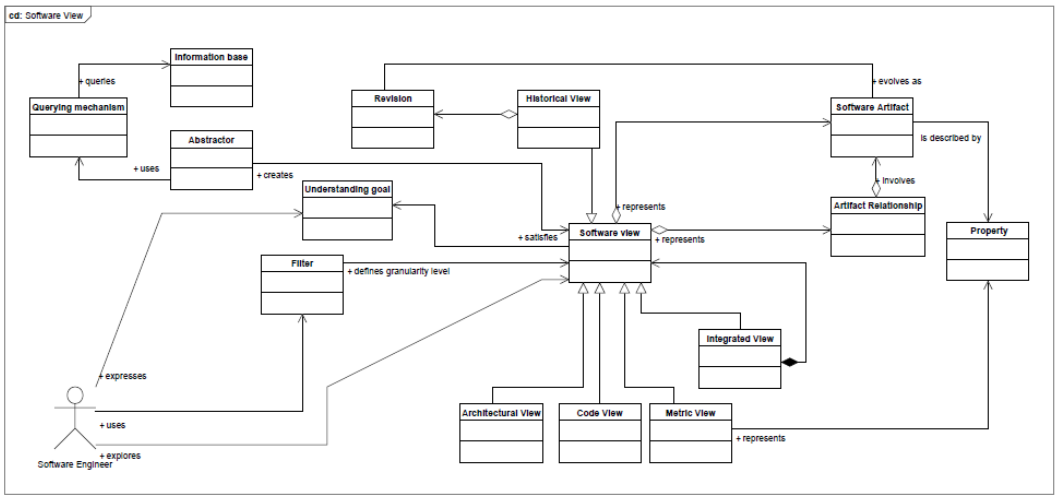


Ilustración 1 Proceso de comprensión del software por parte de ingeniero de software

Utilizar un mecanismo de consulta para el análisis de software, ofrece un nivel más profundo de abstracción sobre la arquitectura completa facilitando su comprensión y aprendizaje, logrando cumplir con el objetivo de compresión por el cual se comenzó a analizar el sistema (Canfora, Di Penta, & Cerulo, 2011), los cuales pueden ser para: satisfacer los nuevos requerimientos de los usuarios, adaptarse a las nuevas y emergentes organizaciones y modelos de negocio, enfrentarse a la innovación de la tecnología o reservar la estructura del sistema del deterioro.

Actualmente existen varias herramientas que permiten realizar consultas para el proceso de análisis de ingeniería directa, enfocándose más que todo lo que se posee en el código fuente del sistema; al igual existen herramientas para el análisis del resultado del proceso de ingeniería inversa, la cual se encuentra representada en una base de información a la cual se realizan consultas.

Existe una gran brecha entre lo que el Ingeniero de software realmente quiere encontrar y lo que realmente obtiene de la consulta que él especificó. Esto se debe a que un concepto en un sistema software puede ser representado por diferentes términos, mientras que el mismo término puede tener diferentes significados en distintas partes (SHAO & SOON, 2002), esto se presenta gracias a la libertad que se posee al momento de diseñar y desarrollar un software independiente que se utilicen o no, buenas practicas.

3.2 ESTADO DEL ARTE

El estado del arte descrito a continuación se realizó a partir de la construcción de un protocolo de mapeo sistemático de literatura, el cual permitió identificar, evaluar e interpretar un gran número de investigaciones pertinentes en base a unas preguntas de investigación definidas.

Con el mapeo de revisión de literatura se encontraron diferentes artículos de gran importancia para el trabajo de investigación, permitiendo resumir toda la información existente sobre los mecanismos de consulta y lo relacionado a estos.

La ingeniería de software implica la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software. Actualmente, existen un amplio grupo de técnicas, metodologías, herramientas y estándares que facilitan el enfrentamiento a cada una de estas actividades de la ingeniería del software (Monroy, Arciniegas , & Rodríguez, 2012).

Sin embargo, hay situaciones que se han convertido en escenarios de investigación, en las cuales es necesario realizar grandes esfuerzos. Entre estas situaciones tenemos: facilitar la recuperación de conocimiento de sistemas implementados con poca o nula documentación, disminuyendo la complejidad de las tareas de mantenimiento (Canfora, Di Penta, & Cerulo, 2011) (Canfora & Di Penta, 2007).

La recuperación de conocimiento ha sido abordada haciendo uso de la ingeniería inversa. Se han logrado importantes avances sobre todo en la recuperación de la parte estructural del sistema (Muller, y otros, 2000), sin embargo, esta disciplina aun cuenta con grandes desafíos que no han sido solucionados, tales como la capacidad de consultar la base de información para recuperar la información de la arquitectura que es de interés para el ingeniero de software y visualizar los resultados de la consulta realizada. A partir de la recuperación de conocimiento se convierte en un aspecto indispensable la visualización en el proceso de análisis de la base de información obtenida, debido a que se debe representar de una forma adecuada con el fin que sea entendible para el ingeniero de software.

La ingeniería inversa se ha centrado tradicionalmente en la recuperación de vistas funcionales y arquitectónicas de los artefactos de un software existente, y es por eso que uno de sus

objetivos principales es producir vistas de software generadas a partir de las abstracciones almacenadas en una base de conocimiento (Chikovsfky & Cross, 1990). Sin embargo, durante este proceso de recuperación de vista, hay una serie de desafíos que los ingenieros de software deben enfrentar.

Uno de estos desafíos es la capacidad de consultar la base de información que contenga datos generados por los analizadores (Canfora, Di Penta, & Cerulo, Achievements and Challenges in Software Reverse Engineering, 2011). Esto implica la necesidad de lenguajes de consultas de gran alcance que permitan la construcción de vistas informativas de los datos almacenados en la base de información.

A mediado de los 70, aparecieron los primeros enfoques de mecanismos de consulta, pero estos no implicaban un lenguaje de consulta, como las herramientas de UNIX, grep (Wellner & Dant, 1989) y awk (Aho, Weinberger, & BrianKernighan, 1988), utilizados para consulta código fuente y permitían hacer comparaciones de expresiones regulares en el texto del lenguaje de programación. Estas herramientas no permitían expresar consultas como asignación a una variable de un determinado tipo de datos.

En los años 80, empezaron a surgir las primeras herramientas de consulta de código. Estos enfoques hacían uso de los siguientes lenguajes de consulta: algebra relacional, cálculo relacional y lógica de predicados. De acuerdo a la tesis de maestría de Peter Rademaker, titulada "Binary relational querying for structural source code analysis" (Rademaker P. , 2008), a través de la historia se han tenido los siguientes mecanismos de consulta.

3.2.1 Álgebra relacional

Omega es la primera herramienta de consulta en la historia, fue realizada por Linton (Linton, 1984), la cual hace uso de SQL, un lenguaje de consulta basado en el álgebra relacional. Esta herramienta permitía almacenar la información extraída del código fuente en una base de datos relacional. Esta información almacenada era muy detallada para permitir la reconstrucción del código fuente desde la base de datos, pero este nivel de detalle aumentaba la magnitud del volumen de la información contenida en la base de datos, por lo tanto el tiempo de respuesta de la evaluación de una consulta aumentaba considerablemente.

Al utilizar el lenguaje de consulta SQL es muy ventajoso porque elimina la necesidad de implementar un nuevo lenguaje de consulta. Además, este lenguaje es conocido por una gran parte de los programadores. Desgraciadamente, SQL no permite la realización de consultas recursivas (Libkin, 2001), las cuales son esenciales para enfrentar las construcciones recursivas en los sistemas software.

CIA (Chen, Nishimoto, & Ramamoorthy, 1990), es otra herramienta de consulta propuesta, al igual que Omega almacena el código fuente en una base de datos, pero a diferencia de esta herramienta, CIA almacena solo una cantidad limitada de información, lo cual mejoró los tiempos de consulta significativamente CIA. Esta herramienta se creó para analizar muchos de los programas de producción en la Universidad de California, Berkeley. Además, fue la primera en proponer la actualización incremental de las relaciones del código fuente en la base de datos, es decir, después de realizar un cambio en el código fuente, sólo el modelo que contiene las partes que fueron cambiadas, se analiza de nuevo y los nuevos hechos se añaden a la base de datos. No es necesario cambiar toda la base de datos, sino solo se realiza el cambio en la parte afectada. Una desventaja de CIA es que algunos comandos requieren que el usuario especifique el tipo de objetos para recuperar información sobre este, pero muchas veces el tipo de objeto no es tan obvio, por ejemplo, las llamadas a macros y las llamadas a funciones se lucen de la misma forma.

Al utilizar el lenguaje de consulta SQL es muy ventajoso porque elimina la necesidad de implementar un nuevo lenguaje de consulta. Además, este lenguaje es conocido por una gran parte de los programadores. Desgraciadamente, SQL no permite la realización de consultas recursivas (Libkin, 2001), las cuales son esenciales para enfrentar las construcciones recursivas en los sistemas software. Este problema es abordado por la herramienta SCA (Source Code Algebra).

SCA es un framework propuesto por Paul y Prakash (Paul & Prakash, 1994), el cual ofrece un modelo formal de datos de código fuente, un lenguaje de consulta basada en expresiones algebraicas, y las oportunidades para la optimización de consultas. El modelo de consulta de SCA combina un alto nivel de integridad conceptual, con una potencia expresiva, clausula

transitiva. Esta última, permite una forma limitada de recursión haciendo posible la representación de consultas recursivas.

Otra dificultad fundamental que se presentaba en el diseño de sistemas para realizar consultas es la falta de modelos adecuados para representar la información del código fuente y de consultas exprés. Un modelo relacional presenta un lenguaje de consulta formal como el de OMEGA y CIA, pero este es muy débil para modelar la estructura detallada y el flujo de información que se presenta en el código fuente. Las herramientas Rigi (Muller, Orgun, Tdey, & Uhl., 1993) y SCAN (Al-Zoubi & Prakash, 1991) cuentan con modelos de representación mediante gráficos o arboles de sintaxis abstracta más sofisticados. Pero estos modelos carecen de un lenguaje de consulta de código con operadores bien definidos, dando lugar a sistemas con poca o ninguna base formal, consultas limitadas y no producen ningún margen para la optimización de consultas.

Para abordar lo anterior se propuso SCA (Paul & Prakash, 1994), como base para la construcción de sistemas de consulta de código fuente. La característica clave de este enfoque es el modelado de código fuente como un álgebra, la cual es un conjunto de estructuras matemáticas que consisten en tipos de datos (clases) y operaciones definidas en los tipos de datos (operadores). Un álgebra define un modelo para representar la información de código fuente y da un conjunto bien definido de los operadores que se pueden utilizar para realizar consultas sobre la información. El objetivo de SCA es modelar los tipos de datos en el dominio de código fuente como clases, y para diseñar las de consultas primitivas de código fuente como los operadores.

El álgebra relacional (Codd, 1970) es la base para los sistemas de bases de datos relacionales. El uso del algebra nos brinda varias beneficios como, la capacidad de proporcionar especificaciones formales para construcciones del lenguaje de consulta, la capacidad de utilizar el mismo algebra como un lenguaje potente de consulta de bajo nivel, y las oportunidades para optimizar las consultas.

Para validar la viabilidad de SCA, se diseñó e implementó un prototipo llamado ESCAPE (Paul & Prakash, 1994), el cual fue construido alrededor de un evaluador de expresiones

SCA, y ofrece una interesante perspectiva sobre cómo se pueden construir sistemas de consulta de código fuente basado en un algebra.

Luego se propuso, PQL (Jarzabek, 1998), un lenguaje de consulta similar a SQL que facilita el diseño de los Analizadores de Programas Estáticos (SPA). SPAs son herramientas interactivas que mejoran la comprensión del programa durante el mantenimiento a partir de la extracción de vistas de los programas y responden diferentes consultas acerca del mismo. PQL se centra en la flexibilidad y por lo tanto no hace ninguna suposición de la implementación del lenguaje, que extrae información desde el código fuente y representa estos hechos.

PQL es un nivel conceptual, independiente de la notación del lenguaje para especificar consultas y vistas del programa. Con PQL se pueden realizar consultas al diseño del programa global, así como búsquedas de los patrones de código. Estas consultas se responden mediante un mecanismo de evaluación de consultas integrado en un SPA. El núcleo del modelo conceptual de un SPA está formado por un programa de diseño de modelos y un PQL. Al separar el modelo conceptual de las decisiones de implementación, se pueden diseñar SPAs personalizables a las necesidades de diferentes proyectos.

PQL ofrece una interfaz de usuario para que los programadores realicen las consultas del programa. Las consultas son realizadas a partir de preguntas las cuales se responden automáticamente basándose en el análisis del código fuente, pero existen límites de acuerdo a lo que puede atender acerca de un programa basados solo en el código fuente. Algunas preguntas sobre decisiones de diseño, requisitos, conceptos del dominio de aplicaciones y vínculos entre esos conceptos y estructura de código no se pueden responder sin conocimiento adicional al código fuente (Jarzabek, 1998).

Las herramientas LaSSIE (Devanbu, Brachman, Selfridge, & Ballard, 1991) y TRANS (Kozaczynski, Ning, & Engberts, 1992) demuestran que lo anterior puede ser realizado. Estas herramientas centran la investigación en la integración análisis de código con representaciones a un nivel más alto del nivel del conocimiento del programa, tal como dominio de aplicación, los requisitos del usuario y la arquitectura del software.

Graph Repository Query Language (GReQL) (Dahm, 1997) y su sucesor GReQL 2 (Bildhauer & Ebert, 2008), al igual que Omega ofrecen un lenguaje similar a SQL. Ambas versiones utilizan una representación gráfica del código fuente, lo cual se refleja en el lenguaje de consulta: en GReQL es posible consultar tanto los nodos del grafo y las relaciones relativas de los nodos, y ofrece expresiones de ruta regulares que hacen posible la búsqueda de caminos en el gráfico.

GReQl es un lenguaje de consulta que trabaja sobre TGraphs (J. Ebert, 2008) como estructura de datos. Tgraphs son gráficos escritos, atribuidos, dirigidos y ordenados (Ebert & Franzke, 1995). Estas representaciones gráficas presentan una ventaja en la representación intuitiva de las relaciones con las entidades y los bordes. Pero estos gráficos y el enfoque de los identificadores de modelado y variables por vértices individuales y sus diferentes apariciones de bordes dedicados conducen a una representación compacta con redundancias.

Un derivado comercial del proyecto académico CodeQuest (Hajiyev, Verbaere, & Moor, 2006) académico se llama SemmleCode (Moor, y otros). SemmleCode utiliza el lenguaje de consulta .QL, similar a SQL con dos extensiones importantes: una operación de cierre transitivo, y otra orientada a objetos. .QL es especialmente adecuado para expresar las tareas de análisis del programa, en particular centrándose en las métricas. SemmleCode presenta una desventaja debido a que no almacena el flujo de control de la información por lo tanto todavía no es posible expresas los análisis que se realicen de una forma adecuada.

3.2.2 Calculo Relacional

El cálculo relacional binario (CRB) permite expresar de forma concisa y a un alto nivel las consultas de relaciones binarias. Este cálculo permite pensar en las relaciones entre elementos, incluyendo operaciones de cierre transitivo para expresar la recursividad pero este es menos expresivo que el álgebra relacional (Holt R. C., 2008).

En 1989, apareció la primera herramienta de consulta de código basada en CRB, llamada RelView, la cual se creó principalmente como una herramienta para la creación de prototipos algoritmos de grafos y facilitar las tareas de análisis del software (Abold-Thalmann, Berghammer, & Schmidt, 1989).

A mediado de los noventa aparece la herramienta Grok, la cual es una calculadora de expresiones CRB (Tarski, 1941). Su uso está destinado al análisis y manipulación de la arquitectura de software. También se tiene una implementación en Java de Grok, llamada JGrok pero esta solo se encuentra en la fase de prototipo (Holt, Winter, & Wu, 2002).

Grok incluye el operador de clausura transitiva. Además, soporta operaciones, tales como, la unión, intersección y sustracción, y es compatible con las operaciones sobre valores primitivos, incluyendo números, booleanos y cadenas. Es un lenguaje de script, el cual es algorítmico (o de procedimiento) que incluye construcciones de control estándar, tales como bucles y así como declaraciones y subprogramas simples. Grok también admite un conjunto de operaciones para modificar las bases de datos, representadas mediante gráficos, tales como, agregar, modificar o eliminar. Estas operaciones se realizan para fines de realizar ingeniería inversa, produciendo formas que son adecuadas para el entendimiento de la visión humana.

Por otra parte se tiene a RSCRIPT (Klint P. , 2005) es un lenguaje de consulta basado en CBR. Una de sus características es que soporta las relaciones n-arias. JRelCal, es un prototipo de biblioteca Java que implementa CBR, además este es una rápida implementación de RSCRIPT.

3.2.3 Lenguajes de programación lógica.

El conjunto de herramientas descritas en esta sección cuenta con lenguajes de consultas escritos o basados en el lenguaje de programación lógica Prolog (Lu & Liu, 2013). Prolog se basa en los fundamentos de la lógica, y fue creado inicialmente para el procesamiento del lenguaje natural.

Prolog es un lenguaje de programación lógica. Se basa en los fundamentos de la lógica, y fue originalmente diseñado para el procesamiento del lenguaje natural (Colmerauer & Roussel, 1993). Actualmente, es utilizado en la búsqueda de la inteligencia artificial, para la construcción de sistemas expertos, sistemas de procesamiento de lenguaje natural y sistemas de conocimiento, etc. Prolog es un lenguaje declarativo con algunas características imperativas que admite la recursividad general. Sin embargo, es un lenguaje desconocido

para un gran número de programadores y las consultas de gran tamaño suelen ser muy detalladas.

GraphLog (Consens, Mendelzon, & Ryman, 1991), hace de uso de Prolog como bases de datos y como motor de consulta, ofreciendo un lenguaje visual de consultas que permite realizar preguntas sobre el código fuente. De igual forma, ASTLog (Crew, 1997), utiliza Prolog para realizar consultas en arboles de sintaxis abstracta.

JQuery, al igual que Prolog, es un lenguaje de consulta lógica. El lenguaje de consulta JQuery se define como un conjunto de predicados TyRuBa que operan en los hechos generados a partir de árboles de sintaxis abstracta (Janzen & Volder, 2003).

Crocopat (Beyer, Noack, & Lewerentz, 2003) es una calculadora relacional, su lenguaje se basa en la lógica de predicados y es por lo tanto similar a Prolog. Una diferencia importante, sin embargo, es que no sigue el paradigma de programación lógica, es decir, es basado en la inferencia, es imperativo y ejecuta la sentencia de programa a partir de la declaración de las mismas.

CodeQuest, (Hajiyev, Verbaere, Moor, & Volder., 2005) combina las ventajas de un lenguaje como Prolog y el uso de RDBMS, mediante la compilación de registros de datos en SQL. Otro leguaje de consulta parecido a Prolog, es Datalog (Ceri, Gottlob, & Tanca, 1989), el cual no cuenta con constructores de control y estructura de datos. En CodeQuest se soluciona esto mediante la implementación del cierre transitivo, lo cual permite que se dé la recursividad en las consultas.

