MARCO DE REFERENCIA PARA LA RECUPERACIÓN Y ANÁLISIS DE VISTAS ARQUITECTÓNICAS DE COMPORTAMIENTO

MARTÍN EMILIO MONROY RÍOS

Tesis de Doctorado en Ingeniería Telemática

Director:
José Luis Arciniegas Herrera
Doctor en Ingeniería de Sistemas Telemáticos

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Doctorado en Ingeniería Telemática
Línea: Servicios avanzados de telecomunicaciones
Popayán, Octubre de 2016

MARTÍN EMILIO MONROY RÍOS

MARCO DE REFERENCIA PARA LA RECUPERACIÓN Y ANÁLISIS DE VISTAS ARQUITECTÓNICAS DE COMPORTAMIENTO

Tesis presentada a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del Título de

Doctor en:
Ingeniería Telemática

Director:
José Luis Arciniegas Herrera, PhD

Popayán
2016

Universidad del Cauca

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programas de Posgrado

FORMATO I
ACTA DE SUSTENTACIÓN DE
TRABAJO DE GRADO DE MAESTRÍA/TESIS DOCTORAL

Los jurados de:

TRABAJO DE GRADO DE MAESTRÍA ( ) TESIS DOCTORAL (X)

Titulado: Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento

Bajo la dirección de: Jose Luis Arciniegas Herrera

HACEN CONSTAR:

Que siendo las 1:40pm del día 23 del mes de Noviembre de 2016 el estudiante: Martin Emilio Monroy Ríos realizó la Sustentación del Trabajo de Grado de Tesis Doctoral, obteniendo la calificación de:

APROBADO (X) NO APROBADO ( ).

Para constancia, se firma en Popayán, a los 23 días del mes de Noviembre de 2016.

JURADO 1:

Nombre: Robson do Nascimento Fidalgo

JURADO 2 (Doctorado):

Kelly Garcés P

Nombre: Kelly Johany Garcés Pernett

JURADO COORDINADOR:

Nombre: Juan Carlos Corrales Muñoz

COORDINADOR DEL PROGRAMA (*):

Nombre: Álvaro Rendón Gallón

(*) Firma el Coordinador del Programa con autorización del(los) Jurado(s):

Dedicado a

mi esposa Alfia,
mis hijos Elina y Paul,
y a mis Padres.

Agradecimientos

Manifiesto mi más sincero agradecimiento al Doctor Jose Luis Arciniegas Herrera por su excelente trabajo como director de la tesis, a la Universidad de Cartagena por brindarme las condiciones para lograr los objetivos planteados, al programa de doctorado en Ingeniería Telemática de la Universidad del Cauca, bajo la dirección del Doctor Álvaro Rendón Gallón y a cada uno de sus docentes, por la gran labor que está realizando bajo estrictos lineamientos de calidad.

De igual forma agradezco al Licenciado Sergio Zapata, director del Instituto de Informática de la Universidad Nacional de San Juan, a la Doctora Myriam Herrera y la Licenciada María Inés Lound, por todo el apoyo brindado durante la pasantía en Argentina. Así mismo agradezco al Doctor Robson Fidalgo, profesor del centro de informática de la Universidad Federal de Pernambuco, por toda la colaboración que me brindó durante la pasantía en Brasil. Igualmente agradezco al Ingeniero Camilo Velásquez por su apoyo en la realización de los estudios de caso.

También agradezco a todos y cada uno de los integrantes del semillero de Investigación de Arquitecturas de software, del programa de ingeniería de Sistemas de la Universidad de Cartagena, y especialmente a Cindy Pacheco, Alejandra Ríos, Ismael Sayas, Alexander Silvera, Esteban Triviño y Jesús Prasca, por su trabajo en la implementación del prototipo.

Resumen estructurado

Antecedentes: La ingeniería inversa es una técnica de mantenimiento que hace posible la evolución de productos software legados, condición indispensable para las organizaciones que soportan sus procesos en este tipo de tecnologías. Si bien existen métodos y herramientas que permiten extraer la estructura y el comportamiento del sistema, y mecanismos de consulta soportados en lenguajes especializados para hacer análisis sobre productos software, la utilidad real de un proceso de ingeniería inversa se evidencia en el uso que se le dé a los artefactos recuperados.

Objetivos: Este proyecto de investigación definió un marco de referencia para la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento de sistemas software, por medio de la definición de un modelo conceptual para abstraer artefactos de bajo nivel y reconstruir información de alto nivel sobre el producto software; y el diseño de un mecanismo de consulta para realizar análisis a nivel de arquitectura, a partir de la base de información obtenida en los artefactos generados por los analizadores en el proceso de ingeniería inversa.

Método: Se caracterizaron las propuestas existentes a nivel de herramientas, métodos y mecanismos de consulta, para determinar los lineamientos del marco de referencia propuesto, que sirvieron como referentes para la definición del modelo conceptual y el diseño del mecanismo de consulta. Los resultados obtenidos se evaluaron por medio de la realización de un estudio de caso único (embebido) con dos unidades de análisis, y la implementación de un prototipo del mecanismo de consulta.

Resultados: Se definió ARCo, un Marco de referencia para la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, conformado por: 1) Un modelo conceptual para abstraer artefactos de bajo nivel y reconstruir información de alto nivel de sistemas software, 2) la Metodología para la recuperación y análisis de vistas arquitectónicas, teniendo en cuenta el contexto en el que se presenta el problema objeto de estudio, 3) el diseño del mecanismo de consulta para realizar análisis a nivel de arquitectura de sistemas software, a partir de la información base obtenida en los artefactos generados por los analizadores en el proceso de ingeniería inversa, y 4) QModel-XMI, un prototipo que implementa el

mecanismo de consulta. El alcance de la tesis se limitó a los contextos de producción de software y educación.

Conclusiones: Se lograron los objetivos propuestos dando respuesta a la pregunta de investigación con la definición de ARCo, lo que permitió concluir que la ingeniería inversa es una disciplina en evolución, que ha ampliado sus campos de acción a otros contextos, exigiendo nuevas funcionalidades y características; y el marco de referencia propuesto constituye un nuevo enfoque, que hace posible abordar el problema objeto de estudio en forma pertinente al contexto, porque brinda una visión holística de la ingeniería inversa y una visión detallada de la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, desde la perspectiva de diferentes contextos de uso como: La producción de software, la seguridad informática, la computación forense y la educación. Como investigación futura se recomienda complementar las técnicas para identificar las abstracciones que representan los elementos de la arquitectura y los aspectos determinantes para cada vista arquitectónica, a partir de las representaciones KDM y UML.

Palabras clave: Arquitectura de software, Recuperación de arquitecturas, análisis de vistas arquitectónicas, vistas de comportamiento, ingeniería inversa.

Structured abstract

Background: Reverse engineering is a maintenance technique, that enables evolution of legacy software products, a prerequisite for organizations that support their processes in these technologies. While there are methods and tools to extract the structure and behavior of the system, and query mechanisms supported in specialized languages for analysis of software products, the usefulness of a reverse engineering process becomes clear in the use that is given to the recovered artifacts.

Aims: In this research project was defined a framework for recovery and analysis of structural and behavioral architectural views of software systems, through the definition of a conceptual model for abstracting low-level artifacts and reconstruct high-level information of a software product; and the design of a query mechanism for analysis of the software systems at architectural level, from the information base obtained in the artifacts generated by the analyzers in the process of reverse engineering.

Method: Existing proposals at the level of tools, methods and query mechanisms were characterized to determine the guidelines of the proposed framework, which was used as a reference for defining the conceptual model and design query mechanism. The results were evaluated by conducting a single case study (embedded) with two units of analysis, and the implementation of a prototype of the query mechanism.

Results: ARCo was defined, a framework for recovery and analysis of structural and behavioral architectural views, consisting of: 1) A conceptual model for abstracting low-level artifacts and reconstructing high-level information about software systems, 2) a methodology for the recovery and analysis of architectural views, taking into account the context in which the problem takes place, 3) the design of a query mechanism for analysis on software systems at architectural level, to from information base obtained on artifacts generated by the analyzers in the reverse engineering process, and 4) QModel-XMI, a prototype that implements the query mechanism. The scope of the thesis was limited to the contexts of software production and education.

Conclusions: The objectives were achieved and the research question was answered with the definition of ARCo, which allowed to conclude that reverse engineering is an evolving discipline, which has expanded its field of action to other

contexts, demanding new features and functionalities; and the proposed framework is a new approach, which makes possible to address the problem under study in a pertinent way to the context, because it provides a holistic view of reverse engineering and a detailed overview of the reconstruction and analysis of structural and behavioral architectural views, from the perspective of different contexts of use, as software production, computer security, computer forensics and education. As future research, we intend to complement techniques to identify abstractions that represent the elements of architecture and the determinants for each architectural view, from the KDM representations and UML.

Keywords: Software architecture, architecture reconstruction, analysis of architectural views, behavioral views, reverse engineering.

Contenido

Lista de Figuras

Pág.

Lista de Tablas

Pág.

Lista de Siglas

ACM: Asociación de Sistemas Informáticos (Association for Computing Machinery)

ARCo: Recuperación de Arquitecturas en Contexto (Architecture Recovery in Context).

ARM: Método de reconstrucción de la arquitectura (Architecture Reconstruction Method)

FAMIX: Familia de Metamodelos para representar modelos relacionados con varias facetas de sistemas software.

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

KDM: Metamodelo del Descubrimiento del Conocimiento, (Knowledge Discovery Metamodel.

MDA: Arquitectura Dirigida por Modelos (Model Driven Architecture)

MDE: Ingeniería dirigida por modelos (Model Driven Engineering)

QModel-XMI: Consultando el Modelo XMI (Querying XMI Model)

RSF: Formato Estándar de Rigi (Rigi Standard Format).

UML: Lenguaje Unificado de Modelado (Unified Modeling Language)

XMI: Intercambio de metadatos en XML (XML Metadata Interchange)

XML: Lenguaje de etiquetado extensible (eXtensible Markup Language)

Capítulo 1

Introducción

En este capítulo se presentan los antecedentes teóricos representados en el estado del arte, los antecedentes prácticos, manifiestos en el planteamiento del problema de investigación, relacionando las circunstancias que lo originan y los escenarios de motivación, formulando la pregunta de investigación y presentando los argumentos que justifican la realización de la tesis. También se plantean los objetivos y se indica en forma resumida cuáles fueron las contribuciones del trabajo realizado. El capítulo finaliza explicando la forma cómo se encuentra estructurado y organizado el informe de la tesis doctoral. Aunque se identificaron cuatro contextos de uso de la ingeniería inversa, la tesis centró su atención en los contextos de producción de software y educación.

1.1. Estado del arte

El estado del arte se construyó a partir de un mapeo sistemático de la literatura (Monroy et al., 2016), que se llevó a cabo aplicando la propuesta metodológica establecida por Kitchenham et al. (2011) y Petersen et al. (2008), consultando las siguientes bases de datos bibliográficas: IEEExplore ACM Digital Library, SpringerLink, ScienceDirect, Scopus, Wiley online Library y Citeseer. Los resultados del mapeo permitieron identificar los antecedentes, los logros y los retos que se describen a continuación.

En los antecedentes se hace énfasis en las técnicas utilizadas para la recuperación de la arquitectura, teniendo en cuenta el aspecto que se recupera: la estructura, el comportamiento o los dos al mismo tiempo. También se relacionan las técnicas de análisis sobre los artefactos recuperados. En los logros se resaltan los avances que hasta el momento se han alcanzado, concretamente en la recuperación del comportamiento del sistema, así como también sobre la capacidad de análisis de la parte dinámica recuperada del sistema. Finalmente se presentan los retos existentes

con respecto a la recuperación y análisis de vistas arquitectónicas de comportamiento, resaltando aquellos que aborda esta tesis.

1.1.1. Antecedentes

Al aplicar el protocolo de la revisión bibliográfica se identificaron 246 documentos clasificados según en propósito como se muestra en la Tabla 1, y como se detallan en la Tabla 2 específicamente para las propuestas conceptuales según el aspecto que recuperan de la arquitectura. El análisis de esta documentación llevó a identificar 106 propuestas conceptuales específicamente para la recuperación de arquitectura, 51 de las cuales se centran en recuperar exclusivamente los aspectos estructurales del sistema, 42 recuperan al mismo tiempo los aspectos estructurales y de comportamiento y sólo 13 se centran exclusivamente en recuperar el comportamiento del sistema.

Propósito	***Detalle del propósito***	***Tipo de estudio***				***Total***	***%***
		RE	***EX***	***EC***	***RS***		
Propuesta conceptual (77.1%)	Proponer método	6	9	18	0	33	14,0%
	Proponer método y desarrollar herramienta	7	6	12	0	25	10,6%
	Definir un marco de referencia	32	20	47	0	99	41,9%
	Definir un algoritmo	2	6	2	0	10	4,2%
	Mejorar algoritmo	0	5	2	0	7	3,0%
	Presentar patrones	1	0	0	0	1	0,4%
	Otra	4	2	1	0	7	3,0%
Prueba de conceptos (18.2%)	Evaluación de métodos	0	6	3	0	9	3,8%
	Aplicar método	7	1	7	0	15	6,4%
	Desarrollo de herramienta	6	5	8	0	19	8,1%
Comparación (3%)	Evaluar herramienta	7	0	0	0	7	3,0%
Revisión literaria (1.7%)	Establecer la situación actual	0	0	0	4	4	1,7%
Totales		72	60	100	4	246	100%
		30,5%	25,4%	42,4%	1,7%	100%	

Tabla 1. Resumen estudios encontrados

Con respecto a la recuperación de los aspectos estructurales del sistema, desde 1993 Müller y Orgun proponen un enfoque para crear representaciones abstractas de alto nivel, que implica la identificación de los subsistemas y sus relaciones utilizando la herramienta Rigi y aplicando técnicas de clustering, que son mejoradas por Mancoridis et al. (1999) involucrando el conocimiento del experto. Tzerpos y Holt (2000) presentan ACDC (Algorithm for Comprehension-Driven Clustering) para identificar subsistemas a partir de posibles patrones, mientras que Maqbool y Babri

(2004) formulan WCA-UE (Weighted Combined Algorithm), un algoritmo para calcular la distancia entre clusters.

Para hacer descomposición eficiente del sistema Andritsos y Tzerpos (2005) proponen el algoritmo LIMBO (scaLable InforMation BOttleneck), en el que aplican técnicas de teoría de la información para optimizar las técnicas de clustering, minimizando la pérdida de información. Por otra parte, Corazza et al. (2010), proponen ZBR (Zone-Based Recovery), un modelo probabilístico para calcular el peso de las zonas, que se basa en análisis léxico y sirve para identificar subsistemas representativos; mientras que Platenius et al. (2012) presentan un enfoque para la re-documentación y evolución basada en componentes, Yuanfang Cai (2013) plantea obtener los subsistemas teniendo en cuenta la aplicación de reglas de diseño, Kobayashi et al. (2013) proponen SArF (Software Architecture Finder), un enfoque que utiliza la metáfora de la ciudad para visualizar la arquitectura, y Zhu (2013) plantea DGHC (Directed Graph Hierarchy Clustering), un algoritmo jerárquico de agrupamiento de grafos que considera tanto las similitudes como la conectividad entre los elementos del programa.

Propósito	*Detalle del propósito*	*Aspecto*			*Total*	*%*
		E	*C*	*A*		
Propuesta conceptual (77.1%)	Proponer método	14	6	13	33	19,0%
	Proponer método y desarrolla herramienta	12	5	8	25	14,4%
	Definir un marco de referencia	48	17	34	99	56,9%
	Definir algoritmo	9	0	1	10	5,7%
	Mejorar algoritmo	7	0	0	7	4,0%
Totales		90	28	56	174	100%
		51,7%	16,1%	32,2%	100,0%	

Tabla 2. Detalle propuestas conceptuales por aspecto recuperado

Además de la técnica de clustering para recuperar los aspectos estructurales de la arquitectura, se han propuesto técnicas basadas en álgebra relacional (Krikhaar, 1999), aplicando inteligencia artificial con algoritmos genéticos (Seriai y Chardigny, 2011) y aprendizaje automático (Bibi y Maqbool, 2011; Maqbool y Bibi, 2007); utilizando el modelo de reflexión (Murphy et al., 1995; Koschke y Simon, 2003; Koschke, 2010; Kim et al., 2010); haciendo análisis de grafos por medio de minería (Pavankumar et al., 2011) y coincidencia de patrones (Sartipi y Kostas, 2001); aplicando análisis de conceptos (Bojic y Velasevic; 2000; Lundberg y Löwe; 2003), minería de patrones (Haqqie y Shahid; 2005), indexación semántica latente (Scanniello et al., 2010; Risi et al., 2012), análisis relacional de conceptos (El Hamdouni, 2010), entre otras técnicas.

La mayoría de propuestas de enfoque hibrido (recuperan al mismo tiempo los aspectos estructurales y de comportamiento) presentan procesos genéricos,

explicando los pasos a seguir para recuperar la arquitectura, como es el caso de: HRY (Harris, 1995), FMAT (Fiutem et al., 1996), ARES (Eixelsberger et al., 1997), ARM (Guo et al., 1999), A Lightweight Architecture Recovery Process (Svetinovic y Godfrey, 2001), MAP (Stoermer y O'Brien, 2001), Architecture reconctruction in practice (Riva,2002), Dynamo (Ivkovic y Godfrey, 2002), QADSAR (Stoermer et al., 2003), Symphony (van Deursen et al., 2004), PuLSE-DSSA (Pinzger et al., 2004), CacOphoNy (Favre,2004), QAR (Arciniegas, 2006), Focus (Medvidovic y Jakobac, 2006), SQUA3RE (Stoermer,2007), ADDRA (Jansen et al., 2008), FLORA (Sözer et al., 2009), FASAR (Kang et al., 2009) y SAROS (Lee y Kang, 2011), entre otras.

Otras propuestas híbridas plantean estrategias heurísticas para la recuperación de la arquitectura (Li et al., 2005; Maweed et al., 2006; Anquetil et al., 2009), inteligencia artificial (García et al., 2011), minería de datos (Vasconcelos y Werner, 2011), análisis dinámico y análisis estático (Patel et al., 2009), la técnica de corte (slicing) (Guo et al., 2006) y análisis de conceptos (Eisenbarth et al., 2003).

Las propuestas orientadas a la recuperación del comportamiento utilizan técnicas como: el análisis probabilístico (Cook y Wolf, 1998), coincidencia sintáctica de patrones (Mendoca y Kramer, 2001), mapeo (Ran y Lencevicius, 2003; Schmerl et al., 2006; Zhou, 2008), reflection (Huang et al., 2006; Yu et al., 2007), anotaciones (Abi-Antoun y Aldrich, 2008; Abi-Antoun y Aldrich, 2009; Abi-Antoun y Barnes, 2010), heurística (de Brito et al., 2010) y análisis dinámico (Callo et al., 2011). La propuestas más recientes proponen procesos genéricos centrados en la recuperación del comportamiento (Peake et al., 2013; Forster et al., 2013).

Por otra parte, la revisión de la literatura reveló que el análisis sobre las arquitecturas recuperadas se logra con el cálculo de métricas, que brindan herramientas como Imagix 4D (Imagix Corporation, 2016) y JDepend (Clarkware Consulting, 2016). También existen técnicas de visualización que facilitan el análisis al presentar los resultados en forma visual por medio de gráficos, texto y otras formas de percepción humana incluyendo la interacción (Van Deursen et al., 2004). De igual forma se utilizan técnicas de navegación que permiten al ingeniero de software desplazarse por las estructuras de información generadas en la fase de organización del conocimiento, facilitando el proceso de análisis de las vistas recuperadas.

Las estrategias usadas para la navegación por los artefactos recuperados son los hipervínculos, el plegado (folding) y colapso (collapsing) de información (Panas et al., 2003) y la técnica de ojo de pescado (Tilley, 1998). Algunas herramientas que permiten la exploración de la información, brindando funcionalidades de visualización y navegación son: VizzAnalyzer (Löwe et al., 2003), VANESSA (Pacione et al., 2005), SHriMP (Storey et al., 1995), CodeCrawler (Lanza, 2003), iCIA (Interactive Change Impact Analysis Tool) (Kim et al., 2010), CCVISU (Beyer, 2005), ArchView (Pinzger, 2005) y CoCA-Ex (Holy et al., 2013).

También se identificó que 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), sin embargo, la revisión de la literatura reveló que los mecanismos de consulta existentes en el contexto de la ingeniería inversa, centran su atención en las consultas sobre el código fuente (De Roover et al., 2011; Antlersoft, 2010; Neo4j, 2010; De Moor et al., 2007; Holt, 1995).

Un análisis más detallado sobre los antecedentes de la recuperación y análisis de arquitecturas se puede encontrar en las publicaciones realizadas por Ducasse y Pollet (2009), Koschke (2009), Pollet et al. (2007) y Koschke (2005). De igual forma, los resultados de la revisión literaria realizada en el desarrollo de la presente tesis se encuentran publicados en la revista información tecnológica (Monroy et al., 2016) y se utilizaron para determinar los logros y los retos que se analizan a continuación.

1.1.2. Logros

Como resultado de la revisión de la literatura se identificaron los siguientes logros, que se utilizan como referentes para la toma de decisiones en la elaboración del modelo conceptual y el diseño del mecanismo de consulta:

1. El proceso canónico para la recuperación de arquitecturas está soportado por técnicas y herramientas que permiten extraer la estructura y el comportamiento del sistema software (Cai et al., 2013).
2. Existen modelos para describir los artefactos recuperados (FAMIX, KDM, RSF, entre otros) (El Boussaidi et al., 2012).
3. La recuperación de la arquitectura se ha extendido a contextos como: La seguridad informática (Treude et al., 2011), la computación forense (Nelson et al., 2014) y la educación (Klimek et al., 2011).
4. Existen mecanismos de consulta soportados en lenguajes especializados que permiten hacer análisis sobre productos software (Ujhelyi et al., 2014; Urma y Mycroft, 2012).

1.1.3. Retos

De igual forma, la revisión de la literatura permitió identificar los siguientes retos, que se utilizan como referentes para definir los lineamientos que rigen la propuesta del presente trabajo, teniendo en cuenta que centra la atención en responder especialmente a los dos últimos:

1. Necesidad de un catálogo de las técnicas de recuperación de arquitectura detallando cuándo y cómo deben ser utilizadas (Boussaidi et al., 2012; Koschke, 2009).
2. Mejorar la capacidad para presentar los resultados del proceso de recuperación (Canfora et al., 2011).
3. Reutilizar la tecnología de los analizadores existentes. (Ducasse y Pollet, 2009; Ducasse et al., 2000)
4. Mejorar la capacidad para consultar la base de información obtenida en los artefactos generados por los analizadores (Canfora et al., 2011).

5. Los nuevos contextos de uso de la recuperación de arquitecturas requieren de nuevos enfoques para lograr este objetivo. (Monroy et al., 2016).

1.2. Justificación de la tesis

Se plantean los antecedentes prácticos que originan el problema que soluciona la tesis doctoral, explicando los escenarios de motivación y presentando la pregunta de investigación. También se expone la relevancia y pertinencia de la investigación, y se plantean los objetivos del trabajo.

1.2.1. Planteamiento del problema

Actualmente la mayoría de los procesos organizacionales e industriales se soportan en sistemas software (Van Geet et al., 2010), que requieren de un profundo conocimiento de su arquitectura e implementación al momento de realizarles mantenimiento para garantizar su evolución (Platenius et al., 2012; von Detten, 2012; Bennett y Rajlich, 2000). Este conocimiento se encuentra registrado en forma explícita en la documentación o de manera implícita en la mente de los expertos que lo desarrollaron. Desafortunadamente, con frecuencia no se dispone de dicho conocimiento, sobre todo cuando se trata de sistemas heredados que carecen de documentación o se encuentra desactualizada y no se cuenta con el equipo de expertos que lo crearon (Nierstrasz et al., 2005).

La situación se hace más compleja debido al ritmo acelerado del cambio, que se manifiesta con el surgimiento de nuevos paradigmas y tecnologías, mayores exigencias de calidad y nuevos requerimientos (Lehman et al., 1997), que obliga a la mayoría de las compañías a enfrentarse al problema de tener que modernizar o reemplazar sus sistemas. Estos sistemas representan inversión económica, de tiempo y otros recursos. Muchos de ellos son críticos para el negocio de la organización y reemplazarlos representa un alto riesgo (Favre et al., 2009).

Cuando se trata de sistemas heredados que han sido desarrollados durante varios años, en oportunidades más de dos décadas, que son complejos por su volumen y su heterogeneidad porque combinan diferentes tecnologías; se debe garantizar la inversión que se ha hecho actualizándolos sin causar traumas en los procesos organizacionales (Jouault et al., 2009) y conservando la familiaridad del producto (Lehman et al., 1997).

Pero el problema no sólo se presenta en el proceso de mantenimiento. Con frecuencia se requiere mantener la trazabilidad de los distintos artefactos que se producen en el ciclo de desarrollo de software, por ejemplo, para efectos de control de calidad se requiere garantizar la coherencia entre el código fuente y los modelos arquitectónicos y de diseño propuestos (Müller et al., 2000).

Adicionalmente, se debe tener en cuenta que en los últimos años la práctica del desarrollo de software incluye la reutilización de código abierto, el cual se caracteriza

por contar con poca o nula información sobre su arquitectura (Constantinou et al., 2011). Esto debe hacerse siempre y cuando se mantenga el debido respeto a los derechos de autor, lo que a su vez ha llevado a la necesidad de contar con herramientas que permitan la detección de clonación de código (Canfora et al., 2011).

Del mismo modo, las actividades ilegales en el ciberespacio afectan la seguridad nacional y amenazan el derecho de los ciudadanos a la privacidad, lo que tiene significantes implicaciones políticas, económicas y sociales (Choo, 2008). Estas actividades han sido patrocinadas por el crimen organizado, que ha contratado hackers que desarrollan software malicioso o malware (virus, troyanos, gusanos, etc.) que se instala en el computador sin autorización del propietario, lo que exige mecanismos rápidos que faciliten el análisis y el entendimiento del código malicioso, para dar una respuesta oportuna sobre cómo bloquear y eliminar una determinada pieza de malware (Treude et al., 2011; Ligh et al., 2010, Sharif et al., 2009).

Por otra parte, existe una nueva área de aplicación en la Ingeniería dirigida por modelos (MDE Model Driven Engineering), denominada modelos en tiempo de ejecución, que consiste en que el sistema y los modelos que lo representan deben coexistir en el tiempo y mantenerse sincronizados (Jouault et al., 2009); a diferencia de la ingeniería directa, donde los sistemas han sido creados a partir de modelos, y de la ingeniería inversa, donde los modelos se extraen a partir del sistema.

Por último, pero no menos importante, la industria del desarrollo de software se encuentra en crecimiento y necesita de especialistas altamente calificados, sin embargo, la gran mayoría de instituciones académicas utilizan métodos tradicionales en los procesos de aprendizaje de la ingeniería de software, desaprovechando la oportunidad de aprender a partir del análisis de productos software existentes (Klimek et al., 2011; Cipresso, 2009; Ali, 2006; Ali, 2005).

La ingeniería inversa se ha convertido en la principal solución a las situaciones expuestas, relacionadas con la adquisición de conocimiento para la reconstrucción de documentación a nivel de diseño y arquitectura, el control de la calidad del producto software manteniendo la trazabilidad de los artefactos que lo conforman, la reutilización de código y el control de clonación no autorizado, el análisis y entendimiento del código malicioso, la coexistencia sincronizada del sistema y los modelos que lo representan, y el apoyo al aprendizaje de la ingeniería de software. Esto ha hecho que la ingeniería inversa sea uno de los mayores retos actuales de la ingeniería de software (Favre et al., 2009).

Las dos últimas décadas han representado grandes avances para la ingeniería inversa (Canfora et al., 2011; Tonella et al., 2007), sin embargo, aún existen desafíos que le impiden dar solución de manera completa a las situaciones expuestas. Aunque se han desarrollado métodos y herramientas que permiten recuperar distintas vistas del sistema, la mayoría lo hacen realizando análisis estático y en menor cantidad mediante análisis dinámico (Monroy et al., 2012; Tonella et al.,

2007). Además, es muy limitada la capacidad para consultar la información obtenida en los artefactos (piezas de información) generados por los analizadores (Canfora et al., 2011).

La utilidad real de un proceso de ingeniería inversa se evidencia en el uso que se le dé a los artefactos recuperados (Canfora et al., 2011; Tonella et al., 2007; Müller et al., 2000), por lo tanto existe la necesidad de mejorar la capacidad para consultar la información obtenida en los artefactos generados, para que sea posible realizar tareas de análisis pertinentes al contexto. Esta situación lleva a plantearse la siguiente pregunta:

¿Cómo extraer información a partir de los artefactos generados en un proceso de ingeniería inversa, para realizar análisis a nivel de arquitectura sobre el comportamiento de un sistema software?

1.2.2. Justificación

En este orden de ideas, al permitir realizar análisis sobre la información obtenida a partir de un proceso de ingeniería inversa se contribuye al logro del principal objetivo de esta disciplina, porque se facilita: la adquisición de conocimiento de sistemas implementados a través de la reconstrucción de documentación a nivel de diseño y arquitectura, el control de la calidad del producto software manteniendo la trazabilidad de los artefactos que lo conforman, la reutilización de código y el control de clonación no autorizado, el análisis y entendimiento del código malicioso, la validación de la coexistencia sincronizada del sistema y los modelos que lo representan, además, se permitiría apoyar el aprendizaje de la ingeniería de software.

Por lo tanto, al responder la pregunta planteada se beneficia la comunidad científica dedicada al estudio de la ingeniería de software, por el aporte que representa para la ingeniería inversa el dar respuesta a uno de los desafíos que afronta actualmente (Canfora et al., 2011). Esto a su vez se traduce en un beneficio para la comunidad dedicada al desarrollo de software, porque se facilita el control de la calidad del producto software al permitir una mejor trazabilidad de los artefactos que lo conforman y la validación de la coexistencia sincronizada del sistema y los modelos que lo representan. De igual manera, facilita la reutilización de código y el control de clonación no autorizado.

Por otra parte, también podrían recibir beneficios los especialistas en seguridad, en la medida en que se facilita el análisis y entendimiento del código malicioso, haciendo más eficiente y productivo su trabajo. No menos importante es el beneficio que recibe la comunidad académica, cuyo tema de estudio es la ingeniería de software, porque se facilita el aprendizaje a través de técnicas de ingeniería inversa sobre productos software existentes, que sirvan como referente de estudio y sobre los cuales se pueden realizar distintos tipos de análisis.

Por todo lo anterior, se beneficia la industria del software, uno de los pocos sectores de la economía que no se ha visto afectado por la crisis (SiliconWeek, 2012; Fedesoft, 2012), lo que la convierte en una posibilidad para las economías emergentes, como es el caso de Colombia, manteniendo coherencia con la política del gobierno, reflejada en el marco del Plan Vive Digital y del Programa Nacional de Gobierno en Línea del Ministerio de Tecnologías de la Información y las Comunicaciones MINTIC, que plantea apoyar el programa de Transformación Productiva para los sectores de software y tecnologías de la información (Ministerio de las TIC, 2011).

1.2.3. Objetivos

Objetivo general

Definir un marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento de sistemas software.

Objetivo específicos

- Definir un modelo conceptual para abstraer artefactos de bajo nivel y reconstruir información de alto nivel sobre el comportamiento de sistemas software, integrando técnicas de recuperación de arquitectura.

- Diseñar un mecanismo de consulta para realizar análisis a nivel de arquitectura sobre el comportamiento de sistemas software, a partir de la base de información obtenida en los artefactos generados por los analizadores en el proceso de ingeniería inversa.

- Evaluar el modelo conceptual y el mecanismo de consulta mediante la construcción de un prototipo y la realización de un estudio de caso de ingeniería inversa.

1.3. Contribución de la tesis

La contribución de la tesis doctoral se clasifica en dos grupos. En el primero se incluyen los aportes teóricos y en el segundo los aportes empíricos, como se explica a continuación. Los aportes teóricos hacen referencia a aquellas contribuciones orientadas a enriquecer o estructurar el conocimiento existente sobre la recuperación y análisis de vistas arquitectónicas de comportamiento, mientras que los aportes empíricos están representados por los instrumentos y herramientas resultado de la investigación.

El principal aporte teórico es el Marco de referencia para la recuperación y análisis de vistas arquitectónicas (ARCo), conformado por:

1. Un modelo conceptual para abstraer artefactos de bajo nivel y reconstruir información de alto nivel sobre la estructura y el comportamiento de sistemas software, fundamentado en el estándar ISO/IEC/IEEE 42010 para la descripción de arquitectura, el estándar ISO/IEC 19506 que define el meta-modelo para representar activos de productos software implementados, y el estándar del Lenguaje Unificado de Modelado UML; utilizando el lenguaje XML para garantizar la integración con otras propuestas (Capítulo 3).
2. La Metodología para la recuperación y análisis de vistas arquitectónicas, teniendo en cuenta el contexto en el que se presenta el problema objeto de estudio, definida a partir de la integración de técnicas identificadas en la revisión de la literatura (Capítulo 5).
3. La definición y caracterización de los contextos de uso de la ingeniería inversa, para guiar el proceso de recuperación y análisis vistas arquitectónicas, atendiendo las particularidades del contexto en el que se genera el problema objeto de estudio (Capítulo 2).
4. El diseño del mecanismo de consulta para realizar análisis a nivel de arquitectura de sistemas software, a partir de la información base obtenida en los artefactos generados por los analizadores en el proceso de ingeniería inversa (Capítulo 4).
5. El modelo ontológico para la selección de herramientas de ingeniería inversa, teniendo en cuenta las características del contexto de uso (Capítulo 5) (Monroy et al., 2016).

En forma complementaria se lograron los siguientes aportes teóricos, que representan contribuciones orientadas a estructurar el conocimiento existente sobre la recuperación y análisis de vistas arquitectónicas:

1. El Modelo de caracterización de los mecanismos de consulta (Capítulo 4).
2. El Modelo de caracterización del proceso de recuperación y análisis de vistas arquitectónicas(Capítulo 5).
3. La estructura de caracterización de las herramientas de ingeniería inversa (Capítulo 5) (Monroy et al., 2012).
4. El inventario de propuestas existentes para la recuperación de arquitecturas (Anexo A.).
5. El inventario de herramientas de ingeniería inversa (Anexo B.).
6. El Modelo de dominio de la recuperación de arquitecturas (Capítulo 3).
7. Documento sobre el marco teórico que incluye la base teórica y conceptual que soportan ARCo (Anexo digital "Marco Teórico,pdf").
8. Un mapeo sistemático de la literatura sobre recuperación de arquitecturas (Monroy et al., 2016).

Por otra parte, los aportes empíricos obtenidos como resultado de la tesis doctoral están representados por:

1. QModel-XMI, un prototipo que implementa el mecanismo de consulta propuesto, que responde a preguntas en lenguaje natural escritas en español, para apoyar el análisis de modelos UML representados en repositorios XMI.
2. El diseño, los resultados y la base de datos del estudio de caso, con sus dos unidades de análisis, que pueden ser usados como guía para la aplicación de ARCo.
3. Las lecciones aprendidas relacionadas en el capítulo de las conclusiones.

1.4. Organización de la tesis

El informe de la tesis doctoral se encuentra organizado de la siguiente manera: El primer capítulo es introductorio, los capítulos 2,3,4 y 5 explican los resultados de la investigación, el capítulo 6 presenta la evaluación de la tesis y en el capítulo 7 se plantean las conclusiones y recomendaciones. En la parte final se relacionan los referentes bibliográficos y se incluyen los anexos que complementan este documento.

Inicialmente, en el primer capítulo se presenta el estado del arte, resaltando los antecedentes teóricos, los logros y los retos. Posteriormente se exponen los antecedentes prácticos en el planteamiento del problema, se señala la pertinencia y relevancia de la investigación en la justificación, y se plantean los objetivos de la tesis. Luego se enuncian las contribuciones del trabajo y se finaliza explicando la forma cómo ha sido organizado el informe.

Posteriormente, en el segundo capítulo se explica el Marco de referencia para la recuperación y análisis de vistas arquitectónicas, presentando una visión general de su estructura, los lineamientos que determinaron su creación y los que se establecieron para su utilización. En la parta final de este capítulo se conceptualiza sobre los contextos de uso de la ingeniería inversa y se caracterizan los contextos que se identificaron como resultado de la investigación.

Luego se expone el Modelo conceptual del Marco de referencia para la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, explicando inicialmente el modelo de dominio de la recuperación de arquitecturas, presentando luego la visión holística del modelo conceptual, y cerrando el capítulo tres con la explicación de la visión detallada del modelo, resaltando los artefactos y las actividades que lo conforman.

A continuación se presenta el mecanismo de consulta en el cuarto capítulo, exponiendo la caracterización que se definió a partir de la revisión de la literatura, y señalando los lineamientos que se establecieron como referentes para el mecanismo. También se explica su diseño, determinando como objetivo principal del mecanismo, el contribuir al análisis de las vistas arquitectónicas, recuperadas en un proceso de ingeniería inversa, teniendo en cuenta el contexto de uso en que origine el problema.

Más adelante se explica la Metodología para la recuperación y análisis de vistas arquitectónicas, en el quinto capítulo presentando los axiomas que la fundamentan y su estructura. También se caracterizan las propuestas identificadas como resultado de la revisión de la literatura, y se detalla el proceso de recuperación y análisis de la arquitectura, en función de sus fases, hitos y actividades, entre otras características. Al final de este capítulo se expone la caracterización de herramientas de ingeniería inversa, que se logró con base en la revisión de la literatura y se explica el Modelo ontológico que se propone para los contextos de uso de este tipo de herramientas.

En el capítulo seis se documenta la evaluación del modelo conceptual y del mecanismo de consulta resultado de la tesis doctoral, explicando el diseño y la implementación del prototipo QModel-XMI, y presentando el diseño del estudio de caso que se realizó, y los resultados que se obtuvieron para cada unidad de análisis. El capítulo finaliza con la presentación del análisis que se hizo sobre los resultados del estudio de caso.

Finalmente, en el capítulo 7 se presentan los juicios que plantea el autor de la tesis al terminar el trabajo de investigación, presentando las conclusiones, en respuesta a la pregunta de investigación y a los objetivos trazados; algunas recomendaciones representadas en trabajos futuros, tendientes a resolver nuevas preguntas de investigación; y las lecciones aprendidas en el desarrollo de la tesis.

