DEFINICIÓN DE UN MÉTODO BASADO EN ÁLGEBRA RELACIONAL PARA REALIZAR ANÁLISIS A ARQUITECTURAS DE SOFTWARE OBTENIDAS A PARTIR DE INGENIERÍA INVERSA

INVESTIGADOR

Jorge Andrés Osorio Romero

UNIVERSIDAD DE CARTAGENA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS, 2014

DEFINICIÓN DE UN MÉTODO BASADO EN ÁLGEBRA RELACIONAL PARA REALIZAR ANÁLISIS A ARQUITECTURAS DE SOFTWARE OBTENIDAS A PARTIR DE INGENIERÍA INVERSA

TESIS DE GRADO

E-Soluciones

Ingeniería Del Software

INVESTIGADORES

Jorge Andrés Osorio Romero

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

UNIVERSIDAD DE CARTAGENA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS, 2014

UNIVERSIDAD DE CARTAGENA

Tesis de Grado: DEFINICIÓN DE UN MÉTODO BASADO EN ÁLGEBRA RELACIONAL PARA REALIZAR ANÁLISIS A ARQUITECTURAS DE SOFTWARE OBTENIDAS A PARTIR DE INGENIERÍA INVERSA

Autores: JORGE ANDRÉS OSORIO ROMERO

Director: Msc. MARTÍN MONROY RÍOS

Nota de Aceptación

_____

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Cartagena de Indias, ____ de _____ de 2014

RESUMEN

El proceso de ingeniería de software produce artefactos útiles para analizar un sistema construido mediante la aplicación de una metodología de desarrollo de software. Estos son diagramados a través de herramientas CASE[1] que generan diagramas UML[2] utilizando ingeniería inversa y generalmente constituyen representaciones del software a nivel de implementación, aunque también pueden representar un nivel de abstracción alto: la arquitectura del software.

Realizar análisis a arquitecturas de software es un reto, determinar métodos que permitan analizar una arquitectura de software representaría grandes beneficios para la ingeniería del software.

Con la finalidad de suplir la necesidad descrita en el párrafo anterior se planteó el presente proyecto, cuyo objetivo radicó en definir un método fundamentado en principios del álgebra relacional que permita realizar análisis a arquitecturas de sistemas software, para ser usado como herramienta de apoyo en procesos de ingeniería inversa.

Para dar cumplimiento a los objetivos del proyecto se realizó una investigación de tipo exploratoria, dado que las temáticas a indagar han sido poco abordadas por los científicos debido a que se desprenden de necesidades relativamente nuevas para las ciencias de la computación. Este tipo de investigación se aplicó centrando sus esfuerzos en el análisis documental de fuentes académicas verificables que permitieron soportar los lineamientos propuestos durante la ejecución del proyecto.

Gracias al trabajo realizado se obtuvo la definición de un método que permite realizar análisis a arquitecturas de software cuyos postulados se basan en álgebra relacional, además de un completo análisis y revisión de literatura que pudo ser conjugada en el estado del arte del presente documento.

[1] Herramientas de diseño asistido por computadora.
[2] UML: lenguaje unificado de modelado, utilizado para representar diagramas de software.

La investigación trajo como conclusión que gracias al enfoque del álgebra relacional se puede realizar análisis a arquitecturas de software, permitiendo generar conocimiento que puede seguir siendo estudiado para ser aplicado en herramientas de diseño asistido por computador aunque, el trabajo invertido en el área debe enfocarse más hacia la parte matemática que fundamenta el método propuesto debido a que aún falta un largo trecho por recorrer antes de llegar a una solución óptima.

Palabras clave: *Arquitectura De Software, Álgebra Relacional, Análisis Arquitectural.*

ABSTRACT

The software engineering process produces useful artifacts to analyze a system built through the application of a software development methodology. These ones are diagrammed through CASE tools that generate UML diagrams using reverse engineering and generally constitute representations of the software in an implementation level, although they also can represent a higher abstraction level: the software architecture.

Perform analysis on software architectures is a challenge, determine methods that enable analyze software architecture would represent big benefits for the software engineering.

In order to supply the need described above this project was raised, which aiming was focused in to define a method based on relational algebra principles to perform analysis on software architectures, to be used as a support tool on reverse engineering processes.

To reach the project aims an exploratory type investigation was conducted, because the main topics have been little addressed by the scientists because these ones are relatively news needs. This type of investigation was applied focusing its efforts on the verifiable academic fonts analysis which allowed supporting the guidelines proposed during the project execution.

Thanks to the work done the definition of a method that allows perform analysis on software architecture which guidelines are based on relational algebra was obtained, and also a complete analysis and literature review that could be conjugated in the state of the art of this document.

The research brought as conclusion that thanks to the relational algebra approach, the software architecture analysis can be performed, allowing to generate knowledge that could be being studied to be applied on computer assisted design tools, although, the work invested on this area must be focused to the mathematical field that supports the proposed method because there is a long way to travel before reaching an optimal solution.

Keywords: *Software Architecture, Relational Algebra, Architectural Analysis.*

DEDICATORIA

A mi madre Viviana De Jesús Romero Tapia, por su paciencia, apoyo y enorme sacrificio realizado durante estos años.

A mi hermano Pedro Rafael Rodríguez Romero.

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

"For the brave, nothing is too difficult" – AW

AGRADECIMIENTOS

En primer lugar quiero agradecer especialmente a mi madre quien ha sido la única persona que ha estado todo el tiempo a mi lado, animándome a seguir adelante cuando las cosas no han ido bien y aconsejándome para ser una mejor persona cada día. Ella junto a mi hermano, han sido una motivación constante para llegar cada vez más lejos.

Me siento agradecido con el programa de Ingeniería De Sistemas de la Universidad De Cartagena, pues en 5 años cultivé muchos conocimientos y mejoré otros más bajo la orientación de grandes docentes que me enseñaron a ser un buen profesional no solo en el campo laboral sino también en la vida. Con los profesores Martín Monroy, Julio Rodríguez, David Franco, Yasmín Moya, Raúl Martelo, Luis Tovar, Plinio Puello, Mónica Ospino y Miguel García me sentiré siempre en deuda pues con sus conocimientos aprendí a desempeñarme en el área de la Ingeniería De Sistemas; así mismo, con Alcibaldo Miranda, Alejandra Bello, Ricardo Galezo, Edil Melo, Dagoberto Morón, Ruperto Ealo, William Wood, Juan Pérez, Yahaira Freile, entre otros, quienes se encargaron de complementar la formación recibida fuera del plano destinado a la Ingeniería De Sistemas

Finalmente agradecer a mis compañeros Andrés Betín, Jhonny Madera, Kevin Sarmiento, Arturo Verbel, Ling Lung, Alejandra Ríos, Cindy Pacheco, David Flórez, Jaider Garcés y María Zabaleta, junto a quienes viví la etapa de ser un universitario durante estos años.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Dentro de los múltiples campos que abarca la Ingeniería De Sistemas se pueden encontrar soluciones útiles a la hora de resolver problemas en campos variados como las redes, telecomunicaciones, soporte y electrónica, lo que finalmente conlleva a la satisfacción de una o más necesidades. La línea de la ingeniería del software no es la excepción y, desde sus inicios, ha ido en un proceso continuo de mejoramiento buscando finalmente producir software de calidad.

La vertiente tradicional -partir de una necesidad y desembocar en su solución-, surgió como respuesta a los inconvenientes que se estaban presentando en el área debido a la crisis del software y fue tan alto su impacto que, hoy día, los productos de alta calidad se concretan en tiempos mínimos, con procesos más manejables y completamente tecnificados y estandarizados, de una forma muy diferente a como se llevaba unos años atrás. Es así que con ayuda de los múltiples patrones de software propuestos por los estudiosos del área (Gamma, Helm, Johnson, & Vlissides, 1995), además de las buenas prácticas de programación y cambios de paradigma -en primer momento estructurales, luego orientados a objetos-, que la ingeniería del software se ha convertido actualmente en una de las áreas en donde más esfuerzo investigativo se emplea y por ende, una de las áreas que más innovación introduce y que más interés despierta (Canfora & Di Penta, 2007).

Los productos software actuales surgen a partir de un gran esfuerzo aplicado por todo un equipo de análisis, diseño, y desarrollo. La calidad de los mismos así lo avala; sin embargo, el paradigma de construcción de estos ha cambiado. Existen soluciones tan funcionales como complejas, que despiertan interés por analizar su funcionamiento interno, el producto final está empaquetado, pero es posible descomponerlo gracias a los procedimientos y técnicas que ofrece la ingeniería inversa. Es esta vertiente cada vez más popular en la actualidad, la que representa una solución para comprender el funcionamiento y composición interna de los productos software ya terminados, todo a partir del análisis del mismo (Canfora, Di Penta, & Cerulo, 2011).

El proceso de software surge a partir de una necesidad y termina en un producto que representa una solución; su contraparte, es decir, la ingeniería inversa inicia en la solución final construida y a través de todo el análisis que se realiza, se obtiene la necesidad inicial a partir de la cual había surgido el proceso de desarrollo del software en cuestión, es un transcurrir completamente inverso al tradicional (Chifosky & Cross, 1990).

La ingeniería inversa puede ver resumida su finalidad en 3 actividades principales: Análisis, Identificación de componentes -con sus relaciones- y representación en un nivel de abstracción alto, todo con miras a obtener nuevas soluciones a partir del análisis realizado a la arquitectura (Chifosky & Cross, 1990).

Existen muchas herramientas que facilitan y agilizan los procesos de ingeniería inversa descritos por Chifosky y Cross. Múltiples software CASE esbozan los diagramas UML de diseño para el producto analizado, algunos incluso extraen diagramas de secuencia y otros pocos están siendo amoldados para la extracción de casos de uso, todo esto representado en el nivel más bajo de abstracción posible, por ende muchas veces los resultados obtenidos a través del uso de estas herramientas tienen un nivel de complejidad alto, siendo difíciles de entender para las personas implicadas en el proceso de software (Canfora, Di Penta, & Cerulo, 2011).

La otra cara de la moneda, los resultados que se dan a un nivel de abstracción alto, son mucho más fáciles de comprender porque tienen mucha similitud con los hechos que se dan a nivel de mundo real, además de tener un bajo nivel de complejidad, pero desafortunadamente hoy día las herramientas existentes dedicadas a esta actividad son pocas y están en una dinámica continua de desarrollo y mejoras con miras a fortalecer el proceso en el que se utilizan. Si bien estas herramientas permiten la obtención de arquitecturas de software a través de procesos de ingeniería inversa, no cuentan con opciones que permitan a los usuarios realizar análisis a los resultados obtenidos –arquitectura de software-, aspecto que permitiría determinar información útil como medición de impactos a la hora de cambiar componentes del sistema, escalabilidad de un

software y tiempos de mantenimiento (Muller, Jahnke, Smith, Storey, Tilley, & Wong, 2000).

Los aspectos mencionados en el párrafo anterior permitieron el surgimiento del cuestionamiento que motivó la presente investigación: *¿Cómo realizar análisis a arquitecturas de software obtenidas como resultado de un proceso de ingeniería inversa?*

Como consecuencia de la pregunta de investigación planteada para el presente proyecto, se tiene el objetivo principal consistente en *Definir un método fundamentado en principios del álgebra relacional que permita realizar análisis a arquitecturas de sistemas software, para ser usado como herramienta de apoyo en procesos de ingeniería inversa*, de manera que se logre sentar un precedente en el campo del análisis arquitectural dentro de la ingeniería de software.

Abordar la problemática que representa el realizar análisis a arquitecturas de software constituye un gran avance en materia de investigación e innovación, debido a que la productividad de soluciones se vería agilizada al contar con una fundamentación encaminada a solventar las situaciones que actualmente limitan el proceso de ingeniería de software tales como mantenimiento de software, escalabilidad de productos software, cambios de componentes, entre otros.