3.2.4 Tipos de consulta de las herramientas identificadas

Tabla 1 Tipos de consultas de las herramientas analizadas

HERRAMIENTA	TIPO DE CONSULTA
OMEGA	De acuerdo a la biografía revisada, no se encontró el tipo de consulta o ejemplos de consulta que soporta esta herramienta. Aunque si se menciona que Omega hace uso del lenguaje de consulta Algebra Relacional, por lo tanto las consultas soportadas por la herramienta seguirán la estructura de las consultas permitidas por dicho lenguaje.
CIA	Las vistas relacionales y textuales se pueden crear mediante la recuperación y procesamiento de la información en la base de datos. Se proporcionan ejemplos de tres tipos principales de la recuperación de información que soporta CIA: o Info: recuperación de información de atributos o Rel: la recuperación de la información de relaciones entre los dos dominios de objetos o View: Ver la definición de un objeto. Para cada tipo de recuperación de información, dan ejemplos para consultas en el visor de información: • INFO (es especialmente útil para encontrar el tipo de datos y la ubicación de un objeto.) o ¿Cuáles son los atributos de la función principal? o ¿Cuál es el tamaño promedio de una función? o ¿Cuál es el número de funciones Static en el programa? o ¿Qué archivo tiene el mayor número de funciones? o ¿Qué funciones en el menú principal devuelven un puntero a una librería? • REL (La información relacional es especialmente útil para la construcción de vistas gráficas y estudian las estructuras del programa.) o ¿Cuáles son las funciones llamadas por el menú principal? o ¿Qué archivos incluyen una librería? o ¿Qué función definida en una clase es referida por funciones en el menú principal? o ¿Qué funciones se refieren a una variable global y a un tipo de datos? • VIEW(Obtiene la información de localización de un objeto a partir de la base de datos del programa y luego extrae el texto del archivo de origen que contiene la definición del objeto)

SCA	• SCA permite tres puntos de vistas de consultas, el primer consiste en realizar consultas acerca de las relaciones de composición. Esto incluye consultas relacionadas con información estructural global, por ejemplo, las relaciones de composición entre entidades de programas tales como archivos, funciones, variables, tipos, etc. Las consultas también se pueden basar en la información estructural a nivel de declaración en el código fuente. • Un segundo punto de vista se basa en el flujo de recursos. Las consultas se pueden basar en la información del flujo derivado por análisis estáticos, tales como flujo de datos y análisis de flujo de control, por ejemplo, para localizar las porciones de programa, para encontrar las variables cuyos valores se ven afectados por una declaración particular, etc • Un tercer punto de vista surge de la jerarquía implícita en lenguajes de programación, por ejemplo, una sentencia while es un tipo especial de declaración. • Ejemplos de consultas: o ¿Cuáles son las funciones definidas en el analizador de una clase? o Mostrar el cuerpo de una función o Buscar el archivo que tiene el número máximo de funciones o Identificar el conjunto de todas las funciones que son directamente o indirectamente invocados por una función. o Cuales funciones en una clase son llamados por funciones del menú principal
PQL:	• En PQL, podemos consultar el diseño del programa global, así como la búsqueda de patrones de código detalle. • Ejemplos de consultas globales: o ¿Qué programas se refieren a una clase? o ¿Qué variables han modificado sus valores en una función? o ¿Qué procedimientos son llamados por un procedimiento determinado?
GreQL	Ejemplos de consultas que permite GreQL: • Encontrar todas las relaciones de generalización con origen o destino en una clase cuyo nombre contenga una cadena determinada. • Encontrar todas las definiciones de clases que definen un tipo que puede ser el tipo de argumento en una invocación de método. • Buscar todas las llamadas directas e indirectas de los métodos que se han marcado como obsoleto.
SemmleCode	Ejemplo de consultas en SemmleCode:

	• Contar el número de líneas en un paquete • Localizar todas las clases que declaran un método

Grok	Grok permite consultar entidades de software, las relaciones y las métricas y hacer transformaciones en la arquitectura y define los siguientes tres tipos de consulta:
	• Consultas de clases:
	  ○ Listar nodos cuya clase es un módulo determinado.
	  ○ Enumerar los nombres de los módulos, junto con sus identificadores.
	  ○ Lista de los bordes de una relación
	  ○ Nombres (no los identificadores) de los elementos relacionados por una relación.
	• Consultas de conectividad: Una consulta conectividad específica las conexiones necesarias entre los nodos, en términos de las relaciones en la base de datos.
	  ○ Listar los archivos de cabecera que se incluyen por exactamente dos módulos. Junto con cada archivo de cabecera, una lista de sus dos módulos incluidos. Enumere los nombres (no los identificadores) de las cabeceras y módulos.
	• Consultas de métricas: determina ciertas propiedades de medición de un sistema, por ejemplo, el número promedio de métodos por clase.
	  ○ ¿Cuantos métodos tiene cada clase?
	  ○ Calcular el número promedio de métodos por clase.

RSCRIPT	Soporta las consultas tipo calculo relacional binario
CodeQuest	• Soporta consultas SQL y tipo Plolog • Todas las consultas se dan en tres formatos diferentes: original consulta Datalog, y dos correspondientes equivalentes de SQL, primero de los cuales se basan en CTE y los segundos usan de procedimientos almacenados
RelView	De acuerdo a la biografía revisada, no se encontró el tipo de consulta o ejemplos de consulta que soporta esta herramienta. Aunque si se menciona que RelView hace uso del lenguaje de consulta Calculo Relacional Binario, por lo tanto las consultas soportadas por la herramienta seguirán la estructura de las consultas permitidas por dicho lenguaje.
JRelCal	No se encontró el tipo de consulta o ejemplos de consulta que soporta esta herramienta. Aunque si se menciona que RelView hace uso del lenguaje de consulta Calculo Relacional Binario, por lo tanto las consultas soportadas por la herramienta seguirán la estructura de las consultas permitidas por dicho lenguaje.

GraphLog	Esta herramienta uso Prolog como bases de datos y como motor de consulta, ofreciendo un lenguaje visual de consultas que permite realizar preguntas sobre el código fuente.
ASTLog	Esta herramienta de uso de Prolog como bases de datos y como motor de consulta, ofreciendo un lenguaje visual de consultas que permite realizar preguntas sobre arboles de sintaxis abstracta.
JQuery.	No se encontró el tipo de consulta o ejemplos de consulta que soporta esta herramienta. Aunque si se menciona que JQuery hace uso del lenguaje de consulta partir de la lógica de predicados, por lo tanto las consultas soportadas por la herramienta seguirán la estructura de las consultas permitidas por dicho lenguaje.
Crocopat	No se especificaba el tipo de consulta o ejemplos de consulta que soporta esta herramienta. Crocopal hace uso del lenguaje de consulta JQuery a partir de la lógica de predicados, por lo tanto las consultas soportadas por la herramienta seguirán la estructura de las consultas permitidas por dicho lenguaje.
JTransformer	No se encontró el tipo de consulta o ejemplos de consulta que soporta esta herramienta. Aunque si se menciona que JTransformer hace uso de la lógica de predicados, por lo tanto las consultas soportadas por la herramienta seguirán la estructura de las consultas permitidas por dicho lenguaje.

3.2.5 Tipos de lenguaje de consulta de las herramientas identificadas

Tabla 2 Tipos de lenguaje de consulta de las herramientas analizadas

HERRAMIENTA	TIPO DE LENGUAJE
OMEGA	Algebra relacional
CIA	Algebra relacional
SCA	Algebra relacional
PQL	Algebra relacional
GreQL	Algebra relacional
SemmleCode	Algebra relacional
Grok	Calculo relacional binario
RSCRIPT	Calculo relacional binario
LaSSIE	Algebra relacional
CodeQuest	Logia de predicados
RelView	Calculo relacional binario
JRelCal	Calculo relacional binario

GraphLog	Lógica de predicados
ASTLog	Lógica de predicados
JQuery.	Lógica de predicados
Crocopat	Lógica de predicados
JTransformer	Lógica de predicados

3.3 MARCO TEÓRICO

3.3.1 Mecanismo

Un mecanismo según la Real Academia Española, es un proceso, y como todo proceso tiene una entrada y una salida (Española, 2014).

3.3.2 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 (Pressman, 2002): 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 además de análisis y diseño orientados a objetos.

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 y retiro

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

3.3.3 Ingeniería inversa

La ingeniería inversa es el proceso de análisis de un sistema para (Chikovsfky & Cross, 1990): identificar los componentes de un sistema y su interrelación y para 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 de código fuente, estos últimos, son considerados como Herramientas CASE o Especializados en Recuperación de Diseño (Monroy, Arciniegas & Rodríguez, 2012).

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, s.f.), SrcML (SDML, 2012), 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 (2007) 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.

3.3.4 UML: Lenguaje de Modelado Unificado

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, 2005):

- 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. Luego, estos modelos sufren modificaciones con el fin de perfeccionarlos durante el proceso de desarrollo del software (Lange, Chaudron & Muskens, 2006). La importancia de los modelos ha incrementado con la llegada de la metodología de la arquitectura basada en modelos (Object Management Group, 2013) .

3.3.5 XMI: XML Metadata Interchange Specification

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) (Object Management Group, 2009)
- UML (Unified Modeling Language) (Object Management Group, 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, 2013), un lenguaje de modelado, como UML se define como un modelo en el lenguaje MOF (Alanen & Porres, 2005).

MOF, es la base del entorno estándar de industria de OMG donde los modelos pueden ser exportados desde una aplicación, importados en otra, transformados a través de una red, almacenados en un repositorio y luego recuperados, mostrados en diferentes formatos (incluyendo XMI, formato estándar basado en XML de OMG para modelo de transmisión y de almacenamiento), transformado, y usado para generar código de aplicaciones (Object Management Group, 2012).

Debido a que el metamodelo UML es una instancia del modelo de MOF y 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).

Cuando se trabaja con XMI se tienen los siguientes beneficios (Grose, Doney & Brodsky, 2002) (Gutiérrez, Escalona, Mejías, Torrres & Villadiego, 2005):

- 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.
- Al ser basado en XML permite realización de consultas internamente por medio de cualquier lenguaje de consulta de XML.

3.3.6 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 (World Wide Web Consortium, 2013) (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 (World Wide Web Consortium, 2013), pero XQuery funciona en una amplia variedad de entornos, con y sin bases de datos o servidores de aplicaciones (Robie, 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, 2007). Diversas expresiones utilizadas en XQuery 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 potentes capacidades de consulta y además es simple, flexible y fácil de implementar (World Wide Web Consortium , 2010).

3.3.7 Métodos de búsqueda inteligente para mantenimiento de software

En la busqueda de información relevante para la creación del mecanismo de consulta, se encontraron unos metodos de busqueda inteligente, basados en un articulo con en mismo nombre[1], realizado por Huixiang Liu y Timothy C. Lethbridge de la universidad de Ottawa en Canada y patrocinado por la CSER (Consorcio para la investigación en ingeniería de software) además de ser apoyado por la NSERC[2] y Mitel Corporation[3] .

Para la elaboración de los metodos se consideró el hecho de que la realización de consultas es una de las tareas más importantes para poder cumplir con las labores de mantenimiento de un software (Lethbridge end Singer, 2001) ya que por medio de ellas se facilita la comprensión de un sistema. Sin embargo, "existe una gran brecha entre lo que las personas quieren buscar y lo que realmente especifican en la consulta, debido a que un concepto en un sistema software puede ser representado por diferentes terminos, mientras que el mismo termino puede tener diferentes significados en diferentes lugares" (Liu & Lethbridge, 2002). Esto se debe a que muchos de los nombres de clases, metodos y elementos que intervienen en el desarrollo de software, son abreviaciones o transformaciones del concepto que representan, por lo que los ingenieros de software tienen que adivinar como especificar su busqueda y frecuentemente tienen que consultar repetidamente antes de obtener lo que ellos quieren encontrar .

Para darle solución a la problemática mencionada anteriormente, los autores utilizaron algoritmos de concatenación y expansión de apreviaciones para la generación de candidatos de busqueda que permitieran encontrar un conjunto de resultados más utiles de los que se encontrarian con la consulta original solamente.

Algoritmo de generación de candidatos de búsqueda

El algoritmo de generación de candidatos de búsqueda consta de la utilización de múltiples algoritmos de concatenación y expansión de abreviaciones. Se tiene en cuenta que para el

[1] Intelligent Search Methods for Software Maintenance
[2] The Natural Sciences and Engineering Research Council of Canada : http://www.nserc-crsng.gc.ca/
[3] Empresa de telecomunicaciones : http://co.mitel.com/

algoritmo el término QueryString se refiere a cada uno de los términos relevantes en una consulta y estos se pueden dividir en dos categorías:

1. Términos multi-palabras: Son cadenas concatenadas con diferentes delimitadores como guiones, caracteres especiales o con una mayúscula en una secuencia de minúsculas. Por ejemplo, el término "ejecutar-inicializador" el cual posee dos cadenas, ejecutar e inicializador, concatenadas por medio del delimitador "-".
2. Términos sencillos: Son cadenas con ningún delimitador. A pesar de que pueden ser palabras sencillas como "realizar" o "llamado", puede darse el caso de ser una cadena con varias palabras concatenadas pero con ningún delimitador identificado como "realizarllamado" o "ejecutarinicializador".

Los pasos para la generación de candidatos de búsqueda son los siguientes:

- *Paso 1: Separación de palabras*

Se comienza la primera división de las palabras cuando se encuentran los delimitadores más comunes, espacios y guiones bajos, luego se realiza la separación de palabras cuando se encuentra una mayúscula en una secuencia de minúsculas, iniciando la palabra por el delimitador y se finaliza separando donde se encuentren números o caracteres especiales. Los resultados se agregan a la *lista de palabras*.

Para evaluaciones posteriores se crea una *lista de términos relacionados* para almacenar todos los conceptos semánticamente relevantes que se encuentran durante el procedimiento de generación de candidatos de búsqueda. Inicialmente se agrega en esta lista el valor del query string.

Por ejemplo: teniendo el query string: "RealizarBusqueda_PorUsuario".

1. Se separa el query string cuando se encuentran delimitadores como espacios y guiones.

Tabla 3 Separación del query string cuando se encuentra delimitadores como espacios y guiones

DELIMITADOR	"_"
RESULTADO	"RealizarBusqueda"," PorUsuario"

2. Se separa el resultado del proceso anterior cuando encuentra una mayúscula en una secuencia de minúsculas.

Tabla 4 Separar el resultado cuando encuentra una mayúscula en una secuencia de minúsculas

RESULTADO ANTERIOR	"RealizarBusqueda", "PorUsuario"
DELIMITADOR	Mayúscula en secuencia de minúsculas
RESULTADO	"realizar", "busqueda", " por", "usuario"

3. Se realiza separación por caracteres especiales y números.

Tabla 5 Separación por caracteres especiales y números

RESULTADO ANTERIOR	"realizar", "busqueda", " por", "usuario"
DELIMITADOR	No se encuentra
RESULTADO	"realizar", "busqueda", " por", "usuario"

4. Se agregan los resultados a la lista de palabras y se agrega el query string a la lista de términos relacionados.

Tabla 6 Lista de palabras

LISTA DE PALABRAS
"realizar"
"busqueda"
" por"
"usuario"

Tabla 7 Lista de términos relacionados

LISTA DE TÉRMINOS RELACIONADOS
"RealizarBusqueda_PorUsuario"

- *Paso 2: Remover palabras no significativas*

Se verifica cada ítem de la *lista de palabras* contra un diccionario de palabras no significativas y se dividen en dos listas, si es una palabra no significativa se agrega a la *lista de palabras no significativas* y si no se agrega a la *lista de palabras clave*.

Entre las palabras significativas se tienen las preposiciones.

Por ejemplo: teniendo el query string: "RealizarBusqueda_PorUsuario" y la siguiente lista de palabras.

Tabla 8 Ejemplo de lista de palabras

LISTA DE PALABRAS
"realizar"
"busqueda"
" por"
"usuario"

Se evalúa cada miembro de la lista de palabras y se identifican las palabras no significativas. En el query string presentado se tiene la preposición "por", el cual es un término no significativo. Los otros términos se agregan a la lista de palabras clave.

Tabla 9 Lista de palabras clave

LISTA DE PALABRAS CLAVE
"realizar"
"busqueda"
"usuario"

Tabla 10 Lista de palabras no significativas

LISTA DE PALABRAS NO SIGNIFICATIVAS
" por"

- *Paso 3: Concatenación de abreviaciones por el query string original (ABBR).*

Los siguientes algoritmos de concatenación de abreviaciones tienen como objetivo producir nuevos candidatos de búsqueda concatenando abreviaciones de cada palabra clave.

Paso 3.1: Calcular el acrónimo (ABBR_ACR).

En este paso los candidatos de búsqueda se generan a partir de la unión de las iniciales de cada palabra de la lista de palabras y de la unión de las iniciales de cada palabra de la *lista de palabras clave*. El resultado se agrega a la *lista de candidatos de búsqueda*.

Por ejemplo: teniendo el query string: "RealizarBusqueda_PorUsuario" y la lista de palabras y palabras clave obtenidas de los pasos anteriores

1. Se obtienen los candidatos de búsqueda de las iniciales de las palabras de las dos listas

Tabla 11 Candidatos te búsqueda ABBR_ACR

CANDIDATOS DE BÚSQUEDA ABBR_ACR
Rbpu
Rbu

Paso 3.2: Concatenar palabras clave en una expresión regular (ABBR_KEY).

Los nuevos miembros de la lista de palabras serán resultado de la expresión regular de la forma:

$$palabraClave(1) * palabraClave(2) * \dots * palabraClave(N) * palabraClave(N+1)$$

Donde * significa cualquier cadena y el resultado se agrega la *lista de candidatos de búsqueda.*

Siguiendo con el ejemplo, se forma la expresión regular con los elementos de la lista de palabras clave y el candidato de búsqueda generado a partir del algoritmo ABBR_KEY es:

Tabla 12 Candidatos de búsqueda ABBR_KEY

CANDIDATOS DE BÚSQUEDA ABBR_KEY
realizar*busqueda*usuario

Paso 3.3 Concatenar las abreviaciones de las palabras clave en un patrón (ABBR_CUT).

El algoritmo calcula una lista de abreviaciones por cada ítem de la lista de palabras. Por cada abreviación se generan nuevos candidatos de búsqueda siguiendo la siguiente formula:

$$abreviacionPalabraClave(1) * abreviacionPalabraClave(2) * \ldots \\ * abreviacionPalabraClave(n) * abreviacionPalabraClave(n+1)$$

Para la generación de abreviaciones se siguen los siguientes dos pasos, por cada palabra clave, para generar las abreviaciones de cada ítem de la lista de palabras clave:

Paso 1: Eliminar las vocales de la palabra clave, excepto si la primera letra de la palabra es una vocal.

Paso 2: Obtener abreviaciones de las primeras letras de la palabra clave y la palabra clave sin vocales, eliminando cada vez un carácter del final y agregándolo a la lista de abreviaciones de la palabra clave. El tamaño de las abreviaciones lo definen en el artículo como de 2 a 6 caracteres pues se considera un tamaño razonable como punto de partida.

Por ejemplo, teniendo en cuanta las palabras clave realizar, búsqueda y usuario, se obtendrían los siguientes resultados.

1. El primer resultado se obtiene de eliminar todas las vocales de la palabra, excepto si la primera letra de la palabra es una vocal de lo cual se obtendrían las abreviaturas.

Tabla 13 Abreviaturas de palabras clave

ABREVIATURAS DE PALABRAS CLAVE	
Para KW[4] = "realizar"	Rlzr
	Bsqd
Para KW = "usuario"	Usr

[4] KW: KeyWord o palabra clave

2. El segundo paso es crear abreviaciones de la palabra sin vocales y de la palabra completa con tamaños de 6 hasta 2 caracteres.

Tabla 14 Abreviaturas de palabras claves sin vocales

ABREVIATURAS DE PALABRAS CLAVE		
Para KW = "realizar"	rlzr	realiz
	rlz	reali
	rl	real
	re	rea
Para KW = "busqueda"	bsqd	busque
	bsq	busqu
	bs	busq
	bu	bus
Para KW = "usuario"	usr	usuari
	us	usuar
	usu	usua

3. Por último se realiza la combinación de los elementos, formando las expresiones regulares para generar los nuevos candidatos de búsqueda.