Adicionalmente se presenta la bibliografía que soporta los resultados presentados en la tesis doctoral, y se incluyen los anexos que complementan la documentación de este trabajo de investigación, organizados en dos grupos. El primer grupo de anexos se incluyen en forma física dentro de este informe, y está conformado por el inventario de propuestas para la recuperación de arquitecturas, el inventario de herramientas de ingeniería inversa, y el resumen de cada una de las fases de la metodología. El segundo grupo de anexos se presenta en formato digital, estructurado en dos carpetas: La primera contiene la base de datos del estudio de caso y la segunda los instaladores y el manual de usuario del mecanismo de consulta. Adicionalmente se incluye una versión digital de la monografía y el archivo que tiene el Marco teórico.

Capítulo 2

El Marco de referencia

Según Goffman (1986) un marco de referencia "es un esquema interpretativo que simplifica y condensa el mundo exterior al señalar y codificar selectivamente los objetos, situaciones, acontecimientos, experiencias y las acciones que se han producido en el entorno". Está conformado por un conjunto de elementos conceptuales e instrumentos, que hacen posible la interpretación de la naturaleza del objeto de estudio dentro del ambiente que le rodea.

En este capítulo se describe el Marco de referencia para la recuperación y análisis de vistas arquitectónicas de sistemas software, que se definió como resultado de la investigación. Inicialmente se presenta la visión general del marco de referencia y posteriormente se describen los instrumentos que lo conforman. Los elementos conceptuales correspondientes a la base conceptual y la base teórica se presentan en el anexo digital "Marco Teórico.pdf", y el modelo conceptual se detalla en el capítulo 3 de este documento.

2.1. Visión general

El marco de referencia que se definió, permite abordar la recuperación y análisis de vistas arquitectónicas desde una perspectiva holística, orientada por el contexto que origina la situación que motiva el proceso de recuperación y análisis, por eso se denomina ARCo, por su sigla en inglés Architecture Recovery in Context . Dispone de un sistema conceptual de tipo descriptivo, conformado por una base conceptual, una base teórica y un modelo conceptual, representados en la Figura 1 de color azul, que se derivan de supuestos teóricos sobre la naturaleza del proceso de recuperación de la arquitectura y la forma de abordarlo.

La parte instrumental del marco de referencia, representada de color gris en la Figura 1, comprende un conjunto de lineamientos, que determinan las directrices que orientaron la definición del marco de referencia y la forma cómo debe utilizarse; una metodología que utiliza técnicas y herramientas de ingeniería inversa, y define la forma cómo realizar el proceso de recuperación y análisis siguiendo lineamientos,

para cumplir los propósitos del contexto específico en el que se realiza el proceso. Para diferenciar los aportes hechos en la tesis, en la Figura 1 el nombre de los elementos de ARCo que representan una contribución al estado del arte, aparecen en color rojo.

El principal objetivo de este marco de referencia es apoyar el trabajo de las personas que realizan actividades de recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, teniendo en cuenta el contexto en el que se presenta la situación objeto de estudio, para que los resultados sean más pertinentes, eficientes y precisos; y asumiendo que no todos los interesados son expertos en ingeniería inversa.

La utilidad del marco de referencia depende de la interpretación que se haga de él, por eso, para evitar ambigüedades se elaboró un glosario que representa su base conceptual, en el que se define cada uno de los términos utilizados (ver anexo digital "Marco Teórico.pdf"). Esta base conceptual también sirve para comprender en forma más clara la base teórica del marco de referencia, presentada en el anexo digital "Marco Teórico.pdf", que a su vez sirve como fundamento para la definición, el entendimiento y la utilización del marco de referencia propuesto.

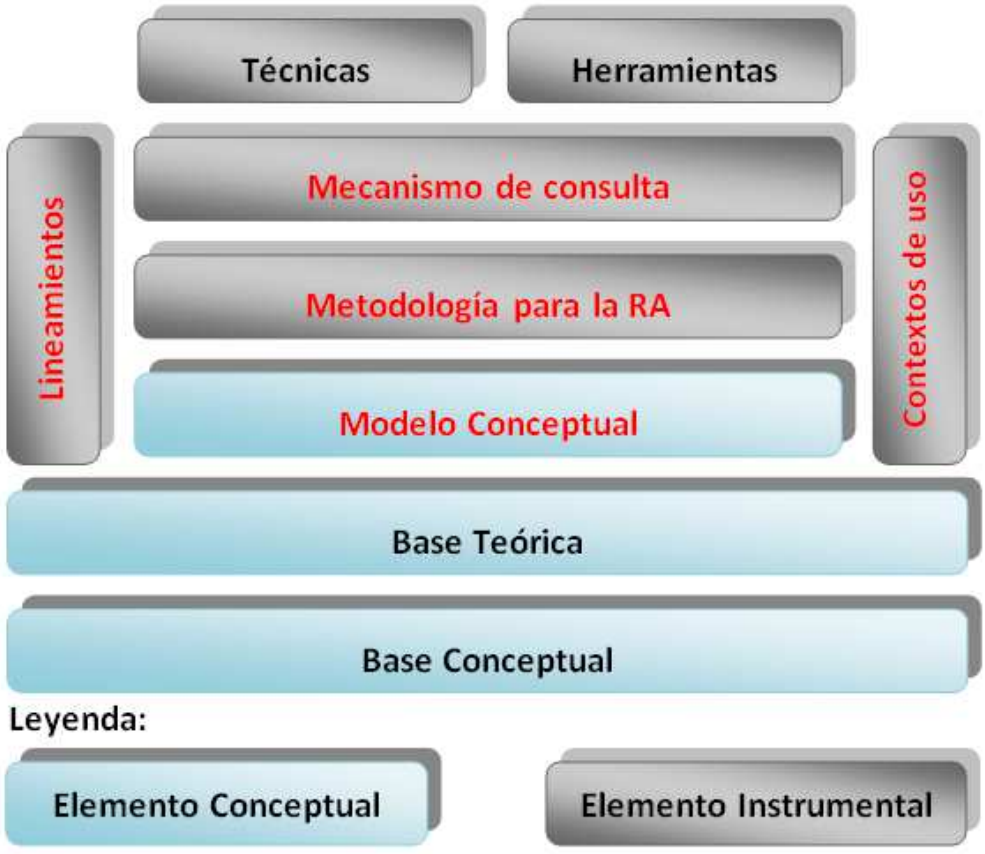

Figura 1. ARCo

Los lineamientos que guían la definición y el uso del marco de referencia se exponen a continuación, más adelante se definen los contextos de uso de la ingeniería inversa. Para facilitar la comprensión del marco de referencia, se simplificó la base conceptual y la base teórica en un modelo conceptual, que se explica detalladamente en el capítulo 3; posteriormente se presenta el mecanismo de consulta en el capítulo 4; y la metodología para la recuperación y análisis de vistas arquitectónicas se expone en el capítulo 5.

2.2. Lineamientos

Los retos identificados en el estado del arte evidencian la necesidad de establecer instrumentos conceptuales y técnicos, que permitan hacer un uso pertinente de la ingeniería inversa en procesos de recuperación y análisis de la arquitectura de productos software. Bajo esta circunstancia se definió el marco de referencia, siguiendo una serie de directrices que se enuncian en la primera parte de esta sección, para garantizar su solidez científica.

De igual forma, para que el marco de referencia definido sea utilizado de manera eficiente, se han establecido un conjunto de lineamientos para su uso, los cuales se presentan en la segunda parte de esta sección. La integración de estas directrices facilita la interpretación, comprensión y utilización del marco de referencia propuesto.

2.2.1. Lineamientos para la definición

El marco de referencia se define como resultado de la revisión de la literatura (Monroy et al. 2016), estableciendo su fundamento epistemológico en la base conceptual, la base teórica y el modelo conceptual. Su principal objetivo es apoyar el trabajo de recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, que realizan aquellas personas involucradas en esta actividad, teniendo en cuenta el contexto en el que se presenta la situación objeto de estudio, para que los resultados sean más pertinentes, eficientes y precisos; y asumiendo que no todos los interesados son expertos en ingeniería inversa.

Su ámbito comprende la recuperación y análisis de vistas arquitectónicas de productos software orientados a objetos, independientemente de las tecnologías utilizadas para su desarrollo y despliegue. Se diferencia de las propuestas identificadas en la revisión de la literatura, porque amplía los contextos de uso de la ingeniería inversa, a campos del conocimiento como la educación, las seguridad informática y la computación forense, sin limitarse exclusivamente a ellos.

Para garantizar todas las ventajas que representan el uso de estándares, el marco de referencia se elaboró con base en los siguientes: ISO/IEC/IEEE 42012:2011 para la descripción de la arquitectura, ISO/IEC 19506:2012 como meta-modelo para el descubrimiento del conocimiento; y las especificaciones: OMG UML 2.5:2015, OMG XML Metadata Interchange 2.5.1:2015. El modelo conceptual y la metodología se definieron como resultado de la integración de las propuestas identificadas en la

revisión de la literatura, asumiendo como pasos centrales del proceso los establecidos por Tilley et al. (1996), porque en esencia todas las propuestas los incorporan.

También se han tenido en cuenta los factores que influyen en la recuperación de la arquitectura (Gall et al., 1996), que se pueden clasificar en: 1) tecnológicos representados por el uso de patrones arquitectónicos, paradigmas de programación, tecnologías específicas, etc.; 2) de carácter humano, manifiestos en las habilidades, la experiencia y el conocimiento sobre el campo del dominio del problema y sobre la ingeniería inversa; 3) la disponibilidad de recursos como herramientas de ingeniería inversa, instrumentos de apoyo para el proceso, documentos, estándares, entre otros; y 4) los contextos de uso (Monroy et al., 2016) explicados más adelante en este capítulo.

Aunque el marco de referencia se definió centrando la atención en la recuperación y análisis de vistas arquitectónicas de comportamiento, también puede ser utilizado para recuperar la estructura del sistema, porque en su formulación se tuvo en cuenta tanto los enfoques que recuperan directamente el comportamiento (Peake et al., 2013; Callo Arias et al., 2011; Abi-Antoun y Barnes, 2010; Abi-Antoun y Aldrich, 2009; Zhou, 2008; Huang et al., 2006; Schmerl et al., 2006; Bojic y Velasevic, 2000; Cook, 1998), como los enfoques que primero recuperan la estructura y a partir de los resultados de esta actividad recuperan el comportamiento (Forster et al., 2013; Mendoca y Kramer, 2001), por eso, en algunas partes de este documento se utiliza el término recuperación del conocimiento, para hacer referencia a la reconstrucción tanto del diseño como de la arquitectura en sus aspectos estructurales y de comportamiento.

La metodología propuesta obedece a un proceso genérico que apoya la recuperación y análisis de vistas arquitectónicas, teniendo en cuenta el contexto en que se presenta la situación objeto de estudio. En su definición se especifican las actividades que la conforman, indicando la secuencia lógica de su ejecución, las metas que se proponen, los recursos que reciben en calidad de entrada y los que generan como salida, así como también los instrumentos que se utilizan y las técnicas que aplican para lograr las metas propuestas. La metodología no define ninguna técnica nueva, sólo integra las existentes para que su uso sea pertinente al contexto y la situación en que se presenta el problema. Los lineamientos del mecanismo de consulta se definen en el capítulo 4 de este documento.

2.2.1. Lineamientos para la utilización

El marco de referencia tiene dos usos principales, el primero corresponde al propósito para el cuál fue elaborado: la recuperación y análisis de vistas arquitectónicas, lo cual se logra aplicando la metodología propuesta y usando las herramientas, instrumentos y técnicas que lo conforman. El segundo uso consiste en servir como instrumento didáctico para el proceso de aprendizaje de la ingeniería inversa. Esto se logra como efecto colateral, demostrado con la segunda unidad de

análisis del estudio de caso realizado para evaluar el marco de referencia (ver resultados del estudio de caso).

Para lograr la recuperación y análisis de las vistas arquitectónicas se usa la estrategia cognitiva dirigida por problemas (Kruger y Cross, 2006), que consiste en centrar la atención directamente en el problema en cuestión, utilizando sólo la información y el conocimiento que sea estrictamente necesario para resolverlo, haciendo énfasis en la definición del problema y en encontrar una solución lo más pronto posible. Por eso la importancia de definir inicialmente el contexto en el que se presenta la situación.

Para lograr mejores resultados en el uso del marco de referencia se recomienda:

1. Tener claridad sobre la base conceptual y la base teórica.
2. Comprender el modelo conceptual propuesto.
3. Conocer las técnicas, herramientas e instrumentos.
4. Aplicar la metodología usando las técnicas e instrumentos propuestos para cada actividad.
5. Contar con expertos en el dominio del problema y de la aplicación.

Por otra parte, se debe tener en cuenta que al ser un marco de referencia que contempla varios contextos de uso, se corre el riesgo de ser muy genérico, por lo tanto, para cada contexto de uso se requiere de un análisis detallado que complemente sus particularidades. También se registra que, aunque el marco de referencia no está diseñado para desarrolladores de herramientas de recuperación de arquitectura, puede servir para identificar nuevas funcionalidades, acordes a los contextos de uso de la ingeniería inversa.

2.3. Contextos de uso

Desde su génesis la ingeniería inversa ha sido utilizada como técnica de mantenimiento de software (Bourque y Fairley, 2014), sin embargo, a lo largo de su evolución ha servido para resolver varios problemas que afronta la ingeniería de software, tales como: la recuperación de arquitecturas y patrones de diseño, la re-documentación de programas y bases de datos, la identificación de activos reutilizables, la definición de la trazabilidad entre artefactos de software, la identificación del impacto de los cambios del producto software, la reestructuración de sistemas existentes, la renovación de interfaces de usuario, la migración hacia nuevas arquitecturas y plataformas, entre otros (Canfora et al., 2011).

A partir de su extensa experiencia Stoermer et al. (2003) identificaron varios contextos en los que se puede aplicar la recuperación de arquitecturas, que incluyen: mejorar el entendimiento de la arquitectura de sistemas software existentes, mejorar la arquitectura como tal, evaluar los atributos de calidad de estos sistemas, mejorar la documentación de la arquitectura de estos sistemas, entre otros. Sin embargo, además del campo del desarrollo del software, la ingeniería inversa también se ha

utilizado en otros contextos, como la educación (Klimek et al., 2011; Cipresso, 2009; Ali, 2006; Ali, 2005), la seguridad informática (Treude et al., 2011; Ligh et al., 2010, Sharif et al., 2009) e incluso en la computación forense (Nelson et al., 2014; Mueller, 2010; Ligh et al., 2010), como se explica más adelante al final de esta sección.

No obstante, hasta ahora, todas las propuestas para recuperar las vistas arquitectónicas están orientadas exclusivamente al ámbito del desarrollo del software, limitando la posibilidad de obtener mayores y mejores beneficios, al analizar la situación dentro del ámbito del contexto particular en el que se presenta la necesidad, para orientar el proceso de recuperación y análisis desde los propósitos del contexto específico del problema, para que la solución sea más pertinente, eficiente y productiva.

Por eso ARCo, el marco de referencia propuesto, está orientado a los contextos de uso de la ingeniería inversa, en respuesta al reto identificado en la revisión de la literatura, referente a la necesidad que existe de nuevos enfoques de recuperación de arquitecturas, teniendo en cuenta los nuevos contextos de uso. Para hacer mayor claridad sobre este aspecto, a continuación se explica lo que significa el contexto para la tesis, y luego se presentan los contextos de uso de la ingeniería inversa, que se identificaron como resultado de la investigación.

2.3.1. El Contexto

Según la real academia de la lengua española, el contexto es el entorno físico o de situación, político, histórico, cultural o de cualquier otra índole, en el que se considera un hecho. El contexto puede ser generalmente descrito como un conjunto de estados físicos y conceptuales de interés a una entidad particular (Pascoe, 1998) y tiene tres aspectos importantes (Schilit et al., 1994): ¿dónde está?, ¿con quién está? y ¿qué recursos hay cerca?. Basado en esto Dey (2001) define el contexto como: "cualquier información que puede ser utilizada para caracterizar la situación de una entidad".

El entorno de un sistema está determinado por el conjunto de circunstancias que influencian el sistema, incluyendo aspectos tecnológicos, organizacionales, de negocio, operacionales, políticos, económicos, legales, económicos e influencias sociales (ISO/IEC/IEEE 42010:2011). El entorno físico o de situación, que responda a las preguntas planteadas por Schilit et al. (1994), implica la existencia de un espacio comprendido dentro de los límites determinados por el contexto, o sea un *ámbito*, que para este caso representa el espacio ideal configurado por las cuestiones y los problemas de una o varias actividades o disciplinas relacionadas entre sí, por ejemplo la seguridad informática, la producción de software, la computación forense y la educación.

La *situación* implica un estado o constitución de las cosas o personas, determinado por el conjunto de factores o circunstancias que afectan a alguien o algo en un determinado momento, por lo tanto existen uno o varios interesados que manifiestan sus propósitos sobre el contexto. Un *interesado* es Individuo, grupo, organización (o

clases de ellos) que tienen un interés sobre lo que sucede en el contexto (ISO/IEC/IEEE 42010:2011). El *propósito* hace referencia al conjunto de intenciones que tienen los interesados con respecto al contexto.

Para el cumplimiento de los propósitos, atendiendo a la tercera pregunta planteada por Schilit et al. (1994), dentro del contexto existen *recursos*, entendidos como un conjunto de elementos disponibles para resolver una necesidad o cumplir con un propósito. En la Figura 2 se presenta un esquema que resalta los aspectos relevantes de este concepto para el marco de referencia.

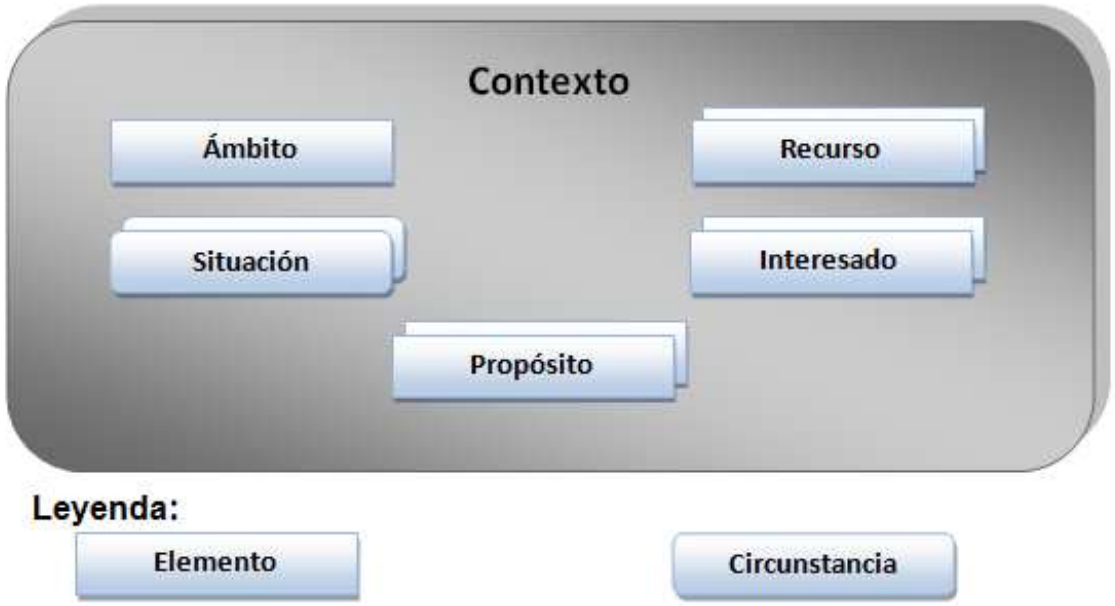

Figura 2. El contexto

2.3.2. Contextos de uso de la ingeniería inversa

La evolución de la ingeniería inversa durante las últimas dos décadas ha permitido ampliar su campo de aplicación, que inicialmente sólo se limitaba al proceso de mantenimiento de software, a distintos contextos de uso como: La producción de software, la seguridad informática, la computación forense y la educación. Para cada uno de estos contextos a continuación se describe la forma como es utilizada la ingeniería inversa (Monroy et al., 2016).

1) *Producción de software*: Comprende todos los procesos que definen el ciclo de vida del software, establecidos en el estándar ISO/IEC 12207:2008. La ingeniería inversa se utiliza concretamente en los siguientes procesos:

i) Mantenimiento de software: El estándar para el mantenimiento de software IEEE – 1219 recomienda la ingeniería inversa como la principal estrategia para tratar

sistemas cuya única representación fiable es el código fuente (Canfora et al., 2011), debido a que la comprensión del software consume una parte sustancial del esfuerzo de mantenimiento. Adicionalmente se utiliza para la identificación de errores, al permitir examinar el sistema con el fin de localizar posibles defectos que presente, facilitando su modificación y actualización al implementar cambios para mejorarlo o ajustarlo a las nuevas necesidades, convirtiéndose en un paso obligatorio de los procesos de reingeniería (Chikofsky y Cross, 1990).

ii) Documentación: La ingeniería inversa se convierte en la principal estrategia para reconstruir la documentación de un sistema software, al lograr una representación de más alto nivel que permita su comprensión al recuperar las vistas de diseño y arquitectura, los requerimientos e incluso los modelos del negocio (Van Geet et al., 2010). También se puede utilizar para mantener el control de versiones en sistemas legados a partir de sus implementaciones, porque facilita la recuperación de artefactos en sistemas que han sido desarrollados hace mucho tiempo y que son difíciles de actualizar, haciendo posible establecer diferencias entre las distintas versiones que ha presentado el producto software (El Boussaidi et al., 2012; Van Geet et al., 2010).

iii) Verificación de software: La ingeniería inversa es una excelente herramienta de control de calidad al momento de verificar la coherencia y trazabilidad entre el código fuente y: los modelos de diseño, la arquitectura, el modelo de negocio y los requisitos establecidos para del producto software. Desde la perspectiva del enfoque conocido como Arquitectura Dirigida por Modelos (MDA), se plantea la necesidad de mantener los modelos en tiempo de ejecución, lo cual consiste en sincronizar los modelos y el código fuente de tal forma que al modificarse uno, el otro se actualice respondiendo a dicho cambio, lo cual es posible gracias a la ingeniería inversa (Jouault et al., 2009).

iv) Reutilización de activos: Este proceso consiste en un conjunto de actividades que soportan la habilidad que tiene la organización para adquirir, desarrollar y gestionar activos software reutilizables. La ingeniería inversa juega un papel importante al momento de identificar piezas de software que puedan ser utilizadas en nuevas soluciones a partir de desarrollos previos, evitando así la necesidad de volver a implementarlas (Vasconcelos y Werner, 2011). Por otra parte, cada vez es mayor la importancia que toma la ingeniería inversa en contextos de líneas de producción de software, sobre todo al momento de mantener la consistencia de la arquitectura de la familia de productos (Wu et al., 2011; Stoermer y O'Brien, 2001).

2) *Seguridad informática*: Como disciplina se encarga de definir normas, procedimientos, métodos y técnicas destinados a garantizar que un sistema de información sea seguro y confiable. En este campo se utiliza la ingeniería inversa con el fin de analizar y entender el código malicioso que pueda estar presente en el sistema, para facilitar la definición de estrategias a seguir en la solución del riesgo e inconvenientes que genera (Treude et al., 2011; Ligh et al., 2010, Sharif et al., 2009). También se puede utilizar para identificar posibles fallas de vulnerabilidad en la seguridad de productos software, como lo demuestran Abi-Antoun y Barnes (2010).

3) *Computación forense*: Noblett y Pollitt (2000) la definen como la ciencia de adquirir, preservar, recuperar y presentar datos que han sido procesados electrónicamente y guardados en un medio computacional, con el fin de interpretarlos y construir evidencia que permita establecer los hechos y formular hipótesis relacionadas con el caso de estudio. Para lograr este propósito es necesario desarrollar múltiples habilidades en áreas como el análisis de código malicioso, la ingeniería inversa de software y análisis forense de dispositivos móviles (Nelson et al., 2014, ; Ligh et al., 2010), como lo evidencia Robert S. Mueller (2010) director de la oficina federal de Investigación del FBI, en la conferencia sobre seguridad cibernética en San Francisco, California, donde mencionó experiencias de uso de la ingeniería inversa para atender casos de terrorismo cibernético.

4) *Educación*: Para el Constructivismo Filosófico de Kant, el principal postulado indica que "el conocimiento humano no se recibe de forma pasiva sino que, más bien, es procesado y construido de una forma activa por el individuo que realiza la actividad de conocer y que, gracias a su aparato cognitivo, puede ir adaptando y modificando el objeto de estudio sobre el cual actúa, lo que permite al conocedor, (alumno o aprendiz, hablando en términos de aprendizaje) organizar su mundo, interactuar con él y registrar sus experiencias desde una perspectiva individual y vivencial" (Flórez, 1994, p.235, citado por Heredia 2012).

En este sentido, la ingeniería inversa se convierte en una herramienta didáctica que posibilita (Klimek et al., 2011; Cipresso, 2009; Ali, 2006; Ali, 2005):

i) El aprendizaje a partir de éxitos y fallas reales, porque hace posible el uso de productos software para mostrarle a los estudiantes el diseño y la arquitectura de los casos de éxito con el fin de replicarlos, así como también los fracasos para evitarlos.

ii) El desarrollo de la curiosidad, estimulando el interés de los estudiantes al evidenciar la importancia de los modelos en los procesos de desarrollo de software, tomando como referentes implementaciones reales.

iii) La recuperación del conocimiento representado en productos software desarrollados, para facilitar la comprensión de conceptos en el campo de la ingeniería de software, como es el caso de la identificación de patrones aplicando técnicas de ingeniería inversa (Tonella et al., 2007).

iv) El desarrollo de habilidades para modelar, programar y hacer mantenimiento a productos software a partir implementaciones reales.

v) La corrección de errores en el proceso de aprendizaje, mediante la utilización de herramientas de ingeniería inversa que permitan identificar posibles errores de diseño, o identificar inconsistencias entre los modelos de diseño y la implementación realizada.

En este orden de ideas, cada contexto de uso tiene un ámbito específico determinado por sus propósitos y situaciones, en el que se generan problemas que

exige a los interesados realizar procesos de recuperación y análisis de la arquitectura de productos software, guiados por los lineamientos definidos para el marco de referencia y aplicando la metodología que se establece en la siguiente sección. En la Tabla 3 se presenta una caracterización de los contextos de uso de la ingeniería inversa identificados en esta investigación, que se obtuvo como resultado del análisis de la literatura.

Contexto	**Producción de software**		
Ámbito	Ingeniería de software		
Propósito general	Construir software de calidad		
Posibles situaciones	**Propósitos específico**	**Interesados**	**Recursos**
Mantenimiento de software	Entender el producto software	Responsables de mantenimiento, arquitecto	El producto software, documentación, IDE
Documentación	Recuperar la documentación del software	Responsables de documentación, arquitecto	El producto software
Verificación	Validar la conformidad entre la arquitectura conceptual y la arquitectura concreta	Responsables de calidad, arquitecto	El producto software, documentación
Reutilización de activos	Identificar activos reutilizables	Arquitecto	El producto software
Contexto	**Seguridad informática**		
Ámbito	Seguridad		
Propósito general	Definir normas, procedimientos, métodos y técnicas destinados a garantizar que un producto software sea seguro y confiable		
Posibles situaciones	**Propósitos específico**	**Interesados**	**Recursos**
Análisis de software malicioso	Comprender la estructura y el comportamiento del software malicioso para solucionar el riesgo que representa	Experto en seguridad	Software malicioso
Problemas de seguridad en productos software	Identificar posibles problemas de seguridad en el producto software	Productor y usuario del software	El producto software
Contexto	**Computación forense**		
Ámbito	Jurídico		
Propósito general	Interpretar y construir evidencia que permita establecer los hechos y formular hipótesis relacionadas con el caso de estudio		
Posibles situaciones	**Propósitos específico**	**Interesados**	**Recursos**
Recuperación de evidencia	Reconstruir evidencias a partir del producto software	Investigador, afectado por el hecho	El producto software
Contexto	**Educación**		
Ámbito	Aprendizaje de la ingeniería de software		
Propósito general	Aprender y desarrollar habilidades		
Posibles situaciones	**Propósitos específico**	**Interesados**	**Recursos**
Herramienta didáctica	Facilitar la comprensión de un tema y desarrollar habilidades	Estudiantes y profesores	Documentación sobre el tema,

	específicas		código fuente
Herramienta de evaluación	Verificar la conformidad entre lo solicitado y lo entregado por el estudiante		Material entregado por el estudiante

Tabla 3. Caracterización de los contextos de uso de la Ingeniería inversa

Para que el proceso de recuperación de la arquitectura del producto software sea posible, es necesario contar con los artefactos que lo representan: Archivos ejecutables, código fuente, archivos de configuración, estructuras de datos y documentación (Canfora et al., 2011); siendo obligatorio tener por lo menos el ejecutable o el código fuente, sin embargo, la revisión de la literatura reveló que la disponibilidad de estos recursos depende del contexto en el que se presente el problema objeto de estudio, como se muestra en la Tabla 4, donde se relacionan las situaciones que se pueden presentar en los contextos de uso y la disponibilidad de los artefactos software. En la tabla se utiliza la letra D para indicar que el artefacto está disponible, la letra N significa no disponible, mientras que la letra P quiere decir que existe la posibilidad de que esté disponible.

Contexto	**Situación**	**Artefactos Software**				
		Ejecutable	**Código fuente**	**Archivos de configuración**	**Estructuras de datos**	**Documentación**
Producción de software	Mantenimiento de software	D	D	D	D	P
	Recuperación de documentación	D	D	D	D	P
	Verificación de software	D	D	D	D	D
	Reutilización de activos	D	D	D	D	P
Seguridad informática	Análisis de software malicioso	D	N	N	N	N
	Identificación de problemas de seguridad en productos software	D	P	P	P	P
Computación forense	Recuperación de evidencia	D	P	P	P	P
Educación	Herramienta didáctica	P	D	P	P	P
	Herramienta de evaluación	P	D	P	P	P

Tabla 4. Disponibilidad de artefactos software en contextos de uso

Como se observa en la Tabla 4, en el proceso de desarrollo de software el único artefacto que posiblemente no esté disponible es la documentación, mientras que en el contexto de la seguridad informática y la computación forense, sólo se cuenta con certeza de la disponibilidad del ejecutable. En el caso del contexto de la educación, es el código fuente el único artefacto que debe estar disponible. Esta situación afecta la forma como se debe realizar el proceso de recuperación de la arquitectura, obligando a aplicar diferentes estrategias para cada contexto y situación.

Por otra parte, aunque en un proceso de ingeniería inversa se pueden recuperar distintas vistas arquitectónicas, la revisión de la literatura permitió identificar que la relevancia de la vista recuperada depende también del contexto en el que se realice el proceso, siendo algunas vistas de carácter obligatorio (O), otras complementarias (C), y otras relativas (R) al problema concreto que se esté tratando, lo que implica que en algunas circunstancias pueden ser obligatorias y en otras complementarias para la misma situación; mientras que hay vistas que definitivamente no son relevantes (--), como se muestra en la Tabla 5. De igual manera, las vistas a reconstruir influyen en el proceso de recuperación de la arquitectura como se explica en el capítulo de la metodología.

Contexto	**Situación**	**Punto de vista**												
		Estructuras modulares					**Componentes y conectores**				**Estructuras de asignación**			
		Descomposición	**Usos**	**Capas**	**Clases**	**Modelo de datos**	**Servicios/Procesos**	**Concurrencia**	**Ejecución de la concurrencia**	**Perfil de la ejecución**	**Implementación**	**Asignación de trabajo**	**Despliegue**	**Uso de recursos**
Producción de software	Mantenimiento de software	O	O	O	O	R	R	R	R	R	O	C	R	R
	Recuperación de documentación	O	O	O	R	R	R	R	C	C	R	C	R	C
	Verificación de software	O	O	O	O	O	R	R	R	R	O	--	R	R
	Reutilización de activos	O	O	C	C	R	C	C	C	C	C	--	C	C
Seguridad informática	Análisis de software malicioso	O	O	C	C	C	--	C	C	O	--	--	R	R
	Identificación de problemas de seguridad en productos software	O	O	C	C	C	R	C	C	O	C	--	R	R
Computación forense	Recuperación de evidencia	O	O	R	R	R	R	R	R	R	R	--	R	R
Educación	Herramienta didáctica	O	O	R	R	R	R	R	R	R	R	--	R	R
	Herramienta de evaluación	O	O	R	R	R	R	R	R	R	O	--	R	R

Tabla 5. Vistas arquitectónicas según contexto de uso

Por lo anterior, se puede inferir que el proceso de recuperación de las vistas arquitectónicas depende del contexto en el que se presente la situación que se desea resolver. Mientras que en un contexto como el desarrollo de software se puede aplicar un enfoque orientado por los atributos de calidad (Stoermer et al., 2003), en contextos como la computación forense y la seguridad informática, cuando se hace análisis de software malicioso, este enfoque no es pertinente, porque no se está revisando el cumplimiento de un requisito específico del sistema (Bass et al., 2013), sino una condición particular del contexto de uso. Para abordar esta situación

se elaboró el modelo conceptual para la recuperación y análisis de vistas arquitectónicas que se explica en el siguiente capítulo.

Capítulo 3

Modelo conceptual

Un modelo conceptual es una representación externa, precisa, completa y consistente con el conocimiento científicamente compartido, que facilita la comprensión o enseñanza de sistemas o estados de las cosas del mundo (Moreira y Palmero, 2002). En consecuencia, el modelo conceptual se definió como resultado del estudio y análisis del material bibliográfico identificado a partir de la revisión de la literatura, y de la base teórica existente sobre arquitectura de software.

El modelo conceptual propuesto permite la abstracción de artefactos de bajo nivel y la reconstrucción de información de alto nivel sobre la estructura y el comportamiento de sistemas software y brinda una visión holística sobre la ingeniería inversa y una visión detallada sobra la recuperación y análisis de vistas arquitectónicas de comportamiento, facilitando el estudio y comprensión de este campo del conocimiento. El modelo conceptual se logra a partir de la integración de varias técnicas de recuperación de arquitectura.

En este capítulo se explica inicialmente el modelo de dominio de la recuperación de arquitecturas, porque es el referente a partir del cual se construyó el modelo conceptual, que se expone posteriormente y ha sido estructurado en dos partes, la primera hace una abstracción de la ingeniería inversa, mientras que la segunda se centra en representar específicamente la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento.

3.1. Modelo de dominio de la recuperación de arquitecturas

En su concepción inicial la ingeniería inversa se entendió 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 y Cross, 1990). Esto significaba que el producto software era procesado por un analizador generando información básica, que posteriormente era trasformada para generar vistas comprensibles para las personas.

Actualmente la ingeniería inversa se entiende 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 et al., 2007) y su modelo de dominio ha sido presentado por Canfora et al. (2011) como se muestra en la figura Figura 3, donde visualmente evidencia cómo el modelo conceptual propuesto por Chikovsfky y Cross (1990) se expande, al incluir la utilidad de la ingeniería inversa para reconstruir la evolución histórica del producto software.

Por eso, en el modelo de Canfora et al. (2011) se representa al ingeniero de software como responsable de atender las solicitudes de cambio del producto, consideradas como artefactos de software. Esta acción genera una evolución histórica del producto software representada en una revisión. Adicionalmente, los autores agregan cuatro tipos de analizadores: dinámicos, estáticos, históricos e híbridos, dependiendo el aspecto del software que se recupera. Para la recuperación de los aspectos dinámicos se utilizan las tazas de ejecución obtenidas a partir del ejecutable, considerado como artefacto software. Finalmente incluyen las recomendaciones del sistema como vistas del software.

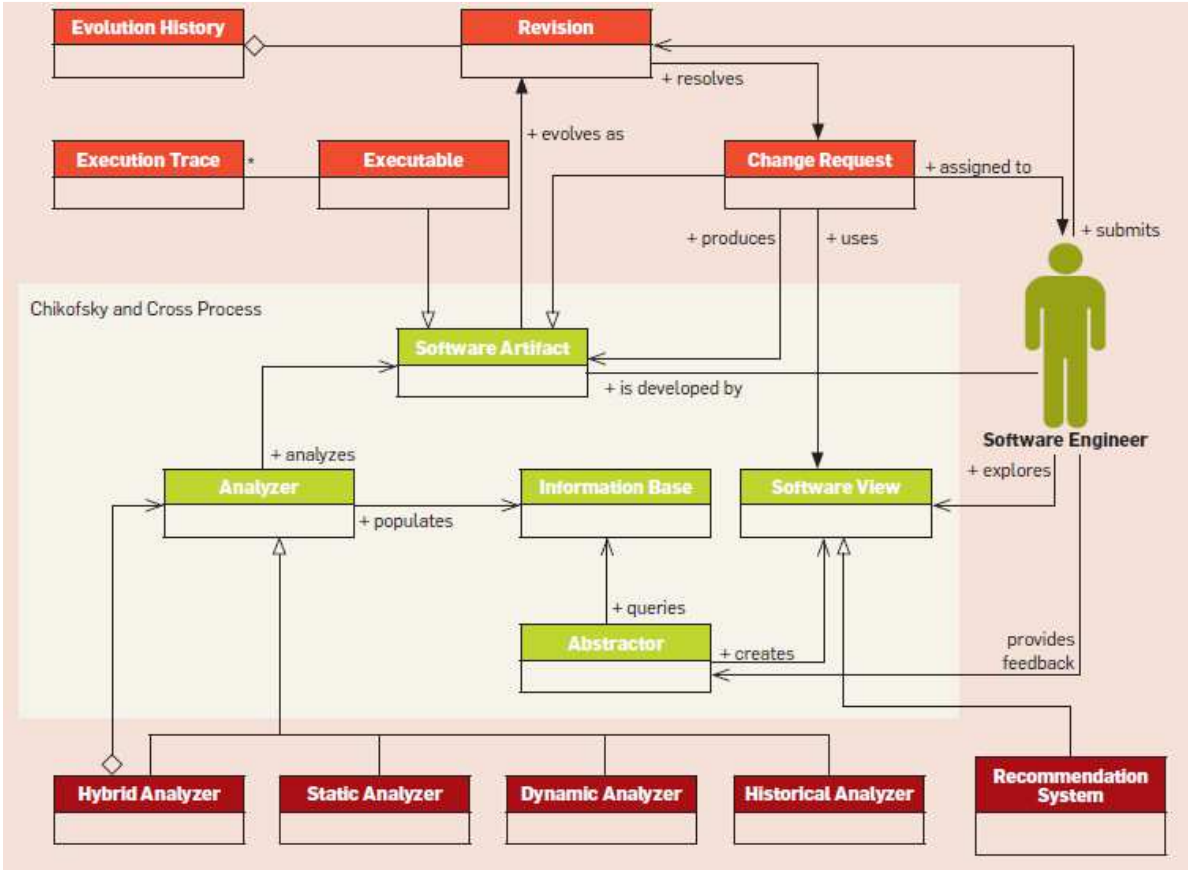

Figura 3. Modelo de dominio de la ingeniería inversa. Fuente Canfora et al.(2011)

Aunque esta representación es más completa, aún existen elementos importantes de la ingeniería inversa que no se evidencian en el modelo, y que dan origen a la propuesta que se explica a continuación y se presenta en la Figura 4. El modelo tiene dos diferencias radicales, la primera consiste en la inclusión del contexto, porque cómo se argumentó en el capítulo anterior, las características y circunstancias del contexto son las que determinan el uso que se da a la ingeniería inversa, por eso todo el modelo está presentado dentro de un límite denominado contexto.