Para el campo de la Ingeniería De Sistemas el presente proyecto trae beneficios conjugados en primer lugar en el campo de la productividad de software, debido a que gracias al análisis de arquitecturas el mantenimiento de una herramienta se verá simplificado siendo más fácil de realizar y disminuyendo los tiempos que se deban invertir en este punto, lo que aumentará el número de horas disponibles para ser dedicadas a nuevos desarrollos que representarían beneficios económicos para los productores de software.

En segundo lugar, la calidad del software producido aumentará puesto que se podrá verificar de una mejor manera la coherencia que exista entre el diseño y la implementación de un software, constatando que la arquitectura de software obtenida luego de realizar procesos de ingeniería inversa concuerde con la implementación a nivel de código de la

solución analizada. A su vez la seguridad de los productos desarrollados se vería potenciada puesto que gracias al análisis arquitectural es más fácil encontrar fallas de seguridad a través de los cuales el software puede presentar mal funcionamiento.

Finalmente el campo académico de la Ingeniería De Sistemas también obtiene beneficios gracias al proyecto realizado, puesto que los resultados obtenidos pueden servir como punto de partida y guía para nuevas iniciativas que se encuentren enmarcadas dentro del área de la ingeniería del software, aumentando la producción científica y el nivel de los productos construidos además de generar interés en la academia para continuar desarrollando investigaciones en el área.

Para llevar a cabo el presente proyecto se ha clasificado la investigación a partir de sus características como una *Investigación Exploratoria* y sus técnicas y postulados reafirmados gracias al *Análisis Documental* y revisión de la literatura realizada a fuentes académicas verificables y científicamente aceptadas que soportan el trabajo realizado.

Inicialmente se construyó un estado del arte que permitió conjugar los antecedentes, investigaciones realizadas y contexto actual de la temática abordada. Posteriormente se realizó un análisis al estado del arte construido, que permitió obtener y clasificar una serie de necesidades que se tienen el campo del análisis arquitectural y que se pueden resolver mediante el álgebra relacional. A partir de las necesidades encontradas se realizó la definición de un método de análisis de arquitecturas de software fundamentado en el álgebra relacional que permite solucionar los inconvenientes encontrados en el campo del análisis de arquitecturas y finalmente se realizó la validación del método propuesto, mediante la aplicación de la solución a la arquitectura de software de un sistema construido durante los estudios de Ingeniería De Sistemas del autor en la Universidad De Cartagena durante el año 2011 y que fue aceptado como solución innovadora en el marco del Congreso Nacional De Ingeniería De Sistemas en la ciudad de Barranquilla en ese mismo año, lo que soporta y valida el escenario de prueba propuesto.

2. OBJETIVOS Y ALCANCE

2.1. OBJETIVO GENERAL

- Definir un método fundamentado en principios del álgebra relacional que permita realizar análisis a arquitecturas de sistemas software, para ser usado como herramienta de apoyo en procesos de ingeniería inversa.

2.2. OBJETIVOS ESPECÍFICOS

- Realizar un estado del arte de los métodos de análisis a arquitecturas de software que actualmente existen, mediante la revisión de la literatura académica y científica disponible.

- Identificar las necesidades existentes en los procesos de análisis de arquitecturas de software a partir de la revisión a la literatura científica

- Definir un método a partir de los postulados del álgebra relacional que permita solventar las necesidades identificadas en los procesos de análisis a partir de arquitecturas de software.

- Realizar la validación de la solución propuesta aplicando el método definido a un escenario de prueba que permita determinar la pertinencia del resultado obtenido como fruto de la investigación.

2.3. ALCANCE

El resultado de la investigación ha sido un método que determina una serie de pasos que debe seguir el experto en software al momento de realizar análisis sobre arquitecturas obtenidas a partir de procesos de ingeniería inversa. El método definido permite al experto en software, realizar análisis a la arquitectura de tipo medición de impacto, pudiendo determinar componentes que se verían afectados a partir de la modificación de entidades relacionadas en la arquitectura analizada. El método también permite al experto en software realizar análisis de tipo estructural, mediante el cual se pueden determinar tipos de relaciones existentes entre entidades pertenecientes a la arquitectura. El resultado de la presente investigación no permite al experto en software realizar análisis de atributos debido a que este aspecto ya se ha abordado con anterioridad por técnicas de evaluación y análisis de arquitecturas como ATAM y QAW.

La presente investigación ha sido enmarcada en el campo de la ingeniería de software, específicamente en las vertientes conocidas como ingeniería inversa y análisis de arquitecturas, cuyos contenidos y herramientas permitieron el cumplimiento de los objetivos de la investigación. Gracias a la realización del estado del arte referente a los métodos de análisis a arquitecturas de software actualmente existentes se pudieron determinar problemáticas y necesidades que buscan ser resueltas mediante la definición del método propuesto en el objetivo general del presente proyecto, que representa el entregable principal de esta iniciativa.

El método descrito en el objetivo general comprende tres aspectos: metodología de la investigación, arquitecturas de software y al álgebra relacional. En primer lugar, la metodología de la investigación propone definiciones de métodos dependiendo de su finalidad, para el caso particular de la presente propuesta se ha optado por la definición de un método deductivo debido a que su característica principal es partir de un marco de referencia general, estableciendo parámetros de comparación que permitan analizar casos

objeto, permitiendo finalmente deducir si un objeto o un conjunto de elementos pertenecen o no a un grupo con el cual habían sido relacionados previamente.

En segundo lugar, la arquitectura de software se relaciona con el método deductivo anteriormente descrito a partir de su estructura: la arquitectura de un sistema es tomada como el marco de referencia, sus entidades se comportan como objetos de análisis y los parámetros de comparación definidos permiten determinar qué tipo de relación existe entre determinados componentes, su pertenencia a un determinado contexto –a un paquete en particular por ejemplo- e incluso, se puede realizar medición de impactos en el sistema a partir de la modificación de componentes de su arquitectura.

Finalmente para lograr materializar el método propuesto se utilizaron componentes del álgebra relacional, este aspecto constituyó la técnica para la obtención de resultados. El álgebra relacional es fácilmente aplicable a las arquitecturas de software y académicamente resulta muy útil para determinar pertenencia a grupos y relaciones entre entidades gracias a las operaciones que propone como lo son –entre otras- el producto cartesiano, proyección y selección.

Resumiendo los párrafos anteriores, el método que se propone como objetivo del presente trabajo tiene como entrada la arquitectura de un sistema software, procesa sus componentes a través de operaciones del álgebra relacional y provee como salida respuestas acerca de la arquitectura del sistema analizado.

El escenario de prueba para validar el método propuesto constó de una arquitectura de software, cuyos desarrolladores necesiten obtener resultados respecto al análisis de la arquitectura del mismo tales como relaciones entre componentes, medición de impactos y pertenencia a contextos por parte de las entidades.

3. MARCO DE REFERENCIA

3.1. CONTEXTUALIZACIÓN: ANTECEDENTES DEL ANÁLISIS ARQUITECTURAL

La ingeniería inversa es uno de los tópicos relacionados con la ingeniería de software, que en la actualidad concentra múltiples esfuerzos de la comunidad científica. Los autores Canfora, Di Penta y Cerulo realizaron un valioso aporte a la temática, que permite contextualizar de manera clara y concisa el estado actual de las necesidades existentes en el campo y trabajos futuros relacionados con la misma.

La ingeniería inversa en primer lugar, se puede contextualizar en sus diversas manifestaciones, con una sola acepción: análisis (Canfora & Di Penta, New Frontiers of Reverse Engineering, 2007). El análisis es un componente fundamental de los procesos de ingeniería inversa, a partir de una información base obtenida gracias a la arquitectura de un sistema, se realizan una serie de procedimientos que buscan la abstracción de la información obtenida y posteriormente el uso de esta como herramienta que permita abordar un sistema de forma precisa, para realizar cambios en este. El aspecto limitado que aparece en la investigación realizada por Canfora et al, es que el análisis realizado a través de ingeniería inversa no llega a la abstracción de alto nivel, sino que por el contrario solo aborda el contexto de bajo nivel, el contexto de implementación.

Para un experto en software que lleve mucho tiempo trabajando en un sistema, el entendimiento del mismo en un contexto de bajo nivel no representa mayor dificultad: el autor de un software conoce a fondo sus componentes, relaciones, estructuras internas y entidades. Sin embargo, para una persona que no lleve un tiempo prudencial trabajando en un sistema pero que deba realizar modificaciones o abordar el mismo para otras finalidades, tratar de entender la estructura de bajo nivel del mismo puede resultar extremadamente complejo (Canfora, Di Penta, & Cerulo, Achievements and Challenges in Software Reverse

Engineering, 2011), debido a que la abstracción de la información no está completa y por ende no se han separado elementos relevantes de irrelevantes para entender el "qué" del sistema analizado sin necesidad de abordar el "cómo".

Los autores Canfora et al, describen de manera muy acertada cómo los análisis de código, análisis estructurales y análisis de diseño realizados en los procedimientos de ingeniería inversa, se han constituido en una herramienta muy útil a la hora de comprender la composición de un sistema. Sin embargo, estos tipos de análisis resultan limitados en el sentido que el interesado experto en software debe ser prácticamente un experto en la solución analizada para poder abstraer información relevante de manera precisa, lo que se constituye como una problemática.

A partir del aspecto planteado en el párrafo anterior, Canfora et al, concluyen que existe la necesidad de abordar un tipo de análisis que desemboque en resultados de alto nivel, análisis que puede ser realizado a arquitecturas de software obtenidas como artefacto resultante de procesos de ingeniería inversa y que puede traer beneficios relevantes al campo de la ingeniería de software, que en últimas podría reflejarse en reducción de tiempos de entrega, aumento de calidad en los productos, aumento de ingresos, fomento de la academia y entendimiento más rápido de sistemas software.

El estudio realizado por Canfora et al, constituye el punto de partida de muchos autores para realizar sus investigaciones concernientes a análisis arquitectural, puesto que ha sido el primero en abordar la temática de forma precisa además de relacionarla con la ingeniería inversa. Sin embargo, el trabajo de Gerardo Canfora y sus colegas deja de lado un aspecto importante: en la actualidad existen metodologías de análisis arquitectural.

Las metodologías de análisis a arquitecturas de software existentes en la actualidad y aceptadas por la comunidad científica presentan una particularidad –que permite entender por qué no son abordadas en el trabajo realizado por Canfora et al-: están orientadas al estudio de atributos de calidad de la arquitectura, no al análisis de su estructura,

componentes y relaciones por lo que la información obtenida aplicando estas técnicas no admite realizar inferencias que permitan al experto en software medir impactos a partir del cambio o definir entidades implicadas a partir de relaciones. Aunque estas metodologías de análisis no representan importancia relevante para el cumplimiento de los objetivos de la presente investigación, se mencionan a continuación, puesto que en un futuro pueden trabajar de la mano con los aportes obtenidos en el presente trabajo.

3.1.1 SAAM, ATAM y QAW: análisis de arquitecturas para evaluar calidad.

Los métodos de análisis a arquitecturas de software no son una novedad. A la hora de determinar la arquitectura de un sistema muchos factores deben ser analizados, propiedades tales como escalabilidad, rehúso, nivel de abstracción, entre otras permiten muchas veces obtener soluciones de mucha mayor calidad y funcionalidad que las construidas sobre la marcha y sin la debida planeación (Gorton, 2006).

En primer lugar, apareció el método de análisis de arquitecturas de software, cuyas siglas en inglés traducen SAAM. El método fue originalmente creado para el análisis de la modificabilidad de una arquitectura, pero en la práctica ha demostrado ser muy útil para evaluar distintos factores de calidad de un sistema, tales como la portabilidad, escalabilidad y sobre todo el grado de modificabilidad del mismo (Kazman, Bass, Abowd, & Webb, 2007). Este método se enfoca principalmente en la enumeración de un conjunto de escenarios que le permiten intuir al experto en software los posibles cambios que el sistema tendrá en un futuro.