Tabla 15 Expresiones regulares para generar los nuevos candidatos de búsqueda

CANDIDATOS DE BÚSQUEDA			
Rlzr*bsqd*usr	Rlz*bsqd*usr	Rl*bsqd*usr	…
Rlzr*bsqd*us	Rlz*bsqd*us	Rl*bsqd*us	…
Rlzr*bsqd*us	Rlz*bsqd*us	Rl*bsqd*us	…
…	…	…	…

Se generan los candidatos de búsqueda y se descartan esos candidatos de búsqueda duplicados (las expresiones regulares en negrita).

Paso 3.4 Buscar cada uno de las palabras clave (ABBR_SNG)

En este paso se buscan todas las entidades que contengan por lo menos alguna de las palabras clave.

Con el ejemplo que se ha trabajado se tendrían los siguientes candidatos:

Tabla 16 Búsqueda de palabras claves

CANDIDATOS DE BÚSQUEDA
realizar
busqueda
usuario

- *Paso 4: Expansión potencial de abreviaciones (EXP)*

Cuando se finaliza el paso 1 y la *lista de palabras* solo posee un elemento, quiere decir que no se encontró ningún delimitador definido en el query string para realizar la división de palabras, por lo cual los pasos siguientes no serían convenientes pues requieren que la lista de palabras posea más de un elemento para que los algoritmos sean aplicables.

Ejemplos de este tipo de query string que darían un solo elemento después de realizar la división de palabras son: "realizarbusquedaporusuario", "conexiondb", "ejecutordeconsultas", etcétera. Aunque los ejemplos dados son una combinación de palabras con ningún delimitador, también se puede dar el caso que se busque una palabra exacta que no necesite delimitadores como el caso de "usuario", "consulta", "helper", etcétera. Los autores asumen que, para que sea aplicable este conjunto de algoritmos de expansión de abreviaciones, el query string es una combinación de abreviaciones con múltiples palabras con ningún delimitador.

Los siguientes pasos son los algoritmos de expansión para encontrar nuevos candidatos de búsqueda:

Paso 4.1 Buscar por de la expresión regular que contiene el query string completo (EXPN_WHL)

El candidato de búsqueda se obtiene de crear una expresión regular de la forma:

$$* \; queryString \; *$$

Por medio de la cual se obtendrán todos esos resultados que posean el query string completo.

Si se tiene, por ejemplo, el query string "conexiondb", la expresión regular sería "*conexiondb*" y los posibles resultados de realizar esta búsqueda en un modelo serían: "realizar**conexiondb**", "**conexiondb**embebida", "finalizar**conexiondb**", entre otros.

Tabla 17 Query String

CANDIDATOS DE BÚSQUEDA
conexiondb

Paso 4.2 Cortar las abreviaciones delante de las vocales (EXPN_VOW)

En este paso se corta la palabra delante de una vocal, ya que ellas normalmente siguen a una consonante, y se forma una expresión regular para agregar a la lista de candidatos de búsqueda.

Si se tiene como ejemplo el query string "conexiondb", el candidato de búsqueda que se obtendría sería:

Tabla 18 Candidato de búsqueda luego de cortar la palabra delante una vocal

CANDIDATOS DE BÚSQUEDA
c*on*ex*i*ondb

Paso 4.3 Cortar la abreviación después de cada consonante (EXPN_CON):

Se forma la expresión regular de la misma forma que se presentó en el paso anterior, pero en este caso se cortara después de cada consonante.

Siguiendo con el ejemplo del candidato de búsqueda "conexiondb", el candidato de búsqueda que se obtiene de esta operación sería:

Tabla 19 Candidato de búsqueda después de cortar la palabra delante una consonante

CANDIDATOS DE BÚSQUEDA

c*on*ex*ion*d*b*

Paso 4.4 cortar la abreviación utilizando una lista de acrónimos (EXPN_ACR).

Debe existir un diccionario de acrónimos con todas esas palabras comunes al desarrollo de software, con sus respectivos acrónimos como por ejemplo: la palabra "estándar" con su relativo en inglés "standard" comúnmente se reduce a "std", también existen acrónimos comunes como "db" para database, "dto" para cuando se utiliza con data transfer objects, "dao" cuando se trabaja con el patrón DAO, entre otros.

Se busca entre el diccionario de acrónimos todos esos que aparezcan en el query string y se forma la *lista de acrónimos*. Estos acrónimos se utilizan para obtener nuevos candidatos de búsqueda, dividiendo el query string antes y después del acrónimo y generando una expresión regular.

Si se tiene por ejemplo la palabra "listdbg" y en nuestro diccionario de acrónimos tenemos "db" y ""std los nuevos candidatos de búsqueda resultantes serían:

Tabla 20 Abreviación cortada por lista de acrónimos

CANDIDATOS DE BÚSQUEDA
list*db*g
li*std*bg

Paso 4.5 Buscar por cada palabra clave (EXPN_KEY)

Un problema que se presenta para aplicar este paso es que en este momento la lista de palabras clave solo posee un elemento. Para llenar la lista se buscan todas esas partes que se encuentren en los candidatos de búsqueda, exceptuando las que tengan un tamaño muy pequeño como uno o dos caracteres. Cuando se obtengan las palabras clave, se crean expresiones regulares de la siguiente forma y se agregan a la lista de candidatos de búsqueda:

$$* \, palabraClave \, *$$

Si se tiene el candidato de búsqueda “conexiondb” y se toman los candidatos de búsqueda “c*on*ex*i*o*ndb” y “c*on*ex*ion*d*b*”, se obtendrían las siguientes palabras clave:

Tabla 21 Lista de palabras claves

PALABRAS CLAVE
Ndb
Ion

Tabla 22 Lista de candidatos de búsqueda

CANDIDATOS DE BÚSQUEDA
* ndb*
* ion*

Paso 4.6 Cortar las palabras clave después de cada consonante (EXPN_KEY_CON)

Se realiza, como en un paso anterior, la división de cada palabra, en este caso palabra clave, después de cada consonante y se obtienen nuevos candidatos de búsqueda.

Con el ejemplo anterior se obtendrían los siguientes candidatos:

Tabla 23División de cada palabra clave después de cada consonante

CANDIDATOS DE BÚSQUEDA
n*d*b*
ion*

- *Diagrama del algoritmo*

El diagrama de actividades que representaría todo el algoritmo sería el siguiente:

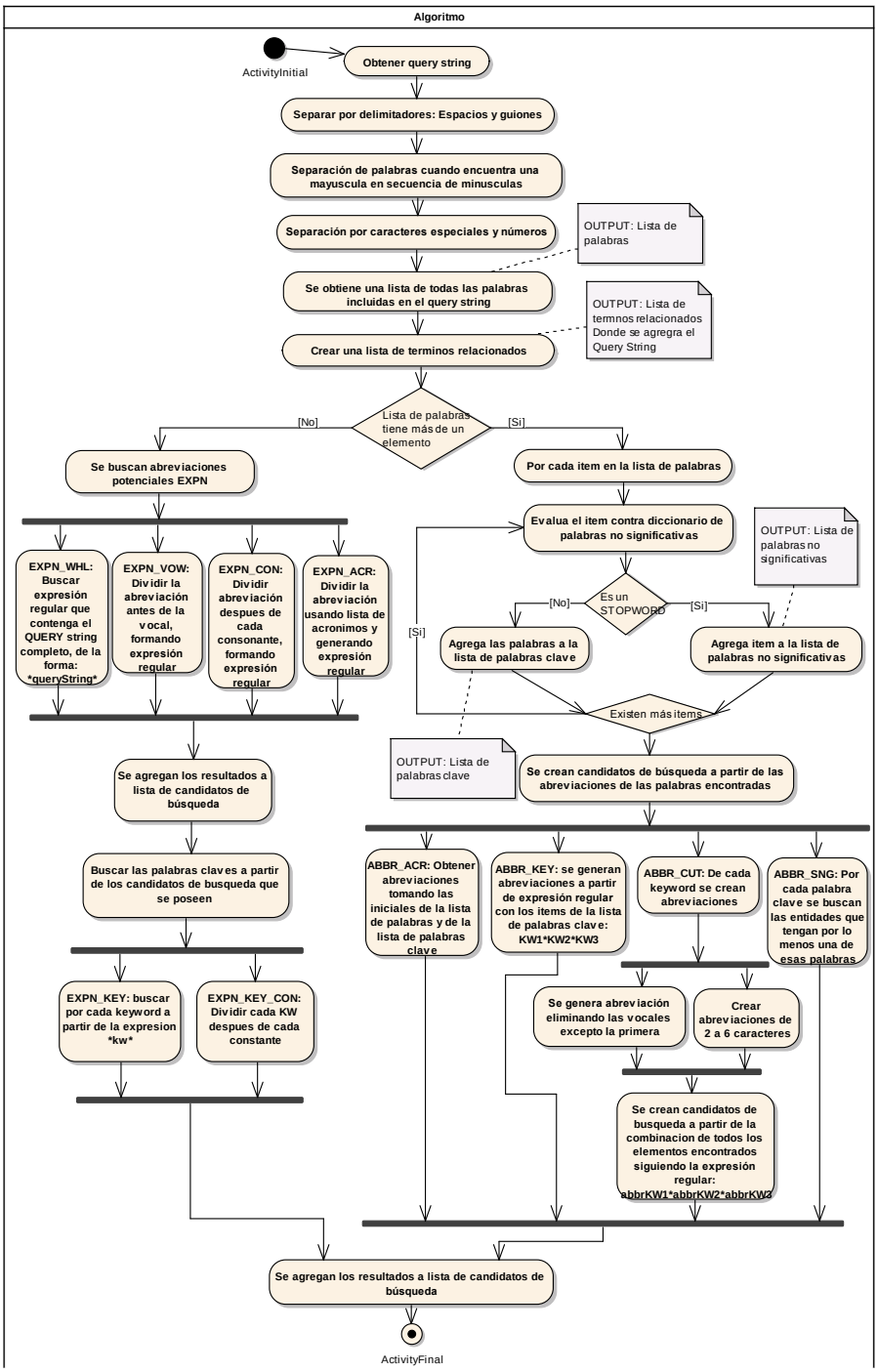


Ilustración 2 Algoritmo de búsqueda inteligente

4 METODOLOGÍA

4.1 ENFOQUE Y TIPO DE INVESTIGACIÓN

Con la finalidad de cumplir los objetivos propuestos del presente trabajo se realizó una investigación de tipo aplicada y exploratoria. Exploratoria debido a que persigue fines en los cuales no se ha trabajado lo suficiente anteriormente y tiene como fin aumentar el grado de familiaridad y contribuir soluciones a una problemática o necesidad. Además, aplicada porque dependerá de los descubrimientos y avances de la investigación básica y se enriquecerá de ellos, caracterizándose en la aplicación y utilización de los conocimientos para actuar, construir y modificar.

La investigación realizada se hizo a partir de un mapeo sistemático de literatura (K. Petersen, 2008), el cual es un tipo de estudio basado en el análisis de investigaciones previas y tiene como objetivo determinar el alcance de la investigación realizada sobre un tema de investigación específico y clasificar el conocimiento conduciendo el análisis a obtener un mapa visual del conocimiento existente dentro un amplio tema (Matturro & Saavedra, 2012).

El objetivo principal de la investigación es la construcción de un mecanismo de consulta para el análisis de la base de información obtenida en el proceso de ingeniería inversa y para proponer el diseño del modelo conceptual de este mecanismo fue necesario conocer los trabajos que se han realizado para así tener claro en que se debe mejorar y que se debe proponer. Lo anterior, se consiguió a partir de la realización del mapeo sistemático de literatura.

Los resultados obtenidos del análisis realizado a la literatura fueron validados a partir de la construcción del diseño del mecanismo de consulta y a su vez se validó este mecanismo con la construcción de un prototipo software que implementó las principales funcionalidades y características del mecanismo. El prototipo software se desarrolló a partir del mecanismo y permite realizar consultas a la base de información permitiendo el análisis del software.

4.2 PROCEDIMIENTO DE TRABAJO

Para dar cumplimiento al presente trabajo de investigación se siguió un procedimiento por objetivos específicos orientados hacia el logro de un objetivo general. Este procedimiento consistió en la planificación y desarrollo de actividades generales relacionadas directamente con cada uno de los objetivos específicos propuestos. Estas actividades generales a su vez se desglosaban en actividades más específicas denominadas sub actividades, que a su vez permitieron generar resultados entregables, los cuales fueron de gran ayuda para la redacción del informe final.

Para cubrir las necesidades de los objetivos propuestos se desarrollaron las siguientes actividades:

- ***Estado del arte***

En esta actividad se realizó un mapeo sistemático de literatura, el cual consistió en un minucioso análisis de la bibliografía correspondiente a los mecanismos de consultas existentes y uno de los retos que estas herramientas presentan: "Mejorar la capacidad para consular la información base obtenida en los artefactos generados por los analizadores" (Canfora, Di Penta, & Cerulo, 2011). El mapeo se realizó haciendo uso de las bases de datos digitales disponibles en la Universidad De Cartagena. Y a partir de este mapeo, se construyó un estado del arte completo y detallado, el cual permitió contextualizar y conocer el estado actual de la temática mencionada. El estado de arte servirá como soporte académico de los resultados del presente trabajo de investigación.

Construir el estado del arte fue una de las actividades que más tiempo tomó debido a la importancia que esta actividad amerita, esta actividad fue el punto de partida para la realización del trabajo de investigación porque permitió identificar los trabajos afines que fueron realizados y cuales faltaban por realizar, por tal razón fue necesario dedicar el tiempo y esfuerzo suficiente en su construcción.

Tabla 24 Subactividades del estado del arte

SUB ACTIVIDADES DE ESTADO DEL ARTE
Revisión de literatura
Recolección de información
Análisis de información
Elaboración de estado del arte

- ***Definición de los requerimientos para la construcción del mecanismo de consulta***

A partir del análisis y la revisión de la documentación generada en la realización del Estado del Arte, se procedió a identificar los requisitos que debe tener el mecanismo de consulta, de acuerdo a las falencias y necesidades que se encontraron en las herramientas existentes. El estado del arte sirvió como punto de partida para la realización de esta actividad porque permitió identificar los inconvenientes que se estaban presentado y los trabajos futuros a realizar.

Tabla 25 Subactividades para la definición de los requerimientos para la construcción del mecanismo de consulta

SUB ACTIVIDADES
Análisis del estado del arte
Definición de requisitos
Elaboración de informe

- ***Diseñar el mecanismo de consulta***

De acuerdo a los requerimientos definidos en la actividad anterior, en esta fase se procedió a proponer un diseño conceptual para el mecanismo de consulta, identificando sus componentes y las relaciones entre ellos y su entorno. Para la representación del diseño se utilizó un lenguaje gráfico, y se aseguró que este mecanismo posea todo los aspectos necesarios con el fin de cumplir el objetivo de esta investigación.

Tabla 26 Subactividades del diseño del mecanismo de consulta

SUB ACTIVIDADES
Diseño de mecanismo de consulta
Elaboración de informe

- ***Validación del diseño del mecanismo de consulta***

Para la validación del mecanismo de consulta se desarrolló un prototipo software que cumpliera con las funcionalidades básicas a evaluar. Inicialmente se identificaron las funcionalidades que debían ser desarrolladas en el prototipo software, a partir de esto y siguiendo la metodología de desarrollo de software RUP, se desarrollaron las fases de inicio, elaboración, construcción y transición. Por último se definió un escenario de prueba donde se pudieran aplicar las funcionalidades del prototipo para validar el mecanismo y establecer conclusiones.

En las fases de inicio y elaboración, se identificaron los requerimientos funcionales y no funcionales que debían ser desarrollados en el prototipo. También, se definió lo relacionado al modelo del negocio del software como modelo de dominio, casos de uso y diagrama de actividades, adicional se definieron los diagramas relacionados con las vistas estáticas del sistema como diagrama de clases y diagrama de paquetes. Una vez establecido qué se iba a desarrollar, se inició una investigación para determinar el uso de tecnologías aplicadas en el prototipo, entre ellos lenguajes de programación, librerías, frameworks, IDE, etc. Ya que el lenguaje de consulta que se seleccionó fue XQuery, se realizó una búsqueda de los ejecutores de estas sentencias en el lenguaje de programación seleccionado, además de buscar ejecutores de sentencias XQuery Online que facilitaran las labores de aprendizaje. Identificadas las tecnologías se realizó el diseño de la arquitectura del prototipo con el objetivo de iniciar la implementación.

En la fase de construcción se desarrolló el prototipo teniendo en cuenta la arquitectura determinada. El inicio del desarrollo se efectuó desde la capa de la lógica de la aplicación, implementando primero las funcionalidades de la lógica del negocio y siguiendo con las de

la capa de presentación, encontrándose en ella controladores y vistas. Para la implementación de los diferentes algoritmos incluidos en el mecanismo, se aplicó la práctica de programación TDD[5] o "Desarrollo Guiado por Pruebas de Software" con el objetivo de validar y verificar que los resultados obtenidos de la ejecución del algoritmo en el prototipo, sean los esperados en el flujo normal del actividades del algoritmo. Esto se realizó por medio de pruebas unitarias aplicadas a los métodos que implementaban los algoritmos. A medida que iba avanzando la fase, se iban tomando en cuenta los cambios funcionales que se le aplicaban al modelo conceptual del mecanismo de consulta, generando cambios en los requerimientos del prototipo.

En la fase de transición se realizaron pruebas funcionales al prototipo y corrección a los bugs encontrados en su ejecución. Las pruebas funcionales se efectuaron creando diferentes tipos de consulta aplicadas a un modelo XMI para evaluar las diferentes salidas que se podían obtener a partir del mecanismo implementado en el prototipo.

En la definición del escenario de prueba para la validación del mecanismo, se tomó un software llamado ***Protocol Route Connect*** desarrollado previamente como proyecto final en las materias de Seminario de actualización y de redes I dictadas por los profesores Luis Carlos Tovar y Raúl Martelo, respectivamente, para realizarle el análisis de su base de información por medio de consultas ejecutadas en el prototipo. El software seleccionado tiene como función simular el intercambio de mensajes entre nodos utilizando distintos protocolos de intercambio de datos. Para poder realizar el análisis de la base de información, el código fuente de la aplicación se importó en una herramienta que ejecutara procesos de ingeniería inversa, como Enterprise Architech 9, obteniendo los resultados en un archivo XMI. Seguido a esto, se definió la siguiente lista de preguntas que se le aplicaron al mecanismo para realizar el análisis del software. Por medio de estas preguntas se pudieron determinar a qué tipo de preguntas le da respuesta el mecanismo.

Tabla 27 Preguntas de análisis del software

PREGUNTAS DE ANÁLISIS DEL SOFTWARE

[5] Test-driven development

¿Cuáles son los nombres de los paquetes que posee el proyecto *PRC?*
¿Cuáles son los nombres de las clases pertenecientes a un paquete*?*
¿Cuáles son los nombres de los métodos que posee una clase*?*
¿Cuántas clases se poseen en el proyecto PRC?
¿Qué clases están relacionadas con un término específico?
¿Qué métodos están relacionados con un término específico?
¿Qué tipo dato retorna un método de una clase?

A partir de las respuestas obtenidas, se realizó el análisis de resultados y definición de conclusiones.