La segunda diferencia radica en la abstracción que se hace del concepto artefacto, que inicialmente fue visto por Chikovsky y Cross (1990) como artefacto software, y posteriormente Canfora et al. (2011) agregan dos tipos de artefactos software: el ejecutable y la solicitud de cambio. Sin embargo, estos no son los únicos tipos de artefactos que existen. Un artefacto es la especificación de una pieza física de información que es usada o producida en el proceso de desarrollo de software, o en el despliegue y operación del sistema (OMG, 2015), por eso se establecen dos clases de artefactos: La documentación y los artefactos software, teniendo en cuenta que un artefacto puede estar conformado por más artefactos.

La documentación comprende a todas las piezas de información que describen las especificaciones del sistema, el diseño y la arquitectura del producto software (vistas del software), incluyendo las solicitudes de cambio. Un artefacto software es una pieza de información que representa al producto objeto del proceso de ingeniería inversa, por ejemplo el código fuente, archivos ejecutables, archivos de configuración, entre otros. Un elemento es una parte constituyente de un sistema complejo . La especificación KDM (OMG, 2011) establece cuatro capas y un conjunto de 12 paquetes para comprender el producto software y los elementos que lo conforman, como se explica en marco teórico.

Todo artefacto es representado por medio de modelos y tiene un nivel de abstracción que corresponde a las capas establecidas por la especificación KDM: infraestructura, elementos de programa, recursos en tiempo de ejecución y abstracciones. Por ejemplo, el código fuente es un artefacto software que tiene un nivel de abstracción correspondiente a la capa de infraestructura, mientras que una solicitud de cambio es un documento que representa un artefacto con un nivel de abstracción correspondiente a la capa de abstracciones.

Adicionalmente, se complementa la semántica del analizador incluido desde el modelo de Chikovsky y Corss (1990), atendiendo el modelo del proceso de recuperación de la arquitectura propuesto por Tilley et al. (1996), según el cual la extracción de datos consiste en identificar a partir de los datos en bruto del sistema objeto de estudio, los artefactos y elementos que lo conforman y las relaciones entre ellos, usando mecanismos apropiados de extracción. Este objetivo se puede lograr utilizando distintos tipos de analizadores y técnicas como la extracción informal de datos a través de entrevistas. Por eso se agregó el extractor de datos y el analizador se definió como un tipo particular de éste.

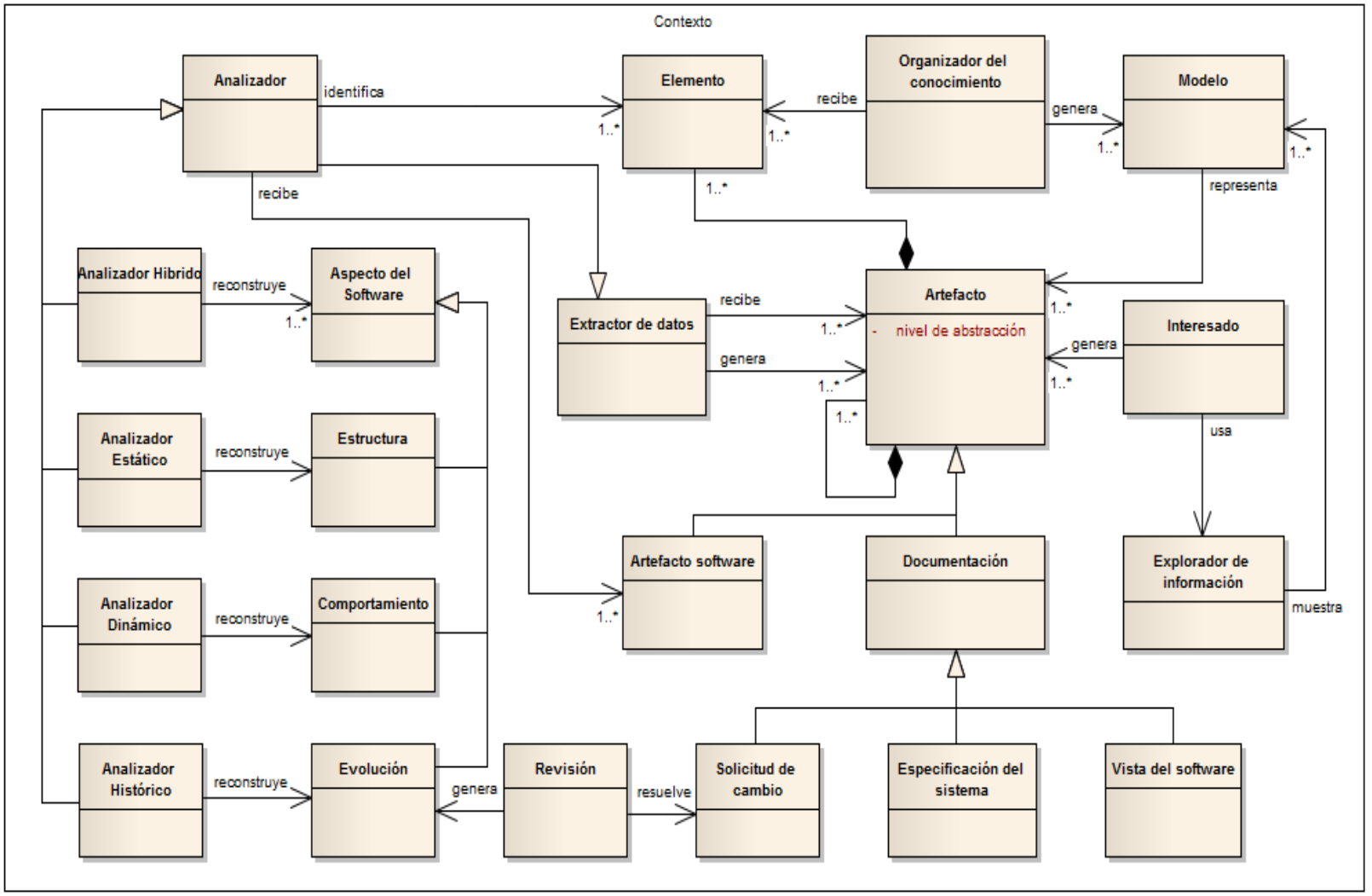

Figura 4. Modelo de dominio de la ingeniería inversa

De igual forma, se sustituye el elemento responsable de la abstracción por el organizador del conocimiento, cuya función consiste en representar a través de modelos los datos obtenidos en la etapa de extracción, haciendo posible su almacenamiento y recuperación de manera fácil y eficiente, permitiendo el análisis de los artefactos y sus relaciones.

Siguiendo el mismo argumento, se incluye el *e*xplorador de la información, que brinda los mecanismos para navegar, analizar y presentar los resultados del proceso de ingeniería inversa. La navegación hace posible el desplazamiento por las estructuras de información generadas en la fase de organización del conocimiento. El análisis implica extraer información que no está disponible en forma explícita, generando vistas interesantes que pueden ayudar a la comprensión del sistema objeto de estudio. La presentación de los resultados utiliza estrategias que facilitan su comprensión y hacen posible la navegación y el análisis.

También se hace una abstracción del ingeniero de software, transformándolo en un interesado, porque en el proceso de ingeniería inversa, además del ingeniero de software participan otras personas (o grupos) que hacen parte del contexto y que tienen responsabilidades y preocupaciones específicas, que se describen con más detalle en el capítulo que explica la metodología para la recuperación de arquitecturas.

Finalmente, se complementa la propuesta de Canfora et al. (2011) incluyendo los aspectos que comprende el producto software: Estructura, comportamiento y evolución, que se constituyen en referente para clasificar en forma correspondiente los distintos tipos de analizadores que existen, cada uno de ellos responsable de reconstruir un aspecto específico. Esta clasificación es muy importante, porque dependiendo las situaciones que se presenten en un contexto específico, será necesaria la reconstrucción de uno o más aspectos del producto software.

3.2. Visión holística

El modelo conceptual se definió desde la perspectiva práctica que representa el proceso de recuperación y análisis de vistas arquitectónicas, con base en el modelo de dominio explicado previamente, para hacer una representación externa, precisa, completa y consistente con el conocimiento científicamente compartido. Por eso, se presenta en dos dimensiones, la primera corresponde a la visión holística de la ingeniería inversa, explicada en esta sección, y la segunda se centra específicamente en la recuperación y análisis de vistas arquitectónicas, expuesta en la siguiente sección.

Al incluir el contexto en el modelo de dominio de la ingeniería inversa se modifica el concepto presentado por Tonella et al. (2007), según el cual, por ingeniería inversa se entiende “todo método destinado a la recuperación del conocimiento de un sistema software, en apoyo a la realización de actividades de ingeniería de software", porque la ingeniería inversa actualmente no se utiliza exclusivamente en el campo de

la ingeniería de software, también tiene aplicación en otros contextos como la seguridad informática, la computación forense y la educación, como se explicó en el capítulo anterior.

En consecuencia la ingeniería inversa incluye todo método destinado a la recuperación del conocimiento de un sistema software, en apoyo a la realización de actividades que requieran de su entendimiento. Esto implica la existencia de dos aspectos relevantes en la ingeniería inversa: El contexto de uso y el proceso de recuperación. Por eso el modelo conceptual se encuentra estructurado a partir de estos dos elementos, como se muestra en la Figura 5.

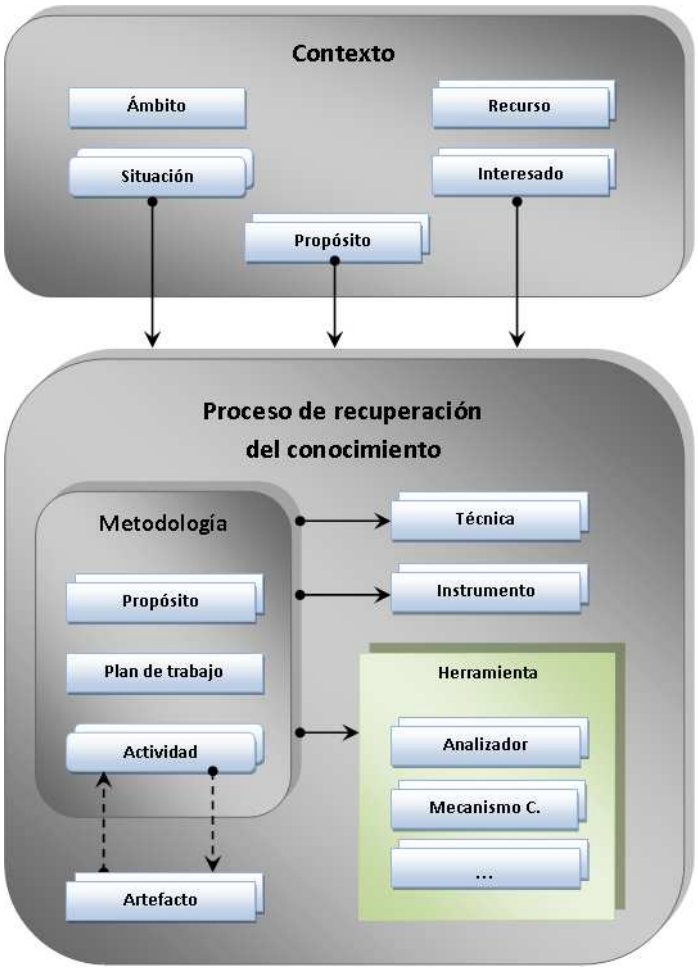

Figura 5. Modelo conceptual. Visión holística

Las situaciones que se presenten en el contexto dan origen al proceso de recuperación de la arquitectura, que es realizado por los interesados teniendo en cuenta los propósitos y los recursos disponibles en el contexto de uso. Este elemento del modelo conceptual es explicado detalladamente en el capítulo anterior.

El proceso de recuperación del conocimiento consiste en la realización metódica de un conjunto de actividades, aplicando técnicas y utilizando herramientas e instrumentos, para generar una nueva representación del producto software objeto de estudio a partir de los artefactos existentes. Esto implica la existencia de una metodología, que establece los objetivos del proceso, el plan de trabajo, la forma cómo se realizan las actividades, y además determina cuáles son las técnicas, herramientas e instrumentos y cómo se utilizan en el proceso. Todos estos aspectos se explican detalladamente en el capítulo 5.

Esta visión holística del modelo conceptual comprende la representación externa de la recuperación de la arquitectura, y es complementada con la visión detallada de la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, que se explica a continuación, para lograr una representación precisa, completa y consistente de este campo del conocimiento.

3.3. Visión detallada

El proceso genérico de la recuperación de la arquitectura comprende tres actividades básicas (Tilley et al., 1996), que se constituyen en el referente principal del modelo conceptual propuesto. Cada una de estas actividades recibe como entrada artefactos que son transformados a un mayor nivel de abstracción, como se muestra en la Figura 6. La visión detallada del modelo conceptual centra su atención en la comprensión de cada uno de estos artefactos y actividades, por eso se explican a continuación.

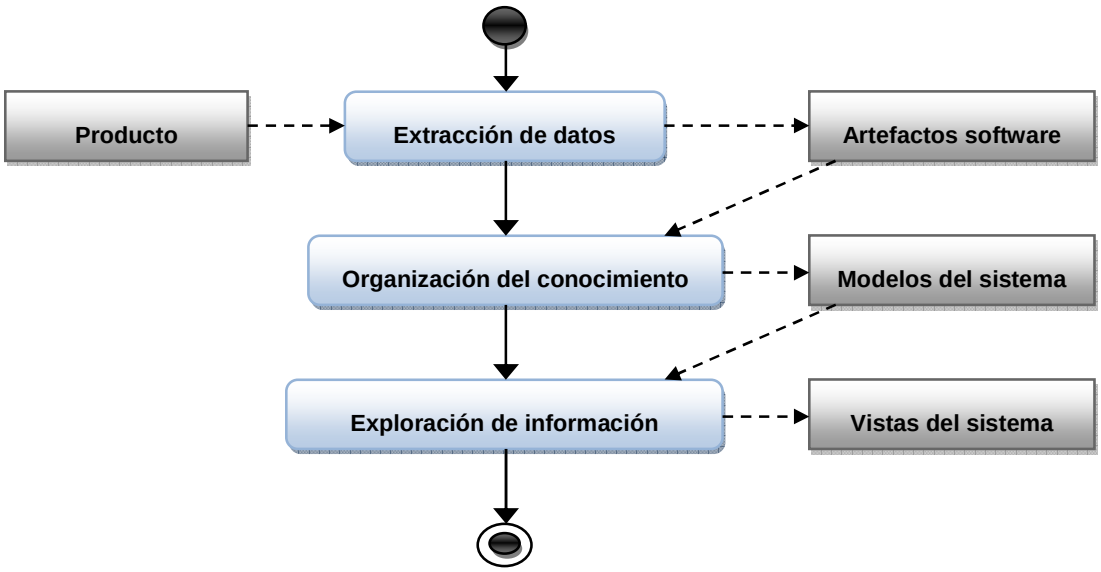

Figura 6. Proceso genérico de recuperación de la arquitectura

3.3.1. Artefactos

Los artefactos representan elementos concretos del mundo físico, pueden tener propiedades que simbolizan las características y operaciones que realizan sus instancias, las cuales pueden ser múltiples y estar desplegadas en varios entornos, cada una por separado (OMG, 2011), por ejemplo los archivos de modelo, archivos fuente, scripts, archivos ejecutables, tablas de bases de datos, entregables, documentos de procesamiento de texto y mensajes de correo, entre otros.

Como se observa en la Figura 7, un artefacto puede ser un documento o una pieza de software y puede estar constituido por otros artefactos, como es el caso del *producto software* que está conformado por archivos ejecutables, binarios, de código fuente, de configuración, descripción de recursos, imágenes, documentación, entre otros entregables. De igual forma, un artefacto puede estar constituido por elementos, y es representado por medio de modelos desde la perspectiva de un conjunto relacionado de intereses, que a su vez conforman las vistas del producto software como tipos de documentación.

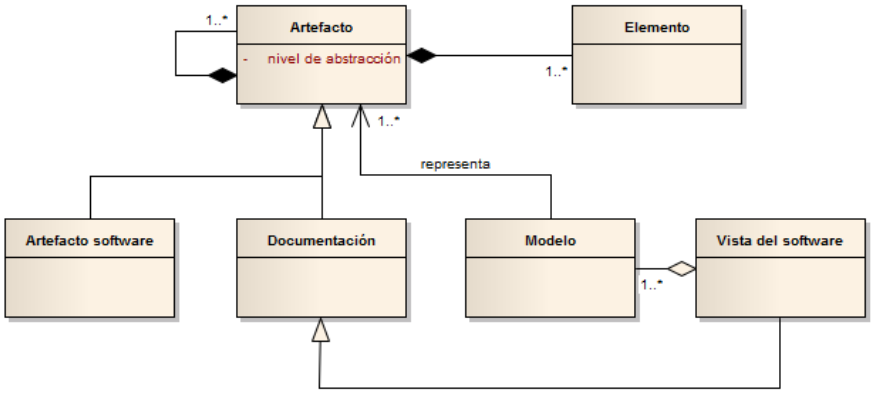

Figura 7. Artefacto

Para recuperar el conocimiento implícito en los artefactos que conforman el producto software, se deben identificar estos artefactos y descomponerlos en las partes que los conforman, utilizando las técnicas de extracción de datos que se explican en el marco teórico. Para garantizar la interoperabilidad entre herramientas de ingeniería inversa, se debe utilizar la especificación KDM para representar el producto software objeto de estudio, sus elementos, asociaciones y entornos de operación. Esto permite atender uno de los mayores inconvenientes encontrados en el estado del arte, referente a la interoperabilidad entre herramientas.

Los elementos del paquete fuente y el paquete código KDM se pueden extraer estableciendo reglas de mapeo a partir del lenguaje (o lenguajes) de programación del producto objeto de estudio, porque están explícitos en la sintaxis, sin embargo, los elementos de las capas superiores (de abstracción y de recursos en tiempo de ejecución) se encuentran implícitos, porque corresponden a niveles más altos de

abstracción, por lo cual requieren de análisis incrementales partiendo de las representaciones KDM primitivas.

Con base en la representación KDM del producto software objeto de estudio, se pueden organizar los elementos identificados formando *modelos del sistema*, aplicando las técnicas de organización del conocimiento expuestas en la base teórica del marco de referencia. Siguiendo el criterio de interoperabilidad, los modelos generados deben representarse a través de mecanismos que hagan posible su interpretación, como es el uso del lenguaje de marcado XML. En el modelo conceptual propuesto se utiliza UML para garantizar esta característica, representando los modelos obtenidos en el proceso de organización del conocimiento por medio de diagramas expresados en XMI.

Las *vistas del sistema* pueden estar conformadas por uno o varios modelos y para su visualización se puede utilizar cualquier herramienta que interprete los modelos UML en XMI. Las vistas que se obtienen dependen de los objetivos establecidos en la metodología explicada en el capítulo 5, y que hace parte del marco de referencia propuesto. Este modelo conceptual se centra, pero no se limita, en las vistas arquitectónicas de comportamiento, que pueden ser representadas en UML por medio de diagramas de actividades, de casos de uso, de máquinas de estado, o de interacción: diagramas de secuencia, diagramas de comunicación y diagramas de tiempos. La trasformación de los artefactos se logra realizando las actividades que se presentan a continuación y se explican en forma más detallada en el capítulo de la metodología.

3.3.2. Actividades

Toda actividad tiene un fin o intención, recibe entradas, aplica técnicas y genera salidas. La *extracción de datos* es la primera actividad del proceso genérico de la recuperación de la arquitectura (Tilley et al., 1996), su principal fin es recolectar los datos requeridos para lograr las vistas arquitectónicas que se pretenden recuperar. Para lograr este propósito se requiere como entrada los datos en bruto del producto software objeto de estudio y el conocimiento del experto.

Esta actividad comprende dos tareas, la primera consiste en identificar los artefactos que constituyen el producto software, para lo cual se aplican técnicas de inspección manual. Como resultado de esta tarea se obtienen los repositorios que contienen los artefactos software, tales como archivos ejecutables, binarios, de código fuente, de configuración, descripción de recursos, imágenes documentación, entre otros; teniendo en cuenta que su disponibilidad depende del contexto en el que surge la necesidad de recuperar las vistas arquitectónicas.

La segunda tarea radica en descomponer los artefactos identificados para lograr una representación KDM del sistema que se conserva en un repositorio XMI. Esto se logra aplicando técnicas como el análisis léxico, análisis sintáctico, análisis difuso,

islas gramaticales, coincidencias sintácticas, análisis de flujos de datos, instrumentación de código y perfilado.

La segunda actividad consiste en *organizar el conocimiento*, tiene como fin representar los datos obtenidos en la actividad anterior en una forma que permita su almacenamiento y recuperación de manera fácil y eficiente, para que se pueda hacer análisis de los artefactos y sus relaciones. Como resultado de esta actividad se obtienen modelos en UML, bajo una representación XMI, de las vistas arquitectónicas que se desean recuperar. Para lograrlo se pueden aplicar técnicas como: Álgebra relacional, álgebra proposicional, álgebra de Tarski, modelo de reflexión, consultas SQL, lenguajes de Consulta ad-hoc y algoritmos de agrupamiento.

La última actividad consiste en la *exploración de la información*, su intención es brindar los mecanismos necesarios para navegar, analizar y presentar los resultados del proceso de recuperación. La presentación y navegación se logran con el uso de herramientas que permitan la visualización de modelos UML en su representación XMI. Para hacer posible el análisis se define un mecanismo de consulta que deriva y extrae información que no está disponible en forma explícita en los modelos recuperados, generando resultados que ayudan a la comprensión del sistema objeto de estudio. Este mecanismo se explica en el capítulo 4.

En este orden de ideas, el modelo conceptual se centra en la representación de los artefactos y actividades que comprenden el proceso de recuperación y análisis de vistas arquitectónicas, como se muestra en la Figura 8.

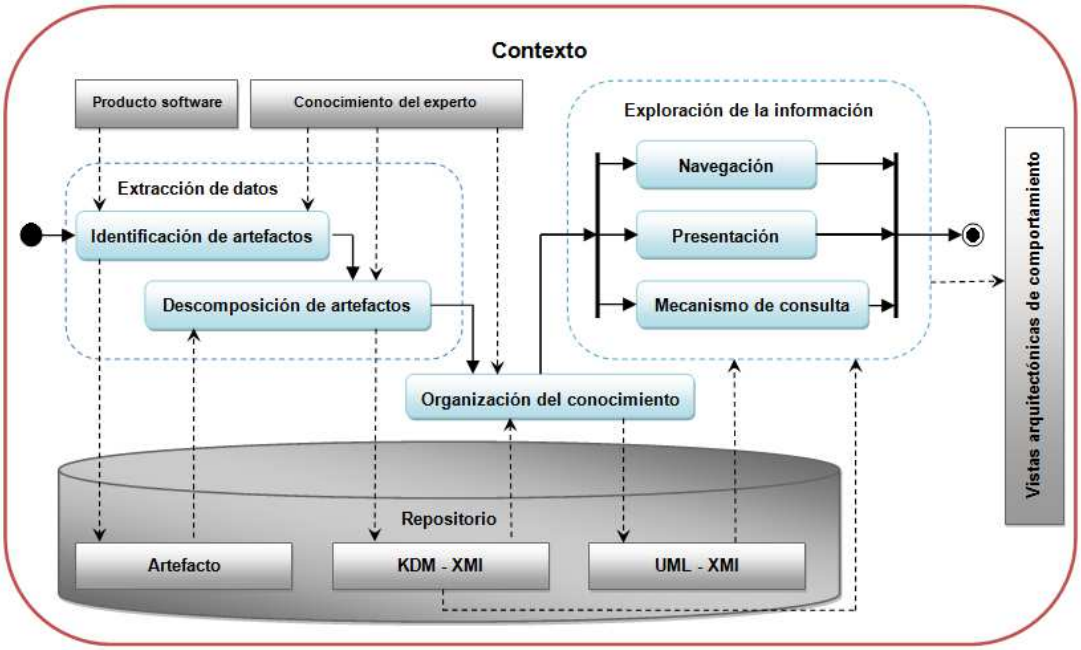

Figura 8. Visión detallada del modelo conceptual

Con base en este modelo se diseña el mecanismo de consulta y se define la metodología que propone el marco de referencia, como se explica en los siguientes capítulos.

Capítulo 4

Mecanismo de consulta

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., 2014; Urma y Mycroft, 2012), 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., 2010; Holt et al., 2002).

En coherencia con los objetivos planteados en la tesis, el mecanismo de consulta que se propone está orientado al análisis del producto software, a partir de las vistas arquitectónicas recuperadas. Por eso, en este capítulo se presenta la caracterización que se hizo de los mecanismos de consulta, que se identificaron en la revisión de la literatura y con base en esto, posteriormente se definen los lineamientos y el diseño del mecanismo de consulta propuesto. A diferencia de las propuestas existentes, el mecanismo de consulta diseñado permite una interacción en lenguaje natural, ampliando su capacidad de uso a contextos en los que no todos los interesados son expertos en ingeniería de software.

4.1. Caracterización de los mecanismos de consulta

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; Heafield et al., 2011; Opoka et al., 2008; Stein et al., 2005; Balsamo y Marzolla, 2005; Lange et al., 2001; Kullbach y Winter, 1999), 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 (Li et al., 2012; Zhang et al., 2012; Störrle, 2011; Choi et al., 2009; Störrle, 2007).

De igual forma hay autores que proponen herramientas de consulta (Noguera et al., 2012; Liepiņš, 2012; De Roover et al., 2011; Mendonça et al., 2004; Paul y Prakash, 1994), o las evalúan (Ujhelyi et al., 2014; Rademaker, 2008; Alves y Rademaker, 2008), 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; Ráth et al., 2012; Bergmann et al., 2012; Acretoaie y Störrle, 2012; Alves et al., 2011; Bergmann et al., 2010; 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, (2012) 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 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., 2014), 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.

Como resultado del análisis de las propuestas identificadas se elaboró el modelo de caracterización de los mecanismos de consulta representado en la Figura 9, según el cual, 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; Störrle, 2011), XQuery (Choi

et al., 2009; Eichberg et al., 2005), 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 6 se relacionan los 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, el año de su creación y los autores que lo proponen.

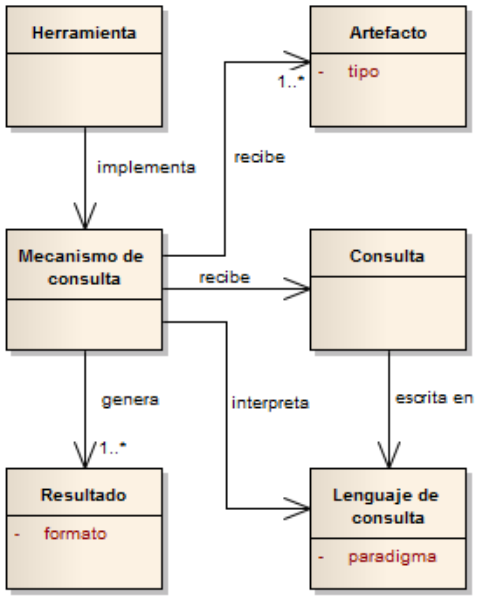

Figura 9. Modelo de caracterización de los mecanismos de consulta.

	Lenguaje	Paradigma	Artefacto	Repositorio	Año	Autores
1	MOCQL	LP	Modelo	Base de datos deductiva	2013	Störrle, H.
2	EMF-IncQuery	OG	Modelo	Grafos	2011	Bergmann et al.
3	SOUL	LP	Código	AST	2011	De Roover et al.
4	VMQL	LP	Modelo	Base de datos deductiva	2011	Störrle, H.
5	BBQ	AR	Código	BDR Bytecode	2010	Antlersoft
6	Cypher	OG	Código	Grafos	2010	Neo4j
7	BPMN-Q	OG	Modelo	Grafos	2007	Awad A.
8	.QL	AR y MO	Código	Base de datos relacional	2007	De Moor et al.
9	BP-QL	OG	Modelo	Grafos	2006	Beeri et al.
10	JTL	LP	Código	BDR Bytecode	2006	Cohen et al.
11	QM	MO	Modelo	Modelo UML	2004	Stein et al.
12	RML	CP	Código	Tuplas	2005	Beyer et al.
13	Rscript	CR	Código	Tuplas	2003	Klint, P.
14	AQL	AR-G	Código	AST y tuplas	2000	Sartipi et al.
15	Visual OCL	MO	Modelo	Modelo UML	2000	Bottoni et al.
16	Xquery	Xpath		XML	2000	W3C

	Lenguaje	Paradigma	Artefacto	Repositorio	Año	Autores
17	GReQL	AR-G	Código	Repositorio Tgraph	1999	Kullbach et al.
18	TyRuBa	LP		Base de datos deductiva	1998	De Volder, K.
19	ASTLog	LP	Código	AST	1997	Crew, R. F.
20	Grok	CR-G	Código	Tuplas	1995	Holt, R.
21	OCL	MO	Modelo	Modelo UML	1995	IBM
22	PQL	AR	Código	Base de conocimiento	1995	Jarzabek, S.
23	SCA	AR	Código	AST	1994	Paul y Prakash
24	GraphLog	LP	Código	Grafos	1991	Consens et al.
25	Datalog	LP	Código	Base de datos deductiva	1989	Ceri et al.
26	QUEL	AR		Base de datos relacional	1976	Stonebraker et al.
27	SQL	AR		Base de datos relacional	1974	Chamberlin y Boyce
28	Prolog	LP		Base de datos deductiva	1972	Colmerauer y Roussel

Tabla 6. Lenguajes de consulta

De igual manera se identificaron 21 herramientas que implementan mecanismos de consulta, relacionadas en la Tabla 7 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).

Id.	Nombre del mecanismo	Artefacto	Propósito	Lenguaje de consulta	Formato salida	Autores/Año
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., 2010
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.
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., 2007
12	JGraLab	CF	CP	GReQL	Texto	Kahle, S., 2006
13	JQuery	CF	CP	TyRuBa	Texto y gráficos	De Volder, K., 2006

Id.	Nombre del mecanismo	Artefacto	Propósito	Lenguaje de consulta	Formato salida	Autores/Año
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 y Lewerentz, 2003
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

Tabla 7. Mecanismos de consulta

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. Los resultados de la revisión bibliográfica, sintetizados en la Tabla 7, permiten 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, M. A., 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 8, 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).

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.

Id.	Nombre del mecanismo	Repositorio	Interface de usuario	Objetivo
1	MACH	Prolog	Texto	Integrar prototipos para permitir el análisis de sistemas implementados.
2	lQuery	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.

Tabla 8. Mecanismos de consulta que procesan modelos

También se resalta que los primeros cuatro mecanismos relacionados en la Tabla 8 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; Ujhelyi et al., 2014; Ráth et al., 2012; Bergmann et al., 2012; Bergmann et al., 2010).

Al analizar el propósito de los mecanismos identificados (ver columna cuatro de la Tabla 7) y el objetivo concreto de aquellos que procesan modelos (ver última

columna de la Tabla 8), 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 8, 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.

4.2. Lineamientos del mecanismo de consulta

Como consecuencia de la caracterización de los mecanismos analizados presentados en el ítem anterior, para el mecanismo de consulta se establecen como referentes los siguientes lineamientos:

1) Se utiliza el mecanismo lingüístico, aprovechando las capacidades descriptivas disponibles en lenguajes de modelado como UML, que registran la estructura, el comportamiento y demás propiedades del sistema (Clements et al., 2002).

2) El mecanismo apoya el análisis de las vistas arquitectónicas recuperadas a partir de productos software implementados, por lo tanto corresponde a la fase posterior al desarrollo (Abowd et al., 1997).

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.

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.

4.3. Diseño del mecanismo de consulta

El diseño del mecanismo de consulta se plantea a partir de los lineamientos establecidos previamente y teniendo en cuenta que su objetivo principal es contribuir al análisis de las vistas arquitectónicas estructurales y de comportamiento, recuperadas en un proceso de ingeniería inversa, que puede presentarse en uno de los contextos de uso expuestos previamente.

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 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 experto en software, brindando la posibilidad de plantear preguntas en lenguaje natural a los modelos recuperados.

Los elementos que conforman el mecanismo de consulta se clasifican en dos grupos que se explican a continuación y se representan en la Figura 10. 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.

El patrón esperado del comportamiento del mecanismo de consulta obedece al siguiente algoritmo, representado en la Figura 11. El ingeniero de software 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.

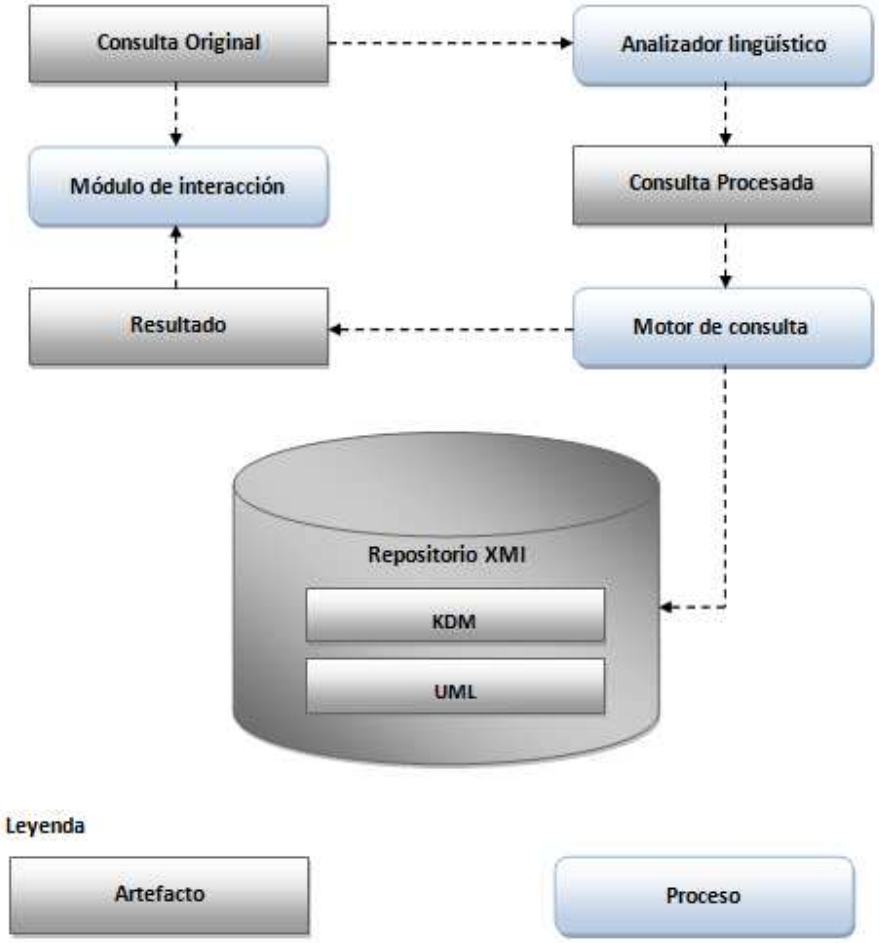

Figura 10. Diseño del mecanismo de consulta.

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 simples. 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 9 se presenta un listado no exhaustivo de preguntas simples 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 la 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.

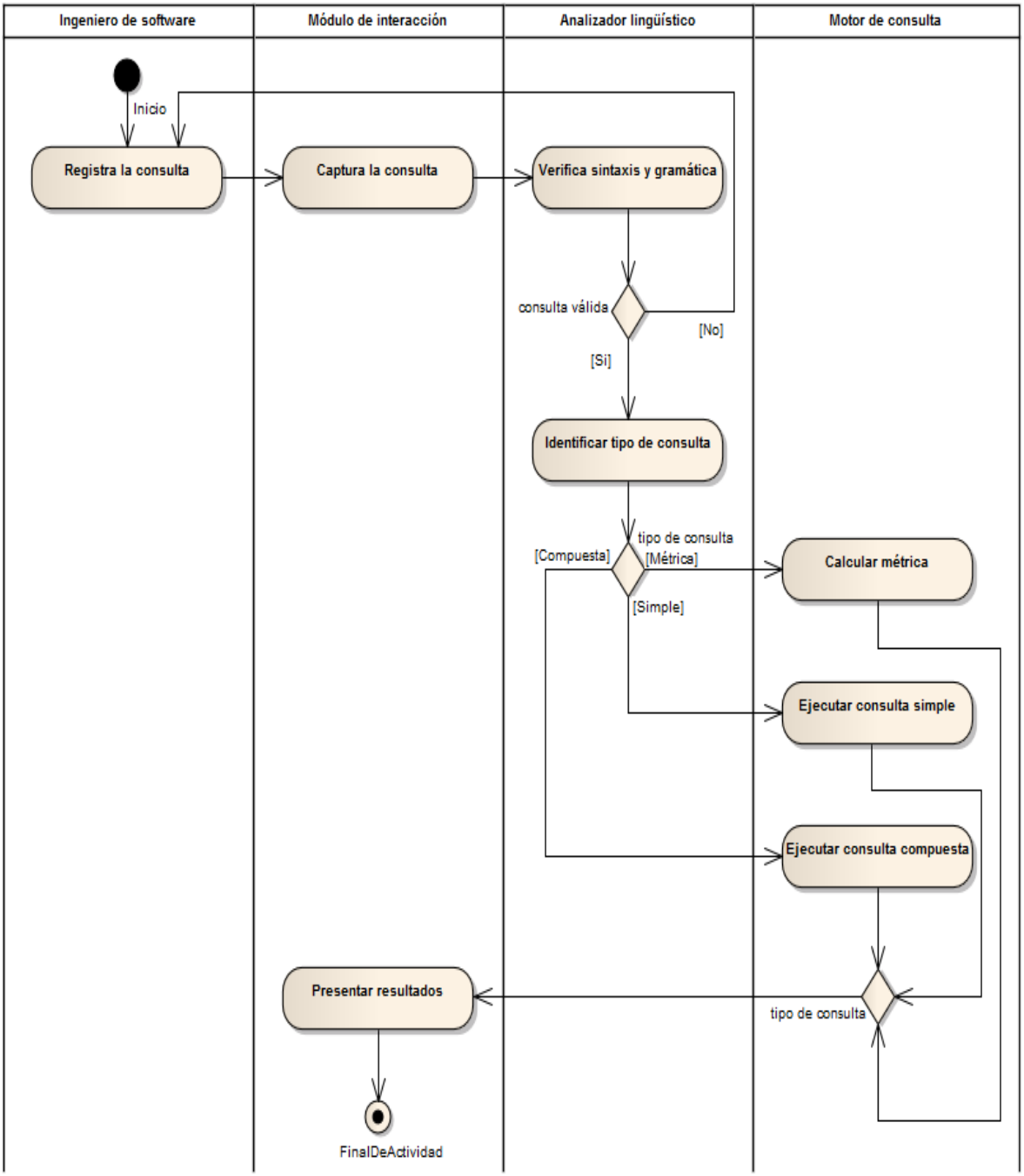

Figura 11. Lógica del mecanismo de consulta.

