Información Tecnológica
Vol. 28(5), 87-100 (2017)
doi: 10.4067/S0718-07642017000500011

Mecanismo de Consulta para el Análisis de Arquitecturas Recuperadas

Martín E. Monroy[(1)], José L. Arciniegas[(2)] y Julio C. Rodríguez[(1)]
(1) Universidad de Cartagena, Facultad de Ingeniería, Programa de Ingeniería de Sistemas, Piedra de Bolívar, Bolívar – Colombia (e-mail: mmonroyr@unicartagena.edu.co, jrodriguezr@unicartagena.edu.co)
(2) Universidad del Cauca, Departamento de Telemática, Sector Tulcán, Cauca – Colombia
(e-mail: jlarci@unicauca.edu.co)

Recibido Dic. 30, 2016; Aceptado Mar. 6, 2017; Versión final Abr. 17, 2017, Publicado Oct. 2017

Resumen

El objetivo de este trabajo fue diseñar un mecanismo de consulta para hacer análisis sobre modelos UML, que representan arquitecturas recuperadas almacenadas en repositorios XMI, resultado de procesos de ingeniería inversa. Esto se logró a partir de: 1) la revisión de la literatura, que hizo posible la caracterización de los mecanismos de consulta; 2) el diseño del mecanismo de consulta y 3) la implementación del prototipo QModel-XMI, que sirvió como estrategia de evaluación de la propuesta planteada. Se concluye que el mecanismo diseñado amplia la capacidad de uso de los mecanismos de consulta existentes, porque extiende su uso a personas que no necesariamente cuentan con conocimientos profundos sobre lenguajes de consulta especializados.

Palabras clave: mecanismo de consulta; arquitecturas recuperadas; análisis de arquitecturas; ingeniería inversa

A Query Mechanism for Analysis of Recovered Architectures

Abstract

The objective of this work was to design a query mechanism to support the analysis on UML models that represent recovered architectures stored in XMI repositories, which are the result of reverse engineering processes. This was achieved by: 1) reviewing the literature, which made possible the characterization of the query mechanisms; 2) the design of the query mechanism and 3) the implementation of the QModel-XMI prototype, which served as a strategy for evaluating the proposed approach. It is concluded that the designed mechanism extends the usability of existing query mechanisms, because it extends its use to people who do not necessarily have deep knowledge about specialized query languages.

Keywords: query mechanism; recovered architectures; architectures analysis; reverse engineering

INTRODUCCIÓN

La evolución de la ingeniería inversa ha permitido extender su aplicación a nuevos contextos de uso, como la educación, la seguridad informática, la computación forense y a distintas actividades del proceso de desarrollo de software (Monroy et al., 2016a); por otra parte, su utilidad real se evidencia en el uso que se dé a los artefactos recuperados (Canfora et al, 2011), por eso se requiere de instrumentos y herramientas que permitan hacer análisis sobre las arquitecturas recuperadas, teniendo en cuenta los diferentes perfiles de los interesados para los nuevos contextos de uso de la ingeniería inversa. Un mecanismo de consulta es una estructura que organiza sus partes constitutivas para permitir la formulación de preguntas, con la respectiva representación de las respuestas. En el campo de la ingeniería de software esta estrategia ha sido muy utilizada sobre todo para apoyar el proceso de mantenimiento (Ujhelyi et al., 2015; Urma y Mycroft, 2015), facilitando el análisis y la comprensión del producto (Alves et al., 2011; Verbaere et al., 2008; Beyer et al., 2005), así como también la validación y transformación de modelos (Bergmann et al., 2012; Holt et al., 2002).

Durante los últimos años las tecnologías de consulta han cobrado importancia para realizar análisis sobre productos software (Urma y Mycroft, 2015; Alves et al., 2011). La revisión de la literatura reveló que algunos autores centran la atención de sus estudios en los leguajes de consulta (Störrle, 2013), otros hacen énfasis en la necesidad de lenguajes de consulta gráficos, que aprovechen las fortalezas de la percepción visual para facilitar la comprensión de las consultas (Zhang et al., 2012; Choi et al., 2009). De igual forma hay autores que proponen herramientas de consulta (Liepiņš, 2012; De Roover et al., 2011; Mendonça et al., 2004; Paul y Prakash, 1994), o las evalúan (Ujhelyi et al., 2015; Alves y Rademaker, 2011), mientras que en otros trabajos se toma como objeto de estudio tanto los lenguajes de consulta como las herramientas (Ujhely et al., 2015; Urma y Mycroft, 2015; Bergmann et al., 2012; Acretoaie y Störrle, 2012; Alves et al., 2011; Verbaere et al., 2008; Beyer et al., 2005; Holt et al.; 2002). En consecuencia, queda claro que todo mecanismo de consulta requiere de un lenguaje y una herramienta que interprete las consultas y presente los resultados de su ejecución.

Con respecto a los lenguajes de consulta Alves et al. (2011) establecen características como: el paradigma, los tipos de datos que soporta, la capacidad de parametrización, polimorfismo, modularidad; y la posibilidad de definir o usar librerías de consultas genéricas para implementarlos. En el mismo sentido Urma y Mycroft, (2015) definen como características la complejidad en la formación de las sentencias, los constructos que soporta (o no) de los lenguajes de programación, el conjunto de operaciones que permite el lenguaje, los IDE (Entornos de desarrollo integrado) para los que existen implementaciones, el soporte para sentencias genéricas y para trabajar con distintos tipos de expresiones; la capacidad de filtrado y sobrecarga de métodos; y la posibilidad de: hacer declaración local de variables, usar sentencias genéricas y de control de flujo, usar variables vinculantes; e identificar errores, código duplicado o expresiones sobre-compiladas.

En relación con las herramientas algunas de las características que se definen son: el lenguaje de implementación, el lenguaje de consulta (Holt et al., 2002), la interactividad de la interfaz de usuario, los formatos de intercambio y de salida, el tipo de licencia (Alves et al., 2011), el rendimiento y uso de memoria (Ujhelyi et al., 2015), y la capacidad para: el cálculo de métricas a partir de la consulta (Beyer et al., 2005), la visualización y manipulación de los resultados (Verbaere et al., 2008). Por otra parte, la revisión de la literatura permitió identificar dos grupos de tecnologías: 1) Las que hacen la consulta sobre el código fuente y 2) las que hace la consulta sobre los modelos. Todas las herramientas identificadas exigen el aprendizaje de lenguajes de consulta especializados, limitando su capacidad de uso exclusivamente a personas con profundos conocimientos en el tema, por lo tanto queda en evidencia la necesidad de un mecanismo de consulta que permita hacer análisis sobre las arquitecturas recuperadas, a aquellas personas que hacen parte de los nuevos contextos de uso de la ingeniería inversa: La educación, la computación forense y la seguridad Informática.