El método SAAM presenta un inconveniente: para obtener resultados mediante su aplicación, es estrictamente necesaria una forma de descripción de la arquitectura a analizar, descripción que muchas veces es desconocida y solamente puede ser obtenida a través del arquitecto de software original de la solución (Kazman, Clements, & Klain, 2005); sin embargo, los resultados de la aplicación del método son bastante útiles, pues permiten deducir los lugares en los que la arquitectura puede fallar o, si se analizan varias

arquitecturas, se obtiene una clasificación que permite observar cuál de las opciones satisface mejor los requerimientos con menor cantidad de modificaciones.

Posteriormente, hace su aparición el método ATAM. Este tipo de análisis, cuyas siglas traducen *"Método de Análisis de Puntos de Sensibilidad de Arquitectura"*, fue desarrollado e impulsado por el Instituto de Ingeniería de Software (SEI) y centra su actividad de análisis y evaluación en la interacción entre los diferentes atributos de calidad arquitectónica y basa sus evaluaciones sobre los escenarios de prueba y un equipo de evaluadores que constantemente vela por encontrar fallas de calidad (Kazman, Klein, & Clements, ATAM: Method for Architecture Evaluation, 2000).

La finalidad del método ATAM es, a partir del análisis realizado a una arquitectura de software y mediante el análisis de los objetivos del modelo de negocio y requerimientos, presentar puntos de sensibilidad en los cuales la arquitectura propuesta podría verse afectada en los diferentes escenarios de prueba para los cuales ha sido diseñada. A su vez, también permite definir la hoja de ruta a seguir para mejorar los riesgos encontrados y de esta manera garantizar que los atributos de calidad como modificabilidad, escalabilidad y usabilidad de la arquitectura estén presentes en la solución construida (Kazman, Clements, & Klain, 2005).

La principal dificultad del método ATAM radica en que los resultados arrojados por el mismo, muchas veces provienen de la intuición de las personas implicadas en el proceso de análisis de la arquitectura, los cuales en la mayoría de los casos no son *stakeholders* del proyecto evaluado, lo que genera que en ciertas ocasiones los resultados obtenidos no sean compartidos por los clientes. Para mitigar esta problemática nace el método QAW como complemento al método ATAM.

QAW –Quality Attribute Workshops-, es un método de análisis que permite identificar atributos de calidad críticos en la arquitectura evaluada, con miras a realizar modificaciones en la misma, mitigando los impactos que puedan tener dichos cambios en el futuro

necesarios para garantizar calidad (Barbacci, Ellison, Lattanze, Stafford, Weinstock, & Wood, 2003). La particularidad que presenta este método es que permite la interacción directa de los *stakeholders* de un proyecto, junto a los integrantes del equipo que realiza el análisis arquitectural, permitiendo de esta manera que los resultados obtenidos sean compartidos y conocidos por todos los implicados en el proceso. Resulta sencillo aplicar la metodología QAW pues se basa en las opiniones que puedan provenir de todos los participantes, gracias a que una de sus actividades es el *brain storming*-lluvia de ideas-.

SAAM, ATAM y QAW constituyen poderosas herramientas de análisis arquitectural, los resultados detallados y precisos que se obtienen al aplicar dichas metodologías sobre arquitecturas de software permiten incrementar los factores de calidad del producto; sin embargo, la desventaja que presentan estos tres métodos radica en que no solo la calidad de la arquitectura es el interés de un experto en software y es este aspecto el centro fundamental de estas metodologías (Störmer, 2007).

Debido a esta situación, los expertos en el área han optado por construir sus propios enfoques, determinados por las necesidades que deseen solventar mediante el análisis arquitectural. Trabajos desde el punto de vista visual, matemático e incluso experimental han sido realizados por diferentes investigadores. A continuación se presenta un recorrido por estos, sus aportes y sus retos a futuro.

3.2. ESTADO DEL ARTE

La arquitectura de un sistema, representa una herramienta bastante útil a la hora de comprender la estructura y elementos que intervienen en un software (Booch, Rumbaugh, & Jacobson, 1999). Sus relaciones entre componentes, representación de entidades y demás artefactos que la componen, facilitan el entendimiento de la disposición de un sistema en un nivel de abstracción alto, es decir, separando "qué" hace el producto de "cómo" lo hace, lo que permite a las personas expertas en software que en un momento dado se encuentran

analizando un determinado sistema, realizar una interpretación precisa y concreta de los datos obtenidos a partir del análisis realizado a una arquitectura de software (Larman, 2003).

El análisis realizado a una arquitectura de software puede ser de muchos tipos y tener diferentes finalidades, todo depende de la necesidad o necesidades que tenga el experto en software para con un determinado sistema (Gorton, 2006). Para alcanzar de manera clara y concisa los objetivos de la presente investigación, se hace necesario definir el uso y alcance del término **análisis** dentro de su relación con las arquitecturas de software.

Para la Real Academia de la Lengua Española –RAE-, la palabra análisis comprende *"la distinción y separación de las partes de un todo hasta llegar a conocer sus principios o elementos"* (Real Academia De La Lengua Española - RAE). Llevada esta definición al contexto de las arquitecturas de software, su similitud con el proceso de análisis que normalmente realiza un experto en software a la arquitectura de un sistema resulta interesante puesto que la finalidad de realizar análisis arquitectural es comprender la estructura del software analizado, la disposición de sus elementos y de esta forma, a partir de la interpretación de estos componentes, entender el sistema analizado como una totalidad (Booch, Rumbaugh, & Jacobson, 1999).

A partir de la definición de análisis de la RAE descrita en el párrafo anterior, se puede aterrizar el alcance de qué comprenderá el término análisis para la presente investigación. Los elementos de la arquitectura de un determinado software, permiten el entendimiento de un sistema a partir de las entidades que lo componen. Estas entidades se encuentran correspondidas entre sí a partir de relaciones, que a su vez, permiten determinar **medición de impacto** gracias a que un cambio realizado en una relación afecta directamente a los componentes implicados con esta. Se define entonces análisis, para el presente trabajo, como el proceso que realiza una persona experta en software, consistente en abordar una arquitectura de software desde el punto de vista estructural teniendo en cuenta los componentes de la misma y las relaciones entre estos, para de esta manera determinar qué

elementos del sistema pueden verse afectados si se realizan cambios a determinadas entidades –que pueden ser relaciones-, permitiendo al interesado tomar cursos de acción para realizar modificaciones, cambios e incluso añadir nuevos componentes al sistema estudiado, permitiendo obtener como beneficio el aumento de la calidad del producto y consecuentemente, el aumento en la calidad de todos los factores que lo componen.

Es importante añadir que las arquitecturas de software pueden ser obtenidas o no a partir de ingeniería inversa (Chifosky & Cross, 1990). Aunque el hecho de que hayan sido o no obtenidas a partir de esta técnica no represente mayores dificultades o aspectos relevantes para el presente trabajo, es importante abordar la temática desde el punto de vista de ingeniería inversa, puesto que es la temática base de la fundamentación teórica de la investigación realizada.

El contexto de análisis propuesto, ha sido abordado por diferentes autores pertenecientes a la comunidad científica y académica, cuyos aportes permiten fundamentar de manera sólida los objetivos de esta investigación. A continuación, se presenta un recorrido por estos.

3.2.1 MÉTODOS DE ANÁLISIS ARQUITECTURAL

3.2.1.1. Enfoque visual: una imagen dice más que mil palabras

El análisis gráfico se constituye como una herramienta bastante didáctica en cualquiera de los campos que sea aplicado, interpretar gráficas y bosquejos permite a una persona sintetizar rápidamente ideas y generar resultados. El mundo del análisis de arquitecturas de software no es la excepción y existe un trabajo en particular que así lo demuestra.

Al obtener arquitecturas de software generadas a través de ingeniería inversa, muchas veces el modelo inicial difiere del obtenido a partir del producto, situación confusa que genera problemáticas al experto en software (Canfora & Di Penta, 2007), por lo que el análisis

visual de las mismas se hace necesario. Los investigadores alemanes Fabian Beck y Stephan Diehl, abordaron esta necesidad y realizaron un detallado estudio de algoritmos de comparación de imágenes que permitieron desarrollar una metodología de análisis arquitectural materializada en una herramienta software que facilita el proceso.

En primer lugar, mediante los artefactos generados durante el proceso de software –diagramas por ejemplo-, se determinan las relaciones que existen entre las entidades de la arquitectura y se realiza una base de conocimiento de las mismas, que permite generar un gráfico que se comparará con el diagrama obtenido a través de las herramientas de ingeniería inversa utilizadas para obtener la arquitectura del sistema. Este tipo de análisis permite obtener puntos de equilibrio y de diferencia entre las arquitecturas comparadas, permitiendo al experto en software determinar puntos críticos de la arquitectura y realizar cambios en la misma –y en la implementación del sistema como tal- con la debida seguridad del caso (Beck & Diehl, 2010). En la ilustración 1 se muestran los resultados arrojados por la herramienta que implementa la solución de Beck y Diehl.

El método propuesto por Beck y Diehl es bastante útil cuando se disponen de las herramientas necesarias para llevarlo a cabo: ordenadores gráficamente poderosos, implementación de los algoritmos propuestos en una herramienta software y un equipo experto en análisis visual que se encargue de estudiar los resultados obtenidos. No todos los laboratorios de investigación tienen acceso a dichos requerimientos, por lo tanto el uso de esta propuesta se ha visto limitado.

El trabajo futuro propuesto por Beck y Diehl radica en la necesidad de estandarizar los métodos visuales de análisis arquitectural que existan, con miras a desarrollar una poderosa herramienta que pueda ser usada en cualquier rincón del mundo; sin embargo, este punto –afirman los autores- no sería posible si antes no existe un método de análisis a arquitecturas de software debidamente estandarizado y aceptado por la comunidad académica y científica del campo de la ingeniería de software.

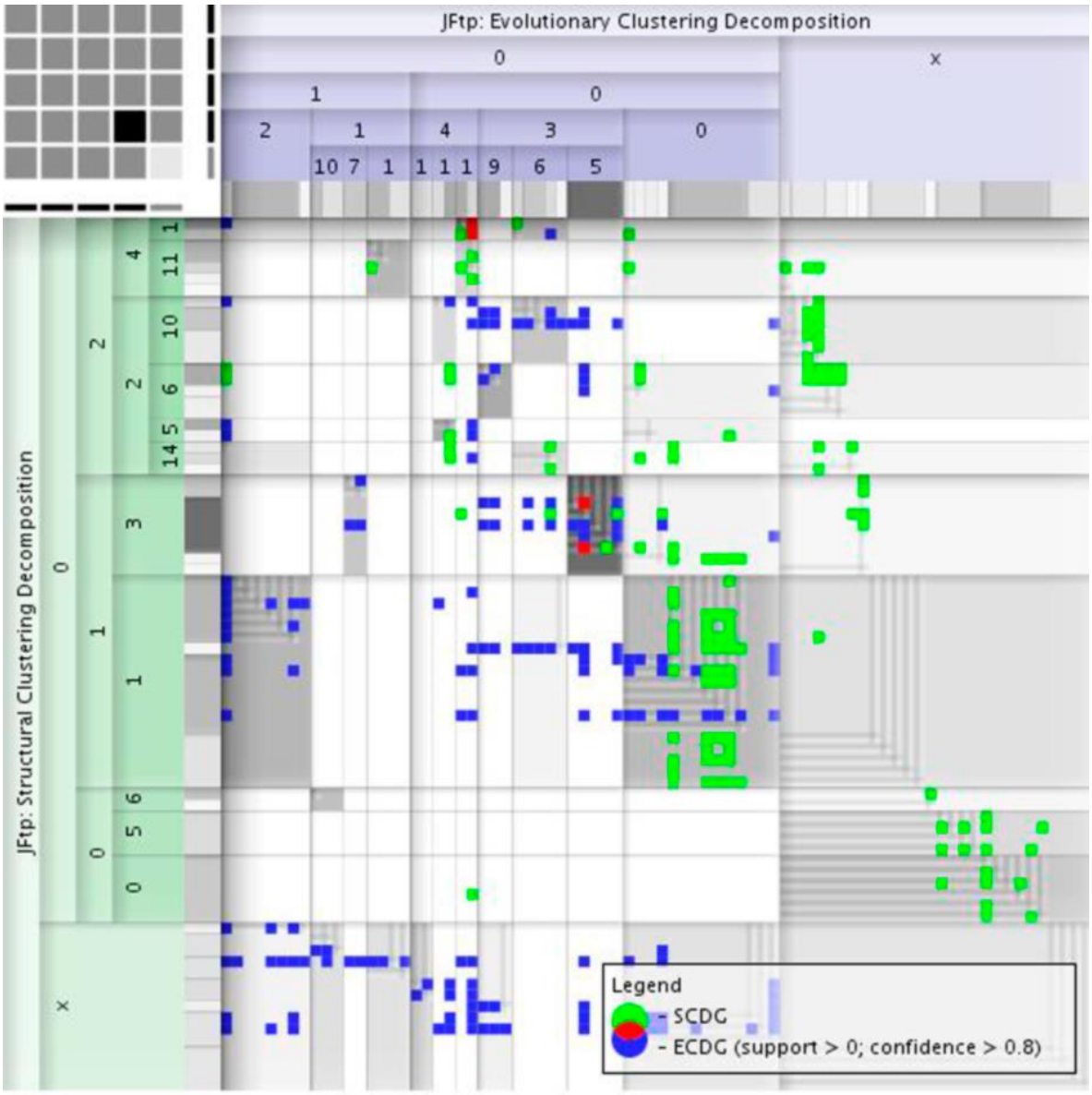

Ilustración 1. Análisis visual para arquitecturas de software generado a partir del método propuesto por Beck y Diehl (Beck & Diehl, 2010)