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?

Tabla 9. Listado de posibles consultas simples

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 10. 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.

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?

Tabla 10. Listado de posibles consultas compuestas

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; Arcelli et al., 2008; Dong et al., 2007); 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; Koschke y Simon, 2003; Murphy et al., 1997).

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.

El mecanismo de consulta diseñado hace parte del modelo conceptual propuesto en el Marco de referencia, en calidad de herramienta que contribuye al proceso de exploración de la información resultante de las actividades de extracción de datos y organización del conocimiento generados en un proceso de ingeniería inversa. En el siguiente capítulo se presenta la metodología para la recuperación de la arquitectura, como instrumento integrante del Marco de referencia para la recuperación y análisis de vistas arquitectónicas objeto de esta tesis.

Capítulo 5

Metodología para la recuperación y análisis de vistas arquitectónicas.

El proceso para realizar la recuperación y análisis de las vistas arquitectónicas se presenta como una metodología, porque no sólo se refiere a los procedimientos técnicos utilizados para lograr el objetivo, también da forma a la diversidad de todo el conjunto de conocimientos que se requieren para lograrlo, estableciendo los cuatro axiomas que diferencian a una metodología (McGregor y Murnane, 2010), como se explican a continuación.

El axioma epistemológico hace claridad sobre lo que cuenta como conocimiento y las formas de conocimiento; está constituido por la base conceptual, la base teórica y el modelo conceptual definidos en el marco de referencia. El axioma ontológico define lo que debería ser el objeto del estudio y la naturaleza de la realidad que lo rodea; está representado por la recuperación y análisis de vistas arquitectónicas como objeto de estudio, que se presenta como necesidad ante una situación bajo un contexto específico, según se explica en el modelo conceptual.

El axioma lógico indica lo que es aceptable en calidad de rigor e inferencia en el desarrollo de los argumentos, los juicios, las reflexiones o las acciones; queda manifiesto en el proceso de la metodología que define el desarrollo de las acciones, y se complementa con el modelo conceptual y los lineamientos del marco de referencia, que establecen las directrices para hacer inferencias, argumentar juicios y reflexiones. Finalmente, el axioma axiológico que revela lo que cuenta como valores fundamentales y lo que es la conciencia, estableciendo elecciones morales, éticas y juicios normativos; está representado por los lineamientos del marco de referencia.

La metodología se define a partir de un análisis detallado de las propuestas existentes, que se sintetiza por medio de un modelo de caracterización explicado en la primera parte de este capítulo y que sirve como referente para definir la estructura de la metodología. Este modelo también se utilizó para caracterizar las propuestas identificadas, como se describe en la segunda parte del capítulo. Posteriormente se

explica en forma detallada el proceso de la metodología y en la última parte se exponen las herramientas que se utilizan.

5.1. Estructura de la metodología

La estructura de la metodología se establece a partir del modelo de caracterización del proceso de recuperación de arquitecturas, que se definió con base en el análisis de las propuestas identificadas en la revisión de la literatura y que se representa en la Figura 12. El *proceso* se ejecuta en un contexto específico y corresponde al conjunto de actividades realizadas por los interesados bajo un orden lógico, siguiendo un enfoque y estableciendo uno o varios objetivos. Existen tres tipos de *enfoques*: Arriba - abajo (de lo abstracto a lo concreto), abajo - arriba (de lo concreto a lo abstracto) e híbrido, explicados en el archivo digital "Marco Teórico.pdf".

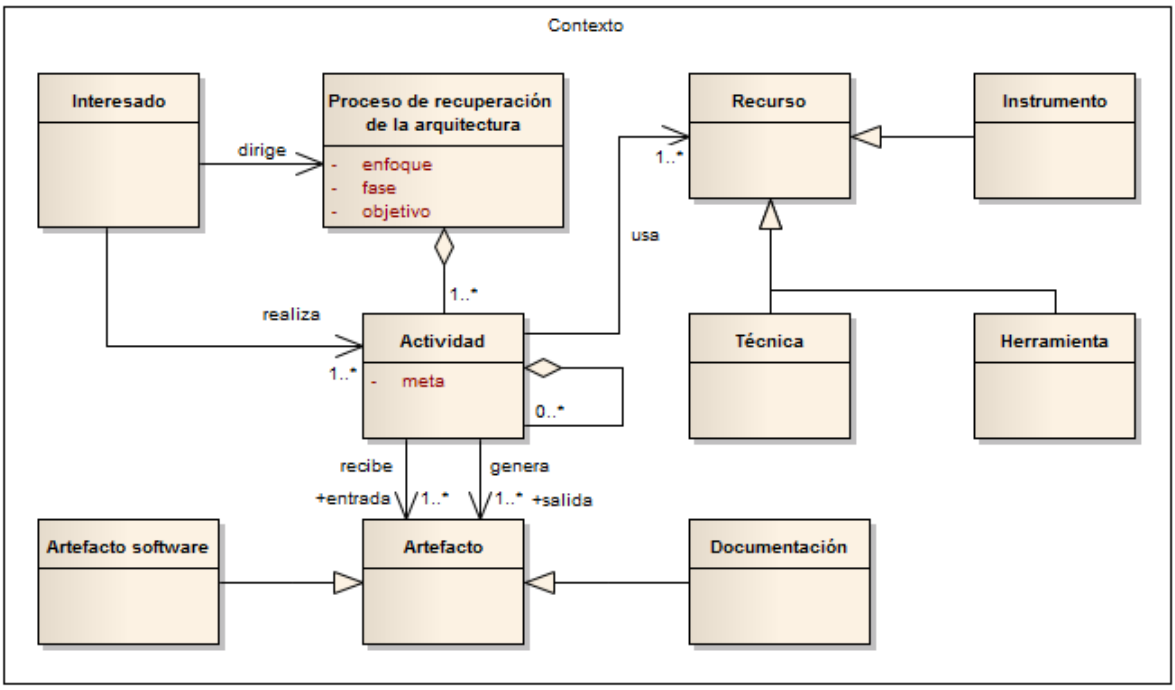

Figura 12. Modelo de caracterización del proceso de recuperación

El *objetivo* del proceso establece el fin que se pretende alcanzar. Algunos posibles objetivos que se pueden plantear en un proceso de recuperación de arquitectura son: reconstrucción de la documentación, identificación de activos para la reutilización, análisis de conformidad, análisis para la identificación de patrones, aspectos, características, roles, colaboraciones, entre otros.

Una *actividad* comprende un conjunto de operaciones o tareas propias de una persona o entidad, en la que se reciben artefactos o piezas de información representadas por documentos o elementos arquitectónicos, que son manipulados

para cumplir las metas de la actividad, con la ayuda de recursos representados por técnicas y herramientas de recuperación de la arquitectura. En las actividades también se utilizan *instrumentos*, elementos de apoyo que sirven para guiar o registrar las actividades y los resultados obtenidos en cada una de ellas.

Una actividad puede estar conformada por varias sub-actividades o tareas organizadas a partir de un *plan de trabajo*, entendido este último como un instrumento en el que se establece bajo un orden lógico y temporal la secuencia de las actividades a realizar, indicando los responsables, los recursos a utilizar y los resultados que se deben obtener. El orden lógico del desarrollo de las actividades se define a partir de la realización de *fases*, que representan los estados por los que pasa el proceso, dependiendo del cumplimiento de los hitos establecidos.

En este orden de ideas, se define la estructura de la metodología como se representa en la Figura 13, entendida como un proceso que utiliza técnicas, herramientas e instrumentos, orientado por los axiomas epistemológico, ontológico, lógico y axiológico, y guiado por un plan de trabajo organizado en fases, en las que se realizan un conjunto de actividades, cumpliendo metas orientadas al logro de los hitos establecidos, para conseguir el cumplimiento de los objetivos formulados para el proceso. Esta estructura y el modelo de caracterización se utilizaron para analizar las propuestas identificadas, como se explica en la siguiente sección.

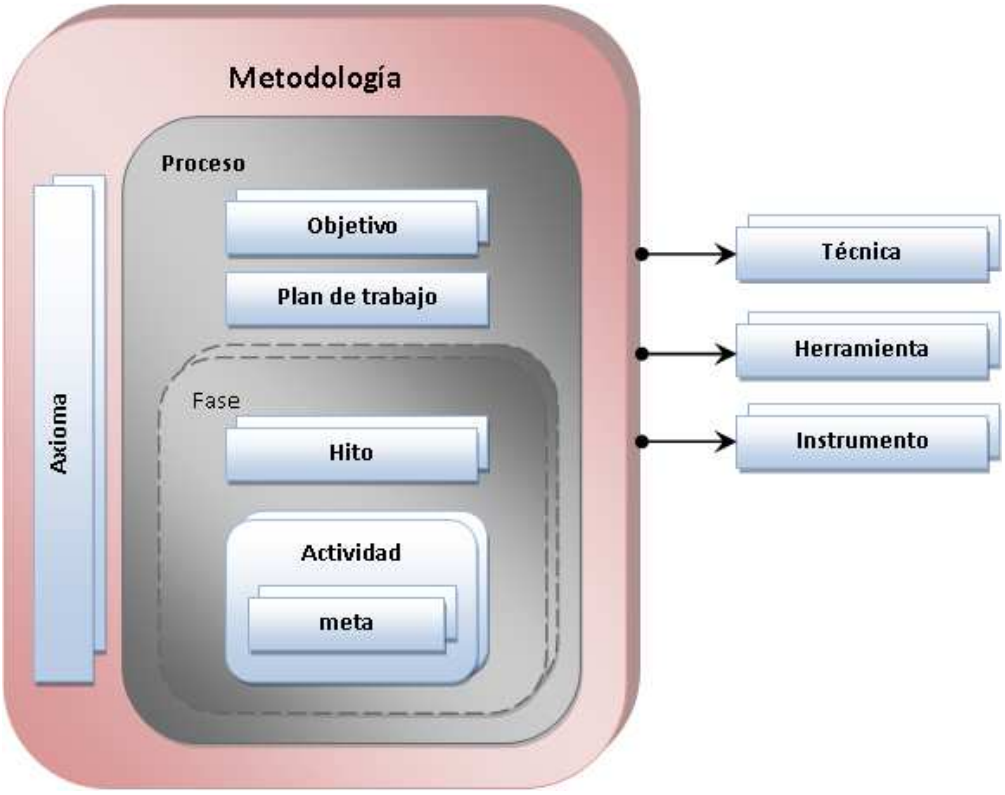

Figura 13. Estructura de la metodología

5.2. Caracterización de las propuestas identificadas

Como resultado de la revisión de la literatura se identificaron 89 propuestas para la recuperación de la arquitectura, que se clasificaron como técnicas o procesos, dependiendo de ámbito establecido por cada una. Se consideró como proceso toda propuesta que contempla las actividades genéricas del proceso de recuperación de la arquitectura: Extracción de datos, organización del conocimiento y exploración de la información. Se asignó la calidad de técnica a aquellas propuestas que sólo se centran en una de estas actividades. En total se identificaron 60 procesos y 29 técnicas (ver Anexo A).

Con el fin de definir la metodología a partir de los logros alcanzados en el campo de conocimiento, para cada una de las 60 propuestas clasificadas como procesos se identificaron las actividades que realizan para lograr la recuperación de la arquitectura. Aunque todas representan aportes significativos, en la Tabla 11 sólo se presenta la caracterización detallada de 13 propuestas, que fueron seleccionadas por incluir actividades, técnicas, o instrumentos complementarios al proceso genérico.

El análisis comparativo entre las propuestas seleccionadas se hizo teniendo en cuenta cada uno de los aspectos usados en la caracterización, obteniendo los resultados que se exponen a continuación. Se evidencia la necesidad de una actividad previa al proceso, para entender el problema, establecer su ámbito, definir el objetivo y diseñar el plan de acción para lograrlo. De igual forma, además de recuperar las vistas del producto software, es conveniente contar con mecanismos que faciliten la interpretación y análisis de los resultados obtenidos en la fase de extracción de datos, para lograr resultados más pertinentes al contexto y la situación que originan el problema objeto de estudio.

También se puede afirmar que las técnicas de inspección manual prevalecen en la ejecución de la actividad previa al proceso, mientras que en la actividad de extracción de datos, las técnicas se clasifican en técnicas de análisis estático y técnicas de análisis dinámico. En las técnicas de análisis estático predominan la inspección manual, el análisis de léxico, el análisis sintáctico, el análisis difuso, la gramática de islas, el análisis semántico (Van Deursen et al., 2004). En las técnicas de análisis dinámico se destacan la instrumentación y el perfilado (Eisenbarth et al., 2003; Kazman et al., 2002; Qiao et al., 2003; Callo et al., 2011).

En la actividad orientada a la organización del conocimiento se aplican distintas técnicas, clasificadas por Van Deursen et al. (2004) en semiautomáticas: Álgebra relacional, álgebra relacional de particiones, modelo de Reflexión, Prolog, transformación de grafos en Rigi, consultas SGL, lenguaje de consulta ad-hoc; y completamente automáticas: basadas en clustering: acoplamiento, nombres de archivos, análisis de conceptos e inferencias de tipos. En la actividad de exploración de la información se observa que la técnica más usada es la visualización, mientras que las propuestas que incluyen la actividad de análisis de los resultados, la realizan usando técnicas como ATAM y escenarios de atributos de calidad.

Nombre	Actividades	Técnicas	Herramientas	Instrumentos
Locating features in source code	Creación de escenarios	Manual - Conocimiento del experto		
	Extracción del grafo estático de dependencias		Bauhaus, Rigi	
	Análisis dinámico	Perfilado/Instrumentación		
	Interpretación del enrejado de conceptos	Mapeo escenarios características	Graph Viz	
	Análisis estático de dependencias			
ARM (Architecture Reconstruction Method)	Definir el plan concreto de reconocimiento de patrones	Reglas abstractas		
	Extracción del modelo fuente	Algebra relacional	SAAMTool,IAPR, LSME, Imagix, SNiFF+	
	Detección y evaluación de instancias de patrones		Dali	
	Reconstrucción y análisis de la arquitectura	Visualización	Rigi	
Symphony	Diseño de la reconstrucción			
	Levantamiento del problema	Talleres, listas de chequeo, juego de roles y análisis de escenarios		Tabla de vistas e interesados
	Determinación de conceptos			
	Extracción de datos	Inspección manual, Análisis de léxico, Análisis sintáctico, Análisis difuso, Gramática de islas, Análisis semántico	grep y perl scripting	
	Inferencia del conocimiento	Semiautomáticas y Completamente automáticas		Clasificación de técnicas
	Interpretación de la información	Presentación (visualización) e interacción		
Architecture reconstruction guidelines	Extracción de información	Análisis de léxico, análisis sintáctico, instrumentación de código, perfilado		Guía para la selección del tipo de extracción
	Construcción de la base de datos (Convertir la información extraída en formato Rigi Standard Format RSF)		ARMIN	Guía para la construcción de la base de datos
	Fusión de vistas		ARMIN	Guía para la fusión de vistas

Nombre	Actividades	Técnicas	Herramientas	Instrumentos
	Composición de vistas arquitectónicas	Visualización		Guía para la visualización
	Análisis de la arquitectura	ATAM		
PuLSE-DSSA	Determinar las vistas y conceptos arquitectónicos			
	Recuperación de la arquitectura (Extracción, abstracción, visualización)	Recuperación semiautomática de componentes, Recuperación de la arquitectura soportada en patrones		
	Análisis de la arquitectura recuperada			Guía comparación de arquitecturas
	Diseño de la arquitectura referencia			Guía proceso de diseño arquitectura de referencia
QADSAR	Identificación del ámbito	Entrevista a los desarrolladores, análisis de documentos		
	Extracción del modelo	Análisis de léxico, perfilado, instrumentación	ARMIN	
	Abstracción del modelo	Álgebra Relacional Particional, álgebra de Tarski, agregación de funcionalidades coherentes, coincidencia de patrones		
	Instanciación de elementos y propiedades		TimeWiz	
	Evaluación de atributos de calidad	Escenarios de atributos de calidad		
Cacophony	Análisis de dominio del metaware		G^{SEE}	
	Análisis de requisitos del metaware			
	Especificación del metaware			
	Implementación del metaware			
	Ejecución del metaware			
	Evolución del metaware			
Reconstructing	Extraer la representación KDM			

Nombre	Actividades	Técnicas	Herramientas	Instrumentos
Architectural Views from Legacy Systems	Seleccionar entidades y relaciones KDM relevantes			
	Seleccionar algoritmo de descomposición y parámetros necesarios			
	Refinar la vista de componentes resultante			
	Documentar los componentes resultantes			
	Construir las vistas utilizando algoritmos de clustering	Clustering		
ArchMine	Extracción de la estructura estática y definición de los escenarios de casos de uso		Odyssey	Guía definición de casos de uso
	Análisis dinámico		TraceMining tool	
	Reconstrucción de los elementos arquitectónicos	Minería de datos	ArchMine	Mapeo de conceptos para la minería de reglas de asociación
	Reconstrucción de la vista dinámica			
	Visualización		ArchMine	
FASAR: The Framework for Automating Software Architecture	Fase de preparación:	Inspección manual		
	Definición del problema			
	Definición de las vistas hipotéticas			
	Fase de reconstrucción			Listado de herramientas para la reconstrucción
	Análisis estático			
	Análisis dinámico			
	Construir las vistas			
	Analizar las vistas			
A top-down approach to construct execution views of a large software-intensive system	Diseño de la reconstrucción			
	Ejecución de la reconstrucción			Meta-modelo de ejecución
	Identificación de tareas	Perfilado/Instrumentación	Python scripts	Vistas de ejecución
	Interpretación de la información en tiempo de ejecución	Reglas de mapeo	Python scripts	

Nombre	Actividades	Técnicas	Herramientas	Instrumentos
	Construcción del modelo en tiempo de ejecución		Python scripts y Graphviz	
A Framework for Obtaining the Ground-Truth in Architectural Recovery	Capturar los principios del dominio y la aplicación	Inspección manual		Principios de mapeo
	Seleccionar una técnica de recuperación genérica			
	Extraer información a nivel de implementación		IBM RSA y Class Dependency Analyzer	
	Aplicar la técnica de recuperación genérica			
	Utilizar los principios de aplicación y dominio			
	Identificar componentes			
	Pasar la revisión autorizada revisada a certificación			
	Hacer las modificaciones sugeridas			
Que-ES Architecture Recovery, QAR	Extracción de al información			
	Definición del modelo conceptual			
	Extracción de la vista estática	Análisis estático		
	Extracción de la vista dinámica	Análisis dinámico		
	Definición de la arquitectura preliminar			

Tabla 11. Caracterización de procesos de recuperación de arquitectura

Por otra parte, el análisis realizado indica una tendencia a utilizar herramientas desarrolladas por los mismos autores, circunstancia que justifica el gran número de herramientas existentes y la complejidad al momento de seleccionar la más conveniente, aspecto que se analiza más adelante en la sección de herramientas. De igual forma, la definición de instrumentos para el desarrollo de las distintas actividades del proceso, facilita su interpretación y ejecución, por lo tanto, en esta propuesta se incluyen una serie de instrumentos que se explican más adelante en forma detallada para cada fase de la metodología.

5.3. El proceso de recuperación de la arquitectura

La caracterización de las propuestas identificadas reveló que todas han sido definidas para el contexto del proceso de desarrollo de software, evidenciando la necesidad de una metodología que guíe el proceso de recuperación de la arquitectura, teniendo en cuenta las características específicas de cada contexto, en el que se presenta la situación que se quiere resolver. Por eso, el proceso se definió a partir de la integración de las propuestas identificadas, relacionadas en el Anexo A., agregando los aspectos pertinentes al manejo de los posibles contextos, y manteniendo la coherencia con el modelo conceptual definido para el marco de referencia.

El proceso está estructurado en cuatro fases, cada una tiene establecido uno o varios hitos y un conjunto de actividades que se realizan bajo una secuencia lógica, como se muestra en la Figura 14. Cada actividad define las metas que se pretenden lograr y las técnicas e instrumentos necesarios para su realización. De igual forma, para cada actividad se identifican los argumentos que la justifican (motivación), los interesados que intervienen, los artefactos que se requieren como entrada y los artefactos que se generan como salida o resultado. Si la actividad conlleva a la realización de tareas, los aspectos anteriormente mencionados se especifican para cada tarea.

En orden de ejecución de las actividades que conforman cada una de las fases se establece en el plan de trabajo, dependiendo el análisis que se haga del contexto y la situación que genera el proceso de recuperación de la arquitectura. En la fase de extracción de datos, la actividad orientada a definir el modelo de dominio de la aplicación, sólo se presenta si se utiliza un enfoque Top-Down. Además, el uso de algunas herramientas de recuperación de arquitectura, como Imagix4D y la plataforma Moose, entre otras, fusionan las actividades correspondientes a la descomposición de los artefactos, la abstracción y la representación de modelos.

No se contempla ninguna actividad que implique la modificación del producto, porque este es un proceso de ingeniería inversa. Cada una de las fases se explica a continuación, teniendo en cuenta que el proceso define lineamientos generales para la recuperación y análisis de vistas arquitectónicas, sin detallar las técnicas que se utilizan para cada situación específica, porque esto se encuentra fuera del alcance

de la tesis y además puede ser investigado en los documentos analizados en la revisión la literatura.

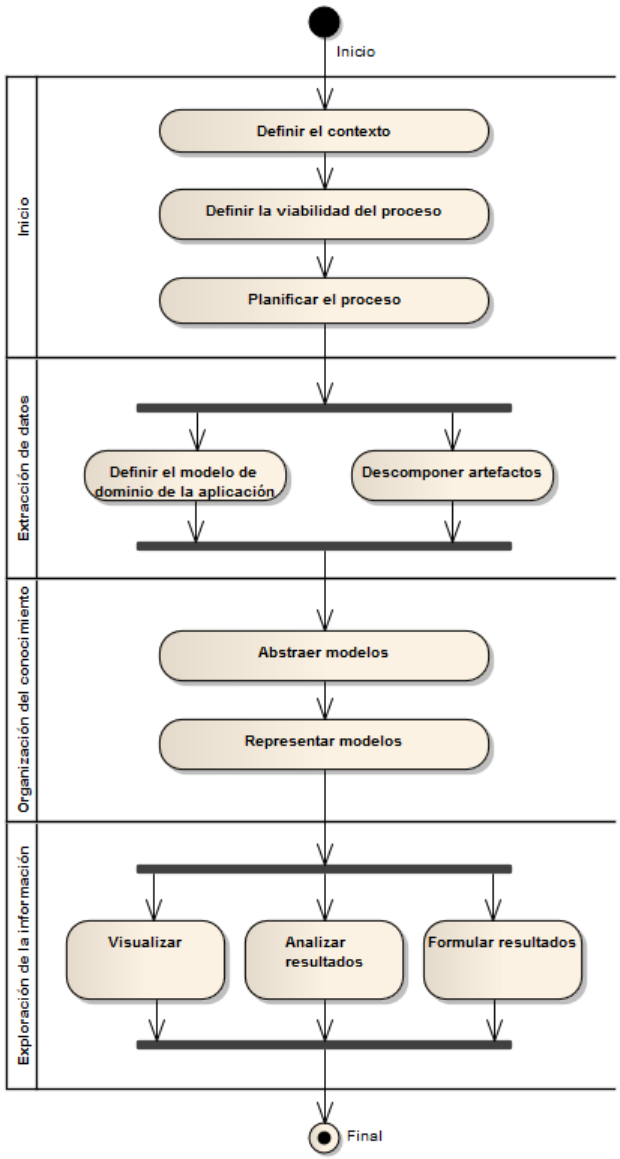

Figura 14. Proceso de la metodología

5.3.1. Fase de inicio

El análisis de las propuestas identificadas a partir de la revisión de la literatura, reveló que sólo una cuarta parte incluye una fase de preparación, previa al proceso de recuperación de la arquitectura, con la intención de: definir el plan concreto de reconocimiento de patrones (Guo et al., 1999), identificar un conjunto de casos de

uso (Bojic y Velasevic, 2000), definir los conceptos arquitectónicos (Riva y Rodríguez, 2002), determinar las vistas y conceptos arquitectónicos (Pinzger et al., 2004), hacer análisis de dominio del metaware (Favre, 2004), definir y seleccionar las versiones (Jansen et al., 2008), o definir el espacio de búsqueda (Chardigny, 2008).

En forma más concreta Stoermer y O'Brien (2001) proponen un fase para preparar el proceso, Van Deursen et al. (2004) definen un paso previo para hacer el diseño de la reconstrucción, que es aplicado por Callo Arias et al. (2011) y que Kang (2009) denomina definición del problema, mientas que García et al. (2012) lo definen como captura de los principios del dominio y la aplicación. Sin embargo, todos asumen que la reconstrucción de la arquitectura se realiza en el contexto del proceso de desarrollo de software, por eso, la fase de inicio se definió complementando las propuestas identificadas, agregando las actividades adicionales que surgen cuando la reconstrucción se hace teniendo en cuenta el contexto de la situación que genera la necesidad.

La fase de inicio tiene como fin comprender el contexto del problema, identificando el ámbito, los actores y sus intereses, los recursos disponibles, la situación que genera la necesidad de recuperar la arquitectura y los propósitos que se desean alcanzar, por eso contempla tres hitos que se logran en forma secuencial, el primero consiste en especificar el ámbito del problema, el segundo en definir la viabilidad de la realización del proceso de recuperación de la arquitectura y el tercero, en establecer el plan de trabajo. Para alcanzar esto hitos se realizan las siguientes actividades: 1) Definición del contexto, 2) Análisis de la viabilidad del proceso y 3) planificación del proceso.

1) Definición del contexto: En el capítulo cuatro se explicó cómo la ingeniería inversa, gracias a su evolución, ha sido utilizada en contextos diferentes al proceso de desarrollo de software, por lo tanto al iniciar un proceso de recuperación de la arquitectura, la primera actividad que se debe realizar es definir en qué contexto se presenta la situación, porque esto determina la orientación del proceso. Esta actividad consiste en determinar el ámbito de la situación que genera la necesidad, para lo cual se establecen dos metas, la primera es identificar el contexto y la segunda plantear el problema.

El cumplimiento de la primera meta radica en definir el contexto describiendo su ámbito, propósitos, recursos e interesados o actores. Para facilitar el cumplimiento de esta meta, en la Tabla 3 se presenta una caracterización de los contextos de uso de la ingeniería inversa identificados en este trabajo de investigación. La identificación del contexto se hace en forma directa por inferencia, a partir de los conocimientos del ingeniero de software experto en ingeniería inversa, y la información básica que se tenga de la situación. Sin embargo, para determinar los demás aspectos que describen el contexto es necesario el apoyo de los expertos en el dominio, por lo cual se recomienda utilizar la entrevista como técnica para lograr este propósito.

El cumplimiento de la segunda meta implica describir la situación que genera la necesidad, indicando sus causas, las circunstancias del contexto y las posibles consecuencias que implican el problema, para realizar el análisis respectivo y establecer el objetivo del proceso de recuperación de la arquitectura. Esto se logra utilizando técnicas como talleres, listas de chequeo, juego de roles y análisis de escenarios (Van Deursen et al., 2004), entrevistas con usuarios, expertos en el dominio y personal técnico, además de la revisión y estudio de documentos, formularios, guías y demás material disponible (Pashov y Riebisch, 2004).

El objetivo del proceso se plantea en función de la solución del problema, manteniendo coherencia con las vistas que se recuperan para darle solución. En la Tabla 12 se presenta una plantilla para la descripción del problema, que sirve como instrumento para cumplir las metas de esta actividad.

Contexto:	
Ámbito:	
Propósito:	
Descripción de la situación	
Circunstancias que rodean el problema Causas del problema El problema Consecuencias	
Objetivo del proceso:	
Actores o interesados	
Rol del actor	Intereses
Listado de recursos disponibles	
Recurso	Descripción

Tabla 12. Plantilla para la descripción del problema

La actividad orientada a la definición del contexto recibe como entrada el conocimiento del experto y de los actores o interesados en el problema, además de la información existente en los documentos disponibles, que hacen parte de los recursos del contexto. Genera como salida el documento de descripción del problema, en el que se describe el contexto, con todos sus aspectos, la situación que genera la necesidad de recuperar la arquitectura, y se plantea el objetivo del proceso, logrando el primer hito de la fase de inicio: especificar el ámbito del problema. Se recomienda que la realice el ingeniero de software con el apoyo de los expertos en el contexto del problema.

2) Análisis de la viabilidad del proceso: En el contexto del desarrollo de software, ante una situación que requiera la recuperación de la arquitectura, no es indispensable determinar si este objetivo es viable o no, porque siempre se cuenta

con todos los artefactos software necesarios para realizarlo (ver Tabla 4). Sin embargo, para los contextos de la computación forense, la seguridad informática y la educación, el escenario es diferente. Por lo tanto se hace necesario realizar esta actividad, que consiste en determinar si se cuenta con todos los recursos necesarios para lograr el objetivo planteado en la descripción del problema.

En este caso el término recurso hace referencia a: 1) los artefactos que representan al software objeto de estudio, 2) las herramientas necesarias para la recuperación de la arquitectura, 3) el talento humano con la formación y capacidades pertinentes, relacionadas con el contexto del problema y la ingeniería inversa; y 4) la capacidad económica. En esta actividad se establece una sola meta: identificar si el proyecto es viable o no. Para lograrla se recomienda realizar los siguientes pasos:

1. Teniendo en cuenta el objetivo planteado en la descripción del problema establecer las vistas objetivo que se deben recuperar. Para facilitar esta tarea, se elaboró la Tabla 5, que representa un instrumento que relaciona para cada contexto de uso las vistas que son obligatorias, complementarias, o que requieren de un análisis particular de la situación para determinar su rol en el proceso de recuperación de la arquitectura.
2. Se deben identificar los artefactos software que se requieren para recuperar cada vista objetivo. Para lograr este fin se recomienda utilizar como instrumento una lista de chequeo.
3. Verificar la disponibilidad de los artefactos software requeridos. En caso de no disponer de estos artefactos se determina que el proceso no es viable y termina.
4. A partir de las vistas objetivo identificar las posibles técnicas y herramientas necesarias para recuperar la arquitectura. Se recomienda identificar varias técnicas y herramientas, si es posible, para que se puedan seleccionar la que mejor se ajuste a las condiciones de la situación que se está solucionando. Para la selección de las herramientas se puede utilizar el modelo ontológico propuesto en este trabajo.
5. Verificar la disponibilidad de las herramientas seleccionadas. Si son herramientas de uso libre, descargarlas, configurarlas y probarlas. Si son herramientas de uso comercial establecer los costos de adquisición. En caso de no disponer de las herramientas se determina que el proceso no es viable y termina.
6. Identificar la necesidad de cualificar el talento humano, o de contratar personal especializado, definiendo los costos pertinentes.
7. Calcular el costo del proceso de recuperación de la arquitectura y determinar su viabilidad a partir de la relación costo beneficio.

Esta actividad recibe como entrada la descripción del problema, el conocimiento de los interesados y documentación; y genera como salida la decisión que determina la viabilidad del proceso de recuperación de la arquitectura, logrando el segundo hito de la fase de inicio. Adicionalmente se establecen las vistas objetivo, el listado de artefactos disponibles y requeridos, el listado de técnicas y herramientas a utilizar y

el costo del proceso. Se recomienda que la realice el ingeniero de software experto en ingeniería inversa, con el apoyo del grupo de interesados.

3) Planificación del proceso: La recuperación de la arquitectura es un proceso complejo, que implica la manipulación de grandes cantidades de información mediante la ejecución de una serie de actividades. Para alcanzar de una manera eficiente y oportuna el objetivo propuesto en la descripción del problema que se va a solucionar, el plan de trabajo se hace en función de las fases de extracción de datos, organización del conocimiento y exploración de la información, siguiendo las secuencia de actividades establecidas para el proceso (ver Figura 14).

En consecuencia, la planificación del proceso tiene como meta definir el plan de trabajo, que sirve como instrumento guía para cumplir el objetivo propuesto, asignando el orden temporal de la realización de las actividades, los responsables y los entregables para cada actividad. La asignación de tiempos se hace a partir del orden secuencial establecido para el proceso, teniendo en cuenta la experiencia de los actores que intervienen. La asignación de los entregables se hace tomando como referencia las salidas que se definen para cada actividad, según lo expresado en este documento. La asignación de los responsables se hace a consideración de quien dirige el proceso, con base en las habilidades de cada actor

Esta actividad recibe como entrada la descripción del problema, las vistas objetivo, el listado de artefactos disponibles y requeridos, y el listado de técnicas y herramientas a utilizar; y genera como salida el plan de trabajo, determinando tiempos, recursos y responsables, cumpliendo el tercer hito establecido en la fase de inicio. En el Anexo C se presenta un resumen de esta fase.

5.3.2. Fase de Extracción de datos

Las fases de extracción de datos, organización del conocimiento y exploración de la información, hacen parte del proceso genérico establecido por Tilley et al. (1996), que siguen todas la propuestas identificadas en la revisión de la literatura, por eso, en la metodología se integran para facilitar la recuperación dependiendo el contexto en que se presente la situación que se desea solucionar. La fase de extracción de datos tiene como fin la recuperación de los elementos estructurales y de comportamiento del sistema (Vasconcelos y Werner, 2011; Kang et al., 2009; Ganesan et al. 2011; Eisenbarth, 2003; Riva y Rodríguez, 2002; Richner y Ducasse, 1999).

De las 89 propuestas identificadas (ver Anexo A), 41 se centran en la recuperación de los aspectos estáticos del sistema, 11 en la recuperación de los aspectos dinámicos y 37 recuperan tanto la estructura como el comportamiento del sistema. El orden de recuperación no es fijo y depende de las técnicas que se utilicen. Las técnicas estáticas sólo realizan la recuperación de la estructura, mientras que en las técnicas dinámicas, unas primero recuperan la estructura y a partir de los resultados obtenidos recuperan el comportamiento (Forster et al., 2013; Mendoca y Kramer,

2001); y otras sólo recuperan el comportamiento, sin necesidad de una recuperación previa de la estructura (Peake et al., 2013; Callo Arias et al., 2011; Abi-Antoun y Barnes, 2010; Abi-Antoun y Aldrich, 2009; Zhou, 2008; Huang et al., 2006; Schmerl et al., 2006; Bojic y Velasevic, 2000; Cook, 1998).

Por lo anterior, en la actividad de extracción de datos se establece como hito la identificación de los elementos estructurales y de comportamiento que conforman el producto software y las relaciones que existen entre ellos en bajo nivel, y para lograrlo se definen dos actividades: 1) La definición del modelo de dominio de la aplicación y 2) la descomposición de artefactos.

1) *Definición del modelo de dominio de la aplicación*: El entendimiento del dominio de la aplicación permite orientar el proceso de recuperación de la arquitectura, focalizando los esfuerzos hacia los propósitos pertinentes del problema (Tilley et al., 1996), haciéndolo más eficiente. Por eso, esta actividad se propone como la primera de la fase de extracción de datos, sin embargo, es necesario aclarar que, dependiendo de las circunstancias y el contexto en el que se presenta el problema, es probable que esta actividad se realice en forma iterativa en cuatro momentos.

El primero corresponde a la definición de la viabilidad del proceso, cuando se identifican las vistas objetivos y los artefactos que se requieren recuperar, porque sirve para orientar al experto en ingeniería de software en estas tareas. El segundo momento es, cuando se establece el plan del proceso, porque sirve de soporte para tomar decisiones sobre las tareas a realizar y la asignación de recursos. El tercer momento se presenta en la fase de extracción de datos, porque ayuda a guiar el proceso, y finalmente, en la fase de organización del conocimiento, porque se puede utilizar como referente para la abstracción de modelos.

La meta de esta actividad consiste en definir el dominio de la aplicación y se logra utilizando técnicas de modelado conceptual, para lo cual requiere como entrada la descripción del problema, el conocimiento del experto, los usuarios y el personal técnico, obteniendo como resultado el modelo de dominio de la aplicación, representado por medio de diagramas UML. Este modelo puede ser complementado con una representación de la arquitectura hipotética del sistema.

2) Descomposición de artefactos: Para llegar a la representación de la arquitectura concreta por medio de las vistas objetivo establecidas en la fase de inicio, es necesario realizar procesos de transformación, que inician desde los artefactos que representan la implementación del sistema. La primera transformación corresponde a la descomposición de los artefactos, y tiene como fin identificar los elementos que conforman el sistema y sus relaciones. Esto se logra aplicando técnicas de análisis estático y dinámico, de acuerdo a lo establecido en el plan de trabajo definido para el proceso. Como resultado de esta actividad se obtiene el modelo del sistema en bajo nivel, para lo cual se pueden utilizar múltiples representaciones como FAMIX, KDM, Rigi Standard Format, entre otras(El Boussaidi et al., 2012).