En este orden de ideas, el principal aporte de la investigación es el diseño de un mecanismo de consulta basado en lenguaje natural, para que las personas involucradas en un proceso de ingeniería inversa puedan hacer análisis sobre las arquitecturas recuperadas, representadas en XMI (Lenguaje extensible de marcado para el intercambio de metadatos), sin tener que aprender complejos lenguajes de consulta especializados. Esto es útil sobre todo en contextos como la educación, porque permite centrar la atención del educando en el aprendizaje del tema específico de ingeniería de software, sin necesidad de conocer lenguajes de consulta especializados. El segundo aporte es la implementación del mecanismo de consulta, representada en el prototipo QModel-XMI (Query Model in XMI), que sirvió como instrumento para evaluar la validez del diseño propuesto. Además, se presenta un modelo de caracterización de los mecanismos de consulta, que puede ser utilizado para analizar, diseñar, comparar y evaluar este tipo de herramientas. Esto se logró aplicando la metodología explicada a continuación. Posteriormente se presentan los resultados, luego se hace un análisis sobre los resultados obtenidos y finalmente se presentan las conclusiones.

METODOLOGÍA

Se hizo una revisión de la literatura aplicando la propuesta metodológica establecida por Kitchenham et al. (2011), para identificar los mecanismos existentes y los lenguajes de consulta que utilizan. Se realizaron los siguientes pasos: 1) definición de las preguntas de investigación, 2) búsqueda de los estudios primarios, 3) selección de los estudios aplicando los criterios de inclusión y exclusión establecidos, 4) clasificación de los artículos y 5) extracción y agregación de datos. El propósito del estudio consistió en identificar las características que describen los mecanismos de consulta existentes hasta el momento. Para orientar la revisión de la literatura se definieron las siguientes preguntas de investigación: 1) ¿Cuáles son los mecanismos de consulta existentes?. Esto permite establecer un inventario sobre los trabajos realizados. 2) ¿Cuáles son las características que describen los mecanismos de consulta existentes?, para identificar las características que debe contemplar el modelo. 3) ¿Qué lenguajes de consulta utilizan los mecanismos de consulta identificados?, para identificar tendencias sobre el uso de lenguajes de consulta especializados.

La búsqueda de los estudios primarios se realizó en las siguientes bases de datos bibliográficas: IEEExplore ACM Digital Library, SpringerLink, ScienceDirect, Scopus, Wiley online Library y Citeseer. Con base en los resultados obtenidos y aplicando técnicas de coincidencia de patrones (Ying, 2013) se estableció el modelo de caracterización, haciendo abstracción de las características que utilizan los autores para describir los mecanismos de consulta. Esto a su vez se utilizó como referente para definir los lineamientos que debe cumplir el mecanismo propuesto. Posteriormente se definió el diseño del mecanismo, identificando los elementos que lo conforman y definiendo sus patrones de comportamiento.

Para la validación del diseño del mecanismo de consulta se implementó el prototipo QModel-XMI y se realizó un estudio de caso aplicando la propuesta de Robert Yin (2013), en función de sus cinco componentes: las preguntas de investigación, las proposiciones, las unidades de análisis, la lógica para enlazar los datos con las proposiciones y los criterios para interpretar los hallazgos. El estudio de caso se aplicó en el contexto Educación, con estudiantes del programa de ingeniería de sistemas de la Universidad de Cartagena. El principal enlace lógico entre los datos y las proposiciones en este estudio de caso, se realizó aplicando la técnica de análisis denominada Modelo lógico, que consiste en observar la coincidencia de los eventos empíricos observados y los eventos teóricos predichos (Ying, 2013).

RESULTADOS

Como resultado de la investigación se logró: 1) un modelo de caracterización de los mecanismos de consulta encontrados en la revisión de la literatura, 2) un inventario de los lenguajes de consulta utilizados, 3) un inventario de las herramientas de consultas encontradas, 4) el diseño de un mecanismo de consulta basado en lenguaje natural, y 5) su implementación en el prototipo QModel-XMI. Cada uno de estos elementos se explica a continuación. El modelo de caracterización está representado en la figura 1, donde se observa que el mecanismo es implementado por medio de una herramienta y recibe como entrada un artefacto, que puede ser el código fuente o un modelo que represente una vista del producto software, también recibe una consulta escrita en un lenguaje especializado, que es interpretada y ejecutada sobre el artefacto de entrada, para obtener un resultado que se representa a través de un formato específico.

Se identificaron 23 lenguajes especializados que se utilizan para hacer consultas, quince de los cuales las realizan sobre artefactos de código fuente y ocho sobre artefactos que representan los modelos. También se identificaron cinco propuestas que utilizan lenguajes como Prolog (Maffort et al., 2013), XQuery (Choi et al., 2009), SQL (Hajiyev et al., 2006), QUEL (Chen et al., 1990) y Tyruba (De Volder, 2006) para hacer consultas sobre el código fuente y sobre los modelos. En la tabla 1 se relacionan los 28 lenguajes de consulta identificados, indicando el paradigma que los fundamenta (Álgebra relacional - AR, modelo de objetos - MO, lógica de predicados - LP, cálculo relacional - CR, cálculo de predicados - CP, orientadas a grafos - OG, Xpath), los artefactos sobre los cuales se hace la consulta, el formato del repositorio que contiene los datos de la consulta y el año de su creación.

De igual manera se identificaron 21 herramientas que implementan mecanismos de consulta, relacionadas en la tabla 2 en orden cronológico descendente, indicando los siguientes datos: En la segunda columna se presenta el nombre del mecanismo, en la tercera se especifica el tipo de artefacto que recibe como entrada, tomando como posibles valores código fuente (CF) y modelo (M). En la cuarta columna se detalla el propósito principal del mecanismo: análisis de código (AC), comprensión de programas (CP), comprensión de programas y reingeniería (CP y R), interpretación de modelos (IM), transformaciones (T), análisis de modelos (AM) y reutilización (R). En la quinta columna se relaciona el lenguaje de consulta que utiliza cada mecanismo, en la siguiente columna se indica el formato de salida de la herramienta y finalmente, en la última columna aparecen los autores y el año de publicación.

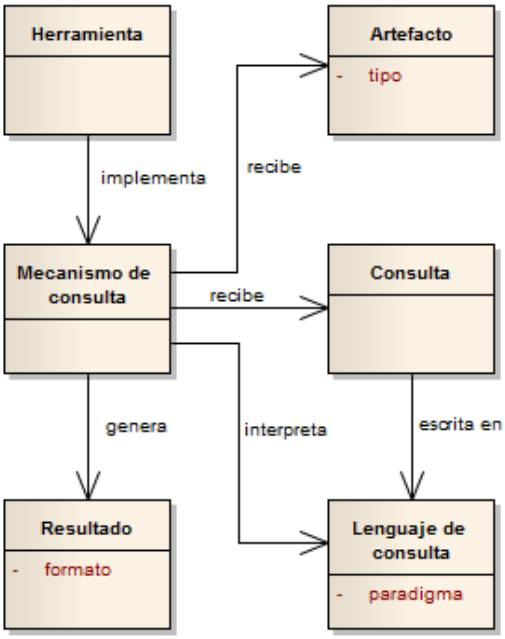

Fig. 1: Modelo de caracterización de los mecanismos de consulta.