El trabajo de Beck y Diehl resalta dos aspectos importantes: el análisis arquitectural como herramienta de trabajo en el proceso de software y –más importante aún- la recuperación de arquitecturas a través de procesos de ingeniería inversa. Este último aspecto es el que ha concentrado la mayor parte de los esfuerzos científicos en el área, con miras a concretar

métodos de análisis arquitectural que permitan obtener resultados útiles a los expertos en software a la hora de trabajar sobre la arquitectura de software de un determinado sistema.

3.2.1.2. Recuperación de arquitecturas a través de ingeniería inversa.

La recuperación de arquitecturas de software con miras a determinar métodos de análisis de las mismas se ha convertido en una tendencia en la actualidad, dentro del campo de métodos de análisis arquitectural (Störmer, 2007). La dinámica con la que cambian los sistemas software es bastante acelerada, por lo que la arquitectura de un sistema sufre constantes cambios que impiden obtener un modelo que pueda ser sometido a análisis, pues no se tiene la certeza de que vaya a mantenerse intacto durante el ciclo de vida del software construido (Acher, Cleve, Collet, Merle, Duchien, & Lahire, 2011).

Debido a la necesidad mostrada en el párrafo anterior, se han propuesto métodos de análisis arquitectural, derivados de la recuperación de arquitecturas. Unos han sido enfocados a sistemas software tradicionales –aplicaciones de escritorio por ejemplo- y otros, a sistemas que operan en la web –cuya dinámica de cambios es acelerada-.

En primer lugar, al recuperar arquitecturas se da un paso fundamental al entendimiento del sistema analizado (Panas, Löwe, & Assmann, 2003), el objetivo del experto en software que trabaja con métodos de análisis arquitectural es comprender el funcionamiento y estructura de un sistema a partir de la abstracción de alto nivel que pueda ser capaz de realizar mediante del estudio de la arquitectura de software (Ducasse & Pollet, 2009). Para lograr estos postulados, los investigadores Panas et al realizaron un detallado estudio de los procedimientos utilizados a la hora de recuperar arquitecturas de software, como se muestra en la ilustración 2.

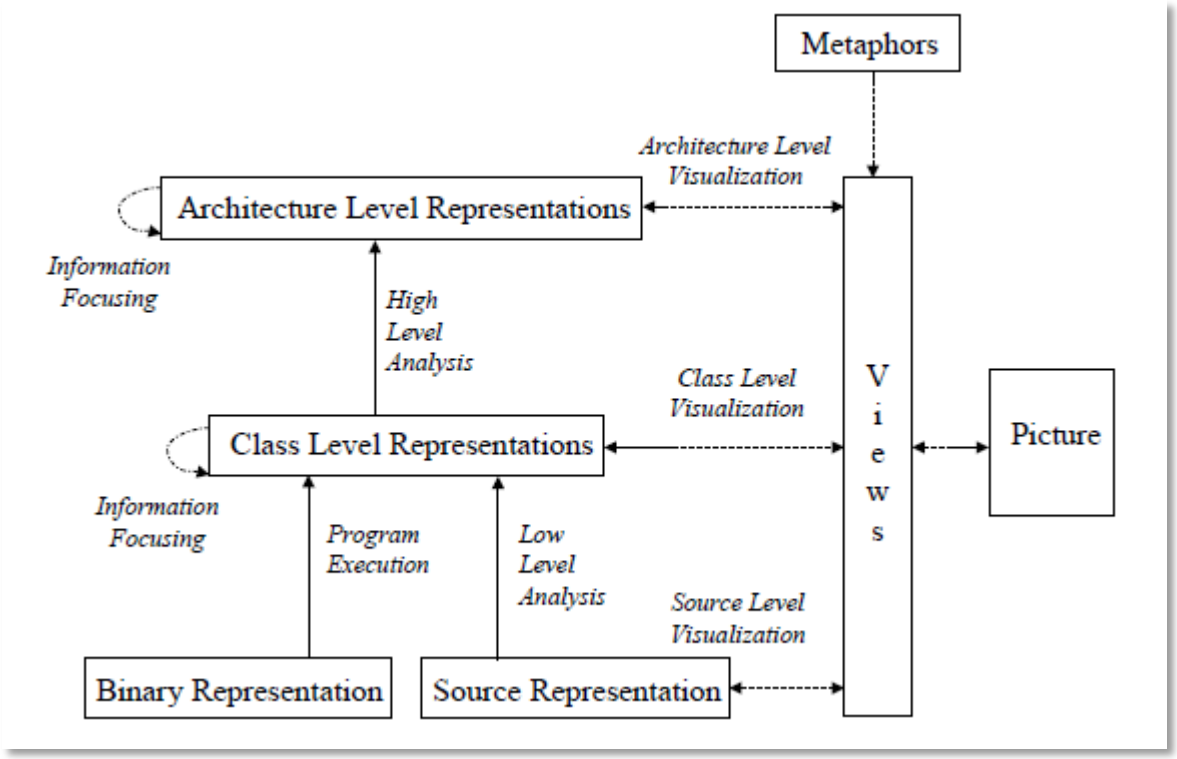

Ilustración 2. Diagrama de flujo necesario para recuperar arquitecturas de software propuesto por Panas et al (Panas, Löwe, & Assmann, 2003)

La propuesta de Panas et al, deja como principal beneficio la clasificación de todos los artefactos generados en un proceso de recuperación arquitectural que pueden ser sometidos a análisis, puesto que dichos artefactos son en últimas representaciones diversas de la arquitectura recuperada. Sin embargo, el método de análisis no está incluido en el alcance de la investigación de Panas et al y se propone como trabajo futuro, la estandarización de un método que permita realizar análisis a las arquitecturas recuperadas y que pueda trabajar de la mano con los métodos de análisis de atributos existentes –SAAM, ATAM, QAW- (Panas, Löwe, & Assmann, 2003).

En este punto es notoria la similitud de la necesidad derivada del trabajo de Panas et al y del trabajo realizado por Beck y Diehl: la búsqueda de un método de análisis arquitectural.

La recuperación de arquitecturas y análisis arquitectural también –como ya se ha mencionado con anterioridad- tiene presencia en el campo de las herramientas cuyo entorno natural de ejecución es la web. Las aplicaciones web son el legado de las aplicaciones del futuro, se desarrollan bajo estrictos periodos de tiempo, con numerosos equipos de desarrolladores y con tendencias al cambio, por lo que las arquitecturas de este tipo de soluciones y su análisis brindan herramientas bastante útiles a los expertos en software para el entendimiento de los sistemas estudiados (Hassan & Holt, 2002).

Para que el análisis arquitectural de aplicaciones web sea efectivo, se necesitan herramientas que en su mayoría no han sido implementadas aún, porque la fundamentación para desarrollarlas no ha sido estandarizada y se plantea como reto a futuro en el trabajo de Hassan et al. Estos autores continúan con la tendencia de los trabajos referenciados con anterioridad en este documento: la necesidad de un método de análisis que permita estandarizar los procesos de recuperación de arquitecturas e implementación de herramientas que faciliten el trabajo a los expertos en software.

3.2.1.3. Enfoques matemáticos de análisis arquitectural.

La necesidad de un método de análisis arquitectural se hace recurrente al analizar los trabajos futuros de los múltiples referentes científicos consultados, tanto así, que incluso se han realizado experimentos con personas para justificar dicha problemática.

El trabajo realizado por Werner Heijstek et al (Heijstek, Kühne, & Chaudron, 2011), toma a un grupo significativo de personas expertos en el área de la ingeniería de software y les pide realizar análisis arquitectural a un sistema con ayuda de artefactos como diagramas UML. Los resultados fueron demasiado tan entre sí, incluso aplicando los mismos métodos de análisis –como QAW por ejemplo- que no existió consenso en la información obtenida.

El trabajo referenciado en el párrafo anterior sirve para demostrar de manera clara y concisa el por qué se necesitan métodos y enfoques de análisis arquitectural que permitan estandarizar los procesos de recuperación de arquitecturas y de esta manera obtener resultados puntuales. Para mitigar esta necesidad, se han hecho aproximaciones a métodos de análisis arquitectural desde el punto de vista matemático, utilizando teoría de conjuntos, análisis de capas e incluso teoría de análisis relacional.

La teoría de conjuntos proporciona conocimientos útiles a la ingeniería de software, que pueden ser aplicados en el campo del análisis arquitectural. Una arquitectura de software puede ser representada como un conjunto de elementos –entidades- que tienen aspectos y propiedades en común y que pueden ser manipuladas y operadas como si fueran elementos matemáticos definidos en un conjunto (Acher, Cleve, Collet, Merle, Duchien, & Lahire, 2011). Gracias a este enfoque, se facilita el trabajo a la hora de recuperar y analizar arquitecturas de software de sistemas que tienen numerosos componentes, pues el manejo matemático proporciona ecuaciones que permiten realizar una representación que permite someter a los distintos componentes de una arquitectura de software a un manejo determinado.

El trabajo de Acher et al puede resultar complejo para los expertos en software que no tengan conocimientos matemáticos sólidos en el campo de la teoría de conjuntos, debido a que este campo de la matemática es bastante complejo y además utiliza la lógica de predicados como se puede observar en la ilustración 3, por ello en la sección de trabajos futuros de su estudio, plantean la necesidad de que los investigadores del campo de la arquitectura de software y de la ingeniería de software en general propongan sus propios métodos de análisis arquitectural matemáticamente fundamentados para de esa forma facilitar su implementación en herramientas de software.