Desde la perspectiva de KDM recuperar los aspectos estáticos del sistema, implica recuperar los elementos del paquete fuente de la capa de infraestructura, los elementos del paquete código correspondiente a la capa elementos del programa y los elementos que representan la parte estructural de los paquetes: 1) plataforma y 2) datos de la capa de recursos en tiempo de ejecución.

Si se desea recuperar los aspectos dinámicos del sistema, desde la perspectiva de KDM, el hito para esta fase implica identificar los elementos del paquete acción de la capa elementos del programa y los elementos que conforman los paquetes de la capa de recursos en tiempo de ejecución. Los elementos concretos que se recuperan dependen de las vistas que se deseen reconstruir, por lo tanto no es indispensable recuperar todos los elementos que conforman el producto software en un proceso específico. En el Anexo D se presenta un resumen de la fase de extracción de datos.

5.3.3. Fase de organización del conocimiento

La mayoría de los autores se refiere a esta fase como la fase de abstracción (Solms, 2013; Rasool y Asif, 2007; Pinzger et al., 2004; Stoermer et al., 2003; Riva, 2002; Riva y Rodríguez, 2002), que implica separar (o identificar) los aspectos relevantes y omitir los detalles para obtener la arquitectura, lo que para Müller et al. (1993), Stoermer y O'Brien (2001) significa composición. Desde la perspectiva de Panas et al. (2003) esta fase se denomina focalización de la información y consiste en gestionar gran cantidad de información de manera apropiada, para lo cual plantean tres técnicas básicas: Abstracción de la información, compresión de la información y fusión de la información. Por otra parte, Van Deursen et al. (2004) denominan esta fase inferencia del conocimiento, porque a partir de los elementos identificados en la fase anterior se derivan las vistas que se desean reconstruir, usando reglas de mapeo y el conocimiento del experto; lo que para Kazman et al. (2002) corresponde a la fase de fusión de vistas.

Esta fase establece como hito convertir el modelo del sistema en bajo nivel en un modelo en alto nivel, logrando una representación y almacenamiento de los datos obtenidos en la fase de extracción, en una forma que permita su recuperación de manera fácil y eficiente, haciendo posible el análisis de los elementos que conforman el sistema y sus relaciones. Esta fase comprende dos actividades: 1) La abstracción de modelos y 2) la representación de modelos.

1) Abstracción de modelos: La gran cantidad de información obtenida en la fase de extracción de datos y la complejidad de su representación, dificulta su interpretación, por eso esta actividad plantea como meta, identificar elementos y relaciones en alto nivel a partir del modelo del sistema en bajo nivel. Esto se logra aplicando reglas de mapeo y utilizando el conocimiento del experto en el dominio del problema y de la aplicación, para relacionar los elementos básicos extraídos con los elementos del sistema en alto nivel que conforman las vistas objetivo. Las técnicas de organización del conocimiento utilizadas en esta actividad se pueden clasificar en (Van Deursen et al., 2004): manuales, semiautomáticas y automáticas.

2) Representación de modelos: Para cumplir con los lineamientos establecidos para el marco de referencia, es necesario garantizar la representación de los modelos atendiendo las especificaciones OMG UML 2.5 de 2015 y OMG XML Metadata Interchange 2.5.1 del mismo año. Por eso se incluye en la fase de organización del conocimiento esta actividad, orientada a representar los elementos y relaciones identificados en el proceso de abstracción, en modelos del sistema en alto nivel representados en el lenguaje XML, bajo la especificación XMI, logrando el hito establecido para esta fase. Esto se logra aplicando reglas de mapeo a partir de las equivalencias semánticas entre los elementos y las relaciones obtenidos en el procesos de abstracción, con sus equivalentes en el lenguaje unificado de modelado. En el Anexo E se presenta un resumen de la fase de organización del conocimiento.

5.3.4. Fase de exploración de la información

Esta fase ha sido denominada de distintas formas por diversos autores: Presentación (Müller y Orgun, 1993; Bowman et al., 1999; Hassan et al., 2002; Riva y Rodríguez, 2002), visualización (Richner y Ducasse, 1999; Pinzger et al., 2004; Panas et al. 2003), instanciación de elementos y propiedades (Stoermer et al., 2003), descripción (Solms, 2013) e interpretación de la información (Van Deursen et al., 2004), mientras que otros le asignan una calidad implícita dentro de la fase de organización del conocimiento sin definirle un nombre específico (Eisenbarth et a., 2003; Boussaidi et al., 2012; Garcia et al., 2012).

Esta fase tiene como fin brindar los mecanismos para visualizar, analizar, navegar y presentar los resultados del proceso de recuperación de la arquitectura, por eso se establece como hito, la elaboración del informe con el análisis de las vistas arquitectónicas recuperadas, que indique el cumplimiento de los objetivos establecidos para el proceso en la fase de inicio. Comprende tres actividades: 1) La visualización, 2) el análisis y 3) la formulación de resultados, que pueden realizarse en forma secuencial o concurrente, dependiendo el estilo de grupo de trabajo.

1) La visualización: Consiste en presentar los resultados obtenidos en la fase de organización del conocimiento, en forma gráfica para facilitar su comprensión e interpretación por parte de los interesados en el proceso. La visualización puede realizarse a nivel de código, a nivel de clases y a nivel de arquitectura, usando editores de código fuente y herramientas CASE de modelado. El proceso de visualización implica utilizar técnicas de mapeo y metáforas para presentar vistas objetivo representadas en UML, a partir de los modelos obtenidos en la fase de abstracción del conocimiento representados en XMI.

La visualización se complementa con estrategias de navegación, que utilizan hipervínculos para facilitar el desplazamiento por las estructuras de información generadas en la fase de organización del conocimiento, haciendo posible la exploración de la información.

2) El análisis de los resultados: En un proceso de recuperación de la arquitectura, generalmente no es suficiente llegar a las vistas objetivo, además es necesario extraer información que no está disponible en forma explícita en los modelos, por lo tanto se requiere de análisis complementarios que contribuyan a su interpretación y por lo tanto a la comprensión del sistema objeto de estudio. En este orden de ideas, se establece como meta interpretar los resultados obtenidos en el proceso, aplicando técnicas de inferencia que permitan hacer análisis de los modelos del sistema en alto nivel, usando el conocimiento del experto sobre el dominio del problema y de la aplicación, para lo cual se puede utilizar como instrumento de apoyo el mecanismo de consulta propuesto.

3) La formulación de resultados: La organización y registro de los resultados bajo una estructura fácil de interpretar, posibilita su comunicación a través del grupo de interesados, sirviendo como soporte para la toma de decisiones en el contexto en el que se presenta el problema. Esta actividad tiene como meta organizar los resultados en un informe final, para que sea posible su interpretación por parte de todos los interesados. El documento se construye a partir del análisis de los resultados hecho por los expertos en el dominio del problema y de la aplicación, dejando registro de las vistas objetivo recuperadas en el proceso. En el Anexo F se presenta un resumen de la fase de exploración de la información.

5.4. Herramientas

Debido a la complejidad de la recuperación del conocimiento y a la gran cantidad de información que se debe procesar para lograr este objetivo, se han desarrollado herramientas que aplican técnicas de ingeniería inversa para facilitar el trabajo al ingeniero de software. La revisión de la literatura permitió identificar 115 herramientas especializadas en ingeniería inversa (ver Anexo B), de las cuales 34 recuperan aspectos relacionados con el diseño, 65 recuperan la arquitectura del producto, 5 recuperan la estructura de la base de datos y 11 son para visualización.

Para un mejor entendimiento sobre las herramientas de ingeniería inversa y por lo tanto facilitar su selección en un proceso de recuperación del conocimiento, en esta sección se presenta la caracterización que se definió en la investigación y la ontología que se creó a partir de dicha caracterización.

5.4.1. Caracterización de herramientas

En un escenario de recuperación de arquitectura, es necesario identificar las herramientas más adecuadas para lograr este objetivo, por lo cual, los productores de herramientas de ingeniería inversa deben identificar nuevas funcionalidades y características para ofrecer valor agregado a los usuarios; mientras que los investigadores en este campo del conocimiento, desean proponer nuevos métodos, técnicas y herramientas, que representen mejoras en los procesos de recuperación de conocimiento de sistemas software, para lo cual requieren identificar en forma precisa las características de las herramientas disponibles.

Por lo tanto, usuarios de herramientas de ingeniería inversa, productores e investigadores, requieren de un instrumento que facilite la toma de decisión ante cada una de las situaciones planteadas. Como respuesta a esta necesidad se propone un modelo de caracterización de herramientas de ingeniería inversa (Monroy et al., 2012), basado en las propuestas de Guéhéneuc et al. (2006), Bellay y Gall (1997).

La caracterización de las herramientas de ingeniería inversa se hace teniendo en cuenta dos criterios: el primero corresponde al aspecto estructural, centrando la atención en los elementos básicos que conforman la arquitectura genérica de este tipo de herramientas: Analizador e Interfaz (Chikofsky y Cross, 1990), mientras que el segundo se centra en las propiedades comunes entre ellas.

Aspecto estructural: Para caracterizar las herramientas de ingeniería inversa, tomando como principal referente su arquitectura genérica, se establece la siguiente estructura: elemento, función, aspecto y característica, como se observa en la Tabla 13 se han definido dos tipos de elementos: Analizador e Interfaz. El primero es el componente responsable de la transformación del software objeto de estudio a un mayor nivel de abstracción, mientras que la interfaz se encarga de permitir la interacción entre el usuario y la herramienta.

Una función corresponde a las tareas que realiza cada elemento de la arquitectura, por ejemplo el analizador cumple con tareas de entrada, proceso y salida. El aspecto indica un matiz o rasgo diferenciador para cada función que cumplen los elementos, como es el caso de la función entrada que cumple el analizador, para la cual se definen cinco aspectos: los supuestos de la entrada, la fuente, la precisión, la automatización y el manejo de versiones. Por último se encuentra la característica, entendida como una cualidad que sirve para distinguir a cada uno de los aspectos identificados para las funciones que cumplen los elementos.

En la Tabla 13 se presenta la estructura de caracterización de las herramientas de ingeniería inversa, tomando como referente principal los aspectos estructurales. Las características marcadas con un signo más (+) han sido establecidas por Guéhéneuc et al. (2006) y las que tienen un asterisco (*) han sido definidas por Bellay y Gall (1997). El principal aporte del este trabajo fue definir la estructura de caracterización para facilitar el análisis, la comprensión y selección de las herramientas, de tal manera que se puedan identificar sus ventajas y desventajas. A continuación sólo se explican aquellas características que se han incluido.

Para la función de entrada del analizador se incluyeron dos características nuevas asociadas a la fuente. La primera hace referencia al tipo de fuente, asumiendo que en la actualidad el proceso de ingeniería inversa incluye, además del código fuente, colecciones de datos y documentación. Adicionalmente, se ha definido la relación entre las características: tipo, modelo y representación de la fuente, teniendo en cuenta que para cada tipo de fuente existe un modelo que la describe y una forma de representación, como se muestra en la Tabla 14.

Elemento	Función	Aspecto	Características
Analizador	Entrada	Supuestos (+)	Estructurales, Semánticos
		Fuente	Tipo, Modelo (+), Representación (+), Origen de datos, Tipos de datos(+) , Tipos de aplicación
		Precisión (+)	Semántica, Sintáctica
		Automatización (+)	Automática, Semiautomática, Manual
		Versiones (+)	Múltiples, Singular
	Proceso	Técnica (+)	Adaptabilidad, Automatización, Complejidad, Determinismo, Explicación, Tratamiento de aspectos indefinidos, Nivel de detalle, Incremental, Iterativo, Independiente del lenguaje, Madurez, Método, Modelo, Escalabilidad, Semántica
		Tipo de analizador	Dinámicos, Estáticos, Híbridos, Históricos
		Aspectos generales	Reconocimiento de patrones, Artefactos que recupera, Análisis incremental (*), Reparsing (*), Preprocesador de comandos configurable (*), Soporte para conmutadores de compilador(*), Capacidad para analizar Funciones/Variables externas (*), Capacidad para trabajar con información incompleta, Capacidad para trabajar con caja negra, Identificación de clonación, Ingeniería inversa de Documentos WSDL en UML
	Salida	Tipo	Modelo, Representación
		Reporte (+)	Legibilidad, Nivel de detalle, Modelo, Calidad, Representación, Tipo de datos
Interfaz	Visualización	Trazabilidad	Horizontal, Vertical
		Tipo (*)	Estática, dinámica
		Tipo de Navegación (*)	Capas, Ojo de pescado, SHriMP: Simple Hierarchical Multi Perspective view
		Calidad (+)	Sintáctica, Semántica
		Niveles de abstracción	Código, Diseño, Arquitectura, Requerimientos, Modelo del negocio
		Análisis de los resultados	Capacidades de hipertexto (*), Capacidad para analizar diferencias, Mecanismos de consulta, Capacidad para agrupar cambios relacionados, Capacidad para clasificar y combinar información desde repositorios de software diferentes y heterogéneos
	Edición	Adaptabilidad del reporte generado	Anotaciones de ayuda (*), Permite organizar el informe (*), Capacidad para mejorar el diseño del reporte de salida (*), Facilidades de edición
		Capacidad de salida (*)	Generación automática de documentación, Formatos de exportación, Impresión de informes
		Generales	Definición de proyectos, Función de búsqueda, Capacidad para seleccionar entradas representativas

Tabla 13. Estructura de caracterización de herramientas de ingeniería inversa

La segunda característica corresponde al tipo de aplicación, los posibles tipos son: aplicación de escritorio, aplicación web, aplicación RIA (Aplicación Rica de Internet), servicio, aplicación móvil y aplicación para televisión digital. Se utiliza para definir la

naturaleza de la tecnología de interacción, según la experiencia de usuario y la tecnología con la que está implementada la aplicación objeto de estudio, con el ánimo de facilitar la identificación del uso de patrones arquitectónicos.

Fuente		
Tipo	**Modelo**	**Representación**
Código Fuente	Paradigma de programación	Lenguajes de programación
Colección de datos	Relacional, jerárquico, etc.	SQL, XML, etc.
Documentación	UML, ADL, Lenguaje natural	Gráficos, texto

Tabla 14. Relación entre el tipo, el modelo y la representación de la fuente

Para la función proceso se ha definido el aspecto denominado tipo de analizador, teniendo en cuenta la propuesta hecha por Canfora et al. (2011), quien distingue cuatro tipos de analizadores: Estático: extrae la estructura del sistema, dinámico: extrae el comportamiento del sistema, híbrido: extrae tanto la estructura como el comportamiento del sistema, y analizador histórico que extrae información que permite establecer la evolución de un producto software.

Adicionalmente, en los aspectos generales de la función proceso se han incluido características, que corresponden a los retos actuales que deben atender las herramientas de ingeniería inversa, tales como la capacidad para trabajar con información incompleta, la capacidad para trabajar con caja negra para el caso concreto de los sistemas orientados a servicios, la capacidad para identificar clonación de código facilitando el trabajo de reutilización y la comprobación del manejo de derechos de autor; así como también la capacidad para realizar tareas de ingeniería inversa a partir de documentos WSDL.

También se agregaron a los aspectos generales características que permiten calificar de manera más detallada al analizador de una herramienta de ingeniería inversa: Reconocimiento de patrones y Artefactos recuperados. El primero indica si la herramienta reconoce o no la existencia del uso de patrones a partir de la entrada que se le suministra, en caso afirmativo se debe especificar el tipo de patrón reconocido (de diseño, arquitectónico, etc.) y los nombres de los patrones concretos que identifica según el tipo (singleton, Abstract factory, etc).

Los artefactos recuperados corresponden a las piezas de información que recupera el analizador, está directamente relacionado con su tipo, el modelo y la representación de la salida. Para el caso de un analizador estático que utilice como modelo UML, la representación de la salida podría ser a través de diagramas de clase, estructuras compuestas, de paquetes, de componentes o de despliegue; mientras que para un analizador dinámico que utilice el mismo modelo, la representación de la salida podría ser diagramas de casos de uso, de secuencia, comunicación, actividades, máquinas de estado, general de interacción, procesos, estado o tiempos.

Para la función que representa la salida del analizador se ha incluido el tipo de salida, que es determinado por el tipo de analizador, por ejemplo, para un analizador estático el tipo de salida puede corresponder a una vista estática, cuyo modelo en UML está representado a por medio de un diagrama de clases.

El segundo componente representativo de cualquier herramienta de ingeniería inversa es la interfaz de usuario, la cual permite realizar funciones de visualización y edición. En el caso de la visualización, se presenta con frecuencia la necesidad de rastrear los modelos obtenidos como resultado de la ingeniería inversa, desde el código fuente hasta el modelo del dominio del problema, pasando por el diseño y los requerimientos, lo que se conoce como trazabilidad horizontal. Del mismo modo, es necesario rastrear los distintos artefactos obtenidos para cada una de estas actividades (trazabilidad vertical), por ejemplo cuando a nivel de diseño se hace el seguimiento de la estructura de un componente por medio de su diagrama de clases y su comportamiento a través de un diagrama de secuencia (Pfleeger, 2010).

También se incluye un nuevo aspecto denominado nivel de abstracción, que hace referencia a la capacidad que tiene la herramienta para permitir visualizar los resultados que arroja, desde el código fuente hasta el modelo del negocio, pasando por el diseño, las vistas arquitectónicas y los requerimientos del sistema. Además, se incorporan nuevas características que permiten establecer la capacidad de análisis de resultados de la herramienta, a través de mecanismos de consulta, de análisis de diferencias, de agrupación de cambios relacionados y de clasificación y combinación de información desde repositorios de software diferentes y heterogéneos.

Con respecto a las capacidades de edición de la herramienta se incluyeron características como la facilidad de edición, que implica acciones como cortar, copiar, pegar y deshacer, entre otras que permiten la adaptabilidad del reporte. Igualmente se agregaron aspectos generales de edición, tales como: la posibilidad de definir proyectos para agrupar resultados según la necesidad del usuario; la capacidad para realizar búsquedas en los artefactos recuperados y la capacidad para seleccionar entradas representativas, dándole mayor libertad al usuario y haciendo más eficiente su trabajo.

Propiedades comunes: Para el segundo criterio que se utilizó en la caracterización se establece un factor y una propiedad, como se observa en la Tabla 15. El factor corresponde a un matiz o rasgo diferenciador de la herramienta, mientras que la propiedad hace referencia a una cualidad que distingue a cada uno de los factores identificados para las herramientas. Todas las propiedades identificadas han sido definidas previamente por Guéhéneuc et al. (2006) y aquellas marcadas con un asterisco (*) en la Tabla 15 han sido descritas por Bellay y Gall (1997). En este trabajo se distinguen cuatro aspectos: Contexto, intención, perfil del usuario y madurez de la herramienta.

El contexto define el entorno y las restricciones externas de la herramienta, la intención establece los objetivos para los cuales ha sido desarrollada la herramienta,

el perfil de usuario indica las habilidades y características que debe cumplir la persona que utilice la herramienta, mientras que la madurez de la herramienta hace referencia al conjunto propiedades que determinan el grado con el cual la herramienta cumple con los requerimientos técnicos del usuario.

Factor	Propiedades
Contexto	Tiempo de vida, Metodología, Campo de acción, Escenario, Universo del discurso
Intención	Objetivos a largo plazo, Objetivos a corto plazo
Perfil del Usuario	Conocimiento, Experiencia, Prerrequisitos, Usuario objeto, Tipo de usuario
Madurez de la Herramienta	Calidad, Tipo de licencia, Independencia del lenguaje, Documentación, Plataformas soportadas (*), Editor Integrado o externo (*), Navegador Integrado o externo (*), Capacidad de almacenamiento (*), Base de usuarios, Soporte a múltiples usuarios

Tabla 15. Propiedades Comunes

El instrumento propuesto además de facilitar el análisis y entendimiento de las herramientas de ingeniería inversa, permite la valoración independiente y conjunta de cada uno de sus componentes. No pretende establecer un modelo terminado de caracterización, sino un instrumento dinámico que debe ser permanentemente actualizado para que atienda a las necesidades del momento. Se recomienda utilizarlo cuando se requiera caracterizar herramientas de ingeniería inversa, teniendo en cuenta que no es indispensable valorar todas las características identificadas, más bien, se sugiere establecer claramente cuáles son pertinentes a partir del objetivo propuesto para la valoración

Es de utilidad para usuarios de herramientas de ingeniería inversa, porque ofrece una plantilla de características que brindan este tipo de herramientas, de tal manera que puede seleccionar aquellas de su particular interés para establecer un comparativo entre las herramientas existentes, sirviéndole de referente y como instrumento de apoyo para la toma de decisión al momento de escoger la más apropiada. Del mismo modo, este instrumento puede ser utilizado por los productores, para caracterizar la herramienta que desarrollan, identificando sus fortalezas para mostrarlas como estrategia de mercadeo y las debilidades para establecer los aspectos a mejorar y futuros desarrollos.

5.4.2. Modelo ontológico para contextos de uso de las herramientas

El número representativo de herramientas de ingeniería inversa identificadas en la revisión de la literatura, evidencia la dificultad para seleccionar las herramientas más adecuadas ante una situación específica, como lo plantea la metodología. Por eso se definió el modelo ontológico que se explica a continuación (Monroy et al., 2016),

usando como referente el modelo de caracterización de herramientas expuesto. La implementación del modelo se realizó en Protégé 5.0 (Stanford, 2015)

El dominio de la ontología está determinado por los contextos de uso de las herramientas de ingeniería inversa en el ámbito de la ingeniería de software. Esta ontología se utiliza para facilitar las tareas de toma de decisión, con respecto al uso de herramientas de ingeniería inversa, a partir de las circunstancias que describen el contexto de uso, por lo tanto, está orientada para ser utilizada por ingenieros de software, docentes y estudiantes en este campo del conocimiento, además de los profesionales relacionados con los contextos de uso establecidos.

En consecuencia, la ontología ha sido diseñada principalmente para responder la pregunta ¿Qué herramientas de ingeniería inversa se pueden utilizar ante un contexto de uso determinado?. Además, puede responder entre otras a las siguientes preguntas: ¿Qué herramientas cumplen con una función específica?, ¿Qué herramientas cumplen con una característica determinada?, ¿Qué funciones se tienen en cuenta en un contexto específico? ¿Qué características se tienen en cuenta en un contexto específico?.

El modelo ontológico se representa en la Figura 15 usando la notación de UML por medio de un modelo de dominio, presentando las clases, su jerarquía y las relaciones entre ellas, manifestando la semántica descrita a continuación. La educación, la producción de software, la seguridad informática y la computación forense son contextos de uso que requieren de herramientas de ingeniería inversa, teniendo en cuenta que la producción de software a su vez tiene procesos como el mantenimiento, la documentación, la verificación de software y la reutilización.

En su estructura, según la caracterización hecha por Monroy et al. (2012), las herramientas de ingeniería inversa tienen dos elementos: el analizador, responsable de la conversión del código fuente a un nivel mayor de abstracción, y la interfaz que permite la interacción entre el usuario y la herramienta. Una función corresponde a las tareas que realiza cada elemento de la arquitectura, por ejemplo el analizador cumple con tareas de entrada, proceso y salida. El aspecto indica un matiz o rasgo diferenciador para cada función que cumplen los elementos. La característica, entendida como una cualidad que sirve para distinguir a cada uno de los aspectos identificados para las funciones que cumplen los elementos, y los factores que distinguen las propiedades comunes a cualquier producto software, como es el caso de las herramientas de ingeniería inversa.

Las características de los contextos de uso se establecen a partir de la propuesta de caracterización de herramientas de ingeniería inversa (Monroy et al., 2012), según la cual se definen 82 características clasificadas teniendo en cuenta el elemento, la función y el aspecto. Debido a que no todas las características son determinantes del contexto de uso, para cada una se evaluó su nivel de pertinencia para cada contexto de uso (M: Mantenimiento, V: Verificación de software, Dc: Documentación, R: Reutilización, S: Seguridad informática, E: Educación y C: Computación forense),

obtenido los resultados relacionados en la Tabla 16, donde se omiten las características valoradas como no determinantes.

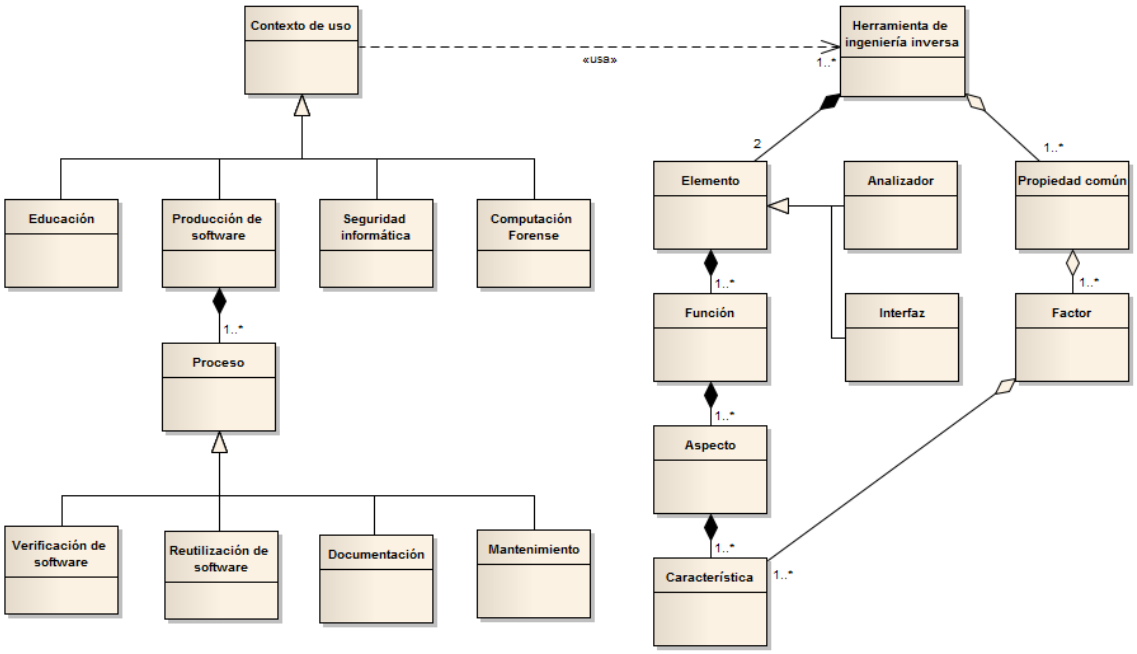

Figura 15. Modelo ontológico

Esto se logró estableciendo la siguiente escala de valoración: La característica es obligatoria (O) para el contexto de uso si al no cumplirse no se le logra el propósito del contexto de uso; la característica es necesaria (N), si contribuye al logro del propósito, pero si no se cumple no implica que no se logre el objetivo del contexto de uso; la característica es deseada (D) si ayuda a mejorar los resultados del propósito del contexto de uso; y finalmente, la característica se valora como no determinante (ND) si no afecta para nada el cumplimiento del propósito del contexto de uso.

	Producción de Sw				***S***	***E***	***C***
Características	***M***	***V***	***Dc***	***R***			
Explicación de la técnica	D	D	D	ND	ND	N	ND
Tratamiento de aspectos indefinidos	D	ND	D	ND	N	ND	O
Tipo de analizador estático	N	N	N	D	N	N	O
Tipo de analizador dinámico	D	D	D	D	D	D	N
Tipo de analizador Híbrido	D	D	D	D	D	D	N
Tipo de analizador Histórico	D	ND	D	D	ND	D	D
Reconocimiento de Patrones	N	N	D	O	D	D	D
Capacidad para analizar Funciones/Variables Externas	D	ND	ND	D	O	ND	D
Capacidad para trabajar con información incompleta	D	D	D	D	O	D	O

	Producción de Sw				***S***	***E***	***C***
Características	***M***	***V***	***Dc***	***R***			
Capacidad para trabajar con caja negra	D	D	D	D	D	D	O
Identificación de clonación	N	D	ND	N	D	D	O
Ingeniería Inversa de documentos WSDL en UML	D	D	D	ND	ND	D	D
Legibilidad del reporte	N	N	O	N	N	O	D
Calidad del reporte	N	N	N	N	D	N	N
Trazabilidad Horizontal	O	O	D	ND	N	D	D
Trazabilidad Vertical	O	O	N	D	N	D	D
Visualización Estática	N	N	N	N	N	N	N
Visualización Dinámica	N	N	N	D	N	N	N
Niveles de Abstracción: Código	O	O	D	N	N	O	D
Niveles de Abstracción: Diseño	O	O	O	N	N	O	D
Niveles de Abstracción: Arquitectura	N	N	N	D	N	D	D
Niveles de Abstracción: Requerimientos	D	D	D	D	D	D	D
Niveles de Abstracción: Modelo del Negocio	D	D	D	D	D	D	D
Capacidades de Hipertexto	N	N	N	D	D	N	D
Capacidad para analizar diferencias	N	N	D	D	ND	D	N
Mecanismos de consulta	N	N	D	D	D	D	N
Capacidad para agrupar cambios relacionados	N	N	N	D	ND	N	D
Anotaciones de ayuda	D	D	D	ND	D	D	ND
Permite organizar el informe	N	N	N	ND	D	N	ND
Generación automática de documentación	N	N	O	ND	D	N	ND
Función de Búsqueda	N	N	N	N	N	N	N
Capacidad para seleccionar entradas representativas	N	N	N	D	D	D	D

Tabla 16. Características de los contextos de uso

Capítulo 6

Evaluación de la propuesta

La evaluación del marco de referencia comprende dos aspectos, el primero corresponde al modelo conceptual y el segundo al mecanismo de consulta, en coherencia con lo establecido en los objetivos de la tesis. Por eso, en este capítulo inicialmente se presenta el prototipo que implementa el diseño propuesto para el mecanismo de consulta, y posteriormente se documenta la realización del estudio de caso sobre la aplicación del modelo conceptual y el mecanismo de consulta en diferentes contextos de uso.

6.1. Prototipo

Con base en el diseño propuesto para el mecanismo de consulta se implementó el prototipo QModel-XMI que se explica en esta sección. Inicialmente se presentan los requisitos del prototipo, resaltando las funcionalidades, características y restricciones que debe cumplir; luego se plantea su diseño mostrando los escenarios de uso, la comunicación entre procesos del sistema por medio de un diagrama de componentes, y finalmente se explica la forma como fue implementado, por medio de la vista de desarrollo y la vista de despliegue.

6.1.1. Requisitos del prototipo

Los requisitos del prototipo se formulan describiendo la perspectiva del producto, las funciones que debe desempeñar, las características que debe tener y las restricciones que debe cumplir, presentadas a continuación. QModel-XMI es un mecanismo de consulta que apoya el análisis de modelos UML representados en repositorios XMI, ofreciendo una interfaz gráfica para la captura de consultas escritas en la lengua castellana, que son respondidas con base en la información contenida en el archivo XML, previamente seleccionado por el usuario.

La función principal del prototipo consiste en responder a consultas en lenguaje natural, como las que se presentan en la Tabla 9, hechas sobre modelos UML en versiones 2.* representados en archivos XMI en versiones 2.1 y posteriores; así

como también a consultas sobre el cálculo de métricas del modelo relacionadas a continuación:

- Métricas de tamaño de clase: Número de métodos públicos que tiene la clase, número de métodos totales que tiene la clase, número de atributos públicos que tiene la clase, número de variables totales que tiene una clase, número de variables estáticas de la clase y número de métodos estáticos de la clase.
- Métricas de herencia de clases: Número de métodos heredados de una clase, número de métodos que ha sobrescrito una clase, número de métodos que no han sido heredados, grado de profundidad del árbol de herencia y número de hijos.
- Métricas internas de clase: Promedio de parámetros respecto a los métodos de una clase, índice de especialización y promedio de métodos por clase.
- Métricas de diseño orientado a objetos: Factor de ocultamiento de los métodos, factor de ocultamiento de los atributos, porcentaje de métodos heredados, porcentaje de atributos heredados, factor de polimorfismo, factor de acoplamiento, factor de agrupación y factor de reutilización.
- Otras métricas del modelo: Número de clases, número de clases abstractas, número de interfaces y número de paquetes.

El prototipo debe ser diseñado para ser usado por personas con conocimientos básicos sobre ingeniería de software, así como también por expertos en el tema, brindando una interfaz gráfica de usuario para la interacción en lenguaje natural (Español) escrito, al momento de hacer consultas a los modelos UML representados en archivos XMI. El resultado de la consulta debe expresarse también en lenguaje natural, con base en los contenidos XMI de los modelos, así como también con etiquetas XMI extraídas directamente de los archivos XML.

No requiere de interfaces hardware específicas, ni tampoco establecer interfaces explicitas con otros sistemas software, sin embargo, para garantizar su interoperabilidad, se establece como restricción de diseño que debe ser implementado como componente independiente, con capacidad de reutilización y de fácil mantenimiento. Del mismo modo, su funcionalidad debe ser garantizada para versiones XMI 2.1 y posteriores, manteniendo los lineamientos establecidos por la respectiva especificación.

Otra restricción de diseño consiste en que se debe utilizar el Lenguaje XQuery, por sus características técnicas favorables y por ser el estándar establecido por el Consorcio W3C. Además, porque existe un amplio número de motores de uso libre y comercial, que permiten la ejecución de consultas escritas en este lenguaje. Por tratarse de un prototipo, para la versión inicial no se establecen atributos de calidad específicos sobre el rendimiento, la eficiencia y la disponibilidad del producto. Sin embargo, se debe garantizar su confiabilidad y capacidad de mantenimiento, por las razones presentadas anteriormente.

6.1.2. Diseño del prototipo

El diseño del prototipo se elaboró con base en el modelo conceptual del mecanismo de consulta, explicado en el capítulo 4 y los requisitos expuestos previamente. Se representa por medio de la vista de escenarios de uso y el modelo de componentes, que se explican a continuación.

En este orden de ideas, se definió que el sistema sólo tiene un actor, que por efectos prácticos se denominó ingeniero de software, aunque para el uso del sistema sólo se requieren conocimientos básicos sobre este campo de conocimiento. Los escenarios de uso del prototipo se establecieron a partir de las funcionalidades que debe cumplir, como se observa en la Figura 16. El actor inicialmente debe seleccionar el archivo XMI que contiene el modelo que se va a analizar. Acto seguido, el actor puede registrar sus consultas en español en forma escrita y dar la orden al sistema para que el sistema que se ejecute la consulta.

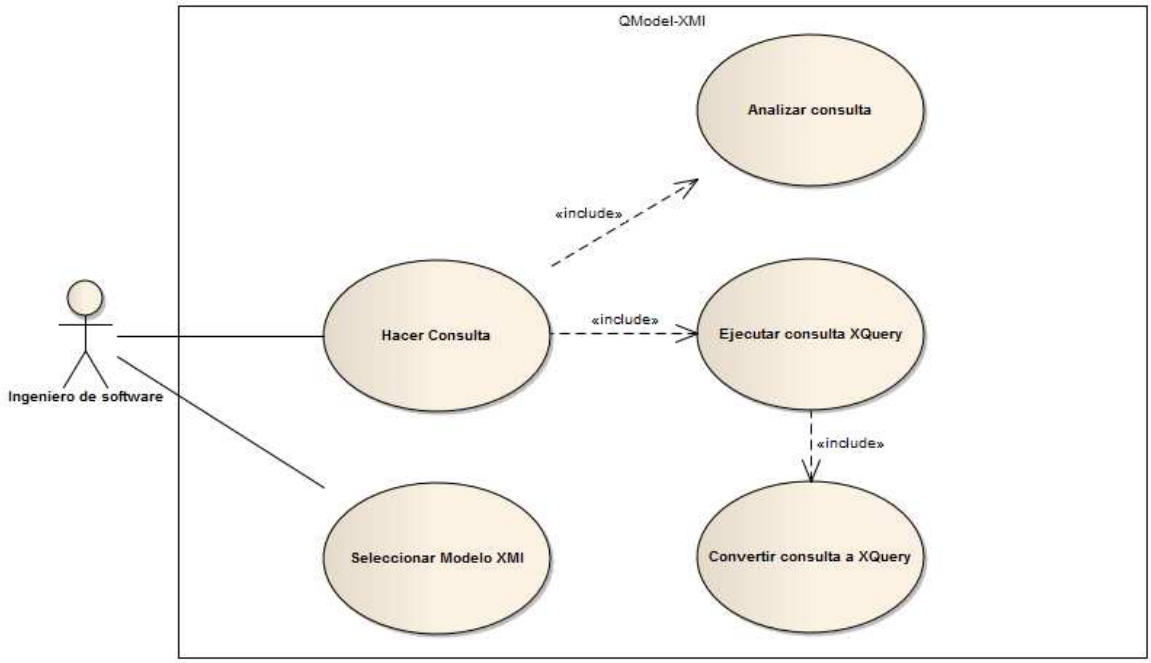

Figura 16 Escenarios de uso de QModel-XMI

El sistema realiza el análisis lingüístico de la consulta, para verificar si a partir de su sintaxis y gramática es una consulta válida. En caso afirmativo, se ejecuta la consulta identificando si se trata del cálculo de una métrica o de una pregunta directa sobre el modelo, para lo cual se hace la respectiva conversión de la consulta del lenguaje natural al lenguaje de consulta XQuery.

Para cumplir con estos escenarios de uso se decidió desarrollar un aplicativo de escritorio, aplicando el patrón arquitectónico Modelo Vista Controlador, para

garantizar la evolución independiente de cada uno de los componentes que lo conforman, como se muestra en la Figura 17. 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 'ContolConsutla' 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 'MotorConsutla' 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.