Tabla 1: Lenguajes de consulta.

	Lenguaje	*Paradigma*	*Artefacto*	*Repositorio*	*Año*
1	MOCQL	LP	Modelo	Base de datos deductiva	2013
2	EMF-IncQuery	OG	Modelo	Grafos	2012
3	SOUL	LP	Código	AST	2011
4	VMQL	LP	Modelo	Base de datos deductiva	2011
5	BBQ	AR	Código	BDR Bytecode	2010
6	Cypher	OG	Código	Grafos	2010
7	BP-QL	OG	Modelo	Grafos	2008
8	BPMN-Q	OG	Modelo	Grafos	2007
9	.QL	AR y MO	Código	Base de datos relacional	2007
10	JTL	LP	Código	BDR Bytecode	2006
11	QM	MO	Modelo	Modelo UML	2005
12	RML	CP	Código	Tuplas	2005
13	Rscript	CR	Código	Tuplas	2003
14	AQL	AR-G	Código	AST y tuplas	2000
15	Visual OCL	MO	Modelo	Modelo UML	2000
16	Xquery	Xpath		XML	2000
17	GReQL	AR-G	Código	Repositorio Tgraph	1999
18	TyRuBa	LP		Base de datos deductiva	1998
19	ASTLog	LP	Código	AST	1997
20	Grok	CR-G	Código	Tuplas	1995
21	OCL	MO	Modelo	Modelo UML	1995
22	PQL	AR	Código	Base de conocimiento	1995
23	SCA	AR	Código	AST	1994
24	GraphLog	LP	Código	Grafos	1991
25	Datalog	LP	Código	Base de datos deductiva	1989
26	QUEL	AR		Base de datos relacional	1976
27	SQL	AR		Base de datos relacional	1974
28	Prolog	LP		Base de datos deductiva	1972

Tabla 2: Mecanismos de consulta.

Id.	Nombre del mecanismo	Artefacto	Propósito	Lenguaje de consulta	Formato salida	Referencias
1	Wiggle	CF	CP	Cypher	Texto y enteros	Urma y Mycroft, 2015
2	MACH	M	AM	MOCQL	Texto y enteros	Störrle, H., 2013
3	lQuery	M	T	LPG	Texto	Liepiņš, R., 2012
4	MQ-2	M	IM	VMQL	Texto y enteros	Acretoaie y Störrle, 2012
5	BARISTA	CF	CP	SOUL	Texto	De Roover et al., 2011
6	EMF-IncQuery Framework	M	T	EMF-IncQuery	Texto y gráficos	Bergmann et al., 2012
7	A framework for querying graph-based BPM	M	R	BPMN-Q	Texto y gráficos	Sakr y Awad, 2010
8	MoDisco	M	IM		Texto	Bruneliere et al., 2010
9	JRelCal	CF	CP Y R	Rscript	Texto y gráficos	Rademaker P., 2008
10	BP-QL Prototype System	M	IM	BP-QL	Texto y gráficos	Beeri et al., 2008
11	SemmleCode	CF	CP	.QL	Texto y enteros	Verbaere et al., 2008
12	JGraLab	CF	CP	GReQL	Texto	Kahle, S., 2006
13	JQuery	CF	CP	TyRuBa	Texto y gráficos	De Volder, K., 2006
14	CodeQuest	CF	CP	Datalog	Texto y enteros	Hajiyev et al., 2006
15	XCARE	CF	AC	Xquery	Texto y enteros	Mendonça et al., 2004
16	CrocoPat	CF	AC	RML	Texto	Beyer et al., 2005
17	SWAG Kit	CF	CP	Grok	Texto	Holt et al., 2002
18	Alborz	CF	AC	AQL	Texto	Sartipi, K., 2001
19	ESCAPE	CF	CP	SCA	Texto	Paul y Prakash, 1994
20	CIA	CF	CP	QUEL	Texto	Chen et al., 1990
21	Omega	CF	CP	QUEL	Texto	Linton, M. A., 1984

Con base en los resultados presentados previamente, se planteó que el diseño del mecanismo de consulta debe cumplir los lineamientos relacionados a continuación, teniendo en cuenta que su objetivo principal es contribuir al análisis de las vistas arquitectónicas recuperadas en un proceso de ingeniería inversa, que puede presentarse en diferentes contextos de uso (Monroy et al., 2016a).

1) Se utiliza el mecanismo lingüístico, aprovechando las capacidades descriptivas disponibles en lenguajes de modelado como UML (Lenguaje Unificado de Modelado), que registran la estructura, el comportamiento y demás propiedades del sistema.

2) El mecanismo apoya el análisis de las vistas arquitectónicas recuperadas, por lo tanto corresponde a la fase posterior al desarrollo.

3) Cada vista determina su objetivo y sus modelos. Se toman como referentes las vistas arquitectónicas que corresponden a la fusión de las propuestas hechas por Bass et al. (2013) y Callo Arias et al. (2011), porque al unirlas comprenden en forma completa todos los aspectos que describen el producto software, pertinentes al proceso de recuperación y análisis. Además, están documentadas según los lineamientos establecidos por el estándar ISO/IEC/IEEE 42010:2011.

4) Los modelos se representan en UML y para garantizar la interoperabilidad y reutilización se guardan en repositorios XMI (Monroy et al., 2016b).

5) Cada modelo puede responder a preguntas específicas, teniendo en cuenta la semántica de los elementos que lo constituyen.

6) El contexto en el que se realiza el análisis determina la pertinencia de la pregunta, a partir de la situación, los recursos existentes, los interesados y sus propósitos.

7) Las preguntas se formulan en lenguaje natural para facilitar el trabajo de los diferentes actores en cada uno de los contextos de uso, simplificando la tarea de análisis sobre las arquitecturas recuperadas.

8) Las métricas hacen referencia a atributos de calidad concretos, por lo tanto responden a preguntas específicas.

El diseño del mecanismo de consulta propuesto se plantea en función de los elementos que lo conforman y

su patrón esperado de comportamiento, según se explica a continuación. Los elementos se clasifican en dos grupos como se representa en la figura 2. El primer grupo corresponde a los artefactos o piezas de información que recibe o genera el mecanismo. En este grupo se encuentra: 1) La consulta original: es una pregunta formulada en lenguaje natural; 2) la consulta procesada: es una sentencia escrita en XQuery; 3) el resultado: es una cadena de texto en lenguaje natural que se obtiene como respuesta a la pregunta formulada inicialmente; 4) el modelo UML: es un archivo XMI que contiene diagramas UML; y 5) La representación KDM: también es un archivo XMI que representa al producto software, sus elementos, asociaciones y entornos de operación usando la especificación KDM. Estos dos últimos artefactos se encuentran almacenados en un repositorio.