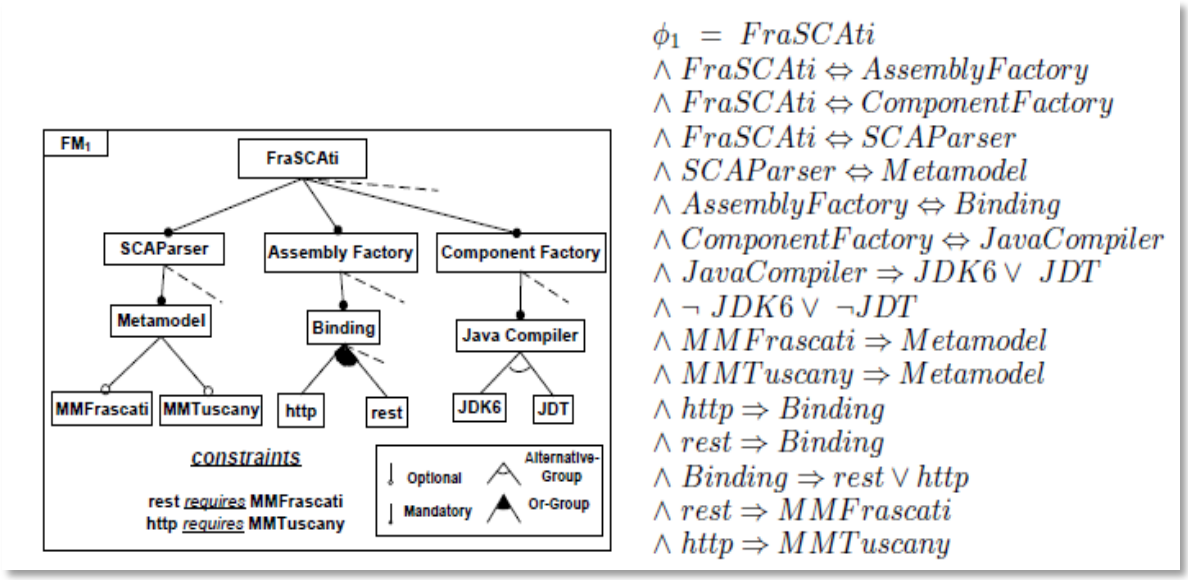

Ilustración 3. Representación arquitectural utilizando lógica de predicados y teoría de conjuntos (Acher, Cleve, Collet, Merle, Duchien, & Lahire, 2011)

Por otro lado, existe el enfoque de métodos de análisis arquitectural desde el punto de vista de capas. Uno de los trabajos que presenta mayor afinidad con la presente investigación es el que ha sido realizado por el departamento de las ciencias de la computación de la universidad de Changsha – China, que centra su interés en realizar análisis a arquitecturas de software desde el punto de vista de capas (Lijun et al, 2012). El análisis desde el punto de vista de capas resulta interesante debido a que se centra en las relaciones de dependencia entre componentes, que representadas una debajo de la otra en función de qué entidades acceden a determinados servicios ubicados en otra capa, pueden representarse de manera jerárquica siendo las capas que están en los niveles más superiores abstracciones de las capas que se encuentran en los niveles inferiores, siendo estas últimas la implementación de sus antecesoras a nivel jerárquico, todo esto sin apartarse del concepto de arquitectura. En la ilustración 4 se puede ver el concepto de manera más clara.

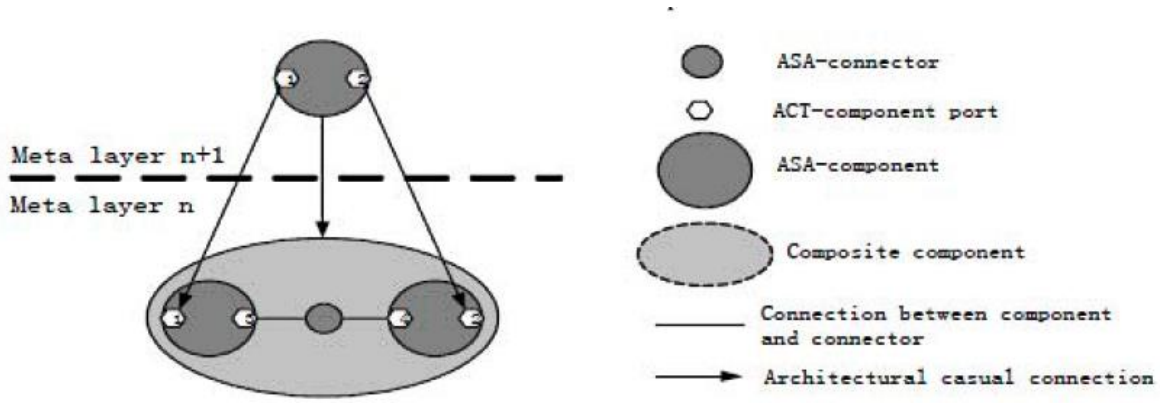

Ilustración 4. Análisis arquitectural desde el punto de vista de capas. (Lijun, Changyun, Gexin, & Zhibing, 2012)

El análisis de capas provee herramientas que pueden ser aplicadas en diseños mejorados y optimizados para nuevas implementaciones de sistemas software que deseen realizarse a futuro, eliminando dependencias innecesarias o recurrentes que reflejarían beneficios palpables en mantenimiento de software y escalabilidad de los mismos; sin embargo, el trabajo realizado por Lijun et al solo se concentra en realizar análisis partiendo de resultados obtenidos de manera gráfica, aspecto que no sería posible realizar en todos los contextos de recuperación y análisis de arquitecturas por diversos motivos como –entre otros- ausencia de herramientas CASE adecuadas y complejidad del sistema a analizar.

3.2.1.4. Análisis relacional

El análisis relacional –que también hace parte del enfoque matemático-, ha sido uno de los más trabajados en el campo de análisis de arquitecturas de software (Podgurski, 2006), debido a que a diferencia de la teoría de conjuntos, toma como referente principal las relaciones entre entidades de la arquitectura y permite la manipulación de los elementos a partir de dichas relaciones. El experto en software posee conocimientos avanzados de las

relaciones entre los componentes de una arquitectura, puesto que es uno de los tópicos de la ingeniería de software (Pressman, 2009).

El principal objetivo del análisis relacional es obtener resultados concentrados únicamente en la abstracción de alto nivel de la arquitectura, apoyándose en el uso de los artefactos generados en el proceso de software (Jarzabek & Woon, 2007). El análisis relacional generalmente utiliza conceptos del álgebra relacional –aunque no es un aspecto obligatorio- y queda a definición del autor utilizar otros componentes de la matemática o no.

Uno de los trabajos de análisis relacional más conocidos a nivel mundial es el trabajo realizado por Zude Li (Li et al, 2010), el cual propone un álgebra relacional extendida para análisis de sistemas software a partir de las relaciones de sus componentes. Se proponen relaciones de tipo "dependencia" y relaciones de tipo "jerárquica" como se muestra en la ilustración 5, donde las relaciones dependencia se simbolizan con la letra *c* y las relaciones jerárquicas con la letra *p*, a su vez las entidades que componen la arquitectura con las letras *c* y *f*.

El enfoque matemático propuesto en el trabajo realizado por Li et al, representa una abstracción bastante útil, debido a que a partir de las relaciones se podrían definir tablas relacionales -de forma similar a lo que manejan las bases de datos relacionales- que permitan realizar consultas y medir impactos a la hora de agregar nuevos componentes a un sistema de software implementado o a la hora de modificar los mismos, a partir de las relaciones establecidas entre cada una de sus entidades. A su vez también se podrían crear bases de datos relacionales a partir de los componentes del sistema y definir métodos de consulta que trabajarían sobre estas, representando una herramienta bastante útil para realizar análisis a sistemas software.

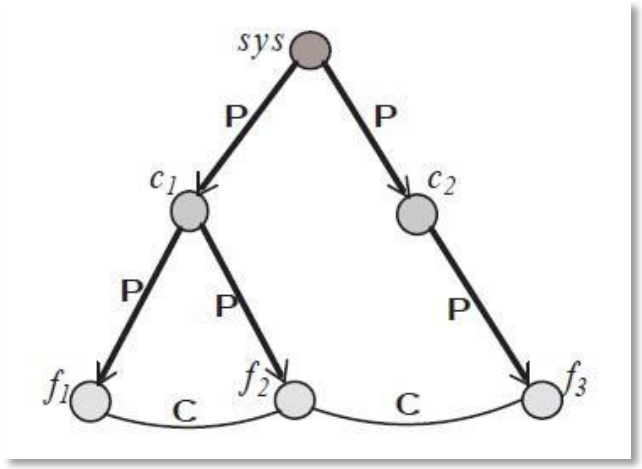

Ilustración 5. Modelo de arquitectura de software desde el punto de vista de grafos y álgebra relacional (Li et al, 2010)

El trabajo de Zude Li toma como punto de partida los postulados realizados por Podgurski y Clarke (Podgurski, 2006) que hace hincapié en las dependencias entre entidades que componen sistemas software y las implicaciones que dichas relaciones pueden tener a la hora de realizar pruebas y mantenimiento a los sistemas. También se menciona el trabajo realizado por Kazmann y Bass (Kazman & Bass, 2002), que se enfoca en los aspectos de las relaciones entre entidades en una arquitectura de software pero desde el punto de vista de mundo real; sin embargo, el trabajo de Li et al se ha enfocado principalmente en soluciones a nivel de diseño e implementación, dejando de lado la aplicación de los principios propuestos a abstracción de alto nivel en sistemas software, es decir, a su arquitectura. Este punto resulta de vital importancia en la realización del presente trabajo puesto que es la principal necesidad que se intenta solventar con el mismo y gracias al análisis realizado en la literatura pertinente se evidencia que, aunque existen bases teóricas y conocimiento científico adecuado, la problemática aún no es abordada en su totalidad o simplemente es mencionada como trabajo futuro en las conclusiones de muchos de los autores consultados (Li et al, 2010).

3.2.2 NUEVOS ENFOQUES Y TENDENCIAS ACTUALES DE ANÁLISIS ARQUITECTURAL

El análisis arquitectural no solo se ha centrado en trabajar sobre las entidades abstractas de un sistema software. En la actualidad la tendencia apunta hacia el trabajo incluso con los elementos físicos que hacen parte del entorno natural de ejecución de un determinado software, puesto que el hardware también hace parte de la solución (Moreland, King, Maynard, & Ma, 2013).

Moreland et al también hacen referencia a las arquitecturas de las aplicaciones desarrolladas para dispositivos móviles como tendencia de análisis arquitectural en la actualidad, aunque no existan muchos avances en el campo puesto que para cada sistema operativo móvil existen ciertas características particulares en la arquitectura de sus aplicaciones.

Las principal opción a futuro que destaca en el mundo del análisis arquitectural y los métodos referenciados a lo largo de esta sección, es la posibilidad de implementar un lenguaje de consultas estandarizado que permita literalmente *"hacer preguntas"* a la arquitectura estudiada, para de esta manera obtener información más puntual acerca del tópico que el experto en software requiera conocer, relacionado con la arquitectura del sistema estudiado. Uno de los avances más conocidos en este tema en particular, es el lenguaje de consultas GrokQL, que ha sido creado a partir de extensas líneas de lógica de predicados y que permite realizar consultas a una arquitectura que previamente haya sido transformada a lenguaje de predicados (Wu & Winter, 2002). El trabajo futuro de esta investigación propone la estandarización de un método de análisis para que los expertos en software no tengan obligación de convertir una arquitectura de software a un lenguaje de lógica de predicados.

A nivel regional y local, los estudios realizados se han enfocado principalmente en caracterizar y catalogar las herramientas CASE usadas en procesos de ingeniería inversa, el

alcance de las mismas, sus ventajas y sus deficiencias, como es el caso del artículo publicado por Martín Monroy (Monroy, Arciniegas, & Rodríguez, 2012) denominado *"Caracterización de herramientas de ingeniería inversa"* en el cual se hace un estudio bastante completo de un conjunto de herramientas de ingeniería inversa a partir de funciones, aspectos y características de las mismas, enfoques que resultan útiles para añadir valor agregado a las nuevas herramientas que pretenden salir al mercado y también para mejorar las ya existentes. En los resultados publicados en el trabajo realizado por Monroy et al, se muestran aspectos fundamentales con los que deben cumplir las herramientas relacionadas con la parte de diseño y se llega a la conclusión de que prácticamente ninguna de las herramientas consultadas ofrece alternativas de utilidad para trabajar abstracción de alto nivel con sistemas software, lo que se plantea como un nuevo reto a tener en cuenta para quienes se dedican a construir herramientas CASE. Cabe anotar que la información publicada en el trabajo mencionado hace parte de los resultados parciales de la tesis doctoral del mismo autor, denominada: *"Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento"*.

El análisis arquitectural representa entonces uno de los principales retos no solo en la ingeniería inversa sino en todo el compendio enmarcado por la ingeniería de software, debido a que actualmente aunque hay avances y trabajos -además de interés-, todavía quedan muchas temáticas por abordar con miras a aportar nuevas alternativas e información que finalmente permitan materializar una solución definitiva a la problemática expuesta.