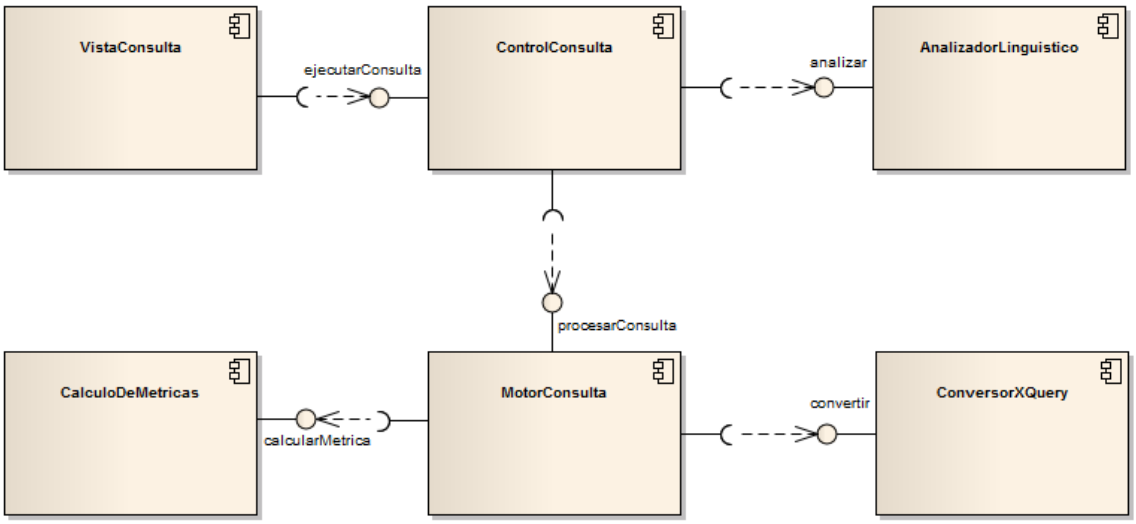

Figura 17 Componentes de QModel-XMI

6.1.3. Implementación del prototipo

El prototipo del Mecanismo de consulta QModel-XMI se implementó usando el lenguaje de programación Java, para aprovechar las ventajas que representa la Máquina Virtual de Java 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 paquetes atendiendo la estructura definida por los componentes que conforman el prototipo, como se muestra en la vista de desarrollo representada en la Figura 18.

Cada componente diseñado para el sistema se implementó en un paquete con el mismo nombre, para mantener la trazabilidad y consistencia entre los modelos. Además se agregó un paquete para el manejo de la persistencia representada por

los archivos XML. La ejecución de las consultas se logra reutilizando la librería BaseX841.jar.

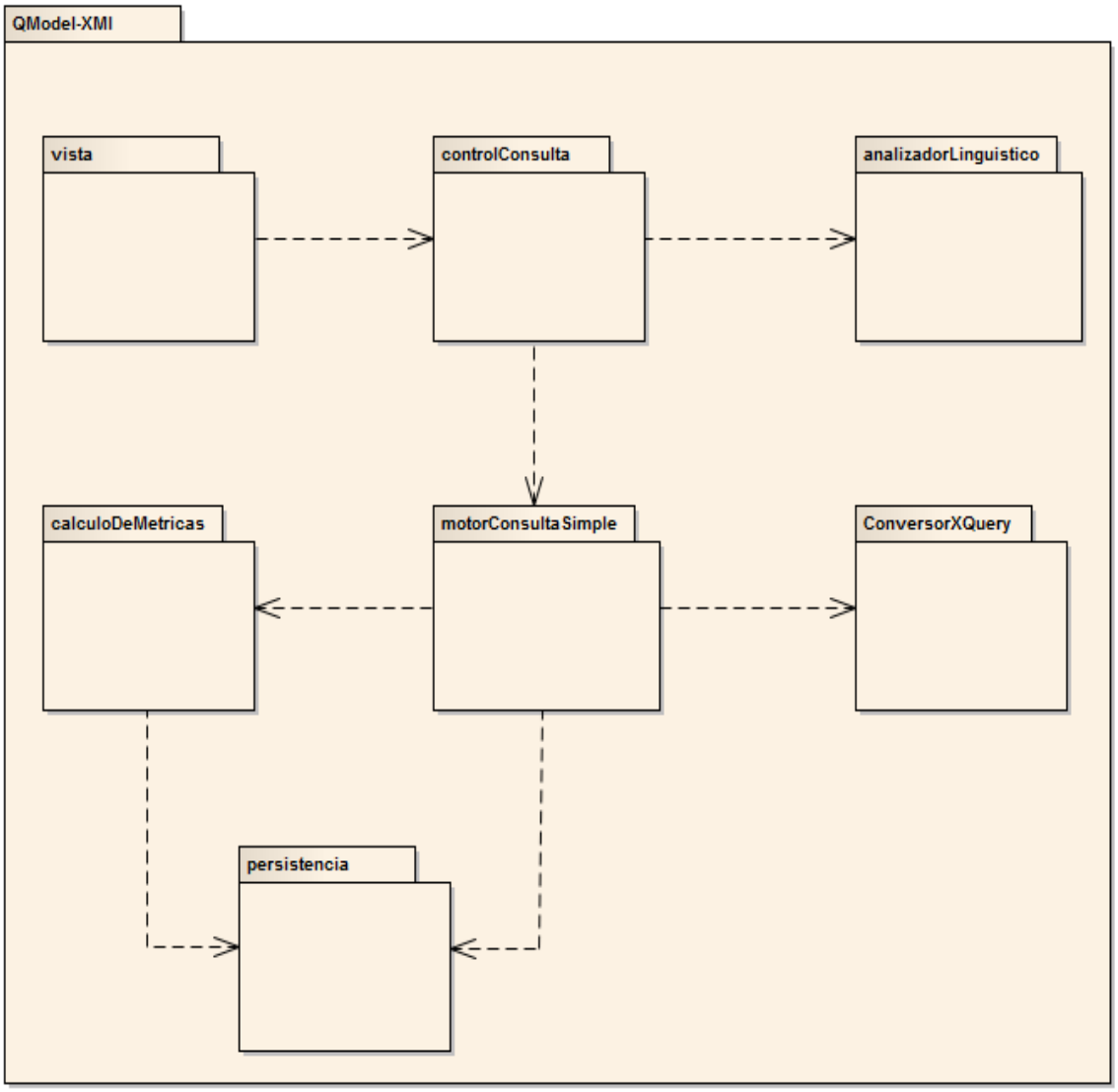

Figura 18 Vista de desarrollo de QModel-XMI

La vista física de QModel-XMI se representa por medio del diagrama de despliegue de la Figura 19. Para el correcto funcionamiento del sistema se requiere de: 1) Un computador personal con una configuración mínima de un procesador de 32 bits, memoria RAM de 4 GB y disponibilidad de almacenamiento de 1 GB. 2) La máquina virtual de Java, en una versión posterior a la 1.6. 3) La librería BaseX como motor de base de datos XML. 4) Los archivos XML que contiene los modelos UML sobre los cuales se van a hacer las consultas.

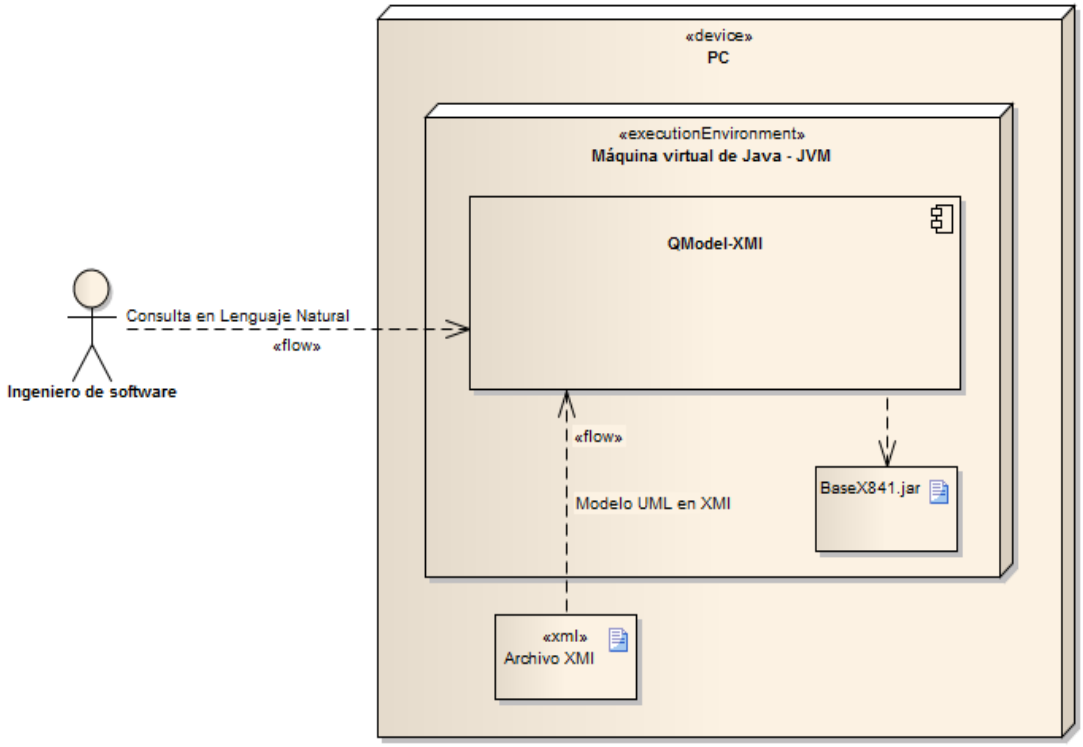

Figura 19 Vista física de QModel-XMI

6.2. Estudio de caso

Un estudio de caso es un método de investigación empírico que indaga sobre un fenómeno contemporáneo dentro de su contexto real, cuando los límites del fenómeno y el contexto no son evidentes y existen múltiples fuentes de evidencia (Ying, 2013). En comparación con otros métodos de evaluación como las encuestas, los experimentos y los cuasi-experimentos, los estudios de caso pueden 1) capturar la complejidad del caso, incluyendo cambios relevantes en el tiempo, 2) atender en forma completa las condiciones del contexto, incluyendo aquellas que potencialmente interactúan con el caso (Ying, 2013).

Además, el estudio de caso permite la investigación de realidades contextuales y establecer la diferencia entre lo que ha sido planeado y lo que realmente sucede (Anderson, 1993), por eso se utilizó como estrategia de evaluación del marco de referencia propuesto, para comparar lo establecido por el modelo conceptual definido y el mecanismo de consulta diseñado, con los resultados obtenidos al utilizarlos en escenarios reales de recuperación y análisis de arquitecturas; para lo cual se realizó un estudio de caso descriptivo, con el ánimo de identificar posibles inconsistencias entre la propuesta teórica y las evidencias recolectadas.

El estudio de caso se realizó aplicando el proceso propuesto por Ying (2013), que consiste en los siguientes pasos: el diseño del estudio de caso, su preparación, la recolección de datos, el análisis sobre los datos recolectados y la formulación de los resultados obtenidos a partir de su realización. Los tres primeros pasos se documentan en el protocolo del estudio de caso presentado en el anexo digital "Protocolo del estudio de caso.pdf", y los dos últimos en el informe de los resultados expuesto a continuación.

El informe del estudio de caso tiene como fin principal mostrar los resultados de la evaluación de ARCo, por eso se estructuró en tres partes. La primera parte presenta el diseño del estudio de caso. En la segunda parte se exponen los resultados de la aplicación del estudio de caso, para cada una de las unidades de análisis. Finalmente, en la tercera parte se hace el análisis sobre los resultados de la aplicación del estudio de caso, tomando como guía las preguntas planteadas a cada unidad de análisis, presentadas previamente.

6.2.1. Diseño del estudio de caso

Se realizó un estudio de caso único (embebido) con dos unidades de análisis porque se prueba una propuesta teórica (el marco de referencia ARCo) a partir de escenarios críticos (Ying, 2013). El primer escenario concierne a la producción de software, porque es el contexto típico para el cual han sido definidas todas las propuestas identificadas en la revisión de la literatura, convirtiéndose en una unidad de análisis obligatoria para evaluar ARCo. Por otra parte, la revisión de la literatura reveló que la ingeniería inversa se puede utilizar como herramienta didáctica en los procesos de enseñanza aprendizaje (Klimek et al., 2011; Cipresso, 2009; Ali, 2006; Ali, 2005), sin embargo, no se encontró ninguna propuesta que guíe el proceso de recuperación y análisis de vistas arquitectónicas en un contexto tan particular como éste, por eso la segunda unidad de análisis corresponde al contexto de la educación.

El diseño del estudio de caso se hizo 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. En primera instancia, las preguntas surgen a partir de la intención del estudio de caso: evaluar ARCo, por eso se elaboraron tomando como referente principal la pregunta de investigación de la tesis (¿Cómo extraer información a partir de los artefactos generados en un proceso de ingeniería inversa, para realizar análisis a nivel de arquitectura sobre el comportamiento de un sistema software?), a la que ARCo debe responder, por lo tanto se afirma que (*Proposición 1*): ARCo guía el proceso de recuperación y análisis nivel de arquitectura de un sistema software, atendiendo las características particulares de la situación que rodea el contexto en el que se realiza dicho proceso.

Para comprobar esta afirmación se estableció como *pregunta de investigación* en el estudio de caso: ¿Cómo contribuye ARCo en el proceso de recuperación y análisis a nivel de arquitectura de un sistema software?. Como este proceso depende del

contexto en el que se realiza (*Proposición 2*), la misma pregunta se formuló en forma más concreta para cada una de las unidades de análisis del estudio de caso, según se indica a continuación: ¿Cómo contribuye ARCo en el proceso de recuperación y análisis a nivel de arquitectura de un sistema software, en un contexto de producción de software? y ¿Cómo contribuye ARCo en el proceso de recuperación y análisis sobre a nivel de arquitectura de un sistema software, en un contexto de educación?.

Para la *primera unidad de análisis* (contexto producción de software) el estudio de caso se realizó en una Empresa del sector Industrial de la ciudad de Cartagena, que cuenta en su estructura organizacional, con un departamento de sistemas que cumple funciones de desarrollo de software y de gestión de la infraestructura tecnológica. Por respeto al convenio de confidencialidad firmado con dicha empresa, no se publica su identidad, por eso en este documento se referencia como Empresa del sector industrial. Para la *segunda unidad de análisis* (contexto educación) el estudio de caso se realizó 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). Se seleccionó éste modelo de análisis, porque el estudio de caso se realiza para evaluar ARCo, demostrando que los eventos empíricos observados al utilizar el Marco de Referencia, coinciden con la propuesta teórica presentada en esta tesis. Para argumentar la *interpretación de los hallazgos* se utilizó como principal estrategia el análisis basado en proposiciones teóricas para hacer inferencias lógicas, a partir de las propuestas conceptuales identificadas en la revisión de la literatura.

Para juzgar la calidad del diseño del estudio de caso se utilizaron los criterios de juicio definidos por Ying (2013): validez de la construcción, validez externa y confiabilidad; y se aplicaron las tácticas que propone el mismo autor para garantizar el cumplimiento de los criterios de juicio, como se indica a continuación. No se tuvo en cuenta el criterio de validez interna, porque éste sólo aplica para estudios de caso explicativos Y(ing, 2013), que mantienen una relación causal, a diferencia de estudios de caso exploratorios y descriptivos. Sin embargo, se resalta que durante el análisis de los datos se utilizó la técnica de análisis denominada Modelo lógico, como se mencionó primitivamente en este documento.

Para lograr la validez de la construcción, durante la recolección de datos se aplicaron los principios: 1) utilización de múltiples fuentes de evidencia (Documentación, artefactos físicos y entrevistas), haciendo posible la aplicación de triangulación de la información recolectada y permitiendo identificar posibles inconsistencias. 2) Preservación de la cadena de evidencia, permitiendo a observadores externos mantener la trazabilidad del estudio de caso, desde la formulación de las preguntas hasta el reporte de las conclusiones, pasando por el protocolo, las evidencias soportadas por las fuentes y registradas en la base de datos del estudio. Y 3)

Creación de la bases de datos del estudio de caso, soportada en archivos de texto manipulados con procesadores de palabras, repositorios de código fuente, repositorios XMI, tablas con datos tabulados manejados con hojas de cálculo, e imágenes.

Para garantizar la validez externa del estudio, se estableció que el domino para el cual los resultados del estudio pueden ser generalizados es: la recuperación y análisis de vistas arquitectónicas. Además, como principal estrategia de análisis para argumentar la interpretación de los hallazgos y hacer inferencias lógicas, se utilizó el análisis basado en proposiciones teóricas identificadas en la revisión de la literatura. Finalmente, para garantizar la confiabilidad del estudio de caso, se diseñó el protocolo del estudio de caso presentado en el anexo digital "Protocolo del estudio de caso.pdf" y se organizó la base de datos de la información recolectada, para cada unidad de análisis, documentada en los archivos "Guía de realización CPDS.pdf" y "Guía de realización CE.pdf", haciendo posible que se pueda repetir llegando a los mismos resultados.

6.2.2. Resultados del estudio de caso

Para cada unidad de análisis se presentan los resultados del estudio de caso, organizados según las actividades que conforman las fases de la metodología propuesta por ARCo, haciendo énfasis en los hitos establecidos para cada fase y las metas de cada actividad. De igual forma se registran los resultados de la percepción de los participantes en el proceso de recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento.

Para la realización del estudio de caso en sus dos unidades de análisis se utilizaron las herramientas Enterprise Architect e Imagix 4D, porque al integrar la funcionalidad que ofrece cada una de ellas con el mecanismo de consulta, permiten el logro de los objetivos planteados, además porque se cuenta con una licencia de uso de la primera y una licencia académica de la segunda.

Unidad de análisis: Contexto Proceso de desarrollo de software

La aplicación de ARCo bajo el contexto del proceso de desarrollo de software, permitió el cumplimiento del objetivo planteado: la recuperación de la documentación del software ICR en la Empresa del sector industrial de la ciudad de Cartagena. El detalle de los resultados presentados en esta sección se encuentra en formato digital en el documento "Guía de realización CPDS.pdf". En este documento sólo se incluyó la información autorizada por la empresa, respetando el convenio de confidencialidad firmado.

La documentación recuperada está representada por: 1) La descripción general del producto, presentada a continuación; 2) el modelo de dominio del problema, anexo en formato digital en el archivo "Dominio del problema.bmp"; 3) el modelo de dominio de la aplicación, anexo en formato digital en el archivo "Dominio de la aplicación.bmp"; y 4) las vistas del sistema: Vista de descomposición (anexo digital

"Vista de descomposición.bmp"), la vista de capas (anexo digital "Capas del sistema.png"), la vista de clases (Carpeta "Diagramas de clase" con 48 archivos que representan los diagramas de clase para cada paquete del sistema), la vista de componentes y conectores (anexo digital " Componentes y conectores.bmp") ; y la vista de estructuras de asignación (anexo digital " Vista física.bmp").

Descripción general del producto

- Nombre: ICR (Nombre ficticio para proteger la identidad de la empresa)
- Propósito: Apoyar tareas asociadas al control de la producción y el área logísticas no atendidas por el ERP de la compañía, en pro de la consecución de los indicadores clave de rendimiento.
- Datos técnicos:
    - Lenguaje de programación: Java
    - Motor de base de datos: MARIADB
    - Interfaces con otras aplicaciones: Servidor de bases de datos SQL Server.
    - Líneas de código fuente: 66.095
    - Líneas de comentarios 9.872
    - Archivos Java 331
    - Paquetes 173
    - Clases 1.300
    - Interfaces 3
    - Métodos 5.273
- Infraestructura hardware para el despliegue: Servidor, cliente, infraestructura de red.

En la realización del estudio de caso, durante la fase de inicio se definió el contexto del problema, identificándolo y planteando el problema, obteniendo como resultado la descripción del problema, resumida en la plantilla para la descripción del problema, presentada como anexo digital en el archivo "Guía de realización CPDS.pdf". También se definió la viabilidad del proceso, estableciendo: Las vistas objetivo, la lista de artefactos disponibles, la lista de artefactos requeridos, las técnicas a utilizar, las herramientas a utilizar y el costo del proceso (ver "Guía de realización CPDS.pdf"). Al determinar que el proceso es viable se elaboró el plan de trabajo, logrando de esa manera los hitos establecidos para la fase de inicio.

En la fase de extracción de datos se definió el dominio del problema (anexo digital "Dominio del problema.bmp") y el dominio de la aplicación (anexo digital "Dominio de la aplicación.bmp"). También se Identificaron los elementos que conforman el sistema y sus relaciones, obteniendo el modelo del sistema en bajo nivel representado en XMI, en el archivo "Elementos_ICR.xml", como resultado de la descomposición del código fuente, cumpliendo con el hito establecido para esta fase.

En la fase de organización del conocimiento, utilizando reglas de mapeo, técnicas manuales y semiautomáticas, se transformó el modelo del sistema en bajo nivel

obtenido en la fase anterior, en el modelo del sistema en alto nivel, identificando los elementos del sistema y sus relaciones; logrando el hito establecido para esta fase, representado en las vistas objetivo planteadas en la descripción del problema, y soportadas en los anexos previamente mencionados.

Finalmente, en la fase de exploración de la información se presentaron los resultados en forma gráfica usando modelos representados en UML, con la herramienta de modelado Enterprise Architect; se hizo la interpretación y análisis de los resultados obtenidos en el proceso, utilizando el mecanismo de consulta QModel-XMI, y se organizó el informe final presentado a la Empresa del sector industrial de la ciudad de Cartagena (que no es incluido por respeto al convenio de confidencialidad firmado con esta empresa), logrando de esta forma el hito establecido para esta fase.

Unidad de análisis: Contexto Educación

ARCo se utilizó como herramienta didáctica en el contexto de la Educación, contribuyendo a la comprensión del concepto de polimorfismo y en el desarrollo de habilidades para utilizarlo, por parte de los estudiantes del programa de ingeniería de sistemas de la Universidad de Cartagena, que participaron en el estudio de caso. El detalle de los resultados presentados para esta unidad de análisis se encuentra en formato digital en el documento "Guía de realización CE.pdf".

Para esta unidad de análisis los resultados se miden no sólo a partir de los artefactos recuperados y el análisis directo hecho sobre las vistas arquitectónicas, sino también en el efecto que tuvo en el aprendiz la aplicación del Marco de referencia ARCo, y la percepción del docente al utilizarlo, por eso, en esta sección se hace énfasis en los resultados obtenidos a nivel de artefactos recuperados, mientras que en la siguiente se resalta la percepción del docente y de los estudiantes que participaron en el estudio de caso.

Teniendo en cuenta las características de este contexto de uso, sólo fue necesario recuperar: 1) La descripción general del producto, presentada a continuación; 2) el modelo de dominio de la aplicación, anexo en formato digital en el archivo "Modelo de dominio JHotDraw.bmp"; y 3) las vistas del sistema: Vista de descomposición (anexo digital "Vista de descomposición.bmp"), la vista de clases (Carpeta "Diagramas de clase" con 14 archivos que representan los diagramas de clase para cada paquete del sistema), y la vista de componentes y conectores (anexo digital "Vista de componentes y conectores JHotDraw.bmp").

Descripción general del producto

- Nombre: JHotDraw versión 7.0.6
- Propósito: Es un framework Java para la creación de gráficos técnicos estructurados. Proporciona soporte a una amplia gama de programas, desde simple editores de estilo de dibujos, hasta programas más complejos que tienen reglas acerca de cómo sus elementos se pueden utilizar y modificar

(por ejemplo, una herramienta de creación de diagramas UML). Proporciona soporte para la creación y edición de formas geométricas definidas por el usuario, estableciendo limitaciones de comportamiento en el editor y las animaciones.

- Datos técnicos:
    - Lenguaje de programación: Java
    - Líneas de código fuente: 32.054
    - Líneas de comentarios 18.931
    - Archivos Java 309
    - Paquetes 122
    - Clases 1.217
    - Interfaces 37
    - Métodos 3.779
- Infraestructura hardware para el despliegue: Computador personal.

Durante la fase de inicio se definió el contexto del problema, y se describió en la plantilla presentada como anexo digital en el archivo "Guía de realización CE.pdf". También se definió la viabilidad del proceso, estableciendo: Las vistas objetivo, la lista de artefactos disponibles, la lista de artefactos requeridos, las técnicas a utilizar, las herramientas a utilizar y el costo del proceso (ver "Guía de realización CE.pdf"). De igual forma se elaboró el plan de trabajo, logrando de esa manera los hitos establecidos para la fase de inicio.

En la fase de extracción de datos se definió el dominio de la aplicación (anexo digital "Dominio de la aplicación.bmp"). Además, se identificaron los elementos que conforman el sistema y sus relaciones, obteniendo como resultado de la descomposición del código fuente, el modelo del sistema en bajo nivel representado en XMI, en el archivo "Elementos_JHotDraw.xml", logrando así el hito establecido para esta fase.

En la fase de organización del conocimiento, se reconstruyó la vista de comunicación entre procesos a través de interfaces, representada por medio del diagrama de componentes, utilizando reglas de mapeo, técnicas manuales y semiautomáticas, identificando los elementos del sistema y sus relaciones en alto nivel (Anexo digital "Diagrama de componentes.bpm"), logrando así el hito establecido para esta fase.

Por último, la fase de exploración de la información fue utilizada para que los estudiantes visualizaran los resultados obtenidos en el proceso, con la herramienta de modelado Enterprise Architect; y para que hicieran interpretación y análisis estos resultados, con la ayuda del mecanismo de consulta QModel-XMI, identificando posibles comportamientos polimórficos, que reportaron como resultado de la realización del laboratorio de clase.

Por su parte, el docente hizo lo propio para tener un referente con respecto al cual evaluó la corrección de las actividades presentadas por cada uno de los estudiantes, cuyos resultados revelaron que la totalidad de los estudiantes entendieron el

concepto de polimorfismo y el 90% demostró que está en capacidad de aplicarlo. Esto quedó registrado en el respectivo reporte sobre la actividad de aprendizaje soportada por el Marco de referencia ARCo (Anexo digital "Guía de realización CE.pdf"), logrando así el hito establecido para esta fase.

Percepción de los participantes en el estudio de caso

En el estudio de caso participaron un arquitecto de software, el Jefe del Departamento de sistemas, un analista de sistemas y un analista junior, para la unidad de análisis correspondiente al contexto del proceso de desarrollo de software; y para la unidad de análisis del contexto educación participaron un docente y 16 estudiantes. Para conocer la percepción de los estudiantes se les aplicó la encuesta diseñada en el protocolo del estudio de caso, mientras que a los demás participantes se les aplicó una entrevista, también diseñada según se indica en el anexo digital "Protocolo del estudio de caso.pdf"; obteniendo los resultados que se presentan a continuación.

El arquitecto del software y el Jefe del departamento de sistemas de la Empresa del sector industrial, han participado previamente en procesos de recuperación y análisis de arquitecturas, sin embargo, coinciden en afirmar que en las oportunidades en que lo han hecho, no han aplicado alguna técnica o método específico, y que se han guiado por su propia experiencia y conocimiento, utilizando visualizadores de código fuente. Entre tanto, el analista de sistemas, el analista junior y el docente no han participado previamente en procesos de esta naturaleza.

Todos los participantes coinciden en afirmar que se lograron los objetivos planteados para el proceso de recuperación y análisis de la arquitectura, porque se pudo reconstruir la documentación del sistema, facilitando la extensión de su funcionalidad a nuevos requisitos solicitados por la alta gerencia, para la primera unidad de análisis; mientras que para la segunda se logró la comprensión del concepto de polimorfismo y el desarrollo de habilidades para aplicarlo por parte de los estudiantes, según lo evidencia los resultados del laboratorio presentado por los estudiantes, la encuesta respondida por ellos al terminar la actividad y el informe del docente sobre la actividad académica realizada. En la encuesta, el 93.8% afirmó que después de realizado el laboratorio logró entender el concepto de polimorfismo; y el 87.5% considera que lo puede aplicar para la solución de problemas prácticos.

De igual manera, los participantes concuerdan en que ARCo contribuye en el proceso de recuperación y análisis a nivel de arquitectura de un sistema software, de la siguiente manera:

1. Sirve como esquema interpretativo que simplifica y condensa el campo del conocimiento de la ingeniería inversa, al señalar y codificar selectivamente los objetos, situaciones, acontecimientos, experiencias y las acciones que se producen en un proceso de recuperación y análisis de arquitectura.

2. La metodología propuesta por ARCo guía el proceso de recuperación y análisis de las vistas arquitectónicas estructurales y de comportamiento, haciéndolo más fácil de realizar, y logrando resultados más pertinentes.
3. El mecanismo de consulta, en su implementación a nivel de prototipo QModel-XMI, permitió el análisis sobre las vistas recuperadas, e incluso hacer inferencias sobre el comportamiento del sistema, a partir de modelos estáticos.
4. El modelo conceptual sirve como instrumento didáctico que contribuye al aprendizaje de la ingeniería inversa en general, y de la recuperación y análisis de vistas arquitectónicas recuperadas en particular.

En forma concreta, el docente afirmó que al haber abordado el proceso de recuperación y análisis de la arquitectura desde la perspectiva del contexto educativo, le permitió centrar más la atención en los objetivos de aprendizaje del proceso académico, que en el proceso técnico como tal, logrando mejores resultados en la actividad realizada, por lo cual considera que ARCo se constituye en una herramienta didáctica que hace posible el aprendizaje a partir de casos de éxito, representados por sistemas diseñados y desarrollados por expertos; posibilita el desarrollo de habilidades para modelar y programar, estimula la curiosidad del estudiante y permite evidenciar la importancia de los modelos.

Al respecto, el 66.7% de los estudiantes afirmó que lograron aprender a partir de un sistema diseñado y desarrollado por expertos, el 60% lograron entender la importancia de los modelos, el 46.7% afirmó que se estimuló su curiosidad por la forma como está diseñado el sistema JHotDraw, el 40% creen que desarrollaron habilidades para programar, el 26.7% consideran que desarrollaron habilidades para modelar, y al 75% le gustaría que esta estrategia didáctica se use con mayor frecuencia. Con relación al mecanismo de consulta, el 81.3% de los estudiantes afirmó que QModel-XMI le sirvió para identificar posibles comportamientos polimórficos a partir del modelo recuperado del sistema JHotDraw; el 75% consideró que el mecanismo le sirvió para entender el sistema JHotDraw; y el 62.5% indicó que el mecanismo le sirvió para analizar la arquitectura del sistema JHotDraw.

Por su parte, el arquitecto de software afirmó que, además de recuperar y hacer análisis sobre el comportamiento del sistema, representado por medio de diagramas de componentes que sirven para describir la comunicación entre procesos a nivel de arquitectura, se logró también la recuperación de: 1) Aspectos estructurales, representados por medio de las vistas de descomposición, las capas del sistema, la vista de clases; y 2) estructuras de asignación expresadas por medio del diagrama de despliegue.

También afirmó el arquitecto del software objeto de estudio, que el mecanismo de consulta, implementado a través del prototipo QModel-XMI, le permitió hace análisis sobre el comportamiento del sistema, facilitándole la identificación de interfaces y de posibles comportamientos polimórficos. Asimismo, manifestó que el mecanismo de

consulta, de igual forma le permitió hacer análisis sobre aspectos estructurales del sistema, manifiestos sobre todo en el tipo de relación existente entre sus elementos.

Para los participantes en el estudio de caso la utilidad de ARCo consiste en: 1) Guía el proceso de recuperación y análisis de vistas arquitectónicas, 2) permite hacer análisis a nivel de arquitectura sobre los artefactos del sistema recuperados, y 3) sirve como instrumento para el aprendizaje de la ingeniería inversa; mientras que sus limitaciones se manifiestan sobre todo al momento de detallar las tareas que se realizan, específicamente en las fases de descomposición de la información y organización del conocimiento, debido a que brinda una visión en alto nivel del proceso, por ser un marco de referencia que es apoyado por un amplio número de técnicas y herramientas que lo soportan.

De igual forma, el arquitecto de software y el docente manifestaron la dificultad que se presenta al momento de seleccionar las técnicas y herramientas que se utilizan durante el proceso, a pesar de los instrumentos que brinda ARCo. Por último, todos los participantes están de acuerdo en que utilizarían ARCo en futuros trabajos de recuperación y análisis de vistas arquitectónicas de comportamiento.

6.2.3. Análisis sobre los resultados del estudio de caso

La validez de los resultados obtenidos en el estudio de caso se consolida al hacer triangulación sobre la fuentes de evidencia, representadas en los documentos anexos, los artefactos físicos representados en los modelos recuperados del sistema, y las entrevistas hechas a los participantes. Al establecer el enlace lógico entre estos resultados, las proposiciones y las preguntas planteadas en el estudio de caso, aplicando la técnica de análisis denominada Modelo Lógico, se puede afirmar que: ARCo guía el proceso de recuperación y análisis a nivel de arquitectura de un sistema software, atendiendo las características particulares de la situación que rodea el contexto en el que se realiza dicho proceso.

En forma más concreta, los elementos conceptuales del Marco de referencia ARCo, representados por la base conceptual, la base teórica y el modelo conceptual, sirvieron como instrumento para el aprendizaje de la ingeniería inversa, por parte del grupo de personas que participaron en las dos unidades de análisis del estudio de caso. Los lineamientos de ARCo y los contextos de uso, permitieron abordar la recuperación y análisis de las vistas arquitectónicas, desde la perspectiva de la estrategia cognitiva dirigida por problemas (Kruger y Cross, 2006), lo que hizo posible asumir la situación objeto de análisis, teniendo en cuenta el contexto en que se presenta, llegando a una solución más pertinente.

Además, la integración de las técnicas de recuperación de arquitectura que dieron origen a la metodología propuesta en ARCo, hizo posible el cumplimento de todos los objetivos planteados, para cada una de las unidades de análisis del estudio de caso, lo cual se evidencia en los documentos anexos, las vistas recuperadas y la percepción de los participantes, lo que comprobó la utilidad de la metodología. Al

mismo tiempo, el mecanismo de consulta, en su implementación a nivel de prototipo QModel-XMI, sirvió para hacer análisis sobre las vistas recuperadas y para hacer inferencias sobre el comportamiento del sistema, a partir de modelos estáticos.

En este orden de ideas, con base en los resultados del estudio de caso para cada unidad de análisis, se deduce que el modelo conceptual para abstraer artefactos de bajo nivel y reconstruir información de alto nivel sobre la estructura y el comportamiento de sistemas software, y el mecanismo de consulta para realizar análisis a nivel de arquitectura, que se proponen en ARCo integrando técnicas de recuperación y análisis de arquitectura, sirven para: 1) Guiar el proceso de recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, logrando resultados más pertinentes, y 2) hacer análisis sobre los artefactos recuperados del sistema. Además, se puede utilizar como instrumento didáctico para el aprendizaje de la ingeniería inversa y de la ingeniería de software. Esto demuestra que los eventos empíricos observados al utilizar el Marco de Referencia ARCo, coincidieron con la propuesta teórica resultado de esta investigación.

Igualmente, se deduce que la aplicación de ARCo requiere del uso de técnicas y herramientas de apoyo, para la realización de tareas específicas en la recuperación y análisis de vistas arquitectónicas, que conforman las actividades de las fases de la metodología, y que no se detallan en este trabajo porque están fuera de su alcance. Sin embargo, se incluye un amplio listado de técnicas (Anexo A) y herramientas (Anexo B), con sus respectivas fuentes bibliográficas para que puedan ser estudiadas, no obstante, se confirma la necesidad de un catálogo de las técnicas de recuperación de arquitectura detallando cuándo y cómo deben ser utilizadas (Boussaidi et al., 2012; Koschke, 2009); al igual que un catálogo de herramientas que se pueda integrar al modelo ontológico propuesto en esta tesis.

Por otra parte, los resultados del estudio de caso revelan que el marco de referencia definido, no solo sirve para la recuperación y análisis de vistas arquitectónicas de comportamiento en forma específica, sino que también se puede utilizar para recuperar y analizar vistas arquitectónicas estructurales, cuando se utilizan técnicas de análisis estático para la recuperación del comportamiento del sistema (Bojic y Velasevic, 2000; Forster et al., 2013).

Capítulo 7

Conclusiones y recomendaciones

En este capítulo se presentan los juicios que plantea el autor de la tesis al terminar el trabajo de investigación, organizados de la siguiente forma: Primero se presentan las conclusiones, en respuesta a la pregunta de investigación y a los objetivos trazados, luego se indican algunas recomendaciones representadas en trabajos futuros, tendientes a resolver nuevas preguntas de investigación y finalmente se relacionan las lecciones aprendidas en el desarrollo de este trabajo.

7.1. Conclusiones

Como resultado del trabajo realizado se concluye que se dio respuesta a la pregunta de investigación planteada definiendo ARCo, un marco de referencia para la recuperación y análisis de vistas arquitectónicas estructurales y de comportamiento, evaluado por medio de la realización de un estudio de caso con dos unidades de análisis, y conformado por:

1. Un modelo conceptual para abstraer artefactos de bajo nivel y reconstruir información de alto nivel de sistemas software, fundamentado en el estándar ISO/IEC/IEEE 42010 para la descripción de arquitectura, el estándar ISO/IEC 19506 que define el meta-modelo para representar activos de productos software implementados, y el estándar del Lenguaje Unificado de Modelado UML; utilizando el lenguaje XML para garantizar la integración con otras propuestas, logrando la re-utilización de analizadores existentes (Ducasse et al., 2000). Este modelo conceptual sirve como esquema interpretativo que simplifica y condensa el campo del conocimiento de la ingeniería inversa, al señalar y codificar selectivamente los objetos, situaciones, acontecimientos, experiencias y las acciones que se producen en un proceso de recuperación y análisis de arquitectura, como quedó demostrado en los resultados del estudio de caso.
2. El diseño de un mecanismo de consulta y su implementación en el prototipo QModel-XMI, utilizado para realizar análisis a nivel de arquitectura sobre la estructura y el comportamiento de sistemas software, a partir de la base de

información obtenida en los artefactos generados por los analizadores en el proceso de ingeniería inversa; que complementa las propuesta existentes soportados en lenguajes especializados (Ujhelyi et al., 2014; Urma y Mycroft, 2012).

3. Una metodología fundamentada en las características del contexto en el que se presenta el problema objeto de estudio, que guía el proceso de recuperación y análisis de vistas arquitectónicas, definida a partir del proceso canónico establecido por Tilley et al. (1996), y la integración de técnicas identificadas en la revisión de la literatura.
4. La definición y caracterización de los contextos de uso de la ingeniería inversa, que sirve para guiar el proceso de recuperación y análisis vistas arquitectónicas, atendiendo las particularidades del contexto en el que se genera el problema objeto de estudio.
5. Un modelo ontológico para apoyar la selección de herramientas de ingeniería inversa, tomando como referente las características del contexto de uso y de las herramientas.

En forma complementario, como resultado del trabajo de investigación realizado se obtuvo: un modelo de caracterización de los mecanismos de consulta, un modelo de caracterización del proceso de recuperación y análisis de vistas arquitectónicas, una estructura de caracterización de las herramientas de ingeniería inversa y un modelo de dominio de la recuperación de arquitecturas. Además, se elaboró un inventario de propuestas existentes para la recuperación de arquitecturas, un inventario de herramientas de ingeniería inversa y un documento sobre el marco teórico que incluye la base teórica y conceptual que soportan ARCo.