El segundo grupo corresponde a los componentes que realizan una tarea específica del mecanismo. Este grupo lo conforman: 1) El módulo de interacción: responsable de la captura de la consulta original y de la presentación de los resultados; 2) el analizador lingüístico: Verifica la sintaxis y la gramática de la consulta original para determinar si es aceptada por el mecanismo, en caso afirmativo transforma la consulta original en consulta procesada. 3) El motor de consulta: Ejecuta la consulta procesada sobre el repositorio que contiene los modelos y emite un resultado.

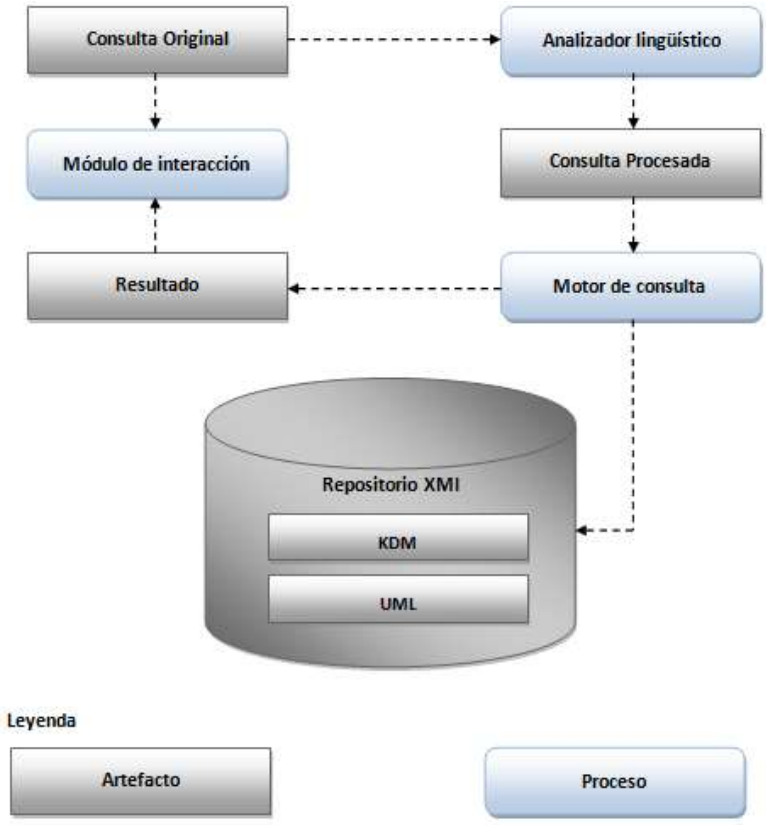

Fig. 2: Diseño del mecanismo de consulta.

El patrón esperado del comportamiento del mecanismo de consulta obedece al siguiente algoritmo, representado en la figura 3. El interesado registra la consulta en lenguaje natural, que es capturada por el módulo de interacción, luego el analizador lingüístico realiza la verificación de la sintaxis y la gramática. Si la consulta es válida y puede ser procesada por el mecanismo, se identifica el tipo de consulta de acuerdo a la siguiente clasificación: 1) cálculo de métricas, 2) consulta simple y 3) consulta compuesta. Posteriormente, el motor de consulta la ejecuta según el tipo identificado y finalmente los resultados son presentados por el módulo de interacción. Los tipos de consulta se determinaron a partir de las propiedades estructurales y semánticas del repositorio XMI. Las propiedades estructurales limitan la realización de consultas representadas por sentencias escritas en el lenguaje XQuery, por eso se denominaron consultas básicas. Este tipo de consultas permite identificar los elementos que conforman el modelo, las relaciones que existen entre ellos y las propiedades que lo describen. En la tabla 3 se presenta un listado no exhaustivo de preguntas básicas teniendo en cuenta los modelos representados en el archivo XMI.

Por otra parte las propiedades semánticas del repositorio XMI dependen del modelo que representan y no se pueden apreciar directamente a partir de su estructura, porque requieren de análisis complementarios para lograr una interpretación. Un caso particular de esta situación obedece al cálculo de métricas que se

pueden realizar a partir de los modelos, por ejemplo, el factor de polimorfismo representa el número de posibles situaciones polimórficas reales, con respecto al número máximo de posibles situaciones no polimórficas (Aggarwal et al., 2006). Este valor se calcula dividiendo el número total de métodos sobrescritos en todas las clases, por la sumatoria del resultado de multiplicar para cada clase, el número de métodos no sobrescritos por el número de descendientes de la clase. Ante esta situación se han definido las consultas de cálculo de métricas, orientadas a responder preguntas sobre métricas que pueden ser calculadas a partir de los modelos representados en los archivos XMI del repositorio.

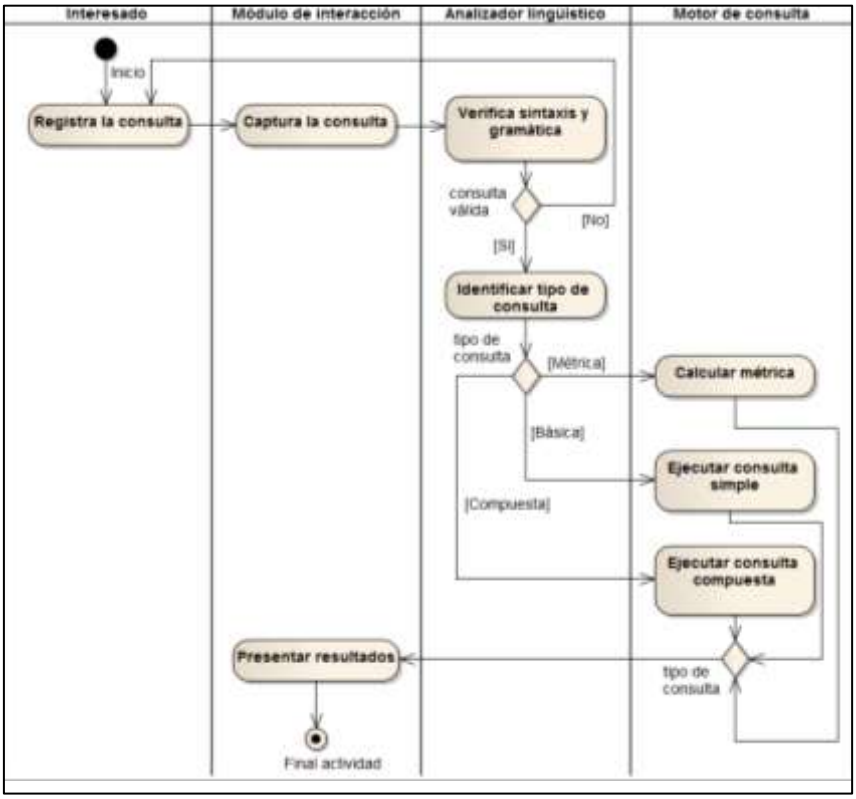

Fig. 3: Lógica del mecanismo de consulta.

Tabla 3: Listado de posibles consultas básicas.