Tabla 28 Subactividades de validación del prototipo

SUB ACTIVIDADES
Identificar y seleccionar patrones arquitectónicos y de diseño que mejor se adapten a las necesidades e utilizarlos en el diseño del prototipo.
Seleccionar el tecnologías aplicables al prototipo
Definición de escenarios de prueba para el prototipo
Implementar el prototipo
Elaboración de informe

- ***Elaborar informe final sobre el mecanismo de consulta***

De acuerdo a la validación del mecanismo de consulta a partir de la implementación del prototipo software, 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. En esta fase se tendrán en cuenta los informes realizados en las actividades anteriores.

Tabla 29 Subactividades de la elaboración del informe final

SUB ACTIVIDADES
Realizar modificaciones y ajustes al mecanismo de consulta
Elaboración del informe final y entrega de resultados

4.3 TÉCNICAS DE RECOLECCIÓN Y ANÁLISIS DE INFORMACIÓN

Para llevar a cabo el primer objetivo del presente trabajo de investigación y de acuerdo a la naturaleza de la investigación que se está abordando, durante el proceso de recolección y análisis se utilizó la metodología llamada Mapeo Sistemático de la información, propuesta por los autores Maturro y Saavedra (Matturro & Saavedra, 2012). Con esta metodología, la recolección de información estuvo centrada en la revisión y análisis sistemático de la literatura. En cuanto a la fuentes de información consultadas, se seleccionaron las bases de datos digitales que facilitadas por la Universidad de Cartagena, estas bases de datos facilitaron encontrar información de tipo académico y científico.

El Mapeo Sistemático de Literatura permitió mejorar la precisión de la información encontrada y que se encontrara más relacionada con la temática del trabajo de investigación. A continuación se detalla la metodología utilizada para la recolección de información y los resultados obtenidos.

Mapeo sistemático de la literatura

Un mapeo sistemático de la literatura es una revisión de literatura que se realiza en forma previa a una revisión completa y sistemática (Matturro & Saavedra, 2012). Este mapeo tiene como objetivo determinar el alcance de la investigación a realizar sobre un tema de investigación específico, es decir, evaluar y encontrar los tipos de estudios realizados hasta el momento, esto permitirá tener claro sobre que se debe trabajar. El mapeo sistemático facilita llevar la investigación de forma organizada obteniendo resultados más pertinentes y estrictamente relacionados con los fines que se están persiguiendo.

Para llevar a cabo el mapeo sistemático se siguió el siguiente proceso teniendo como base al descrito por Marruto y Saavedra (Matturro & Saavedra, 2012):

1. Definir preguntas de investigación.
2. Realizar búsqueda de la literatura relevante.
3. Interpretación de resultados.

Paso 1: Definir preguntas de investigación.

Teniendo en cuenta las necesidades del presente trabajo de investigación se definieron 5 preguntas que abarcan las temáticas a investigar, estas preguntas fueron necesarias para obtener resultados óptimos. Las preguntas pueden ser apreciadas en la tabla 1.

Tabla 30 Preguntas de Investigación para el mapeo sistemático de literatura

N°	PREGUNTA	SECCIÓN QUE RESPONDE A LA PREGUNTA
01	¿Cuáles son los estudios existentes acerca de los mecanismos de consulta sobre una base de información?	3.2. Estado del arte
02	¿Qué tipo de consultas permiten?	3.2.4. Tipos de consulta de las herramientas identificadas
03	¿Qué lenguajes utilizan para realizar las consultas?	3.2.5. Tipos de lenguajes de consultas de las herramientas identificadas
04	¿Cuáles son los principales investigadores del tema?	3.2. Estado del arte
05	¿Cuáles son las principales revistas y conferencias donde se publican estudios sobre el tema?	3.2. Estado del arte

Las preguntas de la tabla 30 permitirán conocer el estado actual de la literatura en cuanto a la capacidad de consulta, mecanismo de consultas, lenguajes de consulta, tipos de consultas entre otros, así mismo como los autores que han tratado estos temas y las principales revistas y conferencias que han publicado estudios referentes a esto. Los resultados que arroje la investigación permitirán construir el soporte conceptual del trabajo de grado.

Paso 2: Realizar búsqueda de la literatura relevante.

Luego de definir las preguntas de investigación, en esta fase se requiere establecer las palabras claves y definir las cadenas de búsquedas a utilizar para realizar las búsquedas en las bases de datos bibliográficas. Las palabras claves deben estar estrechamente relacionadas

con la temática que se está tratando, al igual que las cadenas de búsquedas que serán formadas a partir de las palabras claves definidas.

Al momento de definir las palabras claves se deben tener en cuenta las preguntas de investigación definidas en el primer paso, con el fin de formar cadenas de búsquedas que arrojen resultados pertinentes que den respuestas satisfactorias a dichas preguntas. Las cadenas de búsquedas se formaran uniendo las palabras claves con los conectores lógicos AND y OR.

Las palabras claves que se utilizaron se pueden apreciar en la tabla 31:

Tabla 31 Palabras claves

PALABRA CLAVE
Query
Mechanism
Lenguaje
Analysis
Querying
Xmi

Al escoger las palabras claves se definieron las cadenas de búsqueda, las cuales puede encontrar en la tabla 32.

Tabla 32 Cadenas de búsqueda

CADENA	**CADENA DE BÚSQUEDA**
<cadena1>	“Xmi”
<cadena2>	“mechanism” or “language”
<cadena3>	“querying” or “query”
Cadena general: C1 AND C2 AND C3	

Esta fase también implica especificar cuáles serán los sitios de búsqueda a consultar, es decir las bases de datos bibliográficas que se van a utilizar, en las cuales se realizaran las consultas

con las cadenas de búsqueda definidas. Las bases de datos que se escogieron por su prestigio y por la suscripción que tiene la Universidad de Cartagena a estas. Las bases de datos se encuentran mencionadas en la tabla 33.

Tabla 33 Bases de datos seleccionadas para realizar el mapeo sistemático de literatura

BASE DE DATOS	DIRECCIÓN URL
SpringerLink	www.springerlink.com
ScienceDirect	www.sciencedirect.com
IEEExplore	ieeexplore.ieee.org
ACM Digital Library	portal.acm.org
Scopus	www.scopus.com
Wiley online Library	onlinelbrary.wiley.com
Citeseer	citeseerx.ist.psu.edu

Debido a los recursos económicos estipulados para la realización del trabajo de grado solo se realizaron consultas en las primeras cuatros base de datos presentadas en la tabla 33, las cuales eran las bases de datos que facilitaba la Universidad de Cartagena, por las suscripciones que tiene hasta la fecha activa. Las cadenas de búsqueda utilizadas en las bases de datos durante el proceso del mapeo sistemático se encuentran en la tabla 34.

Tabla 34 Resultados cadenas de búsqueda

BASE DE DATOS	CADENA DE BÚSQUEDA	NO. COINCIDENCIAS
ACM	"xmi" AND ("query" OR "querying") AND ("mechanism" or "language") AND "reverse engineering"	39
SciendeDirect	"xmi" AND ("query" OR "querying") AND ("mechanism" or "language") AND "reverse engineering"	60
Springer	("xmi") AND ("mechanism" OR "querying") AND ("language" OR "query")	247

IEEE	((("xmi") AND ("mechanism" OR "querying")) AND ("language" OR "query")) AND "reverse engineering")	91

- ***Interpretación de resultados***.

Como último paso, se deben interpretar los resultados obtenidos en la búsqueda realizada en las base de datos digitales. Inicialmente se definen los criterios de inclusión y exclusión, los cuales nos permitirán filtran los resultados obtenidos y seleccionando que artículos se van a tener en cuenta para el trabajo de investigación y cuáles no. La definición de criterios nos permitirá disminuir el volumen de la información obtenida en el paso anterior.

Criterios de Inclusión:

- Artículos de revistas o actas de conferencia, capítulos de libros, reportes técnicos y literatura gris publicados en cualquier fecha que presenten resultados de estudios empíricos.

Criterios de Exclusión:

- Artículos publicados en revistas o actas de conferencias no arbitradas.
- Documentos con más de seis años de publicación y sin haber sido citados.
- Documentos de opinión (Position papers)
- Documentos duplicados

Al aplicar los criterios de exclusión e inclusión se obtuvieron un conjunto de artículos descritos en la tabla 35, que proporcionaron las bases teóricas para el presente trabajo de investigación. Es necesario mencionar que además de los artículos obtenidos en el mapeo sistemático de literatura se utilizaron otros referentes distintos, los cuales se consultaron a medida que se encontraron los artículos resultados del mapeo sistemático de literatura o se encontraron en las referencias de los artículos arrojados por el mapeo

Tabla 35 Artículos obtenidos en la revisión de literatura

TÍTULO	AUTOR(ES)	APORTES ÚTILES PARA LA INVESTIGACIÓN	TRABAJOS FUTUROS
Achievements and challenges in software reverse engineering	Gerardo Canfora, Massimiliano Di Penta, Luigi Cerulo	Presenta una descripción detallada de los aspectos que se realizan durante un procesos de ingeniería inversa	Mejorar la capacidad de consulta de la base de la información obtenida en un proceso de ingeniería inversa.
Towards a Common Query Language for Reverse Engineering	Jingwei Wu, Richard C. Holt, Andreas Winter	La investigación presenta un análisis comparativo entre los mecanismos de consulta Grok y GreQl. Y a partir de este análisis proponen un nuevo lenguaje de consulta.	Los autores proponen una serie de requerimientos que consideran, debería abordar el lenguaje de consulta que proponen. Estos requerimientos se tuvieron en cuenta para definición de los requisitos del mecanismo de consulta.
Implementing relational views of programs	Mark A. Linton	El punto de interés de la investigación radica en el análisis de la herramienta OMEGA, la cual permite realizar consultas a la información de un programa representado en una base de datos. Uno de los objetivos de OMEGA es permitir la reconstrucción de objetos de software desde la base de datos del programa.	La base de datos del programa almacena información detallada acerca de elementos, relaciones, variables, etc. Lo anterior, ocasiona que los tiempos de procesamiento de estas bases de datos sean muy altos, incluso para proyectos pequeños. Por lo tanto, el principal punto a mejorar de OMEGA, son los tiempos de evaluación de las consultas.
The C Information Abstraction System	Yih-Far Chen, Michael Y. Nishimoto, C. V. Ramamoorthy	En este estudio los autores proponen el lenguaje de consulta CIA: C Information Abstract. Este lenguaje almacena la información de la	-CIA supone que todos los archivos de origen de entrada son sintácticamente correcta. '

		estructura del programa en una base de datos relacional, esto nos permite utilizar el poder de consulta sofisticado de los sistemas de bases de datos relacionales. CIA puede ser usado para estudiar los siguientes aspectos de estructuras de programas: -Subsystem: Identificar componentes independientes en un sistema grande. -Layering: topológicamente diferentes combinaciones de relaciones de referencia. -Dead code: Detectar código no utilizado en un sistema de software. -Coupling: Analizar la fuerza de unión entre pares de objetos de software. CIA permite detectar debilidades en los programas analizados. Además no requiere gran conocimiento del esquema de bases de datos o del lenguaje de consulta.	-Un usuario tiene que entender el modelo conceptual de CIA para usar los comandos de la herramienta. -Una implementación parcial del modelo conceptual completo limita nuestra capacidad para llevar a cabo varios trabajos de análisis con precisión. -Algunos comandos de InfoView requieren que el usuario especifique el tipo de objeto de un objeto para recuperar su información. Desafortunadamente, cuando el usuario está viendo un programa, el tipo de un objeto por lo general no es obvia. -Los autores proponen que se deben desarrollar herramientas para reestructurar automáticamente los programas en C para reducir el costo de compilación y mejorar la mantenibilidad.
Querying Source Code using an Algebraic Query Language	Santanu Paul, Atul Prakash	Los autores proponen SCA para dar solución a la necesidad de que no se cuenta con herramientas que cuenten con modelos de datos sofisticados y al mismo tiempo con un lenguaje de consulta semánticamente bien definido. SCA incluye operadores potentes para los objetos, conjuntos, secuencias y cierre transitivo	-Los autores mencionan que la ejecución eficiente de las consultas es un problema difícil el cual puede ser abordado a nivel de representación de programas, diseño del lenguaje y de optimización de consultas. -Además, algunas preguntas sobre decisiones de diseño, requisitos, conceptos del dominio de

		utilizados para extraer la información presente en el modelo.	aplicaciones y vínculos entre esos conceptos y estructura de código no se pueden responder sin conocimiento adicional al código fuente
Design of flexible static program analyzers with pql	Stan Jarzabek	-El autor describe la herramienta de consulta :PQL -El principio fundamental del enfoque de autor es la separación de las descripciones conceptuales de Static Program Analyzers (SPA) de las decisiones de implementación. Este enfoque ofrece tres ventajas: 1) las capacidades de la herramienta pueden ser estudiados y definidos en forma aislada a partir de los detalles de implementación de la herramienta, 2) en función de criterios tales como la eficiencia de la consulta evaluación o la simplicidad del diseño de SPA, podemos aplicar las mismas especificaciones funcionales de un SPA en una variedad de representaciones del programa, y 3) un núcleo genérico, personalizable de SPAs se puede implementar basado en los modelos conceptuales, lo que lleva a un sistema de generación de SPA.	Tenemos la intención de perfeccionar PQL con nuevas características (como vistas parametrizadas) y para trabajar en mejores interfaces para ayudar a los programadores ingresen las consultas.

LaSSIE: A Knowledge-Based Software Information System	Premkumar Devanbu Peter G. Selfridge Bruce W. Ballard Ronald J. Brachman	-El lenguaje natural se puede utilizar para formular una consulta o para reformular parte de una consulta anterior. -Los autores proponen una interfaz de lenguaje natural que permite a muchas preguntas que se formulan en inglés.	-El tiempo que toma para realizar las consultas es bastante grande.
Querying Software Abstraction Graphs Bildhauer, Daniel; Ebert, Jurgen	Bildhauer, Daniel; Ebert, Jurgen	Los autores describen el mecanismo de consulta GreQl. Las consultas de GReQL se utilizan para encontrar información en el gráfico del código fuente	-Permite realizar consultas al código fuente y no a la arquitectura del software.
Keynote address: .ql for source code analysis	Oege de Moor, Mathieu Verbaere, Elnar Hajiyev, Pavel Avgustinov, Torbj¨orn Ekman, Neil Ongkingco, Damien Sereni, Julian Tibble	SemmleCode utiliza el lenguaje de consulta .QL, similar a SQL con dos extensiones importantes: una operación de cierre transitivo, y otra orientada a objetos. .QL es especialmente adecuado para expresar las tareas de análisis del programa, en particular centrándose en las métricas. -Los autores han presentado .QL, un lenguaje de consulta orientado a objetos, el cual demostró su idoneidad para el análisis de código fuente, en particular centrándose en las métricas. -Además, su notación para los cálculos globales, ofrece una forma sencilla de expresar operaciones que serían torpe en SQL.	-SemmleCode es solo disponible bajo licencia comercial y no cuenta con una interfaz gráfica. -Permite realizar consultas al código fuente y no a la arquitectura del software.

CodeQuest: scalable source code queries with datalog	Elnar Hajiyev, Mathieu Verbaere, Oege de Moor	-Los autores desarrollaron una herramienta de consulta de código fuente eficiente y escalable, basado en Datalog y RDBMS. -La principal propuesta de los autores es: usar RDBMS para almacenar y consultar una base de código y Datalog para formular consultas.	-El mecanismo de análisis de código fuente no permite actualizar la base de datos sobre la marcha, como el usuario desarrolla su producto de software. Esto mejoraría significativamente la eficiencia de la herramienta.
Manipulation of concrete relations: The relview-system	Hilde Abold-Thalmann, Rudolf Berghammer, Gunther Schmidt	-los autores proponen a RelView como mecanismo de consulta basándose en las relaciones de los elementos.	RelView, se limita a las relaciones binarias, esto hace que el potencial de BDDs no se aprovecha plenamente
A tutorial introduction to Rscript	Paul Klint	-RSCRIPT está basado únicamente en relaciones binarias, sin embargo, existe cierto apoyo sintáctico para relaciones de n-arias traduciendolas internamente a relaciones binarias. -RSCRIPT consiste en una secuencia de declaraciones de variables y / o funciones.	Esta herramienta no tiene soporte directo para las relaciones n-arias.
Prolog with the best search	Benjie Lu, Zhingqing Liu	-Prolog se basa en los fundamentos de la lógica, y fue creado inicialmente para el procesamiento del lenguaje natural. -Prolog es un lenguaje de programación lógica. Se basa en los fundamentos de la lógica, y fue originalmente diseñado para el procesamiento del lenguaje natural.	-Prolog emplea mucho tiempo en las consultas en proyectos grandes.

Visualizing and querying software structures	Mariano Consens, Alberto Mendelzon, Arthur Ryman	- Los autores hablan sobre GraphLog , el cual reúne los dos aspectos de la consulta y visualización juntos. El sistema de software es visto como grafo dirigido. Las consultas se plantearon dibujando patrones de gráfico con un editor gráfico. - Graphlog tiene mayor potencia expresiva de SQL; En particular, se puede expresar, sin necesidad de recursión, las consultas que implican el cálculo de transitiva de operación de gráfico de recorrido similar.	-No maneja un enfoque de recursión
Simple and efficient relational querying of software structures	Dirk Beyer, Andreas Noack,, Claus Lewerentz,	Los autores se centran en el análisis de CrocoPat, el cual es suficientemente expresivo y razonablemente fácil de usar. CrocoPat también puede crear y modificar las relaciones, debido a que se centra en las relaciones	-Carece de capacidad de extracción de resultados

5 RESULTADOS

Los resultados presentados a continuación se encuentran organizados teniendo en cuenta el objetivo general y los objetivos específicos establecidos para la presente investigación. Para la construcción y documentación del estado del arte sobre las herramientas de consulta se presenta el ítem 5.1 *Necesidades encontradas en el estado del arte*; para identificar los requisitos de consulta que debe atender el mecanismo de consulta construido se presenta el ítem 5.2 *Definición de requisitos del mecanismo de consulta*; para los resultados obtenidos a partir del diseño del mecanismo de consulta se presenta el ítem 5.3 *Mecanismo de consulta para el análisis de la base de información resultado de un proceso de ingeniería inversa*; y por último se presenta el ítem 5.4 *Validación del mecanismo propuesto* y 5.5 *Análisis de resultados* para los resultados de la validación del mecanismo a través del prototipo software y análisis de resultados a partir de la validación.

5.1 NECESIDADES ENCONTRADAS EN EL ESTADO DEL ARTE

A partir del mapeo sistemático de literatura realizado fue fácil identificar los aspectos en los que se debería seguir trabajando, debido a que varios de autores proponen trabajos a futuro de sus investigaciones. Estos trabajos a futuro representan las necesidades que deben satisfacerse en cuanto a mecanismos de consulta se refiere. Abordar estas necesidades permitió contribuir conocimiento en el campo de la ingeniería inversa y mantenimiento de software.

Las necesidades encontradas a lo largo del mapeo sistemático se pueden agrupar en un solo aspecto: las consultas. Realizar consultas a la arquitectura de un software representa una de las tareas más importantes del mantenimiento del software, porque estas consultas permiten identificar falencias en el software y realizar mejoras del mismo (Paul & Prakash, 1994). Las herramientas analizadas diferían en el tipo de consultas que realizaban, estas dependían directamente del lenguaje de consulta que estas utilizaban, pero algunas de estas no contaban con un lenguaje de consulta semánticamente bien definido (Paul & Prakash, 1994).