3.3. MARCO TEÓRICO

3.3.1. El método

Método es una palabra que proviene del término griego *methodos* -"camino" o "vía"- y que se refiere al medio utilizado para llegar a un fin. Su significado original señala el camino que conduce a un lugar (Sampieri Hernández, Fernández, & Baptista, 1996). La palabra método puede referirse a diversos conceptos, por ejemplo, a los métodos de clasificación científica. Esta es la disciplina que permite a los biólogos agrupar y separar en categorías a los diversos organismos y conjuntos. El método científico, por su parte, es la serie de pasos que sigue una ciencia para obtener saberes válidos -es decir, que pueden verificarse a través de un instrumento fiable), gracias al respeto por un método científico, un investigador logra apartar su subjetividad y obtiene resultados más cercanos a la objetividad o a lo empírico.

Según el filósofo inglés Francis Bacon (Achinstein, 2004), las distintas etapas del método científico son la observación -que permite analizar un fenómeno según se aparece ante la realidad-; la inducción -para distinguir los principios particulares de cada una de las situaciones observadas-; la hipótesis -la planteada a partir de la observación y de acuerdo a ciertos criterios-; la prueba de la hipótesis mediante la experimentación; la demostración o refutación de la hipótesis; y el establecimiento de la tesis o teoría científica -las conclusiones-

Para definir un método se necesita primero definir su tipo, en función del objetivo requerido, basando esta decisión en los diferentes tipos de métodos existentes y además definir una técnica para alcanzar los objetivos que plantea dicho método (Sampieri Hernández, Fernández, & Baptista, 1996).

Existen métodos que utilizan la lógica -estudio de procedimientos teóricos y prácticos con una explicación racional- para alcanzar el conocimiento. Dichos métodos son la deducción, la inducción, el análisis y la síntesis (López Cano, 1984). Según la deducción, a partir de un

marco de referencia general, se establecen parámetros de comparación que permitan analizar un caso objeto, se trata de descubrir si un elemento dado forma parte o no de un grupo al que se lo había relacionado previamente. Se encuentra también la inducción, cuyo objetivo es conseguir generalizar el conocimiento sobre un tema para prevenir consecuencias que pudieran afectar en el futuro; por otro lado está el análisis que se basa en separar las partes de un todo para conseguir analizar todo por separado y lograr un conocimiento más detallado de cada parte. Finalmente aparece la síntesis, a partir de la cual se reúnen bajo criterios racionales varios elementos que se hallaban dispersos para crear una nueva totalidad.

Ningún método está completamente definido sin una técnica (Ritchey, 1991). La técnica consiste en las acciones precisas para llevar a cabo un método. Un ejemplo donde se entiende claramente este concepto es en el plano deportivo. Todos los tenistas poseen una técnica -revés, servicio, forma de colocar los pies o sostener la raqueta, etc.-, se trata de una habilidad natural o conseguida a partir de un arduo trabajo y que se utiliza en función de un método -fatigar al adversario, jugar desde el fondo o pegado a la red, etc.-. En pocas palabras, en el método se organizan y estructuran las técnicas concretas que servirán para conseguir un objetivo determinado, en el caso del tenis, ganar el partido.

3.3.2. Arquitectura de software

La arquitectura de software se ha constituido desde su aparición en un pilar fundamental para eliminar la concepción de que el desarrollo de software era algo más parecido a un arte que a una ciencia fundamentada. Esto se debe en gran parte a que con las soluciones que plantea, permite entender de una manera muchísimo más rápida y amigable las estructuras internas y composición de los sistemas software que se requieran (Larman, 2003).

Principalmente la arquitectura de software se utiliza para representar el nivel más alto de abstracción en un sistema software, esto es, representar sus procesos y componentes a un

nivel entendible para la persona debido a su similitud con los procesos que se dan a nivel de mundo real.

Formalmente, Boehm et al han realizado una definición de arquitectura de software:

> *"Una arquitectura es el conjunto de decisiones significativas sobre la organización del sistema software, la selección de los elementos estructurales y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplios, y el estilo de arquitectura que guía esta organización -estos elementos y sus interfaces, sus colaboraciones, y su composición".*

El tema fundamental con las arquitecturas de software es la parte relacionada con las ideas y visión a gran escala. Se puede comparar de manera análoga a un proyecto de construcción en donde los arquitectos se encargan de diseñar y visionar una solución a nivel macro y de abstracción alta, que luego se somete a procesos de implementación por parte de los obreros. Formalmente el arquitecto propone varias soluciones y no se comporta de manera estática, esta misma situación se repite en el proceso de software debido a que no solo se debe considerar la arquitectura de un sistema como una representación meramente estructural de las entidades que conforman el sistema, sino también de un proceso investigativo que implica proponer soluciones y estar en constante búsqueda del mejoramiento de las mismas, en el desarrollo de software la arquitectura se considera tanto un nombre como un verbo (Larman, 2003).

3.3.3. Análisis de arquitecturas

Existen métodos que permiten realizar análisis a arquitecturas de software. Dichos métodos ofrecen un marco de referencia que permite evaluar diversos aspectos de la arquitectura

analizada permitiendo determinar si la arquitectura diseñada es pertinente con la solución requerida, si los cambios que sufrirá en el futuro afectarán el funcionamiento del software e incluso determinar costes por implementar una determinada arquitectura (Kazman, Clements, & Klain, Evaluating Software Architectures, Methods and Case Studies, 2005).

Los principales métodos de análisis a arquitecturas de software que hoy en día existen son los siguientes:

- ***Architecture Tradeoff Analysis Method (ATAM)***

En ingeniería de software, ATAM es un proceso de mitigación de riesgo. ATAM fue desarrollado por el Instituto de ingeniería del Software en la Universidad Carnegie Mellon. Su propósito es ayudar a elegir una arquitectura adecuada para un sistema de software mediante el descubrimiento de los compromisos y puntos de sensibilidad del mismo, obtenidos a partir del análisis de la arquitectura. El método ATAM es más beneficioso cuando se realiza temprano en el desarrollo del ciclo de vida del software, cuando el costo de cambiar componentes de la arquitectura es mínimo (SEI, 2013).

- ***Quality Attribute Workshops (QAW)***

Proporciona un método para analizar e identificar en la arquitectura de un sistema atributos críticos de calidad, tales como disponibilidad, rendimiento, seguridad, interoperabilidad y modificabilidad, que se derivan de la misión o los objetivos del modelo de negocio. QAW no asume la existencia de una arquitectura de software. Ha sido desarrollado para complementar el método ATAM en respuesta a las solicitudes de los clientes por un método que sirviera para identificar los atributos de calidad importantes y clarificar los requisitos del sistema antes de que haya una arquitectura de software a la cual se podría aplicar ATAM (Barbacci, Ellison, Lattanze, Stafford, Weinstock, & Wood, 2003).

- ***Software Architecture Analysis Method (SAAM)***

Es un método utilizado en la arquitectura de software para analizar y evaluar la arquitectura del sistema. Fue el primer método documentado para realizar análisis a arquitecturas de software, y fue desarrollado a mediados del año 1990 para analizar y medir el impacto de los cambios que se iban a realizar a un sistema, pero es útil para analizar cualquier aspecto funcional de una arquitectura de software (Kazman, Bass, Abowd, & Webb, 2007).

Además de los métodos mencionados anteriormente, existen múltiples métodos propuestos por diferentes autores, que han sido definidos para atender necesidades particulares de sistemas determinados, basados en distintos campos del conocimiento. En este apartado podemos encontrar, entre muchos otros, los siguientes tipos de análisis brevemente descritos:

- ***Análisis de tipo visual:*** este tipo de análisis arquitectural se encuentra fundamentado en el estudio de patrones de imágenes a través de herramientas de análisis por computadora. Su principio fundamental es comparar dos o más tipos de arquitecturas de software y mediante la aplicación de algoritmos de análisis de imágenes, obtener resultados que proveen al interesado información precisa acerca de las similitudes y diferencias halladas entre las arquitecturas comparadas (Beck & Diehl, 2010). Este tipo de análisis se caracteriza por ser poco preciso y por depender mucho de la calidad con la que hayan sido construidos los diagramas analizados.

- ***Análisis de tipo matemático:*** mediante la aplicación de matemáticas se pueden proponer distintos tipos de análisis basados en algoritmos construidos a partir de demostraciones matemáticas realizadas a modelos propuestos. En este tipo de análisis pueden existir tantas variantes de técnicas como de herramientas matemáticas existan, pudiendo de esta manera encontrar análisis de arquitecturas basado en teoría de conjuntos, en lógica booleana, en lógica de predicados y en el

álgebra relacional, siendo esta última técnica la utilizada en el presente trabajo para la definición del método propuesto en uno de los objetivos planteados.

- ***Análisis de tipo automático:*** este tipo de análisis conjuga las ventajas que brindan las herramientas de tipo software a los procesos, desde el punto de vista del tiempo y la efectividad. Para las técnicas utilizadas pueden utilizarse distintos tipos de enfoque desde matemáticos, lógicos, algorítmicos hasta lógicos, siendo el punto clave la interacción con una máquina que permita reducir sustancialmente el tiempo de procesado de la información y la interpretación de los resultados como por ejemplo una computadora.

3.3.4. Ingeniería inversa

La ingeniería inversa se constituye como un proceso totalmente contrario a lo que normalmente se realiza en la ingeniería de software. En un primer momento se parte de una necesidad, luego se realiza un proceso -con todas las etapas pertinentes- y finalmente se obtiene un producto software que satisface la necesidad desencadenante del problema; sin embargo, la ingeniería inversa parte del producto software obtenido y realiza una serie de pasos y análisis que terminan desglosando el proceso para obtener vistas de software de alto y bajo nivel de abstracción que pueden ser utilizadas para agilizar procesos de mantenimiento al software, cambios de código, agregar nuevos componentes y realizar análisis a las arquitecturas de los sistemas estudiados (Chifosky & Cross, 1990).

En la ilustración 6 se puede apreciar una comparación del proceso de ingeniería inversa con el proceso tradicional de construcción de software, también denominado ingeniería directa.

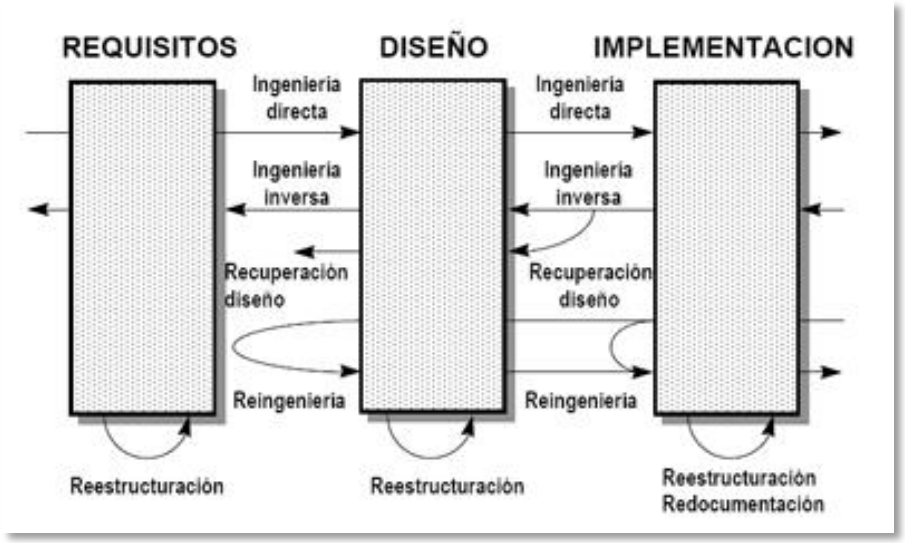

Ilustración 6. Proceso de ingeniería inversa versus ingeniería directa (Sicilia, 2009)

Muchas veces el término ingeniería inversa es tergiversado y se toma para justificar comportamientos poco éticos y en algunos casos ilegales. La finalidad de la ingeniería inversa está sustentada en realizar análisis a los sistemas software (Canfora & Di Penta, New Frontiers of Reverse Engineering, 2007) a partir de la obtención de sus estructuras y componentes, representables mediante vistas de alto nivel -arquitecturas- y de bajo nivel -diseño-, enfocando las soluciones obtenidas a derivar nuevos productos a partir de los que ya se tenían o además a realizar mejoras a estos. Muller et al, sintetizaron esta idea a principios del año 2000, partiendo del hecho de que la información obtenida a partir de los procesos de ingeniería inversa debía servir como soporte para desarrollos futuros, evitando de esta manera incurrir en errores anteriores a la hora de construir software.