Modelo	*Posibles consultas*
Diagrama de Casos de Uso	¿Cuáles son las funcionalidades del sistema? ¿Qué actores tiene el sistema? ¿Qué casos de uso realiza un actor concreto? ¿Qué relación existe entre dos casos de uso?
Diagramas de secuencia	¿Qué objetos interactúan en un escenario de uso específico? ¿Qué mensajes envía el objeto A al objeto B? ¿Qué mensajes recibe un objeto? ¿Qué ciclos de interacción hay? ¿Qué flujos alternos hay en la interacción? ¿Qué objetos interactúan en un ciclo específico? ¿Qué objetos interactúan en un flujo alterno específico?
Diagramas de comunicación	¿Qué objetos interactúan en un escenario de uso específico? ¿Qué mensajes envía el objeto A al objeto B? ¿Qué mensajes recibe un objeto?
Diagramas de actividades	¿Quiénes son responsables de las actividades? ¿Qué actividades se realizan? ¿Qué actividades realiza un responsable concreto? ¿Qué tomas de decisión se presentan en la actividad? ¿Qué actividades se desarrollan en flujos concurrentes?
Diagramas de máquinas de estado	¿Cuáles son los estados del sistema? ¿Cuáles son los eventos que se presentan en el sistema? En un estado determinado, ¿Qué eventos generan cambio de estado?, ¿el evento X a qué estado lleva?
Diagramas de clase	¿Qué clases conforman el sistema? ¿Qué relación existe entre la clase A y la clase B? ¿Qué servicios ofrece la clase A?

Adicionalmente, se pueden plantear preguntas más complejas a los modelos, que involucran propiedades cualitativas individuales y colectivas del sistema y que se pueden responder a partir de su semántica, pero que no consiguen ser resueltas solamente con consultas XQuery, sino que además requieren de la ejecución de elaborados algoritmos. A este grupo pertenecen las consultas compuestas. A partir de la revisión de la literatura, dentro de este tipo de consultas se identificaron los siguientes grupos: 1) Orientadas a establecer aspectos estructurales, 2) orientadas a establecer aspectos de comportamiento, 3) orientadas a establecer el uso de patrones, 4) orientadas a identificar errores y 5) orientadas a identificar la conformidad entre la arquitectura conceptual y la arquitectura concreta.

Cada uno de estos grupos está conformado por un conjunto de consultas, algunas de las cuales se presentan en la tabla 4. Por su naturaleza compleja, prácticamente cada pregunta implica la ejecución de una lógica particular para ser resuelta, por lo tanto, el componente del motor de consulta debe ser implementado aplicando el principio abierto a la extensión cerrado a la modificación (Martin, 1996), de tal manera que pueda ser complementado con nuevas funcionalidades que atiendan las consultas complejas que surjan, brindando además la posibilidad de reutilizar e integrar otros componentes que den respuesta a este tipo de consultas.

Por ejemplo, para dar respuesta a la pregunta ¿Qué patrones de diseño implementa el sistema?, se pueden integrar herramientas para la detección de patrones de diseño (Fontana y Zanoni, 2011); para responder la pregunta ¿Cuál es la secuencia de mensajes entre objetos en un escenario específico?, se puede implementar la gramática para identificar el grafo de flujo de objetos propuesta por Tonella (2005); para solucionar la consulta ¿Existe violación en la comunicación jerárquica entre capas? se puede integrar la implementación de la propuesta de Sarkar et al. (2009); y para detectar desviaciones de la conformidad entre la arquitectura conceptual y la arquitectura concreta, se pueden integrar propuestas que implementen el modelo de reflexión (Buckley et al., 2013).

Por otra parte, muchas de estas preguntas requieren de complejas estructuras de razonamiento y posiblemente no pueden ser implementadas en herramientas concretas que automaticen el proceso, sin embargo, con el uso del mecanismo propuesto se puede descomponer la consulta en partes para posteriormente llegar a una respuesta en forma incremental, involucrando actividades manuales y asistidas por herramientas de cómputo.

Tabla 4: Listado de posibles consultas compuestas.

Aspecto	*Posibles consultas*
Estructura	¿Qué componentes o paquetes conforman el sistema? ¿Cuáles son las capas del sistema?
Comportamiento	¿Cuál es la secuencia de mensajes entre objetos en un escenario específico?, ¿Qué comportamientos polimórficos presenta un componente o paquete?
Uso de patrones	¿Qué patrones arquitectónicos implementa el sistema?, ¿Qué patrones de diseño implementa el sistema?, ¿Qué patrones de programación implementa el sistema?
Identificación de errores	¿Existen violaciones al encapsulamiento del sistema?, ¿Existe violación en la comunicación jerárquica entre capas?
Conformidad	¿Cuál es el grado de desviación de la conformidad entre la arquitectura conceptual y la arquitectura concreta?

Finalmente, para probar el diseño propuesto se implementó el prototipo QModel-XMI, usando el lenguaje de programación Java, para aprovechar las ventajas que representa la Máquina Virtual y las librerías existentes para el manejo de archivos XML y el uso del lenguaje de consulta XQuery. El código fuente generado se distribuyó en componentes atendiendo la estructura definida en el diseño del mecanismo, creando tres componentes: Un módulo de interacción, que corresponde a la interfaz de usuario, un módulo de análisis lingüístico y el motor de consulta.

ANÁLISIS DE LOS RESULTADOS

El análisis se hace inicialmente sobre los resultados de la revisión de la literatura y posteriormente sobre el mecanismo de consulta diseñado. Al revisar detalladamente la tabla 2 se puede inferir lo siguiente:

1) No hay un consenso ni se puede determinar una tendencia sobre el uso de lenguajes de consulta especializados, porque cada mecanismo utiliza un lenguaje diferente, a excepción de las dos primeras propuestas que se presentaron, Omega (Linton, 1984) y CIA (Chen et al., 1990) que utilizaron el lenguaje

QUEL (Stonebraker et al., 1976). Además Liepiņš (2012) plantea la posibilidad de utilizar lenguajes de propósito general (LPG) para lograr éste objetivo.

2) La mayoría de los mecanismos reciben como entrada código fuente, sólo una tercera parte hace la consulta sobre los modelos, sin embargo, en los últimos años se ha centrado más el interés en desarrollar mecanismos de consulta que analicen modelos.

3) Para los mecanismos que procesan código fuente, el principal propósito es la comprensión del programa, sólo tres se utilizan para hacer análisis de código.

4) Los mecanismos analizados utilizan tres formatos para representar el resultado de las consultas: sólo texto, texto y números enteros, y texto y gráficos, siendo este último más usado por aquellas propuestas que reciben como entrada artefactos del sistema que representan modelos.

Un análisis más detallado de los mecanismos que procesan modelos revela que todos utilizan notación formal para realizar las consultas, lo cual implica un esfuerzo adicional para el aprendizaje del lenguaje de consulta. Por otra parte, como se muestra en la tercera columna de la tabla 5, los mecanismos procesan modelos almacenados en repositorios representados en bases de conocimiento de Prolog, bases de datos relacionales, grafos o en el meta modelo de Eclipse (Ecore-Model). Como caso especial está la librería IQuery, que conceptualmente permite cualquier tipo de repositorio, sin embargo, sólo tiene una implementación para repositorios que almacenen datos EMOF (Essential Meta Object Facility) (Acretoaie y Störrle, 2012).