Se encontraron tres tipos de lenguajes de consultas: Algebra relacional (Linton, 1984) (Chen, Nishimoto, & Ramamoorthy, 1990) (Libkin, 2001) (Jarzabek, 1998) (Devanbu, Brachman, Selfridge, & Ballard, 1991) (Dahm, 1997), Calculo relacional (Holt R. C., 2008) (Holt, Winter, & Wu, 2002) (Abold-Thalmann, Berghammer, & Schmidt, 1989) (Klint P. , 2005) y lógica de predicados. Por ejemplo, las herramientas basadas en el álgebra relacional, algunas solo permitían realizar consultas sobre elementos con relaciones binarias, no permitían relaciones de n-arias (Jarzabek, 1998), pero se construyeron otras herramientas que le dieron solución a este problema (Devanbu, Brachman, Selfridge, & Ballard, 1991).

Las herramientas encontradas hacen las consultas directamente código fuente representado en una base de datos (Linton, 1984) (Chen, Nishimoto, & Ramamoorthy, 1990) (Moor, y otros) o en grafos (Bildhauer & Ebert, 2008) (Holt, Winter, & Wu, 2002) (Consens, Mendelzon, & Ryman, 1991). Algunos autores proponen como trabajo futuro, que para realizar algunas consultas sobre decisiones de diseño, requisitos, conceptos del dominio de aplicaciones, entre otros aspectos, es necesario conocer información adicional al código fuente, es decir, proponen que las consultas deben ser realizadas sobre la arquitectura del software (Paul & Prakash, 1994).

En cuanto al tipo de consultas, muchas de los autores no especifican un tipo de consulta, pero los autores que si lo hacen coinciden en que deben ser básicamente estos tres (Holt, Winter, & Wu, 2002):

- Consultas globales: referentes a los atributos de los elementos que hacen parte del código fuente.
- Consultas de conectividad: Este tipo de consulta especifica pregunta sobre las relaciones entre elementos.
- Consultas de métricas: permite determinar ciertas propiedades de medición de un sistema.

Todas las herramientas encontradas suponen que las consultas de entrada son sintácticamente correctas y además de esto realizan las consultas de acuerdo a los términos especificados en la consulta (Chen, Nishimoto, & Ramamoorthy, 1990) (SHAO & SOON, 2002). Lo anterior,

presenta un problema debido a que no se tienen en cuenta la generalidad de términos que se utilizan en la programación, sin tener en cuenta que un término puede tener varios significados a la vez y que múltiples términos puede representar uno solo (Liu & Lethbridge, 2002).

5.2 DEFINICIÓN DE REQUISITOS DEL MECANISMO DE CONSULTA

Las consultas son una técnica que permite la ingeniería inversa (Wu, Holt, & Winter, 2002). A pesar de los diferentes mecanismos de consulta existentes, se tiene la necesidad de mejorar la capacidad de consultar la base de información que contiene datos obtenidos a partir de los analizadores, lo cual implica la necesidad de lenguajes de consulta de gran alcance que permitan la construcción de vistas informativas de los datos almacenados en la base de información, y así facilitar la comprensión de estos al ingeniero de software.

La Ingeniería inversa es un proceso de recuperación de la información de software y los sistemas existentes. Un proceso típico de la ingeniería inversa se compone de tres pasos: extractor, abstractor, y visualizador. En este proceso, una cantidad significativa de tiempo se gasta en la consulta de entidades de software, las relaciones y las métricas en diferentes niveles de abstracción (Panas, Lowe, & Abmann, 2003).

Muchas herramientas de consulta se encontraron, por ejemplo, Grok (Holt R. C., 1998), Rscript (Klint P. , 2003), JRelCal (Rademaker P.), SemmleCode (Verbaere, Hajiyev, & Moor, 2007), JGraLab (Ebert, Bildhauer, Schwarz, & Riediger, 2007), CrocoPat (Beyer, Noack, & Lewerentz, 2005) y JTransformer (Kniesel & Bard, 2006). Estas herramientas utilizan diferentes lenguajes de consultas que proporcionan características lingüísticas distintas.

Las anteriores herramientas de consulta están encaminadas a realizar consultas sobre la base de información del código fuente. Además, estas herramientas a pesar de realizar consultas, no llevan un proceso para tener en cuenta la generalidad de los términos que se utilizan en la programación. (Liu & Lethbridge, 2002).

De acuerdo a las necesidades presentadas en la sección 5.1. NECESIDADES ENCONTRADAS EN EL ESTADO DEL ARTE, se establecieron los siguientes requisitos para el mecanismo de consulta:

Requisito 1: Realizar consultas.

El mecanismo debe permitir realizar consultas al modelo del software representado a partir de un archivo XMI. Las consultas a realizar serán definidas por el usuario de acuerdo a sus necesidades, debido a que el será quien definirá que quiere buscar. Para realizar las consultas no se hace necesario contar con el código fuente del software. Las consultas a realizar se pueden hacer de manera de directa o aplicando los algoritmos de búsquedas inteligentes, de acuerdo a lo que el usuario requiera.

Requisito 2: La base de información a consultar debe estar representada en un esquema XMI.

La base de información a consultar debe estar representado en un esquema XMI. XMI al ser un estándar internacional basado en XML permite el intercambio de metadatos de una manera sencilla entre herramientas relacionadas con el ciclo de vida del desarrollo de software. El hecho de incluir tres estándares como XML, UML y MOF, permite a los desarrolladores de sistemas distribuidos compartir modelos de objetos y otra información. Usar XML admite una gran flexibilidad en la estructura de la información, ya que esta no se especifica como sucede en HTML, lo que permite tener más de una vista de un documento al estar separado el contenido de la estructura (Perez & Garcia). Es decir, XMI ofrece una forma fácil de empaquetar la información y la metainformación, lo cual facilita el uso y comprensión en comparación con las tradicionales tecnologías de metainformación (relacional y repositorios de objetos).

Requisito 3: Representación y visualización de resultados.

La representación del resultado de la consulta dependerá directamente del tipo de consulta que se esté realizando. La persona interesada en realizar el análisis a partir de las consultas

tendrá el poder de decidir cómo quiere representar o visualizar los resultados de sus consultas. Por ejemplo, las consultas podrán ser visualizadas como esquemas XMI, dependiendo el tipo de consulta y si el usuario lo desea.

Requisito 4: El mecanismo contará con un lenguaje de consulta estructurado y bien documentado para realizarlas.

Para la realización de consultas el mecanismo se utilizó el lenguaje de consultas XQUERY, el cual permite realizar búsquedas sobre esquemas XMI, por poseer una estructura similar a la de XML. XQUERY, es un lenguaje de consulta semiestructurado, el cual permite realizar consultas usando el álgebra relacional al estilo SQL proporcionando los medios necesarios para extraer, seleccionar y filtrar información, de una forma sencilla, de documentos XML.

Requisito 5: Implementar algoritmos de búsqueda inteligente con el fin de optimizar los resultados de las consultas.

Con el fin de obtener diferentes resultados se deben implementar algoritmos de búsquedas, los cuales permitan obtener resultados que no se obtienen si se realiza la búsqueda exacta de la consulta. Para implementar estos algoritmos se deben tener en cuenta dos aspectos importantes: obtener los términos relevantes de consulta y generar candidatos de búsqueda

Obtener los términos relevantes de la consulta.

El ingeniero al definir la consulta puede que incluya términos, caracteres especiales o números que no son lo suficiente importantes para agregarlos a la consulta que va a ser procesada por el mecanismo, debido a que son términos que no aportan lo suficiente para obtener óptimos resultados. Los términos relevantes obtenidos, son aquellos a partir de los cuales se va a obtener la información, estos podrían ser nombres de clases, métodos o de cualquier otro elemento que haga parte del código fuente del sistema.

Generación de candidatos de búsqueda para obtener más resultados útiles de los que se obtendrían con la consulta original.

Luego de que el ingeniero del software plasma lo que desea consultar, el mecanismo haciendo uso de un algoritmo de búsqueda inteligente pretende arrojar resultados que se encuentren lo

más estrechamente relacionado entre los términos relevantes de la consulta y lo que el ingeniero desea como resultado. Este proceso se hace por el hecho de que un concepto en un sistema puede ser representado por diferentes términos, debido a que muchas veces para referirse a un elemento se utilizan abreviaciones de la palabra original. Por ejemplo, el término ‘arquitectura’ puede ser representado como: ‘arquitec’, ’rqtctr’,’arq’, entre otros. La generación de estos candidatos se hará haciendo uso de algoritmos de expansión y concatenación de abreviaciones.

Finalmente, luego que se tiene la lista de candidatos de búsqueda obtenidos a partir de los términos relevantes se procede a realizar el proceso de consulta. Aunque en algunas caso el mecanismo generará las consultas sin la lista de candidatos debido a la naturaleza de la consulta.

5.3 MECANISMO DE CONSULTA PARA EL ANÁLISIS DE LA BASE DE INFORMACIÓN RESULTADO DE UN PROCESO DE INGENIERÍA INVERSA

El mecanismo propuesto se basa en el álgebra relacional y en la generación de candidatos de búsqueda para darle solución a los requisitos planteados. Se basa en el álgebra relacional pues la base de información se encuentra almacenada en un documento XMI con elementos que se encuentran relacionados estructuralmente, facilitando operaciones de selección, unión, intersección, agrupación, entre otras. El lenguaje de consulta seleccionado se basa en este tipo de algebra y brinda solución al objetivo de consulta que se planteó.

El mecanismo descrito a continuación se basa en el álgebra relacional al igual que las herramientas CIA (Chen, Nishimoto, & Ramamoorthy, 1990), SCA (Paul & Prakash, 1994), Rigi (Muller, Orgun, Tdey, & Uhl., 1993), SCAN (Al-Zoubi & Prakash, 1991) , LaSSIE (Devanbu, Brachman, Selfridge, & Ballard, 1991), TRANS (Kozaczynski, Ning, & Engberts, 1992), CodeQuest (Hajiyev, Verbaere, & Moor, 2006) , SemmleCode (Moor, y otros), entre otras, que a diferencia del mecanismo propuesto, realizaban las consultas al código fuente; adicionalmente, no poseían un lenguaje de consulta estructurado capaz de

interpretar el contenido de una base de información de una manera estandarizada, como XQuery.

Además, las herramientas no tenían en cuenta la generalidad de los términos en el desarrollo de software al momento de realizar búsquedas. Esto último fue propuesto por Huixiang Liu y Timothy C. Lethbridge de la universidad de Ottawa en Canada y patrocinado por la CSER (Consorcio para la investigación en ingeniería de software) además de ser apoyado por la NSERC[6] y Mitel Corporation[7] (Liu & Lethbridge, 2002).

De acuerdo a lo anterior, el mecanismo de consulta realizado, es la combinación de los resultados de estudios ya realizados, los cuales permiten el cumplimiento del objetivo general del presente trabajo de investigación.

Descripción del mecanismo

Todo mecanismo cuenta con tres aspectos indispensables: una entrada, un proceso y una salida.

- ***Entrada***

En el mecanismo de consulta propuesto, la entrada es proporcionada por el ingeniero de software o interesado en realizar el análisis de software, quien le brinda al mecanismo la base de información, donde se encuentran almacenados los datos relevantes del modelo del software; la sentencia XQuery donde se expresa lo que se quiere obtener de ese modelo y el modo de consulta. Este último, se explicará en los pasos posteriores.

Trabajar con el lenguaje de consulta XQuery resulta ventajoso, porque al definir la sentencia de consulta, también se puede definir la forma de representar los resultados obtenidos (Walmsley, 2007) (W3C, W3C Recommendation - XQuery 1.0: An XML Query Language (Second Edition), 2011) (W3C, 2013), ya sea como cadenas de texto o XML. La determinación de estas dos posibles representaciones de resultados fue identificada en el proceso de aprendizaje del lenguaje de consulta, basándose de la página de la W3C Schools[8],

[6] The Natural Sciences and Engineering Research Council of Canada : http://www.nserc-crsng.gc.ca/
[7] Empresa de telecomunicaciones : http://co.mitel.com/
[8] http://www.w3schools.com/xQuery/default.asp

la definición del lenguaje por la W3C[9] y por el libro Search Across a Variety of XML Data XQuery de Priscilla Walmsley (2007). Al utilizar una representación XML se tiene en cuenta todo lo que se puede crear a partir de este lenguaje, como XMI y XHTML.

- ***Proceso***

El proceso consiste en la ejecución de las consultas realizadas por el ingeniero de software. Las sentencias a ejecutar están definidas en el lenguaje de consulta estructural XQuery. El mecanismo está definido del tal manera, que permite tener en cuenta la generalidad que poseen los términos en el desarrollo de software, gracias a la aplicación de algoritmos de generación de candidatos de búsqueda, propuestos en el artículo Intelligent Search Methods for Software Maintance de la Universidad de Ottawa en Canadá. Los candidatos de búsqueda generados se agregan a la sentencia XQuery creada por el ingeniero de software con lo que se obtienen resultados más útiles que al ejecutar la consulta original.

La generalidad de los términos en el desarrollo de software, se refiere a la diversidad de opciones ofrecidas al momento de determinar los nombres de los elementos que intervienen en la creación de software. Estos elementos pueden ser clases, atributos, métodos, interfaces, paquetes, entre otros. Existen buenas prácticas para la definición de los nombres de estos elementos a nivel de forma, como que los nombres de las clases siempre deben comenzar con mayúscula, los nombres de variables deben utilizar mayúsculas y minúsculas iniciando con minúscula, los nombres que representan constantes deben estar totalmente en mayúsculas utilizando subrayados (guion bajo) para separar las palabras, que las abreviaturas y acrónimos no deberían estar con mayúsculas cuando se usan como nombre, entre otros. Además que existen buenas prácticas a nivel de contenido, que dice que el nombre que se determine a cada elemento debe describir de la mejor manera su objetivo, los nombres que representan tipos deben ser sustantivos, los nombres que representan métodos deben ser verbos, entre otros (Java, 2000). A pesar de esto, es libertad del desarrollador determinar cuál nombre le colocará a los elementos, independientes que sean o no con buenas practicas, no existe un

[9] http://www.w3.org/TR/xquery/

estándar de obligatorio cumplimiento para la definición de nombres o términos utilizados en código.

Los términos en el desarrollo de software son importantes pues pueden expresar relaciones semánticas entre los elementos, como por ejemplo, si se tienen los nombres de elementos “Consulta”, “ejecutarConsulta”, “actualizarCnslta”, “crear-cslta” y “lst-cnslts-realizadas”, se puede identificar que todos están en función de realizar algún objetivo relacionado con las consultas a pesar de que apliquen o no, buenas practicas. Los tipos de relaciones semánticas analizan la correlación que tienen las distintas palabras con su del significado, independientemente del modo en que se expresen (Real Academia Española, 2001). Realizar consultas que tengan en cuenta las relaciones semánticas y la generalidad en que se pueden expresar los términos teniendo el mismo significado, brinda mejores resultados que cuando se realiza la consulta directamente, por ejemplo, si se busca directamente lo relacionado con “consulta”, posiblemente no se obtengan todos los resultados posibles debido a la diversidad de abreviaciones que pueden existir, a menos que el ingeniero determine buscar las diferentes transformaciones que puede tener el término clave del que desea obtener información. Debido a esto, es necesaria la aplicación de los algoritmos de generación de candidatos de búsqueda para agregarlos en las condiciones del XQuery de entrada del mecanismo de consulta.

En el transcurso de la investigación y de la validación del mecanismo se identificó que la generación de candidatos de consulta no era aplicable para todos los casos, siendo aplicable por ejemplo al responder preguntas como: “¿Qué clases están relacionados con el nombre A?”, “¿Qué métodos tienen relación con el nombre A?”, “¿Qué atributos poseen el nombre A en su estructura?”, entre otras, pero siempre con la condición relacionada a un nombre, denominado termino de importancia de la consulta. Los casos en que no es aplicable la generación de candidatos de búsqueda, se dan al responder preguntas como: “¿Cuántas clases se encuentran en el modelo?”, “¿Cuáles son los nombre de todas las clases pertenecientes al modelo?”, “¿Qué métodos posee la clase con nombre A?”, “¿Qué atributos posee la clase con nombre A?”, entre otras, en estos casos se puede poseer o no términos de importancia. Debido a lo anterior, aparece el término de modo de consulta, donde el usuario debe

determinar de qué modo quiere hacer la consulta, si realizando la consulta exactamente como la indico (exacta) o teniendo en cuenta la generación de nuevos candidatos a partir término de importancia que analiza (ampliada).

A partir de las entradas anteriores se determinaron los siguientes pasos que le dan solución a los requisitos de consulta definidos en la sección anterior:

Paso 1: Validación de la consulta

El mecanismo de consulta debe validar que la sentencia XQuery se encuentra expresada de manera correcta. En caso tal, se seguirá con el siguiente paso del proceso. Si el modo de búsqueda no es exacto seguirá al Paso 2, de identificación de los términos importantes de búsqueda, y si el modo de búsqueda es exacto se dirigirá al Paso 4, de representación de resultados.

Paso 2: Identificación de los términos importantes de búsqueda

Los términos de importancia de la consulta son, en el algoritmo de generación de candidatos de búsqueda, aquellos términos clave que son de importancia para la consulta y que intervienen directamente en el posible resultado que expresen. Es el nombre del elemento que se quiere recibir información para su análisis.

Se realiza la identificación de estos términos teniendo en cuenta las condiciones que son expresadas en la sentencia XQuery.

Paso 3: Generación de candidatos de búsqueda.

Se realiza una generación de candidatos de búsqueda para obtener resultados más útiles de los que se obtendrían con la consulta original. Para poder generarlos es necesario poseer términos importantes en la consulta realizada por el usuario. Los términos importantes son esos nombres de elementos que se agregan en las condiciones de la consulta para poder recibir información. Los candidatos generados se agregan a la consulta original para ampliar las opciones de búsqueda.

La generación de candidatos de búsqueda se fundamenta el artículo de Métodos inteligentes de búsqueda para mantenimiento de software (SHAO & SOON, 2002) descrito en el marco teórico de proyecto,

Paso 4: Ejecución de la consulta

Se realiza la ejecución de la consulta especificada ya sea con o sin candidatos de búsqueda. En el caso de que se posean candidatos de búsqueda, se modifica la sentencia XQuery recibida por parte del usuario, agregándole cada uno de los candidatos de búsqueda a las condiciones de la consulta. En el caso de que no se posean candidatos, se ejecuta la sentencia XQuery tal cual como la ingresó el usuario.

- ***Salida***

La salida del mecanismo es el resultado obtenido de la ejecución de la consulta. El formato en que es presentado es definido inicialmente al determinar las entradas del mecanismo. Como se dijo previamente, las salidas pueden ser en XML o en texto.

Por ejemplo, la siguiente secuencia XQuery realiza la búsqueda de todas las clases de un modelo, definiendo diferentes salidas

Tabla 36 Ejemplo de secuencia XQuery

	ENTRADA	**SALIDA**

TEXTO	`declare namespace xmi = "http://www.omg.org/spec/XMI/20110701";`
	`declare variable $doc external;`
	`for $c in $doc//packagedElement`
	`where fn:contains($c/@xmi:type,'Class')`
	`return data($c/@name)`
		Clase1
		Clase2
		Clase3
		Clase4

XML	`declare namespace xmi = "http://www.omg.org/spec/XMI/20110701";`
	`declare variable $doc external;`
	`<resultado>{`
	`for $c in $doc//packagedElement`
	`where fn:contains($c/@xmi:type,'Class')`
		<resultado>
		<nombreClase>Clase1</nombreClase>
		<nombreClase>Clase2</nombreClase>
		<nombreClase>Clase3</nombreClase>
		<nombreClase>Clase4</nombreClase>
		</resultado>

	return <nombreClase>{data($c/@name)}</nombreClase> }</resultado>	

Como se evidencia en la tabla, al momento de realizar la entrada se determina el formato de cómo se representará la salida.