En este orden de ideas, se puede afirmar que la ingeniería inversa es una disciplina en evolución, que ha ampliado sus campos de acción a otros contextos, exigiendo nuevas funcionalidades y características (Canfora et al., 2011) y el marco de referencia propuesto (ARCo) constituye un nuevo enfoque, que hace posible abordar el problema objeto de estudio en forma pertinente al contexto, porque brinda una visión holística de la ingeniería inversa y una visión detallada de la recuperación y análisis de vistas arquitectónicas, desde la perspectiva de diferentes contextos de uso como: La producción de software, la seguridad informática (Treude et al., 2011), la computación forense (Nelson et al., 2014) y la educación (Klimek et al., 2011).

Adicionalmente, los resultados obtenidos en el estudio de caso revelaron que el mecanismo de consulta, bajo su implementación en QModel-XMI, sirvió para hacer análisis sobre las vistas recuperadas y para hacer inferencias sobre el comportamiento del sistema, a partir de modelos estáticos; 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.

Por otra parte, los resultados de la investigación revelaron algunas limitaciones de la propuesta planteada y preguntas abiertas que originan trabajos futuros, que son

explicados a continuación en las recomendaciones. Además, también se identificó un conjunto de lecciones aprendidas que se relacionan al final del capítulo.

7.2. Recomendaciones

En esta sección se presentan las recomendaciones que se deben seguir para la interpretación y utilización correcta de ARCo, también se enuncian las limitaciones de este marco de referencia y finalmente se proponen algunos trabajos futuros, con base en los resultados obtenidos en la investigación.

ARCo está dirigido a profesionales del campo de las ciencias computacionales, que se desempeñan en contextos como el desarrollo de software, la seguridad informática, la computación forense y la educación, y que requieren realizar procesos de recuperación y análisis de arquitecturas. Este marco de referencia constituye un instrumento conceptual que debe ser interpretado bajo los lineamientos expuestos en el capítulo dos de este documento, en el ítem titulado "Lineamientos para la definición". También se precisa que ARCo no define nuevas técnicas, y tampoco altera el proceso canónico para la recuperación y análisis de vistas arquitectónicas, pero si ofrece un nuevo enfoque para abordarlos, que involucra el contexto de uso en el cual se presenta éste proceso, por eso ARCo debe ser utilizado aplicando los "Lineamientos para la utilización", establecidos de igual forma en el capítulo dos.

Por su parte, el mecanismo de consulta diseñado debe ser interpretado bajo los lineamientos presentados en la segunda sección del capítulo cuatro. No se establecieron lineamientos de uso para el mecanismo, porque estos dependen de la implementación que se haga del diseño propuesto. Para el uso del prototipo QModel-XMI, se recomienda leer el manual de usuario anexo en formato digital (archivo "QModel-XMI Manual de usuario.pdf").

Además, la utilización de ARCo en las dos unidades de análisis del estudio de caso realizado, revelo las siguientes limitaciones de este marco de referencia:

1. Depende de las técnicas existentes para la recuperación de vistas arquitectónicas, y aunque en este trabajo se incluye un amplio listado de técnicas (Anexo A) con sus respectivas fuentes bibliográficas para que puedan ser estudiadas, se confirma la necesidad de un catálogo de las técnicas de recuperación de arquitectura detallando cuándo y cómo deben ser utilizadas (Boussaidi et al., 2012; Koschke, 2009).
2. De igual forma, depende de las herramientas disponibles para la recuperación de arquitecturas. A pesar de que se definió un modelo ontológico para la selección de este tipo de herramientas, y se estableció un inventario (Anexo B), se requiere de un catálogo de herramientas para la recuperación de arquitecturas, que sea incorporado al modelo ontológico propuesto en esta tesis.
3. Si bien la especificación KDM ha sido adoptada por la Organización Internacional de estándares como el estándar ISO/IEC 19506 desde el año

2012, es muy reducido el número de herramientas que transforman los activos software en los paquetes que conforman la especificación, sin que ninguna logre hacerlo hasta las capas de "tiempo de ejecución" y "abstracción", para obtener una representación completa del producto, limitando los análisis que se puedan hacer sobre el producto software.
4. La implementación del Mecanismo de consulta (QModel-XMI) sólo permite formular preguntas clasificadas en este trabajo como 'simple' (Tabla 9) y el cálculo de las métricas establecidas en los requisitos del prototipo.

Por todo lo anterior, se propone continuar la investigación con los siguientes trabajos futuros:

1. Complementar las técnicas para identificar las abstracciones que representan los elementos de la arquitectura y los aspectos determinantes para cada vista arquitectónica, a partir de las representaciones KDM (Boussaidi et al., 2012) y UML.
2. Un estudio detallado de las técnicas de recuperación de arquitectura que establezca cuándo y cómo deben ser utilizadas (Boussaidi et al., 2012; Koschke, 2009).
3. Un estudio detallado de las herramientas de recuperación de arquitectura para integrar sus resultados en el modelo ontológico propuesto en esta tesis.
4. Extender el mecanismo de consulta propuesto e implementado en QModel-XMI, para ampliar su capacidad incorporando respuestas a preguntas clasificadas en este trabajo como 'compuestas' (Tabla 10).
5. Complementar las reglas de mapeo para automatizar la conversión de representaciones KDM a modelos UML, para contribuir a la estandarización del proceso de modernización de productos software (Ulrich, 2010).

7.3. Lecciones aprendidas

En forma complementaria a los resultados obtenidos en el proceso de investigación, se relacionan a continuación las lecciones aprendidas, que si bien pueden ser obvias para los expertos en el tema, se plantean para su discusión y se espera sean de utilidad para quienes están incursionando en este campo del conocimiento.

1. El contexto en el que se presenta la necesidad de recuperar y analizar vistas arquitectónicas de productos software, determina el enfoque como debe abordarse el proceso para satisfacer esta necesidad. En esta investigación se caracterizaron cuatro contextos de uso de la ingeniería inversa: La producción de software, la seguridad informática, la computación forense y la educación. Cada contexto cuenta con un ámbito, recursos, situaciones e interesados que definen los propósitos que orientan la intención al momento de recuperar y analizar las vistas arquitectónicas, por eso, el proceso que se realice para lograr dicho objetivo debe ser pertinente a las características del contexto.

2. Las situaciones que se presentan en cada contexto, determinan los recursos disponibles para el proceso de recuperación y análisis de vistas arquitectónicas. Es posible que en un mismo contexto existan múltiples situaciones, y en cada una existan distintos recursos disponibles, por eso en la fase inicial del proceso se deben identificar estos recursos, para determinar la viabilidad del proceso.
3. Las vistas arquitectónicas que se desean recuperar determinan los elementos del sistema que se deben obtener como resultado de la fase de extracción de datos. Esto se debe a que, cada vista arquitectónica describe estructuras modulares, componentes y conector o estructuras de asignación, cada una de las cuales está conformada por un conjunto de elementos que constituyen los artefactos software.
4. El uso de estándares contribuye a la re-utilización, haciendo posible la integración de esfuerzos encaminados a recuperar y analizar vistas arquitectónicas, disminuyendo riesgos, costos y logrando mayor eficiencia. La revisión de la literatura permitió identificar casi un centenar de propuestas para la recuperación de arquitecturas, y más de un centenar de herramientas, sin embargo, se observó un bajo el grado de reutilización e integración.
5. El proceso de recuperación y análisis de vistas arquitectónicas debe ser ligero y flexible. Si bien esta actividad implica un alto nivel de complejidad, no siempre es realizada por expertos en este campo del conocimiento, por eso ARCo brinda elementos conceptuales e instrumentales, que además de guiar el proceso atendiendo las características del contexto, también sirve como instrumento didáctico para comprenderlo, y brinda un mecanismo de consulta con un prototipo que responde preguntas formuladas en lenguaje natural.

Bibliografía

Abi-Antoun, M., & Aldrich, J. (2008). A field study in static extraction of runtime architectures. Paper presented at the ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, 22-28.

Abi-Antoun, M., & Aldrich, J. (2008). Static conformance checking of runtime architectural structure. In Carnegie Mellon University Technical Report, CMU-ISR-08-132.

Abi-Antoun, M., & Aldrich, J. (2008). Tool support for statically checking the structural conformance of an object-oriented system to its runtime architecture. Paper presented at the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 741-742.

Abi-Antoun, M., & Aldrich, J. (2008). Tool support for the static extraction of sound hierarchical representations of runtime object graphs. Paper presented at the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 743-744.

Abi-Antoun, M., & Aldrich, J. (2009). Static extraction and conformance analysis of hierarchical runtime architectural structure using annotations. Paper presented at the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 321-340.

Abi-Antoun, M., & Aldrich, J. (2009, January). Static extraction of sound hierarchical runtime object graphs. In Proceedings of the 4th international workshop on Types in language design and implementation (pp. 51-64). ACM.

Abi-Antoun, M., & Barnes, J. M. (2010). Analyzing security architectures. Paper presented at the ASE'10 - Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, 3-12.

Abowd, G., Bass, L., Clements, P., Kazman, R., & Northrop, L. (1997). Recommended Best Industrial Practice for Software Architecture Evaluation (No. CMU/SEI-96-TR-025). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

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

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

Ali, M. R. (2005). Why teach reverse engineering?. ACM SIGSOFT Software Engineering Notes, 30(4), 1-4.

Ali, M. R. (2006). Imparting effective software engineering education. ACM SIGSOFT Software Engineering Notes, 31(4), 1-3.

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

Anderson, G., 1993. Fundamentals of Educational Research. Falmer Press, London, pp: 152-160.

Andritsos, P., & Tzerpos, V. (2005). Information-theoretic software clustering. IEEE Transactions on Software Engineering, 31(2), 150-165.

Anquetil, N., Royer, J. -., André, P., Ardourel, G., Hnětynka, P., Poch, T., . . . Petraşcu, V. (2009). JavaCompExt: Extracting architectural elements from java source code. Paper presented at the Proceedings - Working Conference on Reverse Engineering, WCRE, 317-318.

Antlersoft (2016). http://browsebyquery.sourceforge.net/

Arcelli, F., Tosi, C., Zanoni, M., & Maggioni, S. (2008, July). The MARPLE project: A tool for design pattern detection and software architecture reconstruction. In 1st International Workshop on Academic Software Development Tools and Techniques (WASDeTT-1) (pp. 325-334).

Arciniegas, J. L., Dueñas, J. C., Ruiz, J. L., Cerón, R., Bermejo, J., & Oltra, M. A. (2006). Architecture reasoning for supporting product line evolution: An example on security

Balsamo, S., & Marzolla, M. (2005, July). Performance evaluation of UML software architectures with multiclass Queueing Network models. InProceedings of the 5th international workshop on Software and performance(pp. 37-42). ACM.

Bass, L.; Clements, P. & Kazman, R. (2013), Software Architecture in Practice. Third edition. Addison-Wesley

Bellay, B., & Gall, H. (1997, October). A comparison of four reverse engineering tools. In Reverse Engineering, 1997. Proceedings of the Fourth Working Conference on (pp. 2-11). IEEE.

Bennett, K. H., & Rajlich, V. T. (2000, May). Software maintenance and evolution: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (pp. 73-87). ACM.

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

Bergmann, G., Horváth, Á., Ráth, I., Varró, D., Balogh, A., Balogh, Z., & Ökrös, A. (2010). Incremental evaluation of model queries over EMF models. In Model Driven Engineering Languages and Systems (pp. 76-90). Springer Berlin Heidelberg.

Beyer, D. (2005). Co-Change Visualization. In ICSM (Industrial and Tool Volume) (pp. 89-92).

Beyer, D., & Lewerentz, C. (2003, May). CrocoPat: Efficient Pattern Analysis in Object-Oriented Programs. In IWPC (Vol. 3, p. 294).

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

Bibi, M., & Maqbool, O. (2011). Version information support for software architecture recovery. Paper presented at the 2011 7th International Conference on Emerging Technologies, ICET 2011

Bojic, D., & Velasevic, D. (2000, February). A use-case driven method of architecture recovery for program understanding and reuse reengineering. In csmr (p. 23). IEEE.

Bourque, P., & Fairley, R. E. (2014). Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.

Boussaidi, G. E., Belle, A. B., Vaucher, S., & Mili, H. (2012, October). Reconstructing architectural views from legacy systems. In Reverse Engineering (WCRE), 2012 19th Working Conference on (pp. 345-354). IEEE.

Bowman, I. T., Holt, R. C., & Brewster, N. V. (1999, May). Linux as a case study: Its extracted software architecture. In Proceedings of the 21st international conference on Software engineering (pp. 555-563). ACM.

Cai, Y., Wang, H., Wong, S., & Wang, L. (2013, June). Leveraging design rules to improve software architecture recovery. In Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures(pp. 133-142). ACM.

Callo Arias, T. B., Avgeriou, P., America, P., Blom, K., & Bachynskyy, S. (2011). 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.

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

Chardigny, S., Seriai, A., Tamzalit, D., & Oussalah, M. (2008, April). Quality-driven extraction of a component-based architecture from an object-oriented system. In Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on (pp. 269-273). IEEE.

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

Chikovsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. Software, IEEE, 7(1), 13-17.

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

Choo, K. K. R. (2008). Organised crime groups in cyberspace: a typology. Trends in organized crime, 11(3), 270-295.

Cipresso, T. (2009). Software reverse engineering education.

Clarkware Consulting (2016). http://clarkware.com/software/JDepend.html

Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., & Little, R. (2002). Documenting software architectures: views and beyond. Pearson Education.

Constantinou, E., Kakarontzas, G., & Stamelos, I. (2011). Towards open source software system architecture recovery using design metrics. Paper presented at the Proceedings - 2011 Panhellenic Conference on Informatics, PCI 2011, 166-170.

Cook, J. E., & Wolf, A. L. (1998). Event-based detection of concurrency. Paper presented at the Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 35-45.

Corazza, A., Di Martino, S., & Scanniello, G. (2010). A probabilistic based approach towards software system clustering. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 88-96.

Corazza, A., Di Martino, S., & Scanniello, G. (2010, March). A probabilistic based approach towards software system clustering. In Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on (pp. 88-96). IEEE.

De Brito, H., Rocha, H., Terra, R., & Valente, M. T. (2010). On-the-fly and non-invasive extraction of runtime architectures using hierarchical object graphs. Paper presented at the Proceedings - 4th Brazilian Symposium on Software Components, Architectures and Reuse, SBCARS 2010, 140-149.

De Moor, O., Verbaere, M., Hajiyev, E., Avgustinov, P., Ekman, T., Ongkingco, N., ... & Tibble, J. (2007, October). Keynote address:. ql for source code analysis. In null (pp. 3-16). IEEE.

De Roover, C., Noguera, C., Kellens, A., & Jonckers, V. (2011, August). 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 (pp. 71-80). ACM.

De Volder, K. (1998, July). Aspect-oriented logic meta programming. In European Conference on Object-Oriented Programming (pp. 414-417). Springer Berlin Heidelberg.

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

Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7.

Ducasse, S., & Pollet, D. (2009). Software architecture reconstruction: A process-oriented taxonomy. IEEE Transactions on Software Engineering, 35(4), 573-591.

Ducasse, S., Lanza, M., & Tichelaar, S. (2000, June). Moose: an extensible language-independent environment for reengineering object-oriented systems. In Proceedings of the Second International Symposium on Constructing Software Engineering Tools (CoSET 2000) (Vol. 4).

Eichberg, M., Haupt, M., Mezini, M., & Schäfer, T. (2005, September). Comprehensive software understanding with SEXTANT. In Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on (pp. 315-324). IEEE.

Eisenbarth, T., Koschke, R., & Simon, D. (2003). Locating features in source code. IEEE Transactions on Software Engineering, 29(3), 210-224.

Eixelsberger, W., Warholm, L., Klosch, R., & Gall, H. (1997). Software architecture recovery of embedded software. Paper presented at the Proceedings - International Conference on Software Engineering, 558-559.

El Boussaidi, G., Belle, A. B., Vaucher, S., & Mili, H. (2012). Reconstructing architectural views from legacy systems. Paper presented at the Proceedings - Working Conference on Reverse Engineering, WCRE, 345-354.

Favre L., Martinez L and C. Pereira (2009). MDA-Based Reverse Engineering of Object Oriented Code. Enterprise, Business-Process and Information Systems Modeling. vol. 29, Springer, , pp. 251–263.

Favre, J. M. (2004, November). Cacophony: Metamodel-driven software architecture reconstruction. In Reverse Engineering, 2004. Proceedings. 11th Working Conference on (pp. 204-213). IEEE.

Fedesoft (2011). Sector de tecnologías quiere superar el 2011. Disponible en http://www.fedesoft.org/novedades/sector-tecnologias-quiere-superar-el-2011 recuperado 24 de noviembre de 2012.

Fiutem, R., Tonella, P., Antoniol, G., & Merlo, E. (1996, November). A cliche-based environment to support architectural reverse engineering. In Software Maintenance 1996, Proceedings., International Conference on (pp. 319-328). IEEE.

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

Forster, T., Keuler, T., Knodel, J., & Becker, M. -. (2013). Recovering component dependencies hidden by frameworks - experiences from analyzing OSGi and qt. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 295-304.

Gall, H., Jazayeri, M., Klösch, R., Lugmayr, W., & Trausmuth, G. (1996, October). Architecture recovery in ARES. In Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints' 96) on SIGSOFT'96 workshops (pp. 111-115). ACM.

Ganesan, D., Lindvall, M., Cleaveland, R., Jetley, R., Jones, P., & Zhang, Y. (2011, June). Architecture reconstruction and analysis of medical device software. In Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on (pp. 194-203). IEEE.

Garcia, J., Krka, I., Medvidovic, N., & Douglas, C. (2012). A framework for obtaining the ground-truth in architectural recovery. Paper presented at the Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012, 292-296.

Garcia, J., Popescu, D., Mattmann, C., Medvidovic, N., & Cai, Y. (2011, November). Enhancing architectural recovery using concerns. In Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (pp. 552-555). IEEE Computer Society.

Goffman, E. (1986). Frame analysis: An essay on the organization of experience. Harvard University Press.

Guéhéneuc Y., Mens K. y R. Wuyts, A Comparative Framework for Design Recovery Tools, Proceedings of the 10th European Conference on Software Maintenance and Reengineering, 134 – 143, Bari, Italia, 22 al 24 de marzo (2006).

Guo, G. Y., Atlee, J. M., & Kazman, R. (1999). A software architecture reconstruction method (pp. 15-33). Springer US.

Guo, J., Liao, Y., & Pamula, R. (2006). Static analysis based software architecture recovery

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

Haqqie, S., & Shahid, A. A. (2005). Mining design patterns for architecture reconstruction using an expert system. Paper presented at the 2005 Pakistan Section Multitopic Conference, INMIC

Harris, D.R.; Reubenstein, H.B. y A.S. Yeh, Reverse Engineering to the Architectural Level, Proc. Int'l Conf. Software Eng. (ICSE), 1995.

Hassan, A. E., & Holt, R. C. (2002). Architecture recovery of web applications. Paper presented at the Proceedings - International Conference on Software Engineering, 349-359.

Heafield, K. (2011, July). KenLM: Faster and smaller language model queries. In Proceedings of the Sixth Workshop on Statistical Machine Translation (pp. 187-197). Association for Computational Linguistics.

Holt, R. C. (1995). Introduction to the Grok programming language. University of Waterloo.

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

Holy, L., Snajberk, J., Brada, P., & Jezek, K. (2013, September). A Visualization Tool for Reverse-Engineering of Complex Component Applications. In 2013 IEEE International Conference on Software Maintenance (pp. 500-503). IEEE.

Huang, G., Mei, H., & Yang, F. -. (2006). Runtime recovery and manipulation of software architecture of component-based systems. Automated Software Engineering, 13(2), 257-281.

Imagix Corpotarion (2016). https://www.imagix.com/

Ivkovic, I., & Godfrey, M. W. (2002). Architecture recovery of dynamically linked applications: A case study. In Program Comprehension, 2002. Proceedings. 10th International Workshop on (pp. 178-184). IEEE.

Jansen, A., Bosch, J., & Avgeriou, P. (2008). Documenting after the fact: Recovering architectural design decisions. Journal of Systems and Software, 81(4), 536-557.

Jouault, F., Bézivin, J., & Barbero, M. (2009). Towards an advanced model-driven engineering toolbox. Innovations in Systems and Software Engineering, 5(1), 5-12.

Kang, S., Lee, S., & Lee, D. (2009). A framework for tool-based software architecture reconstruction. International Journal of Software Engineering and Knowledge Engineering, 19(2), 283-305.

Kazman, R., O'Brien, L., & Verhoef, C. (2002). Architecture reconstruction guidelines (No. CMU/SEI-2002-TR-034). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Kim, T. H., Kim, K., & Kim, W. (2010, July). An interactive change impact analysis based on an architectural reflexion model approach. In Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual(pp. 297-302). IEEE.

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).

Klimek, I., Keltika, M., & Jakab, F. (2011, October). Reverse engineering as an education tool in computer science. In Emerging eLearning Technologies and Applications (ICETA), 2011 9th International Conference on (pp. 123-126). IEEE.

Kobayashi, K., Kamimura, M., Yano, K., Kato, K., & Matsuo, A. (2013, May). SArF map: Visualizing software architecture from feature and layer viewpoints. In 2013 21st International Conference on Program Comprehension (ICPC) (pp. 43-52). IEEE.

Koschke, R. (2005). Reconstruction of software architectures: A literature and methods overview on the state of the science. [Rekonstruktion von Software-Architekturen: Ein Literatur- und Methoden-Überblick zum Stand der Wissenschaft] Informatik - Forschung Und Entwicklung, 19(3), 127-140.

Koschke, R. (2009). Architecture reconstruction: Tutorial on reverse engineering to the architectural level

Koschke, R. (2010). Incremental reflexion analysis. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 1-10.

Koschke, R., & Simon, D. (2003, November). Hierarchical Reflexion Models. In WCRE (Vol. 3, pp. 186-208).

Krikhaar, R. L., de Jong, R. P., Medema, J. P., & Feija, L. M. G. (1999). Architecture comprehension tools for a pbx system. In Software Maintenance and Reengineering, 1999. Proceedings of the Third European Conference on (pp. 31-39). IEEE.

Kullbach, B., & Winter, A. (1999). Querying as an enabling technology in software reengineering. In Software Maintenance and Reengineering, 1999. Proceedings of the Third European Conference on (pp. 42-50). IEEE.

Lange, C., Sneed, H. M., & Winter, A. (2001). Applying the graph-oriented GUPRO Approach in comparison to a Relational Database based Approach. In 9th International Workshop on Program Comprehension.

Lanza, M. (2003, March). Codecrawler-lessons learned in building a software visualization tool. In Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on (pp. 409-418). IEEE.

Lee, S., & Kang, S. (2011). Verifying a Software Architecture Reconstruction Framework with a Case Study.

Lehman, M. M., Ramil, J. F., Wernick, P. D., Perry, D. E., & Turski, W. M. (1997, November). Metrics and laws of software evolution-the nineties view. InSoftware Metrics Symposium, 1997. Proceedings., Fourth International (pp. 20-32). IEEE.

Li, H., Chen, F., Yang, H., Feng, X., & Guo, H. (2012). Software architecture recovery based on weighted hierarchical clustering. International Review on Computers and Software, 7(7), 3654-3659.

Li, Q., Chu, H., Hu, S., Chen, P., & Zhao, Y. (2005). Architecture recovery and abstraction from the perspective of processes. Paper presented at the Proceedings - Working Conference on Reverse Engineering, WCRE, , 2005 57-66.

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

Ligh, M., Adair, S., Hartstein, B., & Richard, M. (2010). Malware analyst's cookbook and DVD: tools and techniques for fighting malicious code. Wiley Publishing.

Linton, M. A. (1984, April). Implementing relational views of programs. In ACM SIGSOFT Software Engineering Notes (Vol. 9, No. 3, pp. 132-140). ACM.

Löwe, W., Ericsson, M., Lundberg, J., Panas, T., & Pettersson, N. (2003). Vizzanalyzer-a software comprehension framework. Software Engineering Research and Practice-SERPS, 3.

Lundberg, J., & Löwe, W. (2003). Architecture recovery by semi-automatic component identification. Electronic Notes in Theoretical Computer Science,82(5), 98-114.

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

Mancoridis, S., Mitchell, B. S., Chen, Y., & Gansner, E. R. (1999). Bunch: A clustering tool for the recovery and maintenance of software system structures. In Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on (pp. 50-59). IEEE.

Maqbool, O., & Babri, H. A. (2004). The weighted combined algorithm: A linkage algorithm for software clustering. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, , 8 15-24.

Martin, R. C. (1996). The open-closed principle. More C++ gems, 97-112.

Maweed, Y., Bouneffa, M., & Basson, H. Using Graph Rewriting Systems for Automating Software Evolution Activities. In Information and Communication Technologies, 2006. ICTTA'06. 2nd (Vol. 2, pp. 2831-2836). IEEE.

McGregor, S. L., y Murnane, J. A. (2010). Paradigm, methodology and method: Intellectual integrity in consumer scholarship. International Journal of Consumer Studies, 34(4), 419-427.

Medvidovic, N. y V. Jakobac, Using Software Evolution to Focus Architectural Recovery, Automated Software Eng., vol. 13, no. 2, pp. 225-256, 2006, doi:10.1007/s10515-006-7737-5.

Mendonça, N. C., & Kramer, J. (2001). An approach for recovering distributed system architectures. Automated Software Engineering, 8(3-4), 311-354.

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

Monroy, M. E., Arciniegas, J. L., & Rodríguez, J. C. (2012). Caracterización de Herramientas de Ingeniería Inversa. Información tecnológica, 23(6), 31-42.

Monroy, M. E., Arciniegas, J. L., & Rodríguez, J. C. (2013). Propuesta Metodológica para Caracterizar y Seleccionar Métodos de Ingeniería Inversa. Información tecnológica, 24(5), 23-30.

Monroy, M., Arciniegas, J.L., & Rodriguez, J. (2016), Ontological model for Contexts of use of reverse engineering tools, Inf. Tecnol. 27(4), in press.

Monroy, M., Arciniegas, J.L., & Rodriguez, J. (2016), Software Architecture Recovery: A systematic mapping study, Inf. Tecnol. 27(5), in press.

Moreira, M. A., Greca, I. M., y Palmero, M. L. R. (2002). Mental models and conceptual models in the teaching & learning of science. Revista Brasileira de Investigação em Educação em Ciências, 2(3), 84-96.

Mueller R. S., RSA Cyber Security Conference San Francisco, California, FBI. Marzo de 2010. Recuperado febrero de 2016, disponible en https://www.fbi.gov/news/speeches/tackling-the-cyber-threat

Müller, H. A., Jahnke, J. H., Smith, D. B., Storey, M. A., Tilley, S. R., & Wong, K. (2000, May). Reverse engineering: A roadmap. In Proceedings of the Conference on the Future of Software Engineering (pp. 47-60). ACM.

Müller, H. A., Orgun, M. A., Tilley, S. R., & Uhl, J. S. (1993). A reverse-engineering approach to subsystem structure identification. Journal of Software Maintenance: Research and Practice, 5(4), 181-204.

Murphy, G. C., & Notkin, D. (1997). Reengineering with reflexion models: A case study. Computer, 30(8), 29-36.

Murphy, G. C.; Notkin, D. y K. Sullivan. Software refexion models: Bridging the gap between source and high-level models. In Proc. of the Third ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 18-28, New York, NY, 1995. ACM Press.

Nelson, B., Phillips, A., & Steuart, C. (2014). Guide to computer forensics and investigations. Cengage Learning.

Neo4j (2016). https://neo4j.com/company/

Nierstrasz, O., Ducasse, S., & Demeyer, S. (2005, September). Object-oriented reengineering patterns—an overview. In International Conference on Generative Programming and Component Engineering (pp. 1-9). Springer Berlin Heidelberg.

Noblett, M.G y M. Pollitt (2000), Recovering and Examining Computer Forensic Evidence, Forensic Science Communication, 2(4).

Noguera, C., De Roover, C., Kellens, A., & Jonckers, V. (2012, June). Code querying by UML. In Program Comprehension (ICPC), 2012 IEEE 20th International Conference on (pp. 229-238). IEEE.

OMG (2011). Architecture-Driven Modernization: Knowledge Discovery Meta-Model (KDM). Version 1.3.

OMG, Diagram Definition, Versión 1.1., 2015. Recuperado 13 de enero 2016, disponible en http://www.omg.org/spec/DD/1.1/

OMG, OMG Meta Object Facility (MOF) Core Specification, versión 2.5, 2015, recuperado 13 de enero 2016, disponible: http://www.omg.org/spec/MOF/2.5/

OMG, Unified Modeling Language. Versión 2.5. 2015. Recuperado 13 de enero 2016, disponible en: http://www.omg.org/spec/UML/2.5/

Opoka, J. C., Felderer, M., Lenz, C., & Lange, C. (2008, April). Querying UML models using OCL and Prolog: A performance study. In Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference on (pp. 81-88). IEEE.

Pacione, M. J. (2005). A novel software visualisation model to support object-oriented program comprehension. University of Strathclyde.

Panas, T., Löwe, W., & Aßmann, U. (2003, June). Towards the Unified Recovery Architecture for Reverse Engineering. In Software Engineering Research and Practice (pp. 854-860).

Pascoe, J. (1998, October). Adding generic contextual capabilities to wearable computers. In Wearable Computers, 1998. Digest of Papers. Second International Symposium on (pp. 92-99). IEEE.

Pashov, I., y Riebisch, M. (2004, May). Using feature modeling for program comprehension and software architecture recovery. In Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the (pp. 406-417). IEEE.

Patel, C., Hamou-Lhadj, A., & Rilling, J. (2009). Software clustering using dynamic analysis and static dependencies. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 273-274.

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

Pavankumar, K., Kolla, H., & R. Manjula. (2011). Software Architecture Recovery through Graph Mining Technique and Reduction of Complexity Involved in A * Algorithm through Stable Marriage Problem. ARPN Journal of Systems and Software, 1(2), 41-46.

Peake, I., Blech, J. O., & Fernando, L. (2013). Towards Reconstructing Architectural Models of Software Tools by Runtime Analysis. In EESSMOD@ MoDELS (pp. 43-48).

Petersen, K.; Feldt R.; Mujtaba S. y M. Mattsson, Systematic Mapping Studies in Software Engineering, 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 1-10, Bari, Italy junio (2008).

Pfleeger S. L, (2010). Software Engineering: Theory and Practice. Prentice Hall. 4a Edition, 792 pp.

Pinzger, M., “ArchView—Analyzing Evolutionary Aspects of Complex Software Systems,” PhD thesis, Vienna Univ. Of Technology, 2005.

Pinzger, M., Gall, H., Girard, J. -., Knodel, J., Riva, C., Pasman, W., . . . Wijnstra, J. G. (2004). Architecture recovery for product families

Platenius, M. C., Von Detten, M., & Becker, S. (2012). Archimetrix: Improved software architecture recovery in the presence of design deficiencies. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 255-264.

Pollet, D., Ducasse, S., Poyet, L., Alloui, I., Cîmpan, S., & Verjus, H. (2007). Towards a process-oriented software architecture reconstruction taxonomy. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 137-148.

Ran, A., & Lencevicius, R. (2003). Making sense of runtime architecture for mobile phone software. Paper presented at the Proceedings of the Joint European Software Engineering Conference (ESEC) and SIGSOFT Symposium on the Foundations of Software Engineering (FSE-11), 367-370.

Rasool, G., and Asif N. Software Architecture Recovery, International Journal of Computer, Information and Systems, Science and Engineering Summer 2007, pg 99-104.

Ráth, I., Hegedüs, Á., & Varró, D. (2012). Derived features for EMF by integrating advanced model queries. In Modelling Foundations and Applications (pp. 102-117). Springer Berlin Heidelberg.

Richner, T., & Ducasse, S. (1999). Recovering high-level views of object-oriented applications from static and dynamic information. Paper presented at the Conference on Software Maintenance, 13-22.

Risi, M., Scanniello, G., & Tortora, G. (2010, September). Architecture recovery using latent semantic indexing and k-means: an empirical evaluation. In 2010 8th IEEE International Conference on Software Engineering and Formal Methods (pp. 103-112). IEEE.

Risi, M., Scanniello, G., & Tortora, G. (2012). Using fold-in and fold-out in the architecture recovery of software systems. Formal Aspects of Computing, 24(3), 307-330.

Riva, C. (2002). Architecture reconstruction in practice. In Software Architecture (pp. 159-173). Springer US.

Riva, C., & Rodriguez, J. V. (2002). Combining static and dynamic views for architecture reconstruction. Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 47-55.

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

Sartipi, K. (2001, May). Alborz: A Query-based Tool for Software Architecture Recovery. In IWPC (pp. 115-118).

Scanniello, G., D'Amico, A., D'Amico, C., & D'Amico, T. (2010). An approach for architectural layer recovery. Paper presented at the Proceedings of the ACM Symposium on Applied Computing, 2198-2202.

Scanniello, G., D'Amico, A., D'Amico, C., & D'Amico, T. (2010). Architectural layer recovery for software system understanding and evolution. Software - Practice and Experience, 40(10), 897-916.

Scanniello, G., D'Amico, A., D'Amico, C., & D'Amico, T. (2010). Using the kleinberg algorithm and vector space model for software system clustering. Paper presented at the IEEE International Conference on Program Comprehension, 180-189.

Scanniello, G., Risi, M., & Tortora, G. (2010). Architecture recovery using latent semantic indexing and k-means: An empirical evaluation. Paper presented at the Proceedings - Software Engineering and Formal Methods, SEFM 2010, 103-112.

Schilit, B., Adams, N., & Want, R. (1994, December). Context-aware computing applications. In Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on (pp. 85-90). IEEE.

Schmerl, B., Aldrich, J., Garlan, D., Kazman, R., & Yan, H. (2006). Discovering architectures from running systems. Software Engineering, IEEE Transactions on, 32(7), 454-466.

Sharif, M., Lanzi, A., Giffin, J., & Lee, W. (2009, May). Automatic reverse engineering of malware emulators. In Security and Privacy, 2009 30th IEEE Symposium on (pp. 94-109). IEEE.

SiliconWeek (2012). La Industria del software en Europa no entiende la crisis. Disponible en http://www.siliconweek.es/noticias/ranking-empresas-software-europa-truffle-100-29192. recuperado 6 de noviembre de 2012

Solms, F. (2013). Experiences with using the systematic method for architecture recovery (SyMAR). Paper presented at the ACM International Conference Proceeding Series, 170-178.

Sözer, H., Tekinerdoğan, B., & Akşit, M. (2009). FLORA: A framework for decomposing software architecture to introduce local recovery. Software: Practice and Experience, 39(10), 869-889.

Stanford University, Protégé, http://protege.stanford.edu/products.php#desktop-protege, acceso: 13 de enero (2015).

Stein, D., Hanenberg, S., & Unland, R. (2005). A graphical notation to specify model queries for MDA transformations on UML models. In Model Driven Architecture (pp. 77-92). Springer Berlin Heidelberg.

Stoermer, C., & O'Brien, L. (2001). MAP-mining architectures for product line evaluations. In Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on (pp. 35-44). IEEE.

Stoermer, C., Bachmann, F., & Verhoef, C. (2003). SACAM: The software architecture comparison analysis method (No. CMU/SEI-2003-TR-006). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Stoermer, C., O'Brien, L., y Verhoef, C. (2003). Moving towards quality attribute driven software architecture reconstruction. In null (p. 46). IEEE.

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

Storey, M. A. D., y Müller, H. A. (1995, October). Manipulating and documenting software structures using SHriMP views. In Software Maintenance, 1995. Proceedings., International Conference on (pp. 275-284). IEEE.

Störmer, C. (2007). Software quality attribute analysis by architecture reconstruction (SQUA 3RE). Paper presented at the Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 361-364.

Störrle, H. (2007). A PROLOG-based Approach to Representing and Querying Software Engineering Models. VLL, 274, 71-83.

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

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

Svetinovic, D., & Godfrey, M. (2001). A lightweight architecture recovery process. Software Architecture Recovery and Modelling, Stuttgart, Germany.

Tilley, S. (1998). A reverse-engineering environment framework (No. CMU/SEI-98-TR-005). Carnegie-Mellon Univ Pittsburgh pa software engineering inst.

Tilley, S.R., Smith, D.B. y S. Paul, Towards a Framework for Program Understanding, Proc. Int'l Workshop Program Comprehension, p. 19, 1996, doi:10.1109/WPC.1996.501117.

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

Tonella, P., Torchiano, M., Du Bois, B., & Systä, T. (2007). Empirical studies in reverse engineering: state of the art and future trends. Empirical Software Engineering, 12(5), 551-571.

Traverso, M., & Mancoridis, S. (2002). On the automatic recovery of style-specific architectural relations in software systems. Automated Software Engineering, 9(4), 331-360.

Treude C., Figueira F. and M. Storey. An Exploratory Study of Software Reverse Engineering in a Security Context. 18th Working Conference on Reverse Engineering, Limerick, Ireland, pp. 184 – 188. (2011).

Tzerpos, V., & Holt, R. C. (2000, November). ACDC: An algorithm for comprehension-driven clustering. In wcre (p. 258). IEEE.

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

Ujhelyi, Z., Horvath, A., Varró, D., Csiszár, N. I., Szoke, G., Vidács, L., & Ferenc, R. (2014, February). Anti-pattern detection with model queries: A comparison of

approaches. In Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on (pp. 293-302). IEEE.

Urma, R. G., & Mycroft, A. (2012, October). Programming language evolution via source code query languages. In Proceedings of the ACM 4th annual workshop on Evaluation and usability of programming languages and tools(pp. 35-38). ACM.

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

Van Deursen, A., Hofmeister, C., Koschke, R., Moonen, L., & Riva, C. (2004, June). Symphony: View-driven software architecture reconstruction. In Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on (pp. 122-132). IEEE.

Van Geet, J., Ebraert, P., & Demeyer, S. (2010, September). Redocumentation of a legacy banking system: an experience report. In Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) (pp. 33-41). ACM.

Vasconcelos, A., y Werner, C. (2011). Evaluating reuse and program understanding in ArchMine architecture recovery approach. Information Sciences, 181(13), 2761-2786.

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

Von Detten, M. (2012). Archimetrix: A tool for deficiency-aware software architecture reconstruction. Paper presented at the Proceedings - Working Conference on Reverse Engineering, WCRE, 503-504.