Tabla 5: Mecanismos de consulta que procesan modelos.

Id.	*Nombre del mecanismo*	*Repositorio*	*Interface de usuario*	*Objetivo*
1	MACH	Prolog	Texto	Integrar prototipos para permitir el análisis de sistemas implementados.
2	IQuery	Heterogéneo	No aplica	Permitir consultas y transformaciones independientemente del repositorio utilizado para guardar los modelos.
3	MQ-2	Prolog	Gráfica	Integrar Prolog a la consola de MagicDraw, permitiendo hacer consultas sobre el modelo usando notación formal, para facilitar su interpretación.
4	EMF-IncQuery Framework	Ecore-Model	Texto	Brindar un framework para consultas y transformaciones de alto rendimiento, fácil de usar y que utilice el formalismo declarativo.
5	A framework for querying graph-based BPM	DB Relacional	Gráfica	Hacer consultas sobre los modelos de proceso de negocio, para facilitar la reutilización.
6	MoDisco	XML	Gráfica	Marco de referencia de ingeniería inversa dirigida por modelos, que permite la interpretación de los modelos por medio de consultas.
7	BP-QL Prototype System	Grafos	Texto	Hacer consultas sobre los modelos de proceso de negocio, para facilitar su interpretación.

La única herramienta que almacena los modelos en repositorios XML es MoDisco (Bruneliere et al., 2010), todas las demás requieren de procesos adicionales y el uso de herramientas especializadas de conversión de los modelos UML representados en XMI a los formatos aceptados por los mecanismos en mención, sumando complejidad al proceso de análisis de los modelos. También se resalta que los primeros cuatro mecanismos relacionados en la tabla 3 procesan modelos software representados en UML, MoDisco lo hace sobre Modelos KDM, mientras que el quinto y el séptimo lo hace sobre modelos de procesos de negocio. Esto permite inferir que el análisis de modelos UML usando mecanismos de consulta es un enfoque relativamente nuevo, con un amplio camino por explorar, en el que, por ahora EMF-IncQuery muestra un proceso de evolución que lo ubica como referente (Ujhelyi et al., 2015; Bergmann et al., 2012).

Al analizar el propósito de los mecanismos identificados (ver columna cuatro de la tabla 2) y el objetivo concreto de aquellos que procesan modelos (ver última columna de la tabla 3), se observa que MoDisco, MQ-2 y BP-QL se utilizan para interpretar modelos, IQuery y EMF-IncQuery Framework, apoyan procesos de transformación, Sakr y Awad (2010) facilita la reutilización, mientras que solamente MACH se usa para hacer análisis de modelos. Con respecto a la facilidad de uso de los mecanismos que procesan modelos, se encontró que todos exigen el conocimiento del lenguaje de consulta y como se observa en la cuarta columna de la tabla 3, únicamente tres cuentan con interfaz de usuario gráfica, tres sólo tienen una consola de texto para la interacción, mientras que IQuery, por ser una librería no tiene.

Con respecto al diseño del mecanismo de consulta, el análisis de los resultados se hace en función del prototipo QModel-XMI, usado en la realización de un estudio de caso como estrategia de evaluación del diseño propuesto. Atendiendo a los lineamientos que deben cumplir el mecanismo de consulta y el patrón de comportamiento establecido, se definió la arquitectura del QModel-XMI como se muestra en la figura 4. El componente 'VistaConsulta' es el responsable de brindar la interfaz gráfica de interacción, para que el actor pueda seleccionar el modelo e ingresar la consulta. El componente 'ContolConsulta' orquesta la ejecución de la consulta, atendiendo la siguiente lógica: 1) Solicita al componente 'AnalizadorLinguistico' verificar la validez de la sintaxis y la gramática de la consulta ingresada por el actor. 2) Si la consulta es válida invoca el servicio procesar consulta que ofrece el componente 'MotorConsulta', de lo contrario retorna un mensaje indicando que la consulta no es válida. El componente 'MotorConsulta' identifica el tipo de consulta, invocando al componente 'CalculoDeMetricas' si se trata de una consulta de métricas, o convirtiéndola al lenguaje XQuery y ejecutándola, si se trata de una pregunta directa sobre el modelo.

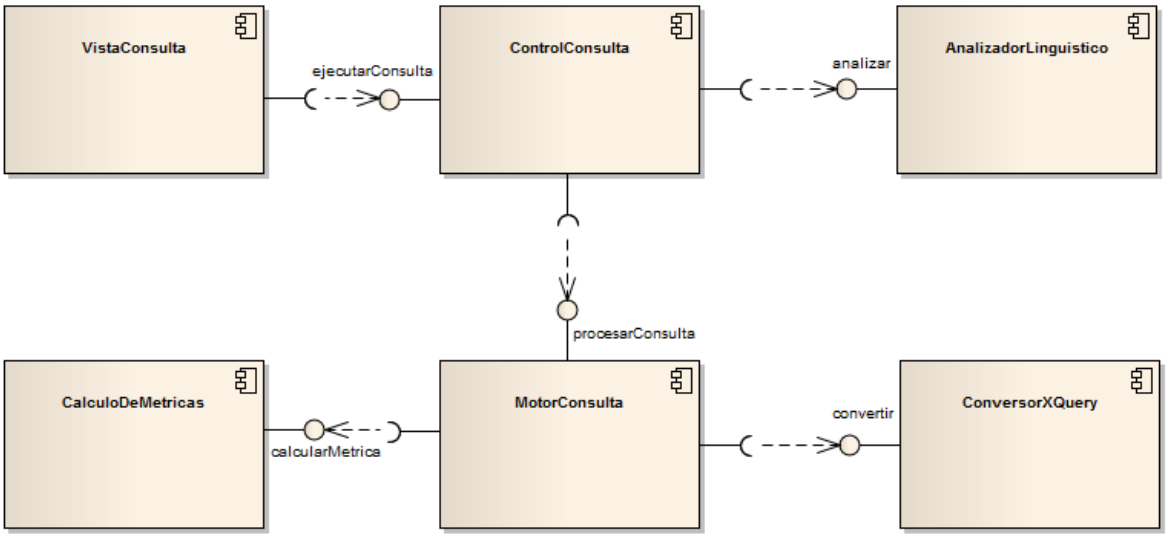

Fig. 4: Arquitectura de QModel-XMI.