En resumen el flujo de actividades que se ejecutan en el mecanismo de consulta se evidencia en el siguiente diagrama:

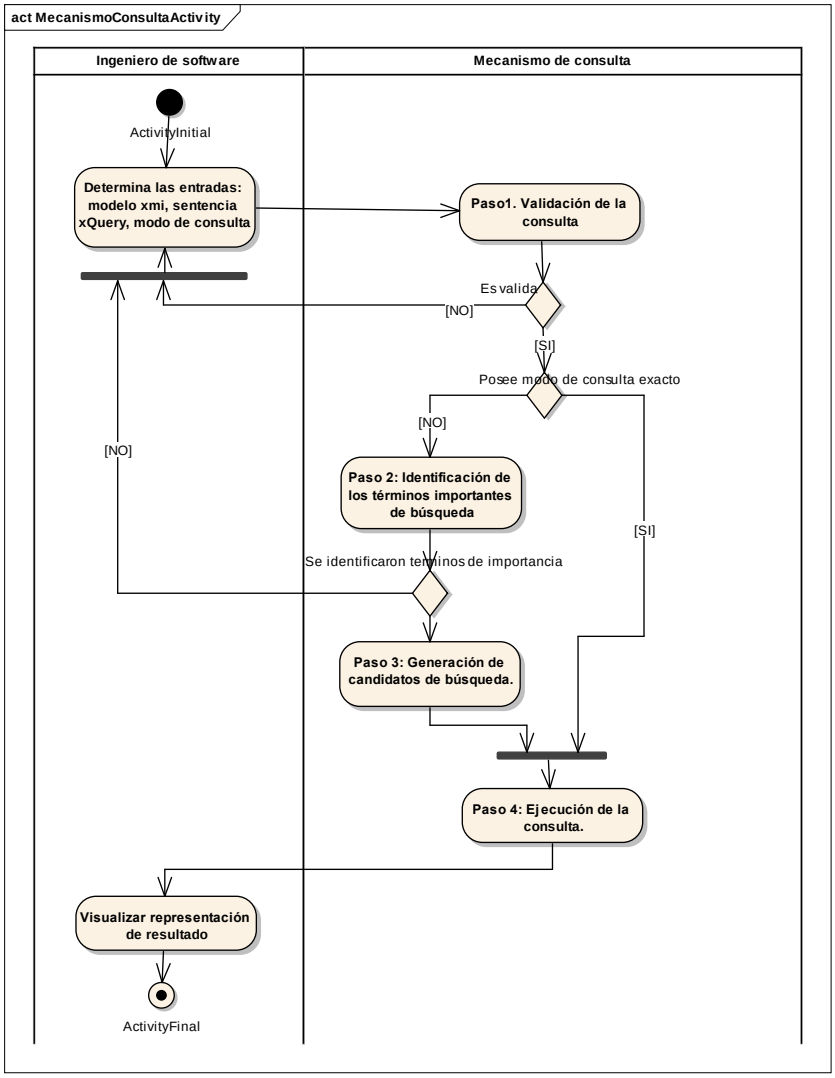


Ilustración 3 Diagrama de actividades

5.4 VALIDACIÓN DEL MECANISMO PROPUESTO

5.4.1 Descripción funcionalidades a validar en el prototipo

El resultado de esta actividad fue un prototipo software para la validación del mecanismo de consulta que permitiera realizar las siguientes funcionalidades:

1. Permitir agregar al mecanismo las entradas del proceso las cuales son:
    a. Modelo XMI
    b. Sentencia XQuery
    c. Modo de consulta
2. Realizar los pasos relacionados con el proceso del mecanismo de consulta:
    a. Realizar validación de la sentencia Xquery
    b. Extraer los términos de importancia de la sentencia XQuery.
    c. Generar candidatos de búsqueda basándose en los algoritmos de generación de los "Métodos de búsqueda inteligente para el mantenimiento de software" (Ver en marco teórico: Métodos de búsqueda inteligente para mantenimiento de software).
    d. Agregar candidatos de búsqueda al XQuery.
    e. Ejecutar consultas XQuery de modo exacto y ampliado.
3. Visualizar el resultado obtenido de ejecutar la consulta

5.4.2 Diseño del prototipo de validación del mecanismo de consulta

- ***Requerimientos del prototipo desarrollado***

Tabla 37 Requerimientos del prototipo de desarrollo

REQUERIMIENTOS		
COD	**REQUERIMIENTO**	**TIPO**
R01	El prototipo debe recibir como entrada por parte del usuario el modelo XMI al cual se le van a realizar las consultas	Funcional
R02	El prototipo debe permitirle al usuario introducir sentencias XQuery	Funcional

R03	El prototipo debe brindarle al usuario la opción de determinar si quiere realizar consultas de modo exacto o ampliado	Funcional
R04	El prototipo debe permitir la validación de sentencias XQuery	Funcional
R05	El prototipo debe permitir extraer los términos de importancia de una sentencia XQuery	Funcional
R06	El prototipo debe permitir la generación de candidatos de búsqueda para cada termino de importancia	Funcional
R07	El prototipo debe agregar los candidatos de búsqueda a las condiciones de la sentencia Xquery	Funcional
R08	El prototipo debe ejecutar consultas XQuery de modo exacto	Funcional
R09	El prototipo debe ejecutar consultas XQuery de modo ampliado	Funcional
R10	El prototipo debe visualizar los resultados obtenidos	Funcional
R11	Se debe poder visualizar el tiempo que se empleó en la ejecución de la consulta	No funcional
R12	Se deben poder visualizar los términos clave que se tuvieron en cuenta para la consulta ampliada	No funcional
R13	Debe mostrar mensajes de alerta en el caso de que se presente algún error	No funcional
R14	Se debe visualizar la sentencia que se agregó como entrada del mecanismo y la que se ejecutó realmente	No funcional
R15	Se deben poder visualizar los candidatos de búsqueda generados por cada término de importancia.	No funcional

- ***Alcances del prototipo***

Tabla 38 Alcances del prototipo

DENTRO DEL ALCANCE	**FUERA DEL ALCANCE**
El prototipo debe recibir como entrada el modelo XMI al cual se le van a realizar las consultas. Se limita al uso de la versión más actual de XMI hasta el momento, la 2.4.1.	El prototipo valida que el documento XMI este bien formado y cumpla con la estructura del estándar. Permitir reconocer la estructura de diferentes versiones del estándar XMI

El prototipo debe permitir introducir sentencias XQuery. La versión de XQuery manejada es la 1.0	Introducir sentencias Xquery con otras versiones. Introducir sentencias en otro lenguaje de consulta para XMI. Consultas en lenguaje natural.
El prototipo debe brindarle al usuario la opción de determinar si quiere realizar consultas de modo exacto o ampliado	Agregar otros tipos de modos de consulta.
El prototipo debe permitir la validación de sentencias XQuery en su versión 1.0	Mostrar sugerencias para verificar la razón del error en la validación
El prototipo debe permitir extraer los términos de importancia de una sentencia XQuery. Los términos de importancia son identificados por medio del atributo name dentro de la lista de condiciones del Xquery. Solo se reconoce la lista de condiciones que se encuentra dentro del where y el return de la sentencia.	Identificar términos de importancia en las condiciones ubicadas en otra parte de la sentencia como en el let.
El prototipo debe permitir la generación de candidatos de búsqueda para cada termino de importancia Permitir la generación de candidatos por algoritmos expansores y de concatenación de abreviaciones.	Crear nuevos candidatos de búsqueda utilizando otro algoritmo diferente al de generación de candidatos de búsqueda de los métodos de búsqueda inteligente para el mantenimiento de software.
El prototipo debe visualizar los resultados obtenidos	Realizar representación gráfica de los resultados obtenidos

- ***Diagrama de casos de uso del sistema***

En la ilustración 3, se observa el diagrama de casos de uso del sistema. Este modelo, ilustra las interacciones entre el usuario y el sistema, por lo cual, indica las funcionalidades que deben ser implementadas y sus dependencias

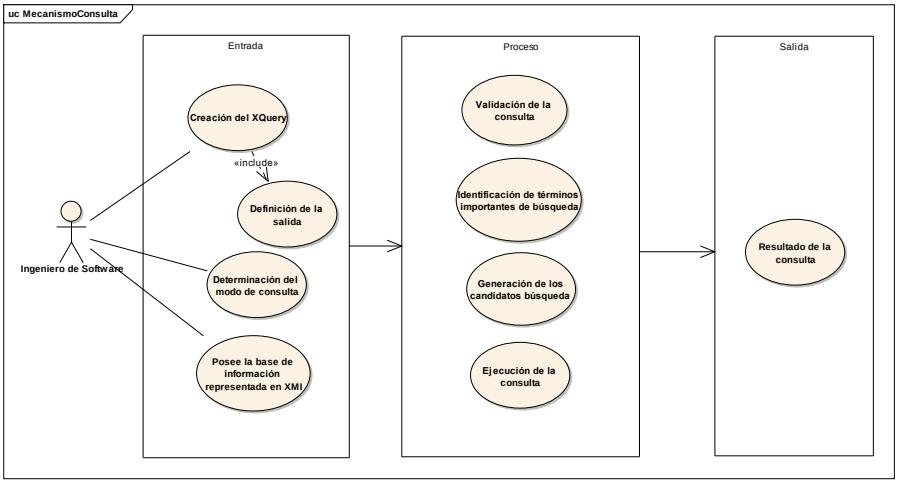


Ilustración 4 Casos de uso

- ***Descripción de la arquitectura seleccionada.***

La arquitectura seleccionada es la multicapas, derivación del paradigma arquitectónico de tres capas, que consiste en usar la filosofía de paquetes con el fin de separar las operaciones del sistema en tres grandes grupos: el primero, es el encargado de soportar las vistas o la interface gráfica de usuario; el segundo grupo maneja la lógica del negocio y los servicios prestados, también es llamado "dominio"; y por última capa, la que contiene las bases de datos y/o los archivos. En la arquitectura multicapas, el dominio es separado en subcapas con el fin de separar la lógica del negocio (transacciones), de los servicios (funcionalidades del sistema). (Larman, 2001)

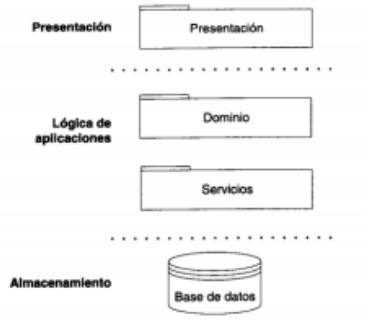


Ilustración 5 Arquitectura del software

- ***Diagramas estáticos***

En esta sección, se encuentran todos los diagramas que intervinieron directamente en el diseño del prototipo. Estos se definieron en la fase de elaboración de la metodología de desarrollo de software RUP

Diagrama de paquetes

La ilustración 5 muestra el diagrama de paquetes, el cual representa la agrupación lógica de las clases que describen el modelo del sistema desde un punto de vista estructural.

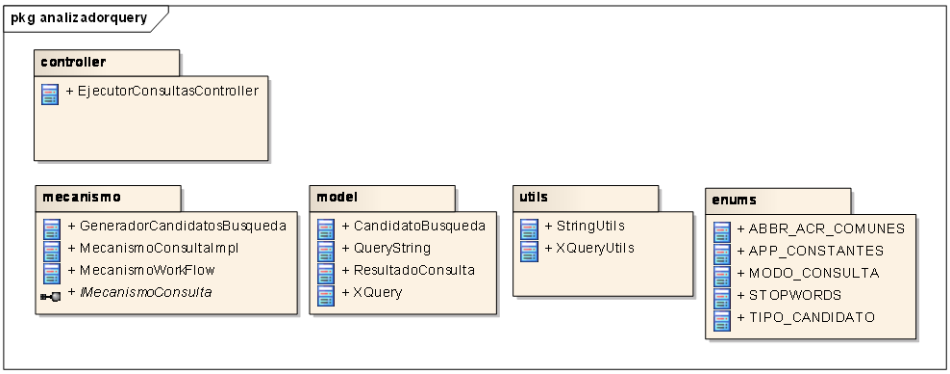


Ilustración 6 Diagrama de paquetes

Diagrama de clases

La ilustración 6 muestra el diagrama de clases del sistema, que es la representación gráfica de los bloques estructurales básicos del mismo

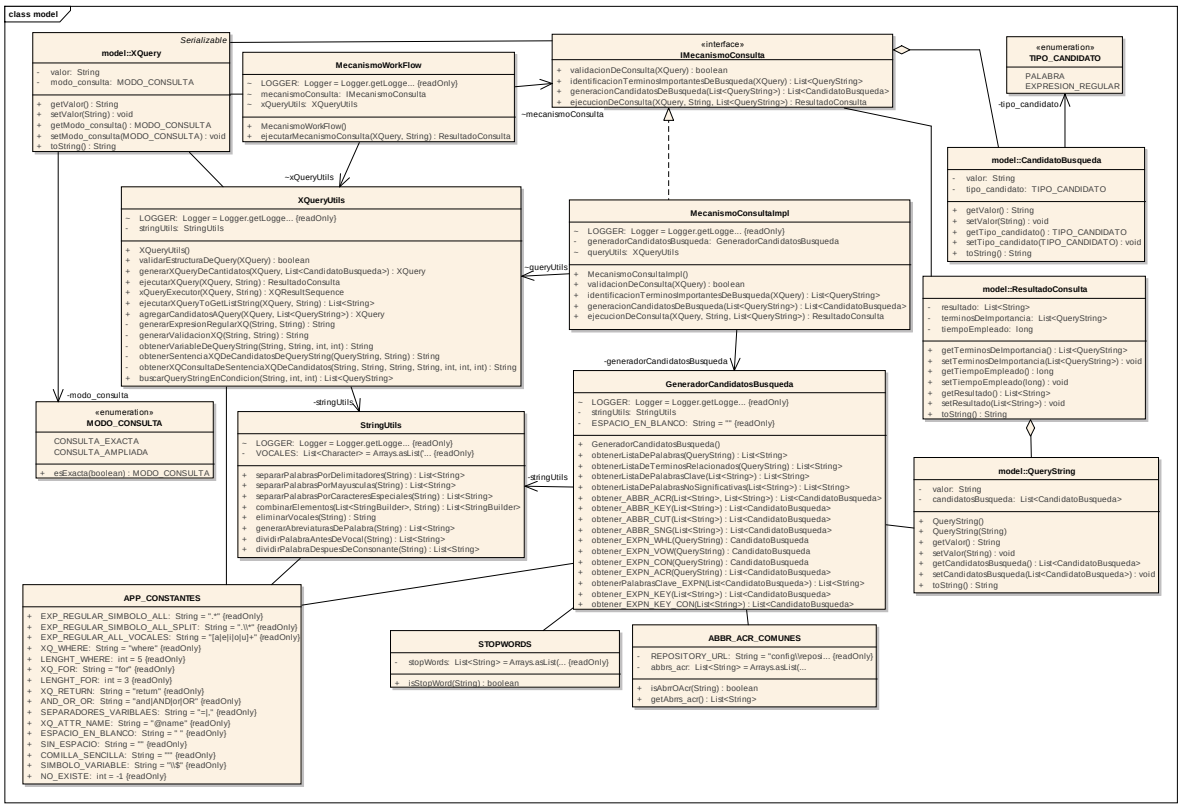


Ilustración 7 Diagrama de clases

- ***Tecnologías utilizadas en el prototipo de validación***

Las tecnologías utilizadas para la construcción del prototipo de validación del mecanismo de consulta son:

1. Java EE 7 Web Profile, como plataforma de programación: brinda todas las funcionalidades necesarias para crear una aplicación web de manera fácil y ligera, además de ofrecer la oportunidad de utilizar dependencias a librerías de fácil manejo de XQuery. Se decidió realizar una aplicación web por facilidad y conocimientos por parte de las investigadoras, además de posibilitar la creación de una interfaz agradable para la realización de la validación del mecanismo y evitar instalaciones del prototipo en cada equipo que vaya ser utilizado.
2. Saxón 9.1.0.8, como procesador de XQuery: es una librería open source que provee implementación para XSLT 2.0, XQuery 1.0, and XPath 2.0 con todos los niveles de

cumplimiento definidos por la W3C (SAXON, 2013). Además está bien documentada y con disponibilidad de ejemplos para su utilización.

3. Primefaces, como librería de componentes visuales.
4. JUnit, como framework de pruebas unitarias.

Adicional a las tecnologías, se utilizaron herramientas para la creación del prototipo, sirviendo también de apoyo en el cumplimiento de su objetivo de la validación del mecanismo de consulta:

1. Netbeans IDE 8.0, como entorno de desarrollo integrado.
2. Enterprise Architect 10, como entorno para realizar funciones de ingeniería inversa.
3. Notepad ++, como visualizador de la base de información.

A partir de todo lo anterior, se obtuvo el prototipo con los requerimientos definidos, permitiendo realizar consultas a la base de información por medio de la aplicación del mecanismo de consulta propuesto. Gracias a esto, se pudo efectuar la aplicación del escenario de prueba que permitió la validación del mecanismo de consulta y el establecimiento de conclusiones.

5.4.3 Descripción del prototipo.

La herramienta prototipo ha sido diseñada de una forma usable, reduciendo la cantidad de información y opciones mostradas en pantalla. Ésta consta de tres entradas necesarias para ejecutar el proceso relacionado al mecanismo de consulta (dirección donde se encuentra la base de información, sentencia XQuery a ejecutar y modo de consulta), un botón de ejecución, un panel que muestra la sentencia XQuery que fue ejecutada por el mecanismo y un panel de resultados o salidas del proceso del mecanismo (Ver Anexo I).

El panel de resultado muestra las salidas del proceso del mecanismo dependiendo de la forma como lo define el usuario, las cuales pueden ser cadena de texto o xml (Ver Anexo II). Si el modo de consulta es ampliado aparece la opción de visualizar los términos clave encontrados con sus candidatos de búsqueda (Ver anexo III).

5.4.4 Descripción del escenario de pruebas.

En las asignaturas de Seminario de Actualización y de Redes I dictadas en la Universidad de Cartagena en el segundo periodo del año 2012, se realizó un proyecto de aula titulado ***Protocol Route Connect*** bajo la dirección y supervisión de los docentes de la Universidad de Cartagena Luis Carlos Tovar Garrido y Raúl José Martelo Gómez y desarrollado por el grupo de trabajo conformado por los estudiantes del programa de Ingeniería De Sistemas de la Universidad De Cartagena Kevin Rafael Sarmiento Mendoza, Jaider Garcés Carrascal, Arturo Verbel de León y las autoras del presente documento Cindy Margarita Pacheco Álvarez y Alejandra Inés Ríos Rodríguez. La funcionalidad del proyecto consistió en simular protocolos de redes de telecomunicaciones, en el que es posible diseñar una red utilizando dispositivos pasivos y activos configurables, de tal manera que se asemeje al mundo real.

Para el escenario de pruebas se tomó como punto de partida que solo se tenía el código fuente y ninguna documentación en el análisis del software de simulación de redes. Por lo tanto, fue necesario aplicarle procesos de ingeniería inversa al código fuente de dicho aplicativo para obtener la base de información de su modelo. La herramienta que se utilizó para cumplir esta función fue Enterprise Architect 9, exportando los resultados del proceso de ingeniería inversa (la base de información) en la versión 2.4.1 de XMI. El resultado del proceso anterior arrojo un documento XMI con 20910 líneas de código XML.

A continuación, se enuncian los pasos del proceso a seguir durante la validación del mecanismo:

1. Se realizaron equivalentes en XQuery de las preguntas que se plantearon para el análisis del software.
2. Se realizaron las consultas en el prototipo con las entradas definidas.
3. Se tabularon los resultados obtenidos al ejecutar las sentencias XQuery con modo de consulta exacta y ampliada. Adicionalmente se almacenan los tiempos de ejecución del mecanismo en cada uno de los dos modos.