El análisis realizado gracias a los postulados de la ingeniería inversa, permite aumentar la calidad de los productos debido a que se tienen puntos de referencia fuertes para afrontar nuevos desarrollos gracias a toda la información extraída a los sistemas software que se analizan.

La ingeniería inversa también ha evolucionado con el pasar del tiempo, buscando facilitar sus procesos. Es por ello que se plantean nuevos retos en el estudio de la misma, tales como

automatizar los análisis realizados, apuntar a ingeniería verde -ecológica- y plantear métodos que permitan realizar análisis a las arquitecturas de sistemas software, obtenidas a través de ingeniería inversa (Canfora & Di Penta, New Frontiers of Reverse Engineering, 2007).

3.3.5. Álgebra relacional

El álgebra relacional es uno de los tantos campos matemáticos que existen y ciertamente provee mucha utilidad a las ciencias de la computación. En un primer vistazo a este campo, el álgebra relacional no es más que un conjunto de tablas llenas de datos denominadas relaciones, compuestas a su vez por tuplas o registros que mediante un conjunto de operaciones derivadas como el producto cartesiano por ejemplo, se operan entre sí para dar resultados de consultas. Normalmente no se llega a medir el gran número de posibilidades que esto ofrece, dando un vistazo general a las bases de datos por ejemplo, los motores de consultas de las mismas y los mecanismos de relaciones entre tablas están estrictamente basados en el álgebra relacional.

Tal cual como la aritmética que es conocida desde los estudios primarios y sus cuatro operaciones -suma, resta, multiplicación y división-, el álgebra relacional también define un conjunto de operaciones que se aplican entre relaciones -o tablas- cuyo resumen puede ser consultado en la ilustración 7. Estas operaciones son el producto cartesiano, que es una combinación tupla a tupla o registro a registro de cada tabla dando como resultado la combinatoria entre relaciones o también relaciones que tengan elementos en común; también se encuentra la proyección, que permite seleccionar un subconjunto vertical de una tabla y obtener el resultado en otra relación. Por otro lado está la operación de selección, que sirve para obtener tuplas que satisfagan cierto predicado.

Estos conceptos matemáticos son de gran utilidad a la hora de analizar elementos estructurales representados en relaciones. En el campo de la arquitectura de software son de gran utilidad debido a que toda arquitectura de software puede ser representada como un

grafo de aristas y nodos que finalmente constituye un conjunto de entidades relacionadas entre sí (Gorton, 2006), situación que tiene importantes implicaciones para el análisis y mantenimiento de sistemas software (Kazman & Bass, 2002) (Podgurski, 2006).

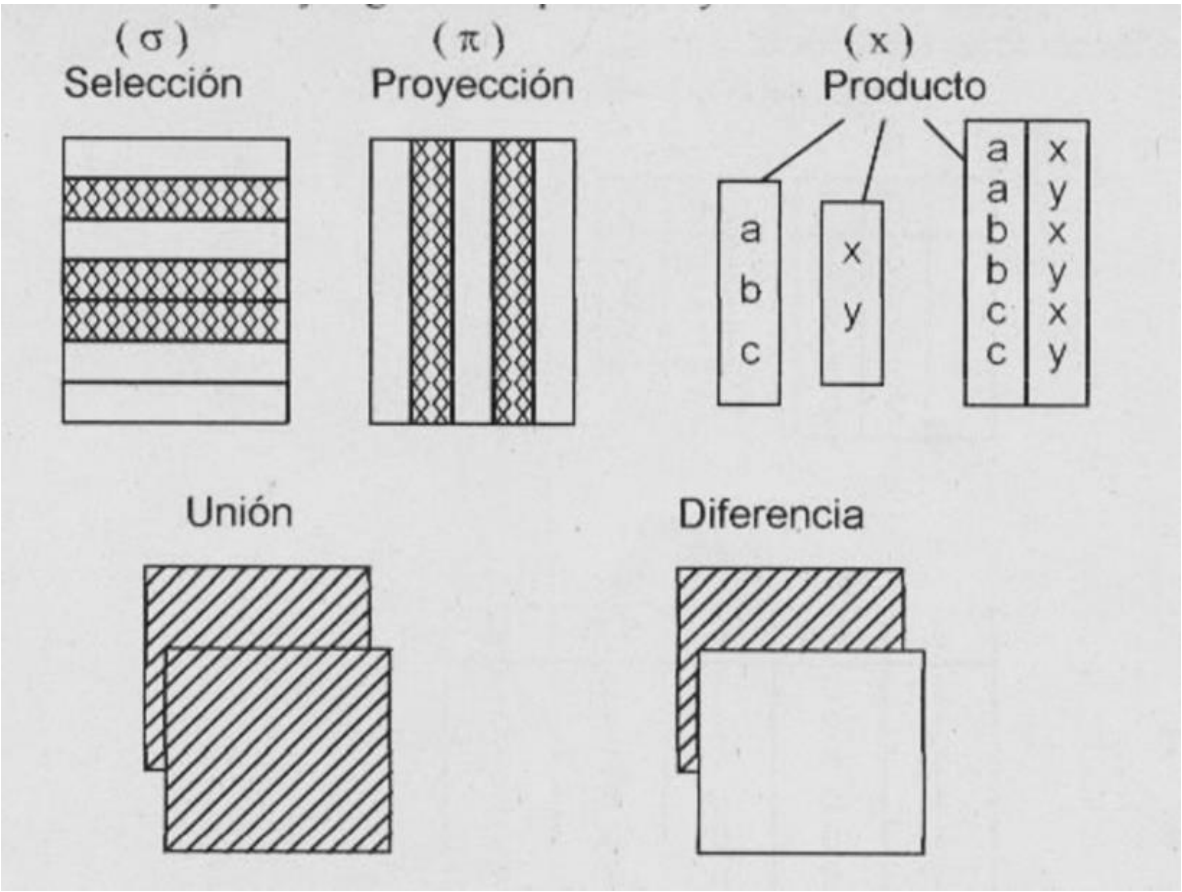

Ilustración 7. Operaciones básicas del álgebra relacional (Universidad De Sevilla, 2012)

El álgebra relacional puede ser fácilmente extendida y amoldada a distintas situaciones que requieran su uso, debido a que los operadores básicos muchas veces no alcanzan a enmarcar todas las necesidades que se tienen. Por ejemplo, muchas veces existen datos que no pueden ser representados en una tupla ordinaria, sino que necesitan el uso de "tuplas de tuplas". De hecho, existen versiones de álgebra relacional extendida enfocadas a las ciencias de la computación, caso concreto a la arquitectura del software en donde se estudian no sólo relaciones sino también entidades relacionadas entre sí (Li et al, 2010).

4. METODOLOGÍA

4.1 ENFOQUE Y TIPO DE INVESTIGACIÓN

Para dar cumplimiento a los objetivos propuestos en el presente proyecto se llevó a cabo una investigación de tipo exploratoria y documental, puesto que las temáticas que representan interés para el trabajo a realizar han sido abordadas con anterioridad pero de manera reducida y por pocos autores, además se pretende con la presente iniciativa sentar una base académica documentada para que sea utilizada en futuras investigaciones cuyas temáticas estén relacionadas con el área de estudio abordada en este trabajo.

A partir de las técnicas de recolección y análisis de la información recolectada, la investigación realizada se fundamenta en el análisis documental (Rojas, 2005) debido a que los lineamientos de este tipo de técnica propician el uso de múltiples registros legibles que contengan diagramas, gráficos, tablas e imágenes que resultaron bastante útiles para la comprensión de las distintas temáticas abordadas cuyas teorías no fueron tan sencillas de entender pero cuya valía académica es tan alta que permiten soportar de una manera robusta y científicamente aceptada los distintos lineamientos propuestos a lo largo del presente documento.

Mediante los resultados obtenidos, la investigación realizada puede enmarcarse como cualitativa, puesto que aunque su principal resultado está encaminado a proponer un método de análisis matemático, su fundamentación académica estuvo basada en la revisión sistemática de la literatura que se realizó con miras a justificar todos los resultados que surgieron a partir de las propuestas hechas a lo largo del trabajo realizado.

Los resultados obtenidos fueron validados a través del diseño y uso de un escenario de prueba experimental, que permitió verificar la pertinencia del estudio realizado. El escenario de prueba fue obtenido a partir de la investigación *Detección y Notificación de Eventos Fortuitos a través de Dispositivos Móviles* realizada con éxito en los años 2010 y

2011 como proyecto de las asignaturas Ingeniería De Servicios De Internet e Ingeniería De Software y aprobado en ese momento por los docentes que estaban encargados de impartir dichas cátedras.

El presente proyecto fue realizado en la ciudad de Cartagena De Indias, Bolívar durante los años 2013 y 2014 siguiendo los lineamientos estipulados por la Universidad De Cartagena para presentar proyectos de investigación y por las directrices establecidas por el Programa De Ingeniería De Sistemas adscrito a la Facultad De Ingeniería para la elaboración de proyectos de grado. La investigación fue dirigida por el ingeniero Msc. Martín Emilio Monroy Ríos quien actualmente se desempeña como profesor e investigador del Programa De Ingeniería De Sistemas de la Universidad De Cartagena y adelanta estudios de doctorado en la Universidad Del Cauca.

4.2 PROCEDIMIENTO DE TRABAJO

Para llevar a cabo el presente proyecto se siguió un procedimiento de trabajo por objetivos, consistente en la planificación y ejecución de actividades generales denominadas hitos determinadas a partir de cada uno de los objetivos propuestos en la investigación. Los hitos a su vez contienen actividades más específicas denominadas sub actividades, que a su vez generaron resultados entregables y contribuyeron a la elaboración del informe final.

Para cada uno de los hitos y actividades realizadas se presenta un resumen a continuación:

- ***Estado del arte***

Se realizó una revisión minuciosa de la literatura mediante el uso de las bases de datos proporcionadas por la Universidad De Cartagena para construir un estado del arte completo y detallado que permitió contextualizar y concretar la actualidad de la temática abordada, con miras a contar con un soporte académico robusto que permitió sustentar los postulados propuestos que surgieron como resultado de la investigación. Este hito constituye uno de

los objetivos del proyecto que más tiempo requirió debido a que a partir de los resultados de esta actividad se realizaría el resto del proyecto y fue fundamental revisarlo minuciosamente. En la ilustración 8 se aprecia la representación esquemática de este paso.

Sub actividades para este hito:
-Revisión de la literatura.
-Recolección de información.
-Análisis de información.
-Elaboración de estado del arte.

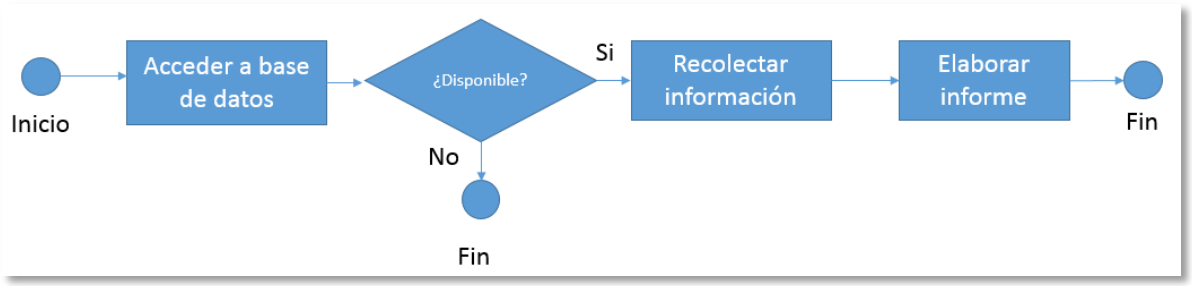

Ilustración 8. Representación esquemática para la actividad Estado Del Arte (Elaboración propia)

- ***Identificación de necesidades relacionadas con análisis de arquitecturas de software***

Tomando como base la documentación generada en la fase anterior, se procedió a identificar las necesidades que surgieron como resultado de las problemáticas y trabajos futuros encontrados en la literatura, propuestos por los autores que han abordado la temática con anterioridad. Posteriormente dichas necesidades se clasificaron siguiendo un orden prioritario de acuerdo a la recurrencia de las mismas en el estado del arte realizado. En la ilustración 9 se aprecia la representación esquemática de este paso.