Se realizó un estudio de caso con una unidad de análisis (Ying, 2013), correspondiente a un escenario de aprendizaje en la Universidad de Cartagena - Colombia con un grupo de 22 estudiantes, en el que se utilizó QModel-XMI como herramienta para apoyar la comprensión del concepto de polimorfismo y el desarrollo de habilidades para aplicarlo. Se entregó a cada estudiante el modelo de clases del sistema JHotDraw (JHotDraw org, 2007), recuperado y representado en XMI. Utilizando QModel-XMI identificaron atributos y métodos de las clases, además de las relaciones de generalización existentes, discriminando las clases base y las clases derivadas, como se muestra en la figura 5. Con los resultados obtenidos los estudiantes aplicaron el concepto de polimorfismo, atendiendo los lineamientos establecidos por el docente. Al hacer triangulación sobre las fuentes de información utilizadas en el estudio de caso (informe de la actividad presentado por cada estudiante, modelo XMI de JHotDraw, patrón de respuestas esperadas en la actividad académica), se reveló que QModel-XMI contribuyó en el logro de los objetivos de aprendizaje de la actividad académica realizada, al permitirle a los estudiantes interactuar con modelos UML representados en XMI, haciendo posible aprendizajes significativos a partir de la integración del conocimiento teórico y su aplicación en la práctica. Por lo tanto, quedó demostrada la validez del diseño del mecanismo propuesto, porque el prototipo QModel-XMI sirvió a los estudiantes para hacer consultas sobre modelos UML representados en XMI, utilizando el lenguaje natural. Esto también demuestra que se amplía la capacidad de uso de los mecanismos de consulta, a aquellas personas que no tienen conocimientos específicos sobre lenguajes de consulta especializados, como lo exigen las herramientas encontradas en la revisión de la literatura.

Al comparar el diseño del mecanismo de consulta y su implementación con las propuestas encontradas en la revisión de la literatura, específicamente con aquellas que procesan modelos (Ver tabla 5), se observa que, aunque su propósito general es el mismo difieren en su objetivo particular. MACH, lQuery, MQ-2, EMF-IncQuery Framework, MoDisco y QModel-XMI procesan modelos software representados en UML, las demás lo hacen sobre modelos de procesos de negocio. La diferencia más importante de QModel-XMI con todas las demás, radica en que requieren del aprendizaje de lenguajes especializados para hacer las consultas, mientras que QModel-XMI utiliza el lenguaje natural, lo que facilita su uso y amplia el campo de acción de este tipo de herramientas a otros contextos de uso de la ingeniería inversa (Monroy et al., 2016a).

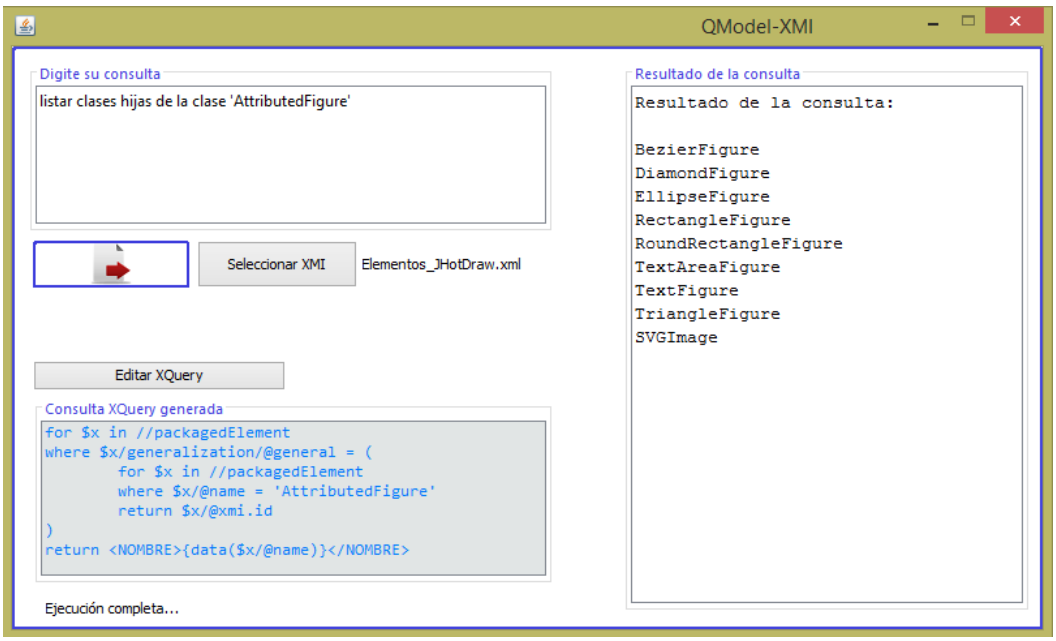

Fig. 5: Ejecución de QModel-XMI.

El mecanismo diseñado es una propuesta conceptual que se limita a modelos representados en archivos XMI, que correspondan a diagramas UML y/o representaciones KDM (Metamodelo de Descubrimiento de Conocimiento) de productos software, por lo tanto su capacidad está determinada por las posibilidades de consulta que brinda el lenguaje XQuery. El mecanismo por sí mismo no realiza ningún tipo de análisis, sólo sirve como instrumento de ayuda que facilita el análisis que pueda hacer el interesado, brindando la posibilidad de plantear preguntas en lenguaje natural sobre los modelos recuperados. Para mejorar los resultados obtenidos se propone complementar la capacidad del analizador lingüístico, para que las preguntas se puedan formular de distintas maneras ante un mismo valor semántico, por ejemplo, la pregunta ¿qué clases conforman el sistema?, también se puede formular ¿cuáles son las clases conforman el sistema?, ¿cuáles son las clases del sistema?, etc. De igual forma se propone mejorar el prototipo QModel-XMI, para que el resultado de las consultas se pueda visualizar en forma gráfica con diagramas UML, para aquellas preguntas que así lo requieran.

CONCLUSIONES

Los resultados obtenidos en el estudio de caso revelaron que el mecanismo de consulta, bajo su implementación en QModel-XMI, sirvió como herramienta de apoyo para hacer análisis sobre las vistas recuperadas y para hacer inferencias sobre el comportamiento del sistema, a partir de modelos estáticos, ampliando la capacidad de uso de los mecanismos de consulta existentes, porque extiende su aplicación a personas que no necesariamente cuentan con conocimientos profundos sobre lenguajes de consulta especializados. Sin embargo, se mantiene el reto trazado por Canfora et al. (2011), sobre la necesidad de mejorar la capacidad para consultar la base de información obtenida en los artefactos generados por los analizadores, en los procesos de recuperación de arquitecturas.

AGRADECIMIENTOS

Agradecemos a la Red Latinoamericana de Coordinación y Cooperación para unificar buenas prácticas de desarrollo de Software CoCoNet-LA por su apoyo; a los estudiantes Ismael Sayas, Alexander Silvera, Esteban Triviño y Jesús Prasca, integrantes del semillero de Investigación de Arquitecturas de software de programa de ingeniería de Sistemas de la Universidad de Cartagena, por su participación en la construcción del prototipo.