El resultado del proceso anterior se puede observar en la siguiente tabla, se respondieron las preguntas que se plantearon en la metodología aplicada, además que se colocaron nombres de clases y métodos para realizar casos específicos en el análisis. Los resultados esperados

se obtuvieron analizando el modelo del software de prueba con herramientas como Enterprise Architect, Visual Studio 2012 y NotePath++.

- ***Escenario 1 - Pregunta: ¿Cuáles son los nombres de los paquetes que posee el proyecto PRC?***

Tabla 39 Respuesta a pregunta: ¿Cuáles son los nombres de los paquetes que posee el proyecto PRC?. CE = "Consulta de monodo exacto", CA = "Consulta de modo ampliado"

CASO	XQUERY	RESULTADOS ESPERADOS	RESULTADOS		TIEMPO	
			CE	CA	CE	CA

NOMBRE DE TODOS LOS PAQUETES	`declare namespace xmi="http://www.omg.org/spec/XMI/20110701";`
	`declare variable $doc external;`
	`for $c in $doc//packagedElement`
	`where fn:contains($c/@xmi:type,'Package')`
	`return data($c/@name)`
		Archivos
		Automatas
		Auxiliares
		BGP
		Configuracion
		Dispositivos
		Inundacion
		OSPF
		Properties
		Protocolos
		RIP
		SemiRedes
		Simulacion
		Vista
			Archivos
			Automatas
			Auxiliares
			BGP
			Configuracion
			Dispositivos
			Inundacion
			OSPF
			Properties
			Protocolos
			RIP
			SemiRedes
			Simulacion
			Vista
			EA_C#_Types_Package
			EA_PrimitiveTypes_Package
				No fue posible realizar	73	-

- ***Escenario 2 - Pregunta: ¿Cuáles son los nombres de las clases pertenecientes a un paquete?***

Tabla 40 Escenario 2 - Pregunta: ¿Cuáles son los nombres de las clases pertenecientes a un paquete?

CASO	XQUERY	RESULTADOS ESPERADOS	RESULTADOS		TIEMPO	
			CE	CA	CE	CA
PAQUETE: ARCHIVOS	`declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $paquete in $doc//packagedElement where $paquete/@xmi:type='uml:Package' and $paquete/@name='Archivos' return $paquete/packagedElement[@xmi:type='uml:Class']/@name`	Archivo ManejoArchivos	Archivo ManejoArchivos	Archivo ManejoArchivos	51	102
PAQUETE: AUXILIARES	`declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $paquete in $doc//packagedElement where $paquete/@xmi:type='uml:Package' and $paquete/@name='Auxiliares' return $paquete/packagedElement[@xmi:type='uml:Class']/@name`	Funciones	Funciones	Funciones	47	95

PAQUETE: DISPOSITIVOS	`declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $paquete in $doc//packagedElement where $paquete/@xmi:type='uml:Package' and $paquete/@name='Dispositivos' return $paquete/packagedElement[@xmi:type='uml:Class']/@name`	Consola Direccion Dispositivo Puerto	Consola Direccion Dispositivo Puerto	Consola Direccion Dispositivo Puerto	64	81

- ***Escenario 3 - Pregunta: ¿Cuáles son los nombres de los métodos que posee una clase?***

Tabla 41 Escenario 3 - Pregunta: ¿Cuáles son los nombres de los métodos que posee una clase?

CASO	**XQUERY**	**RESULTADOS ESPERADOS**	**RESULTADOS**		**TIEMPO**	
			CE	**CA**	**CE**	**CA**
CLASE: MANEJOARCHI	`declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $clase in $doc//packagedElement where $clase/@name='ManejoArchivos' return $clase/ownedOperation/@name`	Abrir Crear	Abrir Crear	Abrir Crear AutomataRIP AutomataRed	49	105
CLASE: CONSO	`declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external;`	Instancia ListaDispositivos Protocolo	Instancia ListaDispositivos Protocolo	ExtraerPalabras Instancia ListaDispositivos	42	86

	for $clase in $doc//packagedElement where $clase/@name='Consola' return $clase/ownedOperation/@name	Evaluar	Evaluar	Protocolo Evaluar Dispose InitializeComponent FrmConfigurarSwitch btnGuardar_Click Dispose InitializeComponent FrmConsola ActivarInput txtConsola_SelectionChanged txtConsola_KeyDown txtConsola_PreviewKeyDown		
CLASE: PUERT	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external;	Puerto Direccion IDDispositivo	Puerto Direccion IDDispositivo	Puerto Direccion IDDispositivo	43	73

	for $paquete in $doc//packagedElement where $paquete/@xmi:type='uml:Package' and $paquete/@name='Dispositivos' return $paquete/packagedElement[@xmi:type='uml:Class']/@name	IDPuerto Conexion NodoPadre Disponible Conectar	IDPuerto Conexion NodoPadre Disponible Conectar	IDPuerto Conexion NodoPadre Disponible Conectar		

- ***Escenario 4 - Pregunta: ¿Cuántas clases se poseen en el proyecto PRC?***

Tabla 42 Escenario 4 - Pregunta: ¿Cuántas clases se poseen en el proyecto PRC?

CASO	XQUERY	RESULTADOS ESPERADOS	RESULTADOS		TIEMPO	
			CE	CA	CE	CA
PROYECTO	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; let $c := $doc//packagedElement[@xmi:type='uml:Class'] return count($c)	58	58	No fue posible realizar	68	-

- ***Escenario 5 - Pregunta: ¿Qué clases están relacionadas con un término específico?***

Tabla 43 Escenario 5 - Pregunta: ¿Qué clases están relacionadas con un término específico?

CASO	XQUERY	RESULTADOS ESPERADOS	RESULTADOS		TIEMPO	
			CE	CA	CE	CA
TERMINO: protocolo	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $c in $doc//packagedElement where fn:contains($c/@xmi:type,'Class') and $c/@name='protocolo' return data($c/@name)	Protocolo	-	Protocolo	37	64
TERMINO: msj	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $c in $doc//packagedElement where fn:contains($c/@xmi:type,'Class') and $c/@name='msj' return data($c/@name)	-	-	-	40	61
CONF	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $c in $doc//packagedElement	ConfiguracionDispositivo ConfiguracionPC	-	ConfiguracionDispositivo ConfiguracionPC	42	62

	where fn:contains($c/@xmi:type,'Class') and $c/@name='conf' return data($c/@name)	ConfiguracionRouter FrmConfigurarPC FrmConfigurarPC FrmConfigurarRouter FrmConfigurarRouter FrmConfigurarSwitch FrmConfigurarSwitch		ConfiguracionRouter FrmConfigurarPC FrmConfigurarPC FrmConfigurarRouter FrmConfigurarRouter FrmConfigurarSwitch FrmConfigurarSwitch		

- ***Escenario 6 - Pregunta: ¿Qué métodos están relacionados con un término específico?***

Tabla 44 Escenario 6 - Pregunta: ¿Qué métodos están relacionados con un término específico?

CASO	XQUERY	RESULTADOS ESPERADOS	RESULTADOS		TIEMPO	
			CE	CA	CE	CA

TERMINO: envioDeMensaje	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; let $c := $doc//packagedElement[@xmi:type='uml:Class'] return count($c)	EnviarMensaje mcEnviarMensaje_Click envio	-	EnviarMensaje mcEnviarMensaje_Click obtenerRouterMarcadoNoVisitadoMenor envio	39	125
TERMINO: simularMovimiento	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $c in $doc//packagedElement/ownedOperation where $c/@name='simularMovimiento' return data($c/@name)	simular	-	simular	44	127
TERMINO: ejecutarComando	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $c in $doc//packagedElement/ownedOperation where $c/@name='ejecutarComando' return data($c/@name)	EjecutarComando EjecutarComando EjecutarComando EjecutarComando EjecutarComando	-	EjecutarComando EjecutarComando EjecutarComando EjecutarComando EjecutarComando	40	129

Escenario 7 - Pregunta: ¿Qué tipo dato retorna un método de una clase?

Tabla 45 Escenario 7 - Pregunta: ¿Qué tipo dato retorna un método de una clase?

CASO	XQUERY	RESULTADOS ESPERADOS	RESULTADOS		TIEMPO	
			CE	CA	CE	CA
CLASE: Automata – METODO:	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $clase in $doc//packagedElement where $clase/@name='Automata' return $clase/ownedOperation[@name='AgregarNodo']/ownedParameter [@name='return']/@type	Nodo	EAID_97507C4E_A49A_4876_83E0_0DA972EDCDD0	EAID_97507C4E_A49A_4876_83E0_0DA972EDCDD0	47	57
CLASE: Automata – METODO: AgregarNodo	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $clase in $doc//packagedElement where $clase/@name='Automata' return let $a := $doc//packagedElement[@xmi:id = $clase/ownedOperation[@name='AgregarNodo']/ownedParameter [@name='return']/@type] return $a/@name	Nodo	Nodo	Nodo	52	90

CLASE: Puerto **METODO: IDPuerto**	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $clase in $doc//packagedElement where $clase/@name='Puerto' return $clase/ownedOperation[@name='IDPuerto']/owne dParameter [@name='return']/@type	String	EAC__string	EAC__string	58	66
CLASE: Puerto **METODO: IDPuerto**	declare namespace xmi="http://www.omg.org/spec/XMI/20110701"; declare variable $doc external; for $clase in $doc//packagedElement where $clase/@name='Puerto' return let $a := $doc//packagedElement[@xmi:id = $clase/ownedOperation[@name='IDPuerto']/owne dParameter [@name='return']/@type] return $a/@name	String	String	String	65	70

5.5 ANÁLISIS DE RESULTADOS

A partir de las pruebas realizadas al mecanismo de consulta por medio del prototipo desarrollado, se analizó lo siguiente:

1. *Análisis N° 1*: En los escenarios 1 y 4, donde se le realizaron preguntas abiertas de generalidades del proyecto, sin tratarse de un elemento específico, se evidenció que a este tipo de preguntas solo puede darse solución de modo exacto. El modo ampliado no fue posible ejecutarse por la falta de términos de importancia en la consulta. Además, si se va a preguntar sobre la totalidad de algo en el proyecto, no es necesario buscar posibilidades de obtener más opciones de resultados.
2. *Análisis N° 2*: En el escenario 1, a pesar de que se preguntó de la totalidad de paquetes en el proyecto, se obtuvieron más de los que posee en proyecto real. Esto se debe a que la herramienta con la que se obtuvo la base de información (Enterprise Architec), agrega paquetes adicionales para agregar los tipos de datos primitivos. Lo anterior se evidenció al analizar el XMI y la información que poseían internamente esos paquetes adicionales retornados.
3. *Análisis N° 3*: En los escenarios 2, 3 y 7, se hacen preguntas sobre cuáles son los elementos que contiene un elemento dado, buscando información sobre clases, métodos y atributos en un elemento. Se evidencio que al realizar las pruebas, 2 de los 9 casos pertenecientes a este escenario no coinciden con los resultados esperados, presentándose estos casos cuando se utiliza el mecanismo de modo ampliado. Este tipo de preguntas donde, a pesar que se posea un término de importancia, se quiere buscar información específica de un elemento, no es necesario que el mecanismo proponga o incluya otros posibles resultados relacionados, evidenciados en ese 22% de casos fallidos en consultas de modo ampliado donde los resultados esperados no fueron los obtenidos. Dentro de lo anterior se pueden incluir los casos cuando se hacen preguntas de cantidad al mecanismo sobre un elemento específico.
4. *Análisis N° 4*: En el escenario 5, en el caso que se utilizó el término “protocolo”, a pesar de que se posee en el proyecto una clase “Protocolo” esta solo se obtuvo al aplicar el modo de consulta ampliado pues XQuery tienen en cuenta mayúsculas y

minúsculas al momento de ejecutar las consultas (Walmsley, 2007), es decir, para el lenguaje de consulta no es lo mismo “protocolo” que “Protocolo”.

5. *Análisis N° 5*: En los escenarios 5 y 6, se realizan preguntas donde no se tienen conocimiento de cómo se llamen los elementos en el modelo o se quieren obtener resultados donde el término sea una funcionalidad de la que queramos saber que elementos pueden estar relacionados. En este tipo de preguntas el 0% de los casos relacionados obtuvieron resultados de modo exacto, la única manera que se pueda dar resultados en estos escenarios es que se especifique el nombre exacto de lo que se busca. En los casos donde se realizó la consulta de modo ampliado, el 83% de los resultados obtenidos coincidieron con los resultados esperados, el porcentaje restante demuestra el margen de error que se tiene al realizar búsquedas donde el mecanismo proponga diferentes resultados, como en el caso del escenario 6 donde el termino es “envioDeMensaje” que se obtuvo el resultado no esperado “obtenerRouterMarcadoNoVisitadoMenor”, esto se presentó que aunque el termino semánticamente podría no estar relacionado, coincide con una de las expresiones regulares de los candidatos de búsqueda generados.
6. *Análisis N° 6*: En el escenario 7, de hacen preguntas sobre los tipos de datos que se retornan. Para estos casos la sentencia XQuery que pregunta sobre el tipo de los datos en la base de información retorna un tipo de datos propio de la herramienta que exportó el documento XMI. Para obtener los resultados esperados, tocó replantear la sentencia de consulta y realizar una búsqueda adicional para obtener el valor de los tipos de datos. Incluso, cuando se buscaban los resultados donde los resultados eran tipos de datos primitivos, retornaba una simbología propia de la herramienta de exportación y se obtenían los nombres reales al buscarlos en los paquetes propios de la herramienta, mencionados en el análisis N°2.

Otros resultados que se obtuvieron en la ejecución de pruebas fue que a pesar que se mencionó que XQuery podía exportar sus resultados en cualquier formato XML como XMI, no funciona la visualización de los resultados en herramientas CASE aunque el XMI se encuentre bien estructurado. Esto se debe a que las consultas realizadas no hacen referencia

a los elementos que estén incluidos en la estructura de los elementos retornados en la consulta. Por ejemplo si la consulta retorna solamente la “ClaseA” y esta posee un atributo de “ClaseB”, la definición de la “ClaseB” no está incluida en el XMI de resultado por lo cual impediría la visualización en una herramienta CASE. Esto se puede solucionar con consultas mucho más especializadas.

6 CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

La realización de consultas a una arquitectura de software, se considera un aspecto importante para el análisis y comprensión del mismo, los múltiples resultados obtenidos en la presente investigación, fundamentados en referentes bibliográficos científicos, así lo demuestran.

Los objetivos planteados para la investigación realizada se cumplieron. Inicialmente, se obtuvo como resultado la documentación del estado del arte, debidamente estructurado y organizado, en el cual se analizaron los estudios de las diferentes herramientas de consultas existentes, dando cumplimiento al primer objetivo de la investigación. El estado del arte permitió identificar, que a pesar que existían estudios de herramientas de consultas y análisis del software, todas estos se encuentran basados en la realización de consultas al código fuente y no a la arquitectura del software, además no existe una clasificación estandarizada del tipo de consultas, cada autor define los tipos de consultas de acuerdo a sus necesidades, aunque en algunas ocasiones tales clasificaciones guarden relación entre ellas. Este estado del arte servirá como punto de partida para la realización de futuras investigaciones relacionadas con la temática.

A partir del estado del arte, se definieron las necesidades de los mecanismos de consultas existentes, lo cual permitió definir los requerimientos del mecanismo de consulta a construir, lo que constituye el cumplimiento del segundo objetivo de la presente investigación. Teniendo en cuenta estos requisitos, se logró la definición del mecanismo de consulta haciendo uso del lenguaje de consulta XQuery, el cual fue validado mediante la construcción de un prototipo software, que a su vez fue validado con escenarios de pruebas dando cumplimiento al tercer y cuarto objetivo de la investigación.

Los resultados obtenidos con el cumplimiento de los objetivos, se relacionan con las investigaciones científicas realizadas por algunos autores como (SHAO & SOON, 2002), (Chen, Nishimoto, & Ramamoorthy, 1990), (Paul & Prakash, 1994), (Muller, Orgun, Tdey, & Uhl., 1993), (Devanbu, Brachman, Selfridge, & Ballard, 1991), (Moor, y otros), (Linton,

1984) (Jarzabek, 1998) (Dahm, 1997), entre otros; debido a que el mecanismo de consulta definido proporciona aspectos que se basan en los resultados de los estudios realizados por estos autores, coincidiendo en algunos aspectos, como el uso del algebra relacional en sus mecanismos de consulta. Pero, no existe ningún estudio que se relacione exactamente con todos los requisitos del mecanismo planteado en el presente trabajo de investigación.

Un mecanismo basado en algebra relacional y la generación de candidatos de búsqueda permite realizar consultas a la arquitectura representada en una base de información obtenida en un proceso de ingeniería inversa, lo que a su vez facilita el análisis de esta base de información, de la cual no se tiene conocimiento previo, permitiendo generar conocimiento que puede ser utilizado para realizar mantenimiento al software. Además, debido a que la base de información se encuentra representada como un archivo XMI, se puede hacer uso de un lenguaje de consulta robusto y bien estructurado basado en el álgebra relacional, como, XQuery.

De acuerdo a todo lo anterior, se cumplió con el objetivo de permitir el análisis de la base de información del resultado de un proceso de ingeniería inversa por medio del mecanismo realizado, permitiendo realizar consultas a esa base de información por medio del lenguaje de consulta XQuery.

XQuery está basado en el álgebra relacional, la cual facilita la obtención de información del resultado del proceso de ingeniería inversa. Una de las ventajas de haber trabajado con el lenguaje de consulta XQuery es que está bien documentado, posee muchas funcionalidades para facilitar el trabajo, es ágil y rápido al navegar en el XML.

Por otra parte, el mecanismo de consulta realizado permite la generación de candidatos de búsqueda, esta es una solución óptima para facilitar el análisis de software, la cual presenta ventajas en la sugerencia de resultados cuando no se posee conocimientos de los nombres de los elementos que se van analizar, aunque resulta poco conveniente de usar cuando se busca información puntual de la base de información. También, se debe tener en cuenta que presenta un margen de error que puede incluir información basura que no posee relación con lo que se analiza.

Los resultados de esta investigación permiten mejorar la capacidad de consulta, lo cual implicaría mejoras en la producción de software y el ámbito académico. La producción de software se verá beneficiada porque se facilitará el mantenimiento continuo de las vistas arquitectónicas del sistema, por lo tanto se esperaría que se disminuyera la probabilidad de error en el desarrollo de software. Además, se espera obtener beneficios en la producción literaria, debido a que con esta investigación se estima contribuir al aumento de la bibliografía de la temática tratada, a partir del mejoramiento a la clasificación realizada de los mecanismos de consultas existentes en el estado del arte. En términos generales al mejorar la producción de software se espera mejorar la calidad de los productos garantizando la evolución de estos a través del tiempo.

Por otra parte, es posible que tengan beneficios económicos para la comunidad de desarrollo de software y la comunidad de ingeniería del software, debido a que si se mejora la capacidad de análisis de una arquitectura por parte de los usuarios, esto implicaría disminución en el tiempo empleado para realizar esta actividad y por consiguiente los costos de la misma al momento de realizar actualizaciones y mantenimiento al software.

De igual forma en el ámbito académico se esperan tener beneficios con el desarrollo de esta investigación, debido a que la definición conceptual de un mecanismo de consulta se puede constituir como la base para la realización de herramientas de ingeniera inversa que permitan mejorar la capacidad de consulta y análisis de los usuarios. Además, porque la investigación realizada puede ser usada para impartir conocimiento relacionado con los conceptos que hacen parte tanto del proceso de ingeniería directa como del proceso de ingeniería inversa.