Wu, Y., Yang, Y., Peng, X., Qiu, C., & Zhao, W. Recovering object-oriented framework for software product line reengineering. Top Productivity through Software Reuse. Springer Berlin Heidelberg, 119-134. (2011).

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

Yu, P., Ma, X., & Lu, J. (2007). Expectation, experiment and reflection on internet software evolution. ACM SIGSOFT Software Engineering Notes, 32(3), 1-5.

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

Zhou, Y. (2008). A runtime architecture-based approach for the dynamic evolution of distributed component-based systems. Paper presented at the Proceedings - International Conference on Software Engineering, 979-982.

Zhu, J., Huang, J., Zhou, D., Yin, Z., Zhang, G., & He, Q. (2013). Software architecture recovery through similarity-based graph clustering. International Journal of Software Engineering and Knowledge Engineering, 23(4), 559-586.

Anexo A.

Propuestas Recuperación de arquitecturas

Id	Año	Nombre de método	P/T	AR	Pr.	Autores
1	2013	Recovering Component Dependencies Hidden by Frameworks--Experiences from Analyzing OSGi and Qt	P	C	R	Forster, T. et al.
2	2013	SyMAR (Systematic method for architecture recovery)	P	E	R	Solms, F.
3	2013	Towards Reconstructing Architectural Models of Software Tools by Runtime Analysis	P	C	R	Peake, I. et al
4	2013	ArchDRH clustering (surge de DRH Design Rule Hierachy)	T	E	R	Cai, Y et al.
5	2013	SArF (Software Architecture Finder)	T	E	R/V	Kobayashi, K. et al.
6	2013	DGHC (Directed Graph Hierarchy Clustering)	T	E	R	Zhu, J et al.
7	2012	A Framework for Obtaining the Ground-Truth in Architectural Recovery	P	EC	R	Garcia, J. et al.
8	2012	Reconstructing Architectural Views from Legacy Systems	P	E	R	Boussaidi, G.E. et al.
9	2012	Extracting and Facilitating Architecture in Service-Oriented Software Systems	P	EC	R	Weinreich, R. et al.
10	2012	Archimetrix: Improved Software Architecture Recovery in the Presence of Design Deficiencies	P	E	R/E	Platenius, M.C. et al.
11	2012	Using fold-in and fold-out in the architecture recovery of software systems	T	E	R	Risi, M. et al.

Id	Año	Nombre de método	P/T	AR	Pr.	Autores
12	2011	SAROS (A Software Architecture Reconstruction Framework)	P	EC	E	Lee, S. y Kang, S.
13	2011	A top-down approach to construct execution views of a large software-intensive system	P	C	R	Callo, T. et al.
14	2011	An Automatic Architecture Reconstruction and Refactoring Framework	P	E	E	Schmidt, F.et al.
15	2011	A genetic approach for software architecture recovery from object-oriented code	T	E	R	Seriai, A. y S. Chardigny
16	2011	Architecture Reconstruction and Analysis of Medical Device Software	P	EC	R	Ganesan, D. et al.
17	2011	Towards Open Source Software System Architecture Recovery Using Design Metrics	P	E	R	Constantinou, E. et al.
18	2011	ARC (Architecture Recovery using Concerns)	P	EC	R	J. Garcia, D. et al.
19	2011	ArchMine	P	EC	C/R	Vasconcelos, A y Werner, C.
20	2010	RCA (Relational Concept Analysis)	P	E	R	El Hamdouni, A. E. et al.
21	2010	ZBR (Zone-Based Recovery)	T	E	R	Corazza, A. et al.
22	2010	Using the kleinberg algorithm and vector space model for software system clustering	T	E	R/E	Scanniello, G. et al.
23	2010	An Interactive Change Impact Analysis Based on an Architectural Reflexion Model Approach	P	E	R/E	Kim, T. H., et al.
24	2010	Architecture Recovery Using Latent Semantic Indexing and K-Means: An Empirical Evaluation	T	E	R	Risi, M. et al.
25	2010	Software architecture reconstruction: An approach based on combining graph clustering and partitioning	T	E	R	Şora, I. et al.
26	2010	Secoria	T	C	A	Abi-Antoun, M. y Barnes, J. M.
27	2010	Incremental Reflexion Analysis	T	E	C	Koschke, R
28	2009	FASAR: The Framework for Automating Software Architecture	P	EC	E	Kang, S. et al.
29	2009	FLORA (Framework for decomposing software architecture to introduce local recovery)	P	EC	R	Sözer, H., et al.
30	2009	A biting-down approach to hierarchical decomposition of object-oriented systems based on structure analysis	T	E	R	Zhang, L. et al.

Id	Año	Nombre de método	P/T	AR	Pr.	Autores
31	2009	Software Clustering Using Dynamic Analysis and Static Dependencies	T	EC	R	Patel, C. et al.
32	2009	Discovery of architectural layers and measurement of layering violations in source code	T	E	R/E	Sarkar, S. et al.
33	2009	Scholia	T	C	R	Abi-Antoun, M. y Aldrich, J.
34	2009	JavaCompExt: Extracting Architectural Elements from Java Source Code	T	EC	R	André, P. et al.
35	2008	ADDRA (Architectural Design Decision Recovery Approach)	P	EC	R	Jansen, A. et al.
36	2008	ROMANTIC	P	E	E	Chardigny, S. et al.
37	2008	A runtime architecture-based approach for the dynamic evolution of distributed component-based systems	P	C	E	Zhou, Y.
38	2007	SQUA3RE	P	EC	A	Störmer, C.
39	2007	Software Architecture Recovery	P	EC	R	Rasool, G. y N. Asif
40	2007	Static Analysis of Programs with Graphical User Interface	P	E	R	Staiger, S.
41	2007	Architecture recovery and evaluation aiming at program understanding and reuse	P	EC	R	Vasconcelos, A. y Werner, C.
42	2007	Extending the reflexion method for consolidating software variants into product lines	P	E	C	Frenzel, P. et al.
43	2007	Bayesian Learning for Software Architecture Recovery	T	E	R	Maqbool, O. y H. A. Babri
44	2006	PKUAS Runtime recovery and manipulation of software architecture of component-based systems	P	C	E	Huang, G. et al.
45	2006	An Orchestrated Multi-view Software Architecture Reconstruction Environment	P	EC	R	Sartipi, K. et al.
46	2006	DiscoTect	P	C	R/C	Schmerl, B. et al.
47	2006	Package patterns for visual architecture recovery	P	E	R/V	Lungu, M. et al.
48	2006	Focus	P	EC	R/E	Medvidovic, N. y Jakobac, V.
49	2006	QAR (QUE-es Architecture Recovery)	P	EC	Reu/E	Arciniegas, J.L.
50	2006	RAAM (recovering architectural assumptions method)	P	EC	E	Roeller, R. et al.
51	2005	Model-centric software architecture reconstruction	P	EC	A	Stoermer, C. et al.

Id	Año	Nombre de método	P/T	AR	Pr.	Autores
52	2005	PSG Process Structure Graph	T	EC	R	Qingshan, L. et al.
53	2005	Coping with legacy system migration complexity	P	EC	E	Wu, L. et al,
54	2005	LIMBO (scaLable InforMation BOttleneck)	T	E	R	Andritsos, P. y Tzerpos, V.
55	2004	CacOphoNy	P	EC	R	Favre, J. M
56	2004	Extraction and Visualization of Architectural Structure Based on Cross References among Object Files	P	E	Reu	Teng, Q. et al.
57	2004	Feature oriented architecture recovery process	P	EC	R	Pashov, I. y Riebisch, M.
58	2004	PuLSE-DSSA	P	EC	E	Pinzger, M. et al.
59	2004	Symphony	P	EC	R/C/E	Van Deursen, A. et al.
60	2004	W4	P	E	C/E	Hassan, A. E. y Holt, R. C.
61	2004	Architecture-aware adaptive clustering of OO systems	T	E	R	Bauer, M. y Trifu, M.
62	2004	The Graph Iterative Analysis Method	T	E	R	Jing, L. et al.
63	2004	WCA-UE (Weighted Combined Algorithm)	T	E	R	Maqbool, O. y Babri, H. A.
64	2004	Refining code-design mapping with flow analysis	T	E	E	Zhang, X. et al.
65	2003	QADSAR (Quality Attribute Driven Software Architecture Reconstruction)	P	EC	A	Stoermer, C. et al.
66	2003	Bridging legacy systems to model driven architecture	P	E	E	Qiao, B. et al.
67	2003	Architecture reconstruction guidelines	P	EC	R/C/A/E	Kazman R. et al.
68	2003	Architecture Recovery by Semi-Automatic Component Identification	T	E	R	Lundberg, J. y W. Löwe
69	2003	Reconstructing Software Architecture for J2EE Web Applications	P	EC	R	Han, M. et al.
70	2003	The Graph Clustering Method	T	E	R	Chiricota, Y. et al.
71	2003	Hierarchical Reflexion Models	T	E	C	Koschke, R. y D. Simon
72	2003	Towards the Unified Recovery Architecture for reverse engineering	P	EC	R	Panas, T. et al.
73	2003	Locating features in source code	P	EC	R	Eisenbarth, T. et al.
74	2002	Dynamo	P	EC	E	Ivkovic, I. y Godfrey, M. W.

Id	Año	Nombre de método	P/T	AR	Pr.	Autores
75	2002	Architecture reconstruction in practice	P	EC	R	Riva, C.
76	2002	Architecture recovery of Web applications	P	EC	R	Hassan, A.E. y R.C. Holt
77	2002	Combining static and dynamic views for architecture reconstruction	P	EC	R	Riva, C. y J.V. Rodriguez
78	2001	X-ray	P	C	R	Mendoca N. y Kramer J.
79	2001	A graph pattern matching Framework	P	E	R	Sartipi, K. y K. Kostas
80	2001	MAP (Mining architectures for product line evaluations)	P	EC	E	Stoermer, C. et al.
81	2000	URCA A Use-Case Driven Method of Architecture Recovery for Program Understanding and Reuse Reengineering	P	C	R/E	Bojic, D., y Velasevic, D.
82	2000	ACDC (Algorithm for Comprehension-Driven Clustering)	T	E	R	Tzerpos, V. y Holt, R. C.
83	1999	SAR (Software Architecture Reconstruction Method)	P	E	E	R. L. Krikhaar
84	1999	Recovering high-level views of object-oriented applications from static and dynamic information	P	EC	R	Richner, T. y Ducasse, S.
85	1999	ARM (Architecture Reconstruction Method)	P	EC	C	G. Guo, J. et al.
86	1999	Bowman and Richard	P	E	R	Bowman, I. T. et al.
87	1999	Bunch-NAHC / SAHC	T	E	R	Mancoridis, S. et al.
88	1998	Event-based detection of concurrency	T	C	R	Cook, J. E. y A.L. Wolf
89	1993	Techniques for discovering, restructuring, and analyzing subsystem structures in software systems using Rigi	P	E	E/A	Müller, H.A y M. Orgun

Convenciones utilizadas		
Símbolo	**Significado**	**Posibles valores**
P/T	Proceso o Técnica	P: Proceso, T: Técnica
AR	Aspecto recuperado	E: Estructura, C: Comportamiento, EC: Estructura y comportamiento
Pr.	Propósito	A: Análisis, C: Conformidad, E: Evolución, R: Re-documentación, Reu: Reutilización, V: visualización

Anexo B.

Inventario herramientas de ingeniería inversa

Id.	Herramienta	Primera versión	Ultima versión	As.	Autores
1	Archimetrix	2014	2014	A	von Detten et al.
2	ARTs	2013	2013	A	Ioana, S.
3	CoCA-Ex	2013	2013	V	Holy et al.
4	Code Visual	2013	2015	A	Nielsen et al.
5	DynaRIA	2013	2013	A	Amalfitano et al.
6	JITTAC	2013	2014	A	Buckley et al.
7	Lego	2013	2013	A	Srinivasan y Reps
8	ACTool	2012		A	Ali et al.
9	Senseo	2012	2012	A	Röthlisberger et al.
10	SAMS	2012	2012	A	Su y Yeh
11	ArchRecJ	2011	2011	A	Abi-Antoun y Hailat
12	URCA	2011	2011	A	Bojic y Velasevic
13	VisCad	2011	2011	D	Asaduzzaman et al.
14	iCIA	2010	2010	V	Kim et al.
15	KADRE	2010	2010	A	Woollard et al.
16	Reclipse	2010	2014	D	von Detten et al.
17	ROPTool	2010	2015	A	Miao et al.
18	SAME	2010	2010	A	Vazquez et al.
19	SM@RT	2010	2010	A	Zhang et al.
20	MoDisco	2009	2015	A	Bruneliere et al.
21	phpModeler	2009	2009	D	Maras et al.
22	PRECISO	2009	2009	BD	Perez-Castillo et al.
23	ReAjax	2009	2012	A	Marchetto et al.
24	SAVE	2009		A	Duszynski et al.
25	CCVISU	2008	2012	V	Beyer, D.
26	DeMIMA	2008	2008	D	Guéhéneuc y Antoniol
27	ESS-Model	2008	2003	D	Eldean AB
28	MARPLE	2008	2010	A	Fontana y Zanoni
29	Q-ImPrESS	2008	2011	A	Koziolek et al
30	SOLIDFX	2008		A	Telea y Voinea
31	SoMoX	2008		A	Klatt et al.

Id.	Herramienta	Primera versión	Ultima versión	As.	Autores
32	SQL2XMI	2008		BD	Alalfi et al.
33	DP-Miner	2007	2007	D	Dong et al.
34	WSAD extractor	2007		A	Kienle et al.
35	Artemis-ARC	2006	2006	A	Yu et al.
36	CPP2XMI	2006	2006	D	Korshunova et al.
37	DPD	2006	2010	D	Tsantalis et al.
38	FINT	2006	2006	A	Marin et al.
39	PINOT	2006	2006	D	Shi y Olsson
40	SMArTIC	2006		A	Lo y Khoo
41	Softwarenaut	2006		A	Lungu et al.
42	ArchView	2005	2005	V	Pinzger, M.
43	ARMIN	2005		A	Kazman et al.
44	ARTISAn	2005	2005	A	Jakobac et al.
45	DPRE	2005	2005	D	De Lucia et al.
46	DPVK	2005	2005	D	Wang y Tzerpos
47	iPlasma	2005	2005	D	Marinescu et al
48	IRiSS	2005	2009	D	Poshyvanyk et al.
49	ReArchJBs	2005		A	Sun et al.
50	REGoLive	2005	2005	A	Gui et al.
51	SrcML	2005	2015	D	Collard, M. L.
52	VANESSA	2005		V	Pacione et al.
53	XDRE	2005		A	Li et al.
54	ArchVis	2004		V	Hatch, A.
55	artDECO	2004	2004	A	Peltonen y Selonen
56	Cacophony	2004	2004	A	Favre, J-M
57	DEREF	2004		A	Teng et al.
58	DiscoTect	2004		A	Yan et al.
59	DMS	2004	2015	A	Semantic Designs inc.
60	Lattix	2004	2016	A	Lattix
61	Nimeta	2004	2004	A	Riva, C.
62	PKUAS	2004	2005	A	Huang et al.
63	PTIDEJ	2004	2014	D	Guéhéneuc et al.
64	SA4J	2004	2004	A	alphaWorks
65	UFJ	2004	2012	A	Scientific Toolworks, Inc.
66	CodeCrawler	2003	2003	V	Lanza, M.
67	JSPick	2003	2013	D	Draheim et al.
68	SPQR	2003	2003	D	Mc Smith y Stotts
69	VizzAnalyzer	2003	2003	V	Löwe et al.
70	ARIS	2002		A	Traverso y Mancoridis
71	CodeSurfer	2002	2002	D	Grammatech Inc.
72	Columbus	2002	2002	D	Ferenc et al.
73	Maisa	2002	2002	A	Paakki et al.
74	Revealer	2002		A	Pinzger et al.

Id.	Herramienta	Primera versión	Ultima versión	As.	Autores
75	SWAG KIT	2002	2005	A	SWAG: Software Architecture Group
76	WARE	2002		A	Di Lucca et al.
77	Alborz	2001	2001	A	Sartipi, K.
78	CCFinder	2001	2008	D	Kamiya et al.
79	Daikon	2001	2015	D	Ernst et al.
80	DynaSee	2001	2002	D	Zayour y Lethbridge
81	JBOORET	2001	2001	D	Hong et al.
82	Jinsight	2001	2002	A	De Pauw et al.
83	MAP	2001	2001	A	Stoermer y O'Brien
84	Reportal	2001		D	Mancoridis et al.
85	VAQUISTA	2001		D	Vanderdonckt et al.
86	DejaVu	2000	2000	D	Alpern et al.
87	Moose	2000	2000	A	Ducasse et al.
88	ART	1999		A	Fiutem et al.
89	Bauhaus	1999	2004	A	Koschke, R.
90	Dali	1999	1999	A	Kazman y Carrière
91	InSight	1999	1999	D	Rajala et al.
92	Jdepend	1999	2009	A	Clarkware Consulting, Inc.
93	SPOOL	1999	1999	D	Keller et al.
94	URSA	1999		A	Bril et al.
95	Acacia	1998	1998	D	Chen et al.
96	ARES	1998		A	Eixelsberger et al.
97	GRASP	1998	2015	D	Cross et al.
98	ISVis	1998	2008	V	Jerding
99	Refine/C	1998	1998	D	Software Research Laboratory
100	SARTool	1998		A	Krikhaar, R.
101	SCED	1998		A	Systa, T.
102	Doxygen	1997	2015	D	Dimitri van Heesch
103	Imagix4D	1997	2015	A	Imagix Corporation
104	ITOC	1997	1997	D	Berglas y Harrison
105	PBS	1997	1997	A	Prechelt y Krämer
106	DB-MAIN - CASE	1996	2015	BD	Hainaut et al.
107	Mandrake	1996	1996	D	Morris y Filman
108	ManSART	1996	1996	A	Chase et al.
109	Rigi	1996	2010	A	Storey et al.
110	SNiFF+	1996	1996	V	Bischofberger, W.R.
111	August-II	1995	1995	BD	Davis, K.H.
112	RMTool	1995		A	Murphy eta al.
113	SHriMP	1995	1995	V	Storey et al.
114	Veda	1994		BD	Chiang et al.
115	PHSE	1993	1993	A	Tilley et al.

As.: Hace referencia al aspecto que aborda la herramienta. Los posibles valores son: A: arquitectura, D: Diseño, BD: Bases de datos y V: Visualización.

Anexo C.

Fase de Inicio

Hitos	Especificar el ámbito del problema				
	Establecer la viabilidad de la realización del proceso				
	Establecer el plan de trabajo				
Actividad	**Metas**	**Entrada**	**Técnicas**	**Instrumentos**	**Salidas**
Definir el contexto	Identificar el contexto	Conocimiento del experto en ingeniería inversa	Inferencia	Caracterización de los contextos de uso	Descripción del problema
	Plantear el problema	Conocimiento de los interesados, documentación	Entrevista	Plantilla para la descripción del problema	
Analizar la viabilidad del proceso	Definir la viabilidad del proceso	Descripción del problema, Conocimiento de los interesados, documentación	Algoritmo de decisión	Relación de vistas arquitectónicas según contexto, listas de chequeo, ficha de caracterización de herramientas de Ingeniería Inversa	Decisión sobre la viabilidad del proceso, Las vistas objetivo, Lista de artefactos disponibles, Lista de artefactos requeridos, Técnicas a utilizar, Herramientas a utilizar, Costo del proceso
Planificar el proceso	Definir el plan de trabajo	Descripción del problema, vistas objetivo, listado de artefactos disponibles y requeridos, y listado de técnicas y herramientas a utilizar	Asignación de recursos		Plan de trabajo

Anexo D.

Fase de extracción de datos

Hito	Identificar de los elementos que conforman el producto software y las relaciones que existen entre ellos en bajo nivel				
Actividad	**Metas**	**Entrada**	**Técnicas**	**Instrumentos**	**Salidas**
Definir el modelo de dominio de la aplicación	Definir el dominio de la aplicación	Descripción del problema, conocimiento del experto, usuarios y personal técnico	Modelado conceptual	Diagramas UML	Modelo de dominio de la aplicación
Descomponer artefactos	Identificar los elementos que conforman el sistema y sus relaciones	Artefactos software	Análisis estático, análisis dinámico	KDM	Modelo del sistema en bajo nivel

Anexo E.

Fase de organización del conocimiento

Hito	Convertir el modelo del sistema en bajo nivel en un modelo en alto nivel				
Actividad	**Metas**	**Entrada**	**Técnicas**	**Instrumentos**	**Salidas**
Abstraer modelos	Identificar elementos y relaciones en alto nivel a partir del modelo del sistema en bajo nivel	Modelo del sistema en bajo nivel, modelo de dominio de la aplicación y conocimiento del experto	Manuales, semiautomáticas y automáticas	Reglas de mapeo	Elementos y relaciones en alto nivel
Representar modelos		elementos y relaciones en alto nivel	Mapeo	UML, reglas de mapeo	Modelo del sistema en alto nivel

Anexo F.

Fase de exploración de la información

Hito	Elaborar el informe con el análisis de las vistas arquitectónicas recuperadas, indicando el cumplimiento de los objetivos establecidos para el proceso en la fase de inicio				
Actividad	**Metas**	**Entrada**	**Técnicas**	**Instrumentos**	**Salidas**
Visualizar resultados	Presentar los resultados en forma gráfica	Modelo del sistema en alto nivel	Metáforas, hipervínculos	UML	Vistas objetivo representadas en UML
Analizar resultados	Interpretar los resultados obtenidos en el proceso	Modelo del sistema en alto nivel, modelo de dominio del problema y de la aplicación, y conocimiento del experto	Inferencia	Mecanismo de consulta	Análisis de los resultados
Formular resultados	Organizar los resultados en un informe final para que sea posible su interpretación por parte de todos los interesados	Vistas objetivo representadas en UML, Análisis de los resultados, modelo de dominio del problema y de la aplicación, y conocimiento del experto	Redacción del documento		Informe del proceso

Anexo G

Protocolo del estudio de caso

1. Visión general del estudio de caso

Los estudios de caso permiten la investigación de realidades contextuales y establecer la diferencia entre lo que ha sido planeado y lo que realmente sucede (Anderson, 1993), por eso se utiliza como estrategia de evaluación del marco de referencia propuesto, para comparar lo establecido por el modelo conceptual definido y el mecanismo de consulta diseñado, con los resultados obtenidos al utilizarlos en escenarios reales de recuperación y análisis de arquitecturas, para lo cual se realiza un estudio de caso descriptivo, con el ánimo de identificar las coincidencias y posibles inconsistencias entre la propuesta teórica y las evidencias recolectadas.

Se utiliza un estudio de caso porque en comparación con otros métodos de evaluación como las encuestas, los experimentos y los cuasi-experimentos, los estudios de caso pueden 1) capturar la complejidad del caso, incluyendo cambios relevantes en el tiempo, 2) atender en forma completa las condiciones del contexto, incluyendo aquellas que potencialmente interactúan con el caso (Ying, 2013).

Se realiza un estudio de caso único (embebido) con dos unidades de análisis porque se va a probar una propuesta teórica (el marco de referencia ARCo) a partir de escenarios críticos (Ying, 2013). El primer escenario concierne a la producción de software, porque es el contexto típico para el cual han sido definidas todas las propuestas identificadas en la revisión de la literatura, convirtiéndose en una unidad de análisis obligatoria para evaluar ARCo. Por otra parte, la revisión de la literatura reveló que la ingeniería inversa se puede utilizar como herramienta didáctica en los procesos de enseñanza aprendizaje (Klimek et al., 2011; Cipreso, 2009; Ali, 2006; Ali, 2005), sin embargo, no se encontró ninguna propuesta que guié el proceso de recuperación y análisis de vistas arquitectónicas en un contexto tan particular como éste, por eso la segunda unidad de análisis corresponde al contexto de la educación.

El diseño del estudio de caso se describe a continuación, 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. En primera instancia, las preguntas surgen a partir de la intención del estudio de caso: evaluar ARCo, por eso se elaboran tomando como referente principal la pregunta de investigación de la tesis (¿Cómo extraer información a partir de los artefactos generados en un proceso de ingeniería inversa, para realizar análisis a nivel de arquitectura sobre el comportamiento de un sistema software?), a la que ARCo debe responder, por lo tanto se afirma que (*Proposición 1*): ARCo guía el proceso de recuperación y análisis sobre el comportamiento a nivel de arquitectura de un sistema software, atendiendo las características particulares de la situación que rodea el contexto en el que se realiza dicho proceso.

Para comprobar esta afirmación se estableció como *pregunta de investigación* en el estudio de caso: ¿Cómo contribuye ARCo en el proceso de recuperación y análisis a nivel de arquitectura sobre el comportamiento de un sistema software?. Como este proceso depende del contexto en el que se realiza (*Proposición 2*), la misma pregunta se formula en forma más concreta para cada una de las unidades de análisis del estudio de caso, según se indica a continuación: ¿Cómo contribuye ARCo en el proceso de recuperación y análisis sobre el comportamiento a nivel de arquitectura de un sistema software, en un contexto de producción de software? y ¿Cómo contribuye ARCo en el proceso de recuperación y análisis sobre el comportamiento a nivel de arquitectura de un sistema software, en un contexto de educación?.

Para la *primera unidad de análisis* (contexto producción de software) el estudio de caso se realiza en una Empresa del sector Industrial de la ciudad de Cartagena, que cuenta en su estructura organizacional, con un departamento de sistemas que cumple funciones de desarrollo de software y de gestión de la infraestructura tecnológica. Por respeto al convenio de confidencialidad firmado con dicha empresa, no se publica su identidad, por eso en este documento se referencia como Empresa del sector industrial. Para la *segunda unidad de análisis* (contexto educación) el estudio de caso se realiza 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 realiza 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). Se selecciona éste modelo de análisis, porque el estudio de caso se realiza para evaluar ARCo, demostrando que los eventos empíricos observados al utilizar el Marco de Referencia, coinciden con la propuesta teórica presentada en esta tesis. Para argumentar la *interpretación de los hallazgos* se utiliza como principal estrategia el análisis basado en proposiciones teóricas para hacer inferencias lógicas, a partir de las propuestas conceptuales identificadas en la revisión de la literatura.

Para juzgar la calidad del diseño del estudio de caso se utilizan los criterios de juicio definidos por Ying (2013): validez de la construcción, validez externa y confiabilidad; y se aplican las tácticas que propone el mismo autor para garantizar el cumplimiento de los criterios de juicio, como se indica a continuación. No se tiene en cuenta el criterio de validez interna, porque éste sólo aplica para estudios de caso explicativos Y(ing, 2013), que mantienen una relación causal, a diferencia de estudios de caso exploratorios y descriptivos. Sin embargo, se resalta que durante el análisis de los datos se utiliza la técnica de análisis denominada Modelo lógico, como se menciona primitivamente en este documento.

Para lograr la validez de la construcción, durante la recolección de datos se aplican los principios: 1) utilización de múltiples fuentes de evidencia (Documentación, artefactos físicos y entrevistas), haciendo posible la aplicación de triangulación de la información recolectada y permitiendo identificar posibles inconsistencias. 2) Preservación de la cadena de evidencia, permitiendo a observadores externos mantener la trazabilidad del estudio de caso, desde la formulación de las preguntas hasta el reporte de las conclusiones, pasando por el protocolo, las evidencias soportadas por las fuentes y registradas en la base de datos del estudio. Y 3) Creación de la bases de datos del estudio de caso, soportada en archivos de texto manipulados con procesadores de palabras, repositorios de código fuente, repositorios XMI, tablas con datos tabulados manejados con hojas de cálculo, e imágenes.

Para garantizar la validez externa del estudio, se establece que el domino para el cual los resultados del estudio pueden ser generalizados es: la recuperación y análisis de vistas arquitectónicas de comportamiento. Además, como principal estrategia de análisis para argumentar la interpretación de los hallazgos y hacer inferencias lógicas, se utiliza el análisis basado en proposiciones teóricas identificadas en la revisión de la literatura. Finalmente, para garantizar la confiabilidad del estudio de caso, se diseña este protocolo y se organiza una base de datos de la información recolectada, haciendo posible que se pueda repetir llegando a los mismos resultados.

2. Procedimiento de recolección de datos

El procedimiento de recolección de datos se centra en registrar las evidencias que sirven para evaluar ARCo, con el propósito de identificar las coincidencias y posibles inconsistencias entre la propuesta teórica y las evidencias recolectadas al momento de aplicarla en escenarios reales de uso. Por eso la información recolectada se organiza en dos grupos, el primero corresponde al registro de los resultados de la aplicación del modelo conceptual y la utilización del mecanismo de consulta; y el segundo grupo lo conforma la percepción de los participantes en el proceso de recuperación y análisis de vistas arquitectónicas de comportamiento, para cada una de las unidades de análisis definidas.

Para valorar la percepción de los participantes se utiliza como fuente de evidencia la entrevista, mientras que para el registro de los resultados se utiliza como fuente de evidencia artefactos físicos y documentación. Adicionalmente, las fuentes de evidencia se organizan en una base de datos, en la que se registra para cada unidad de análisis los distintos tipos de evidencia que se utilizaron. Para orientar el orden cronológico de la recolección de evidencias y la forma como se construye la base de datos del estudio de caso, el plan de recolección de datos se hace con base en las actividades propuestas por la metodología que forma parte de ARCo, según se indica más adelante en esta misma sección. Además, se elaboran los documentos "Guía de realización" para cada una de las unidades de análisis, en los cuales se registra los resultados obtenidos a partir de la realización del estudio de caso.

El plan de recolección de datos obtenidos como resultado de la aplicación de ARCo, se presenta en la Tabla 17, indicando las fuentes de evidencia que se utilizan para cada actividad que se realiza al emplear la metodología propuesta. Por otra parte, para el registro de la percepción de los participantes al final del proceso se aplican las entrevistas relacionadas los anexos H e I, diseñadas para cada unidad de análisis.

Por cumplimiento con el convenio de confidencialidad firmado con la Empresa donde se realizó el estudio de caso, en este documento no se utiliza su nombre, y se hace referencia a ella con la expresión: Empresa del sector industrial. De igual forma, para no atentar contra el derecho a la intimidad de los estudiantes que participaron, y garantizar la reserva de la información del proceso de aprendizaje del grupo en el que se realizó el estudio de caso, no se hace público el nombre del docente ni de los estudiantes, sólo se hace referencia al grupo en general indicando que pertenece al programa de Ingeniería de Sistemas de la Universidad de Cartagena en Colombia.

Actividad	Fuentes de evidencia
Definición del contexto	Documento: Descripción del problema
Análisis de la viabilidad del proceso	Documentos: Listado de vistas objetivo, Lista de artefactos disponibles, Lista de artefactos requeridos, Listado de técnicas y herramientas a utilizar
Planificación del proceso	Documento: Plan de trabajo
Definición del modelo de dominio de la aplicación	Documentos: modelo de dominio de la aplicación y arquitectura hipotética
Descomposición de artefactos y Abstracción de modelos	Artefacto físicos: Diagramas de clase, Modelo del sistema representados en XMI
Representación de modelos	Artefacto físico: Modelos del sistema en alto nivel. representación de las vistas recuperadas
Análisis de resultados	Documento de análisis de resultados
Formulación de resultados	Documento informe del proceso.

Tabla 17. Plan de recolección de datos

3. La recolección de datos

El propósito de esta sección en orientar al investigador para que mantenga el horizonte del estudio de caso sin desviarse de su objetivo (Ying, 2013), por eso se formulan los aspectos que sirven de guía al investigador para identificar la información que debe recolectar y cómo lo debe hacer. En este orden de ideas, y teniendo en cuenta que el objetivo del estudio de caso es evaluar ARCo, se establece que los aspectos que se deben tener en cuenta en la recolección de datos son:

- La recolección de datos se realiza a partir de la ejecución cronológica de las actividades de la metodología propuesta en ARCo, utilizando como fuente de evidencia directa los artefactos que se generan como resultado de la ejecución de cada actividad.
- Sobre las evidencias recolectadas se hace un análisis con respecto a la forma como contribuyen a la recuperación y análisis de las vistas arquitectónicas de comportamiento, teniendo en cuenta el contexto en el que se realiza el estudio de caso.
- Se identifican las mejoras y los aportes que representa ARCo citando las evidencias que lo soportan.

Para que el estudio de caso cumpla con su objetivo principal, cada unidad de análisis debe responder a las siguientes preguntas:

- ¿Lo que plantea ARCo se evidencia en la práctica?
- ¿Permite ARCo hacer recuperación de arquitecturas teniendo en cuenta el contexto en el que se presenta la situación?
- ¿Permite ARCo hacer análisis sobre el comportamiento de sistemas software, teniendo en cuenta el contexto en el que se presenta la situación?
- En comparación con otras propuestas para la recuperación y análisis de vistas arquitectónicas de comportamiento, ¿Cómo mejora ARCo este proceso?

4. Guía para la presentación del informe

El informe del estudio de caso tiene como fin principal mostrar los resultados de la evaluación de ARCo, por eso se estructura en tres partes. La primera parte es introductoria y sirve para presentar brevemente el diseño del estudio de caso. En la segunda parte se presentan los resultados de la aplicación del estudio de caso, para cada una de las unidades de análisis. Finalmente, en la tercera parte se expone el análisis sobre los resultados de la aplicación del estudio de caso, tomando como guía las preguntas planteadas a cada unidad de análisis, presentadas previamente.

Anexo H.

Diseño entrevistas. Unidad de análisis: Contexto desarrollo de software

Objetivos: Identificar la percepción de los participantes en el proceso de recuperación y análisis de vistas arquitectónicas de comportamiento, realizado en la Empresa del sector industrial de la ciudad de Cartagena.

A continuación se presentan las preguntas que se formulan a cada rol que participa en el proceso.

1) Entrevista al arquitecto de software

1. ¿Ha participado usted en procesos de recuperación y análisis de arquitecturas?
2. En caso de respuesta afirmativa, ¿qué técnicas y herramientas ha utilizado?
3. ¿Considera usted que la aplicación de ARCo permitió el logro de los objetivos propuestos para el proceso?
4. ¿Qué aspectos de la arquitectura recuperó aplicando ARCo?
5. ¿Qué tipo de análisis pudo realizar al aplicar ARCo?
6. ¿Cómo contribuye ARCo en el proceso de recuperación y análisis a nivel de arquitectura sobre el comportamiento de un sistema software?
7. ¿Qué utilidad tiene ARCo?
8. ¿Qué limitaciones tiene ARCo?
9. ¿Utilizaría usted ARCo en futuros trabajos de recuperación y análisis de vistas arquitectónicas de comportamiento?

2) Entrevista al Jefe del departamento de sistemas, al Analista de sistemas y al analista Junior

1. ¿Ha participado usted en procesos de recuperación y análisis de arquitecturas?
2. En caso de respuesta afirmativa, ¿qué técnicas y herramientas ha utilizado?

3. ¿Considera usted que al terminar el proceso de recuperación y análisis de las vistas arquitectónicas de comportamiento, se logró el objetivo planteado?
4. ¿Cómo contribuye ARCo en el proceso de recuperación y análisis a nivel de arquitectura sobre el comportamiento de un sistema software?
5. ¿Qué utilidad tiene ARCo?
6. ¿Qué limitaciones tiene ARCo?
7. ¿Utilizaría usted ARCo en futuros trabajos de recuperación y análisis de vistas arquitectónicas de comportamiento?

Anexo I.

Diseño entrevistas. Unidad de análisis: Contexto Educación

Objetivos: Identificar la percepción de los participantes en el proceso de recuperación y análisis de vistas arquitectónicas de comportamiento, realizado en la Universidad de Cartagena - Colombia.

A continuación se presentan las preguntas que se formulan para cada rol que participó en el proceso.

1) Entrevista al Docente

1. ¿Ha participado usted en procesos de recuperación y análisis de arquitecturas?
2. En caso de respuesta afirmativa, ¿qué técnicas y herramientas ha utilizado?
3. ¿Considera usted que al terminar el proceso de recuperación y análisis de las vistas arquitectónicas de comportamiento, se logró el objetivo de aprendizaje planteado?
4. ¿Qué aspectos de la arquitectura recuperó aplicando ARCo?
5. ¿Qué tipo de análisis pudo realizar al aplicar ARCo?
6. ¿Cómo contribuye ARCo en el proceso de recuperación y análisis a nivel de arquitectura sobre el comportamiento de un sistema software?
7. ¿Qué utilidad tiene ARCo?
8. ¿Qué limitaciones tiene ARCo?
9. ¿Utilizaría usted ARCo como herramienta didáctica en futuros escenarios de aprendizaje?

1) Entrevista a los estudiantes

Como participan 16 estudiantes las preguntas se formulan aplicando una encuesta, para obtener la percepción de todos.

1. ¿Considera usted que entiende el concepto de polimorfismo en el contexto de la programación orientada a objetos?
    a. Si ____
    b. No ____
2. ¿Considera usted que puede aplicar el concepto de polimorfismo para la solución de problemas prácticos?
    a. Si ____
    b. No ____
3. Considera usted que el laboratorio que realizó le permitió (puede seleccionar varias opciones):
    a. Aprender a partir de un sistema diseñado y desarrollado por expertos (JHotDraw)
    b. Comprender el concepto de polimorfismo
    c. Desarrollar habilidades para modelar
    d. Desarrollar habilidades para programar
    e. Entender la importancia de los modelos
    f. Estimular su curiosidad por la forma como está diseñado el sistema JHotDraw
    g. Hacer mantenimiento al sistema JHotDraw
    h. Ninguna de las anteriores
    i. Otra opción. Cuál?_____
4. ¿El mecanismo de consulta QModel-XMI le sirvió para identificar posibles comportamientos polimórficos a partir del modelo recuperado del sistema JHotDraw?
    a. Si ____
    b. No ____
5. ¿El mecanismo de consulta QModel-XMI le sirvió para analizar la arquitectura del sistema JHotDraw?
    a. Si ____
    b. No ____
6. ¿El mecanismo de consulta QModel-XMI le sirvió para entender parte del sistema JHotDraw?
    a. Si ____
    b. No ____
7. ¿En el transcurso de su formación profesional había utilizado la ingeniería inversa como estrategia didáctica?
    a. Si ____
    b. No ____
8. ¿Le gustaría que esta estrategia didáctica se use con mayor frecuencia?
    a. Si ____
    b. No ____