REFERENCIAS

Acretoaie, V., y H. Störrle, MQ-2 A Tool for Prolog-based Model Querying. In 8th European Conference on Modelling Foundations and Applications (ECMFA), 328-331 (2012)

Aggarwal, K.K., Singh, Y., Kaur, A., y R. Malhotra, Empirical Study of Object-Oriented Metrics. Journal of Object Technology, 5(8), 149-173 (2006)

Alves, T.L., Hage, J., y P. Rademaker, A comparative study of code query technologies. In Source Code Analysis and Manipulation (SCAM), 11th IEEE International Working Conference on, 145-154 (2011)

Bass, L., Clements, P. y R. Kazman, Software Architecture in Practice. 3ª Ed., Addison-Wesley (2013)

Beeri, C., Eyal, A., Kamenkovich, S., y T. Milo, Querying business processes with BP-QL. Information Systems, 33(6), 477-507 (2008)

Bergmann, G., Hegedüs, Á., Horváth, Á., Ráth, I., Ujhelyi, Z., y D. Varró, Integrating efficient model queries in state-of-the-art EMF tools. Objects, Models, Components, Patterns, 1-8 (2012)

Beyer, D., Noack, A., y C. Lewerentz, Efficient relational calculation for software analysis. Software Engineering, IEEE Transactions on, 31(2), 137-149 (2005)

Bruneliere, H., Cabot, J., Jouault, F., y F. Madiot, MoDisco: a generic and extensible framework for model driven reverse engineering. In Proceedings of the IEEE/ACM international conference on Automated software engineering, ACM, 173-174 (2010)

Buckley, J., Mooney, S., Rosik, J., y N. Ali, JITTAC: a just-in-time tool for architectural consistency. In Proceedings of the 2013 International Conference on Software Engineering, IEEE Press, 1291-1294 (2013)

Callo Arias, T.B., Avgeriou, P., America, P., Blom, K., y S. Bachynskyy, A top-down strategy to reverse architecting execution views for a large and complex software-intensive system: An experience report. Science of Computer Programming, 76(12), 1098-1112 (2011)

Canfora, G., Di Penta, M., y L. Cerulo, Achievements and challenges in software reverse engineering. Communications of the ACM, 54(4), 142-151 (2011)

Chen, Y. F., Nishimoto, M.Y., y C.V. Ramamoorthy, The C information abstraction system. IEEE Transactions on software Engineering, 16(3), 325-334 (1990)

Choi, B.J., Kim, Y.S., y Y. Ha, UML for XML-GL query using class diagram and OCL. In 2009 Seventh ACIS Inter. Conference on Software Engineering Research, Management and Applications, IEEE, 179-185 (2009)

De Roover, C., Noguera, C., Kellens, A., y V. Jonckers, The SOUL tool suite for querying programs in symbiosis with Eclipse. In Proceedings of the 9th International Conference on Principles and Practice of Programming in Java, ACM, 71-80 (2011)

De Volder, K. JQuery: A generic code browser with a declarative configuration language. In Practical Aspects of Declarative Languages, Springer, 88-102 (2006)

Fontana, F.A., y M. Zanoni, A tool for design pattern detection and software architecture reconstruction. Information Sciences, 181(7), 1306-1324 (2011)

Hajiyev, E., Verbaere, M., y De Moor, O. Codequest: Scalable source code queries with datalog. In ECOOP 2006-Object-Oriented Programming, Springer, 2-27 (2006)

Holt, R.C., Winter, A., y J. Wu, Towards a common query language for reverse engineering. Fachberichte Informatik, 8 (2002)

JHotDraw Org, JHotDraw as Open-Source Project. (En línea: http://www.jhotdraw.org/, acceso: 13 de julio de 2016), JHotDraw Org (2007)

Kitchenham, B.A., D. Budgen, y O.P. Brereton, Using mapping studies as the basis for further research. A participant-observer case study, Information and Software Technology, 53(6), 638-651 (2011)

Liepiņš, R., Library for model querying: IQuery. InProceedings of the 12th Workshop on OCL and Textual Modelling, ACM, 31-36 (2012)

Linton, M.A., Implementing relational views of programs. In ACM SIGSOFT Software Engineering Note, ACMs, 9(3), 132-140 (1984)

Maffort, C., Valente, M.T., Bigonha, M., Hora, A., Anquetil, N., y J. Menezes, Mining architectural patterns using association rules. In International Conference on Software Engineering and Knowledge Engineering (SEKE'13) (2013)

Mendonça, N. C., Fonseca, L. A., y P.H.M. Maia, Towards Reusable Code Analysis Tools Using Standard XML Technologies (2004)

Monroy, M.E., Arciniegas, J.L., y J.C. Rodríguez, Modelo Ontológico para Contextos de uso de Herramientas de Ingeniería Inversa. Información Tecnológica, 27(4), 165-174 (2016a)

Monroy, M. E., Arciniegas, J. L., y J. C. Rodríguez, Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura. Información Tecnológica, 27(5), 201-220 (2016b)

Paul, S., y A. Prakash, Querying source code using an algebraic query language. In Software Maintenance, 1994. Proceedings, International Conference on, IEEE, 127-136 (1994)

Rademaker, P., Binary relational querying for structural source code analysis. Master's Thesis, University Utrecht, Netherlands. (En línea: https://goo.gl/Y7yCKr, acceso: abril 28 de 2016) (2008)

Sakr, S., y A. Awad, A framework for querying graph-based business process models. In Proceedings of the 19th international conference on World Wide Web, ACM, 1297-1300 (2010)

Sarkar, S., Maskeri, G., y S. Ramachandran, Discovery of architectural layers and measurement of layering violations in source code. Journal of Systems and Software, 82(11), 1891-1905 (2009)

Sartipi, K., Alborz: A Query-based Tool for Software Architecture Recovery. In IWPC, 115-116 (2001)

Stonebraker, M., Held, G., Wong, E., y P. Kreps, The design and implementation of INGRES. ACM Transactions on Database Systems (TODS), 1(3), 189-222 (1976)

Störrle, H., Mocql: A declarative language for ad-hoc model querying. In Modelling Foundations and Applications, Springer, 3-19 (2013)

Störrle, H., VMQL: A visual language for ad-hoc model querying. Journal of Visual Languages & Computing, 22(1), 3-29 (2011)

Tonella, P., Reverse engineering of object oriented code. In Proceedings of the 27th international conference on Software engineering, ACM, 724-725 (2005)

Ujhelyi, Z., Bergmann, G., Hegedüs, Á., Horváth, Á., Izsó, B., Ráth, I., y D. Varró, EMF-IncQuery: An integrated development environment for live model queries. Science of Computer Programming, 98, 80-99 (2015)

Urma, R.G., y A. Mycroft, Source-code queries with graph databases with application to programming language usage and evolution. Science of Computer Programming, 97, 127-134 (2015)

Verbaere, M., Godfrey, M. W., y T. Girba, Query Technologies and Applications for Program Comprehension (QTAPC 2008). In Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on, IEEE, 285-288 (2008)

Yin, R. K., Case study research: Design and methods. Sage publications (2013)

Zhang, L., Sun, Y., Song, H., Wang, W., y G. Huang, Detecting anti-patterns in java EE runtime system model. Paper presented at the 4th Asia-Pacific Symposium on Internetware, ACM, 21 (2012)