Es importante mencionar que la investigación se vio limitada por algunos factores. Uno de estos factores fue el económico, el cual restringió el acceso únicamente a las bases de datos gratuitas proporcionadas por la Universidad de Cartagena, que aunque fueron de vital importancia en la construcción del estado de arte, seguramente no contienen trabajos que pudieron haber sido incluidos en la investigación. Otro factor limitante, fue la falta de disponibilidad de los mecanismos de consultas encontrados en el estado del arte para realizar pruebas.

6.2 RECOMENDACIONES

Para trabajos futuros es necesario que se tenga en cuenta lo relacionado con las entradas del mecanismo de consulta, este aspecto representa una de las limitaciones más notorias, la necesidad que la entrada sea una sentencia XQuery, lo cual implica una inversión de tiempo bastante grande para poder ejecutar el mecanismo. Se necesitaría tener un conocimiento amplio del formato del documento XMI y de las sentencias XQuery para agilizar este proceso. Esta limitación se evidenció al realizar las validaciones del mecanismo por medio del prototipo, pues era mucho mayor el tiempo definiendo las sentencias que en la misma ejecución del mecanismo, incluso teniendo en cuenta candidatos de búsqueda. Se propone, para futuras investigaciones, la creación de un traductor de preguntas del lenguaje natural a sentencias XQuery, brindando un ambiente de análisis más completo para el ingeniero de software y eliminando la necesidad de tener conocimientos en XQuery y el formato del documento XMI.

El mecanismo desarrollado se centró en los modelos estáticos de un sistema, de acuerdo a esto se recomienda la creación de un mecanismo que tenga en cuenta los modelos dinámicos para poder realizar análisis de la interacción de los diferentes elementos de un sistema por medio de consultas.

Finalmente el diseño y desarrollo de una herramienta de software que aplique el mecanismo obtenido como resultado de la investigación realizada, constituiría un excelente valor agregado a los proyectos realizados dentro del área del análisis arquitectural y arquitecturas de software puesto que implicaría la aplicación de muchos de los conocimientos adquiridos durante la formación como Ingeniero De Sistemas en el Programa De Ingeniería De Sistemas de la Universidad De Cartagena. Adicionalmente, para este software se debería invertir esfuerzos en la visualización grafica de los resultados obtenidos en el mecanismo de consulta propuesto para facilitar al ingeniero de software la interpretación de los resultados. También, se pueden realizar mejoras sobre el prototipo construido para validar el mecanismo ampliando los escenarios de consulta del mismo.

7 BIBLIOGRAFÍA

Programming Language Evolution via Source Code Query Languages . (2012).

A., C., & A., R. (1993). The birth of Prolog. *ACM*.

Abold-Thalmann, H., Berghammer, R., & Schmidt, G. (1989). Manipulation of concrete relations: The relview-system.

Aho, A., Weinberger, P., & BrianKernighan. (1988). *http://c2.com/cgi/wiki?AwkLanguage*.

Alanen, M., & Porres, I. (2005). Model Interchange Using OMG Standards. *IEEE Computer Society Digital Library*.

Alves, T., Hage, J., & Rademaker, P. (2011). A Comparative Study of Code Query Technologies.

Al-Zoubi, R., & Prakash, A. (1991). Software Change Analysis via Attributed Dependency Graphs.

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

Beyer, D., Noack, A., & Lewerentz, a. C. (2005). Efficient relational calculation for software analysis.

Beyer, D., Noack, A., & Lewerentz, C. (2003). Simple and efficient relational querying of software structures. *IEEE*.

Bildhauer, D., & Ebert, J. (2008). Querying Software Abstraction Graphs.

Canfora, & Penta, D. (2008). Frontiers of Reverse Engineering: a Conceptual Model.

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. *ACM*.

Canfora, G., Di Penta, M., & Cerulo, L. (2011). Achievements and Challenges in Software Reverse Engineering. *ACM*.

Ceri, S., Gottlob, G., & Tanca, L. (1989). What you always wanted to know about datalog (and never dared to ask). *IEEE*.

Chen, Y.-F., Nishimoto, M. Y., & Ramamoorthy, C. V. (1990). The C Information Abstraction System . *IEEE*.

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

Codd, E. F. (1970). A relational model for large shared data banks.

Colmerauer, A., & Roussel, A. (1993). The birth of Prolog. *ACM*.

Consens, M., Mendelzon, A., & Ryman, A. (1991). Visualizing and querying software structures.

Crew, R. F. (1997). Astlog: a language for examining abstract syntax trees.

Dahm, P. (1997). Architektur des GReQL–Auswerters.

Devanbu, P., Brachman, R., Selfridge, P., & Ballard, B. (1991). LaSSIE: A Knowledge-Based Software Information System. *ACM*.

Ebert, J., & Franzke, A. (1995). A declarative approach to graph based modeling.

Ebert, J., Bildhauer, D., Schwarz, H., & Riediger, V. (2007). Using difference information to reuse software cases.

Española, R. A. (2014). *http://lema.rae.es/drae/?val=mecanismo*.

Hajiyev, E., Verbaere, M., & Moor, O. d. (2006). CodeQuest: scalable source code queries with datalog. *ACM*.

Hajiyev, E., Verbaere, M., Moor, O. d., & Volder., K. D. (2005). Codequest: querying source code with Datalog. *ACM*.

Holt, R. C. (1998). Structural manipulations of software architecture using Tarski relational algebra.

Holt, R. C. (2008). Grokking Software Architecture . *IEEE*.

Holt, R. C., Winter, A., & Wu, J. (2002). Towards a common query language for reverse engineering.

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

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

J. Ebert, V. R. (2008). Graph Technology in Reverse Engineering, The TGraph Approach. *IEEE*.

Janzen, D., & Volder, K. D. (2003). Navigating and querying code without getting lost. *ACM*.

Jarzabek, S. (1998). Design of flexible static program analyzers with pql. *IEEE*.

Java. (2000). *Java Coding Style Guidelines.* Obtenido de iwombat.com: http://www.iwombat.com/standards/JavaStyleGuide.html

K. Petersen, R. F. (2008). Systematic mapping studies.

Klint, P. (2003). How understanding and restructuring differ from compiling—a rewriting perspective.

Klint, P. (2005). A tutorial introduction to Rscript.

Kniesel, G., & Bard, U. (2006). An analysis of the correctness and completeness of aspect weaving.

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.

Kozaczynski, W., Ning, J., & Engberts, A. (1992). Program Concept Recognition and Transformation. *IEEE*.

Larman, C. (2001). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process.*

Libkin, L. (2001). Expressive power of SQL.

Linton, M. A. (1984). Implementing relational views of programs.

Liu, H., & Lethbridge, T. (2002). Intelligent Search Methods for Software Maintenance.

Liu, H., & Lethbridge, T. C. (2002). Intelligent Search Methods for Software Maintenance. *Information Systems Frontiers*, 409-423.

Lu, B., & Liu, Z. (2013). Prolog with the best search. *IEEE*.

Matturro, G., & Saavedra, J. (2012). Factores que Inciden en la Mejora de Procesos Software. Un mapeo sistemático de la literatura.

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

Moor, O. d., Verbaere, M., Hajiyev, E., Avgustinov, P., Ekman, T., Ongkingco, N., . . . Tibble, J. (s.f.). Keynote address: .ql for source code analysis. *IEEE*.

Muller, H., Jahnke, J., Smith, D., Storey, M., Tilley, S., & Wong, K. (2000). Reverse Engineering: A Roapmap. *IEEE*.

Muller, H., Orgun, M., Tdey, S., & Uhl., J. (1993). A Reverse Engineering Approach to Subsystem Structure Identification.

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

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

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

Object Management Group. (2012). *OMG's MetaObject Facility*. Obtenido de http://www.omg.org/mof/

Object Management Group. (2013). *Object Management Group*. Obtenido de MDA - The Architecture Of Choice For A Changing World. OMG Model Driven Architecture: http://www.omg.org/mda/

Object Management Group. (2013). *Object Management Group*. Obtenido de About OMG: http://www.omg.org/gettingstarted/gettingstartedindex.htm

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

Pagano, D., & Anne, B. (2009.). Engineering Document Applications - From UML Models to XML Schemas. *IEEE Computer Society Digital Library*.

Panas, T., Lowe, W., & Abmann, U. (2003). Towards the Unified Recovery Architecture for Reverse Engineering. *IEEE Computer Society*.

Paul, S., & Prakash, A. (1994). Querying source code using an algebraic query language. *IEEE*.

Perez, J., & Garcia, M. (s.f.). *XMI: XML Metadata Interchange.*

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

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

Rademaker, P. (2008). *Binary relational querying for structural source code analysis.*

Rademaker, P. (s.f.). Binary relational querying for structural source code analysis.

Rasool, G., & Asif, N. (2007). Software Architecture Recovery.

Real Academia Española. (2001). Obtenido de rae.es: http://lema.rae.es/drae/?val=semantica

Resnick, P., & Varian, H. R. (1997). Recommender systems. *ACM*.

Robie, J. (2007). Procesamiento de XML y la integración de datos con XQuery. *IEEE Computer Society Digital Library*.

Robie, J. (2007). XML Processing and Data Integration with XQuery. *IEEE Computer Society Digital Library*.

SAXON. (21 de Junio de 2013). *SAXON The XSLT and XQuery Processor*. Obtenido de SAXON The XSLT and XQuery Processor Web site: http://saxon.sourceforge.net/

SDML. (2012). *SDML*. Obtenido de srcML: A document-oriented XML representation of source code: http://www.sdml.info/projects/srcml/

SHAO, W. Z., & SOON, H. S. (2002). Intelligent Search Methods for Software Maintenance.

Software Reverse Engineering: Achievements and Challenges . (2010).

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

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

Tarski, A. (1941). On the Calculus of Relations. *The Journal of Symbolic Logic 6*.

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

Verbaere, M., Hajiyev, E., & Moor, O. d. (2007). Improve software quality with SemmleCode: an Eclipse plugin for semantic code search.

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

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

W3C. (3 de Enero de 2011). *W3C Recommendation - XQuery 1.0: An XML Query Language (Second Edition)*. Obtenido de World Wide Web Consortium (W3C): http://www.w3.org/TR/xquery/

W3C. (2013). *Introduction to XQuery*. Obtenido de w3schools: http://www.w3schools.com/xquery/xquery_intro.asp

Walmsley, P. (2007). Search Across a Variety of XML Data XQuery. En P. Walmsley, *Search Across a Variety of XML Data XQuery* (págs. 1-2). O'Reilly.

Wellner, B., & Dant, M. (1989). *http://pages.cs.wisc.edu/~mdant/cs520_4.html*. Obtenido de The Unix "GREP" Utility.

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

World Wide Web Consortium. (2013). *World Wide Web Consortium*. Obtenido de http://www.w3.org/TR/xquery/

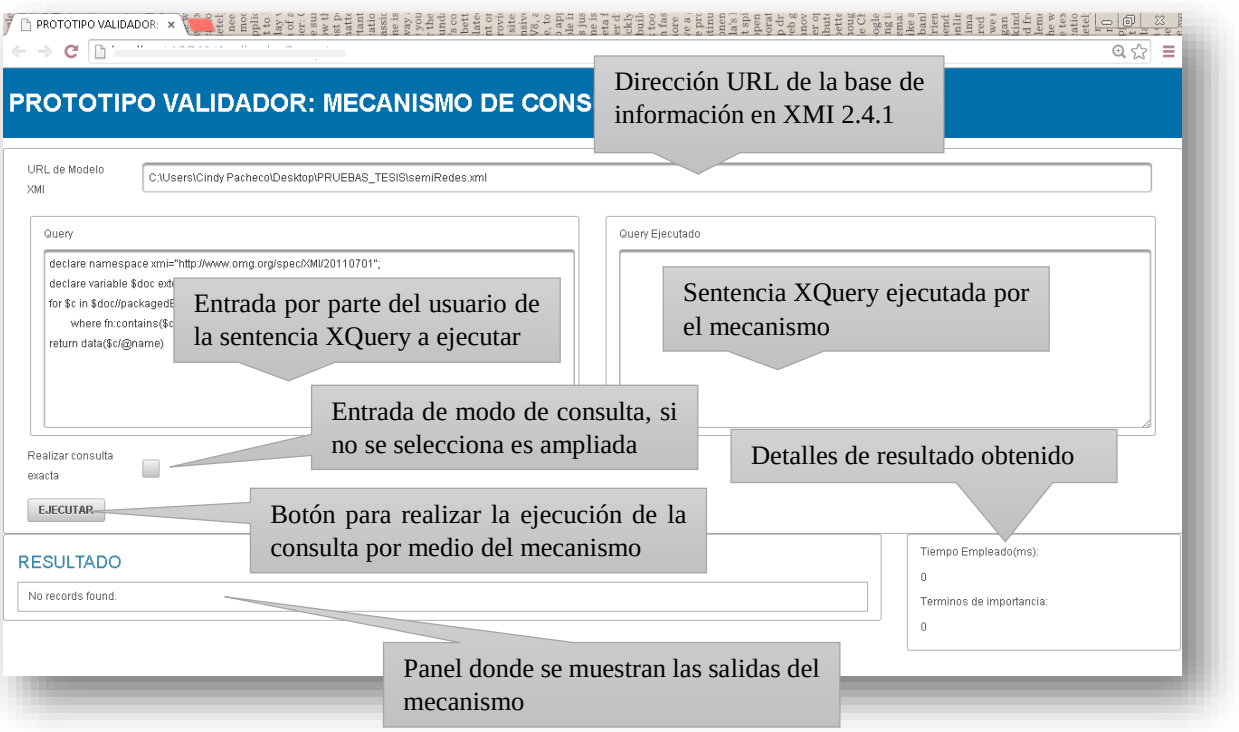
Wu, J., Holt, R. C., & Winter, A. (2002). Towards a Common Query Language for.

Yan, W., & Du, Y. (2010). Research on reverse engineering from formal models to UML models. *IEEE Computer Society Digital Library*.

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

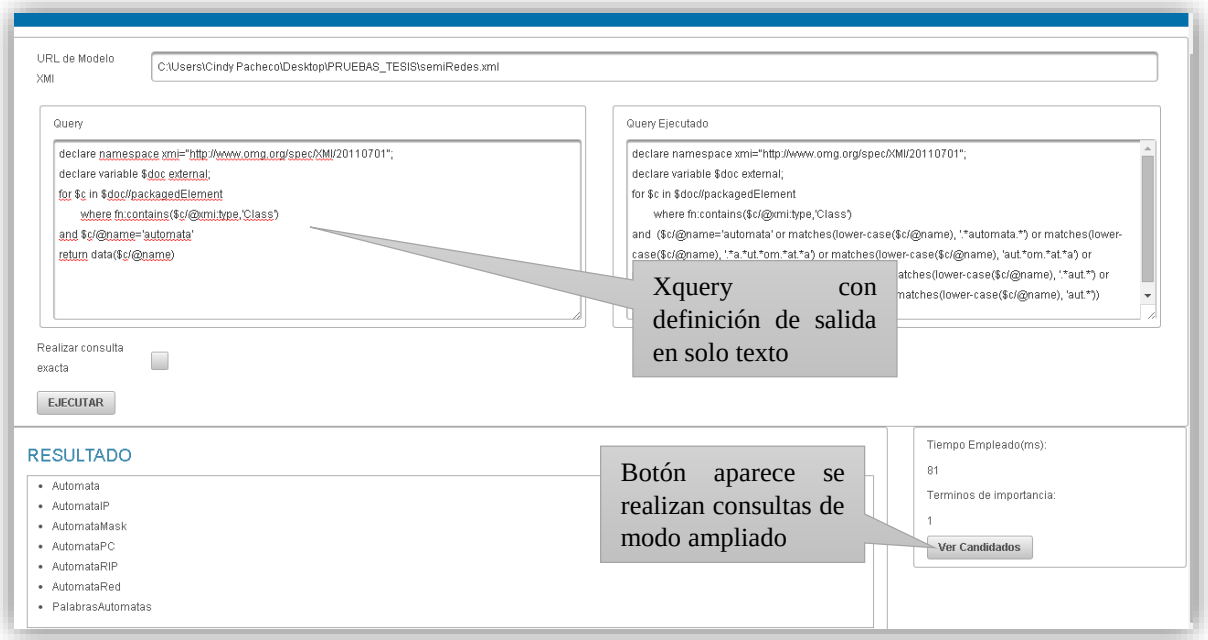
8 ANEXOS

8.1 ANEXO I: DIVISIÓN DE LA PANTALLA DEL PROTOTIPO.

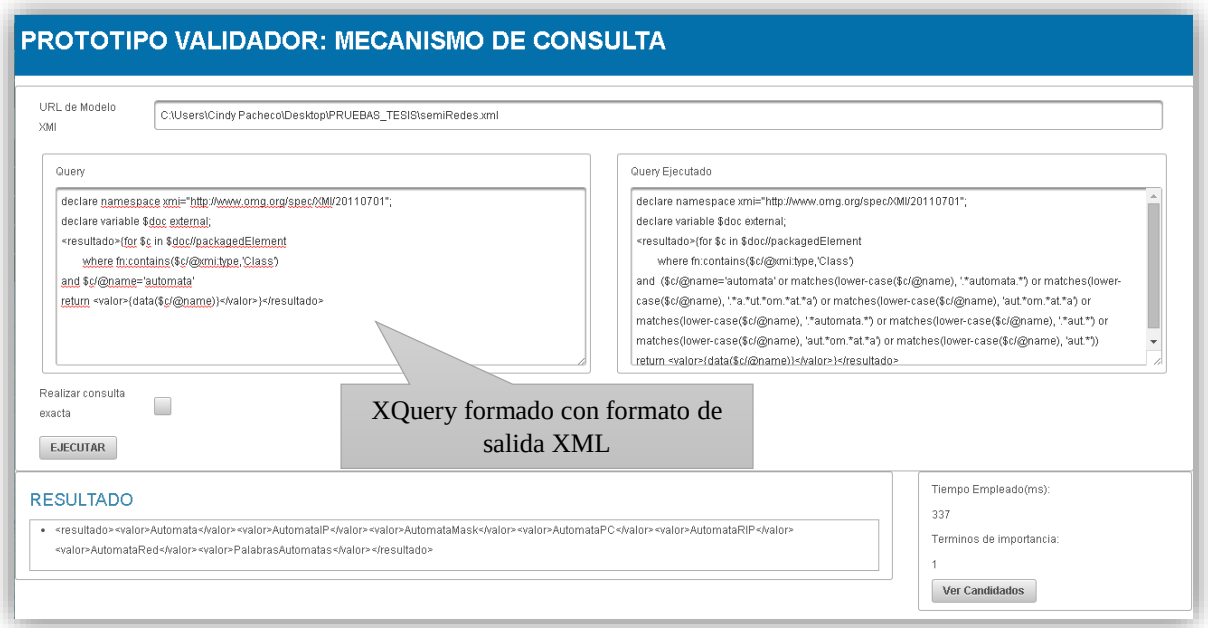


8.2 ANEXO II: PRESENTACIÓN DE RESULTADOS

- Para resultados en texto



- Para resultados en XML



8.3 ANEXO III: VISUALIZACIÓN DE LOS CANDIDATOS DE BÚSQUEDA DE LOS TÉRMINOS DE IMPORTANCIA

8.4 ANEXO IV: ARTÍCULOS INVOLUCRADOS EN EL PROCESO DEL MAPEO SISTEMÁTICO

Este anexo lo podrá encontrar en la carpeta Anexos de la entrega del proyecto de grado.