Sub actividades para este hito:
-Análisis del informe del estado del arte.
-Definición de necesidades encontradas.
-Elaboración de informe.

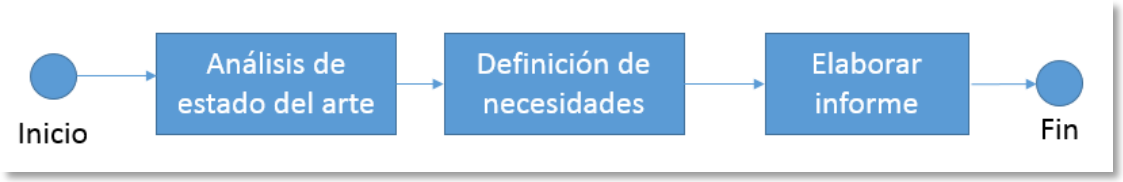

Ilustración 9. Representación esquemática para la actividad de Identificación de Necesidades (Elaboración propia)

- ***Relación de necesidades con posibles soluciones basadas en álgebra relacional***

Las necesidades identificadas en el paso descrito anteriormente fueron relacionadas con posibles soluciones a las mismas, estando dichas soluciones basadas en los postulados matemáticos del álgebra relacional. Los soportes académicos que permitieron sustentar el uso del álgebra relacional y sus temáticas están disponibles como resultado de la fase estado del arte debido a que su principal componente fue la revisión literaria que permitió clasificar numerosos autores y temáticas útiles para el desarrollo de la investigación. La ilustración 10 muestra la representación esquemática de este hito.

Sub actividades para este hito:

-Relación de necesidades con posibles soluciones basadas en álgebra relacional.

-Modelado de posible solución basada en álgebra relacional.

-Elaboración de informe.

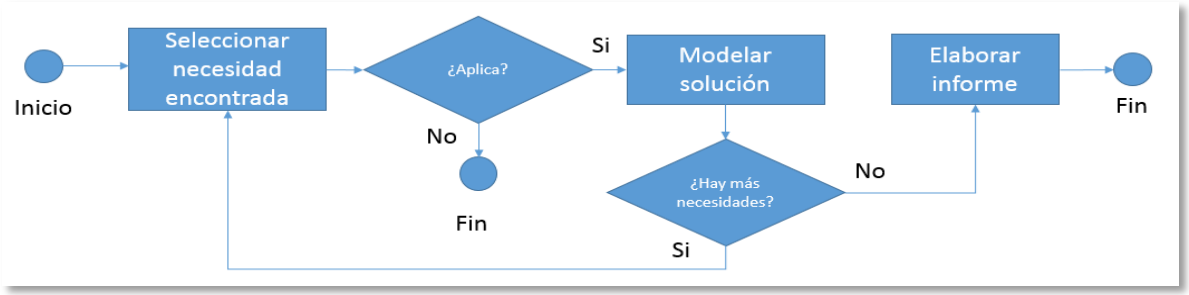

Ilustración 10. Representación esquemática para actividad Relación de Necesidades con posibles soluciones (Elaboración propia)

- ***Definición de método para realizar análisis a arquitecturas de software***

En esta fase se propuso un método que conjuga los resultados obtenidos en el paso anterior buscando generalizarlos para poder ser utilizados en situaciones no puntuales pero que tengan la misma necesidad de análisis arquitectural como punto de partida. En esta fase se obtuvo como resultado una serie de pasos definidos, que teniendo en cuenta las características de los métodos deductivos, parten de un marco de referencia –arquitectura de software- y que mediante el análisis realizado a sus componentes –entidades- permiten finalmente determinar características útiles a los desarrolladores de software y arquitectos de software como el tipo de relaciones existente entre determinadas entidades, medición de impactos, pertenencia a paquetes de software e incluso determinar generalización de objetos. El análisis realizado a los componentes de la arquitectura se hizo con ayuda de los resultados del hito anterior, específicamente en lo que concierne al álgebra relacional. La ilustración 11 muestra la representación esquemática de esta actividad.

Sub actividades para este hito: -Definición del método propuesto.
-Elaboración de informe.

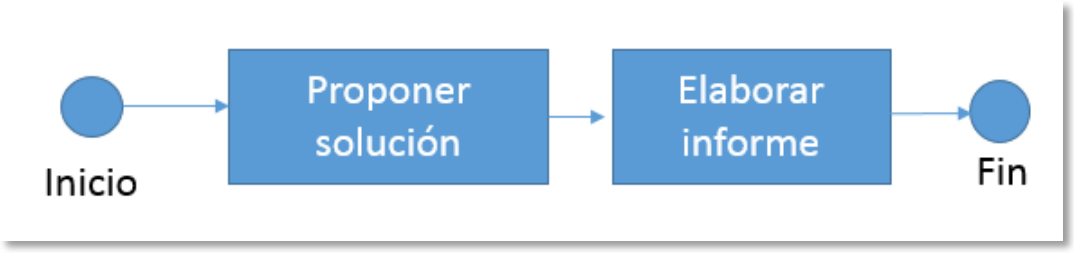

Ilustración 11. Representación esquemática para actividad Definición de Método (Elaboración propia)

- ***Validación de la solución propuesta***

En la fase de validación se puso a prueba el método mediante la aplicación de este en un escenario de prueba que permitió verificar su pertinencia con el área de análisis a arquitecturas de software. El escenario de prueba utilizado fue la arquitectura de software del sistema *Detección y Notificación de Eventos Fortuitos a través de Dispositivos Móviles*

realizado en los años 2010 y 2011 en el programa de Ingeniería De Sistemas de la Universidad De Cartagena y que fue presentado como poster en el Congreso Nacional De Ingeniería De Sistemas en la ciudad de Barranquilla en el año 2011. La ilustración 12 muestra la representación esquemática de esta actividad.

Sub actividades para este hito:

-Definición de caso de prueba.

-Aplicación de método al caso de prueba definido.

-Elaboración de informe.

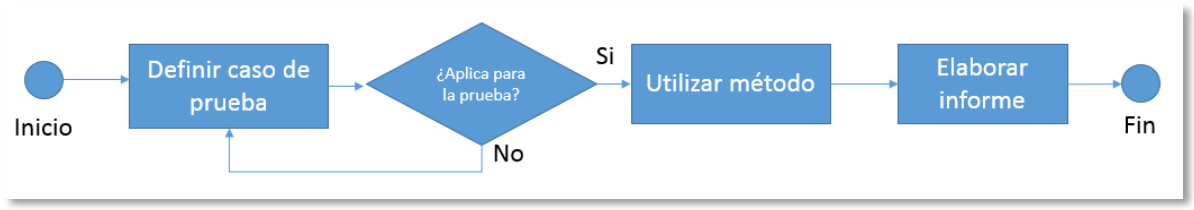

Ilustración 12. Representación esquemática para la actividad Validación de Solución Propuesta (Elaboración propia)

4.3 TÉCNICAS DE RECOLECCIÓN Y ANÁLISIS DE INFORMACIÓN

Por la naturaleza de la investigación realizada anteriormente descrita en la sección *enfoque y tipo de investigación*, la recolección de información estuvo centrada exclusivamente en la revisión y análisis sistemático de la literatura. Las fuentes consultadas permitieron fundamentar la investigación realizada y los resultados obtenidos, debido a que en primer lugar se restringió la búsqueda a bases de datos académicas que hacen parte de la comunidad científica y además, se aplicaron una serie de técnicas pertenecientes al proceso de revisión sistemática de literatura que permitieron discriminar los resultados obtenidos durante las búsquedas realizadas obteniendo información mucho más precisa y relacionada con las temáticas del proyecto.

Durante el proceso de recolección y análisis de información no se realizó una clasificación de fuentes según su prioridad –primarias o secundarias por ejemplo- debido a que a partir de la naturaleza de una investigación de tipo exploratorio todas las fuentes deben tratarse de

la misma manera por tratarse del paso inicial para abordar una temática poco conocida (Rojas, 2005).

La técnica utilizada durante el proceso de recolección y análisis fue la propuesta por Maturro y Saavedra, investigadores uruguayos, cuya metodología denominada *mapeo sistemático de la información* fue exclusivamente diseñada para ser aplicada a investigaciones de tipo exploratorio y análisis documental.

A continuación se describe la técnica especificada utilizada para la recolección de información y sus resultados.

4.3.1 Mapeo sistemático de la información

El principal objetivo de esta técnica de recolección de la información es proveer un marco de referencia organizado al investigador que permita realizar búsquedas organizadas que arrojen resultados pertinentes y relacionados con el objetivo del trabajo que se esté realizando.

Las revisiones de literatura generalmente se constituyen en búsquedas sin un objetivo definido debido a que el volumen de información que se maneja es tan grande que es imposible demarcarlo en un solo contexto. Gracias al mapeo sistemático de la información se puede llevar a cabo un proceso de recolección y análisis de información de forma organizada y obteniendo resultados relevantes.

El resultado esperado al realizar un mapeo sistemático es obtener una visión general de un área de investigación y determinar la cantidad y el tipo de investigación, y los resultados disponibles (Maturro & Saavedra, 2012) a partir de preguntas que permitan resolver el cuestionamiento de investigación principal de un proyecto.

Según el postulado establecido por Maturro y Saavedra, como primer paso se debe proceder a la definición de preguntas para dar inicio al mapeo sistemático.

Paso 1: Definición De Preguntas Para Búsqueda

Para el proyecto realizado se definieron 9 preguntas que enmarcan todas las temáticas relevantes que fueron necesarias para realizar búsquedas pertinentes durante la revisión literaria, siguiendo el proceso de mapeo sistemático. Las preguntas definidas pueden ser consultadas en la tabla 1.

Tabla 1. Preguntas de investigación para mapeo sistemático de la literatura

No.	Pregunta	Sección que responde a la pregunta
1	¿Cuáles son los estudios existentes acerca del análisis de vistas arquitectónicas de software?	Sección 3.2: Estado del arte
2	¿Qué tipo de análisis permiten?	Sección 3.2.1: Métodos de análisis arquitectural
3	¿Qué técnicas de análisis utilizan?	Sección 3.2.1: Métodos de análisis arquitectural
4	¿Qué falencias presentan los actuales estudios existentes acerca del análisis de vistas arquitectónicas de software?	Sección 5.1: Necesidades encontradas en el campo del análisis arquitectural: estudio de las falencias halladas en el estado del arte
5	¿Cuál es el método más completo que actualmente existe para análisis de vistas arquitectónicas?	Sección 5.3: Método de análisis arquitectural basado en álgebra relacional
6	¿Existe alguna fundamentación matemática para este método?	Sección 5.3.1: fundamentación propuesta en el trabajo de Zude Li et al.
7	¿Existe un campo matemático que pueda fundamentar un método de análisis a vistas arquitectónicas, como el álgebra relacional?	Sección 5.3.1: fundamentación propuesta en el trabajo de Zude Li et al.
8	¿Cuáles son los principales investigadores en el área?	Sección 3: Marco de referencia Sección 4.3.1: mapeo sistemático de la información
9	¿Cuáles son las principales revistas y conferencias donde se publican estudios sobre el tema?	Sección 4.3.1: mapeo sistemático de la información

Paso 2: Palabras Clave y Definición De Cadenas De Búsqueda

A partir del establecimiento de las preguntas para iniciar la búsqueda en el mapeo sistemático de la información, se deben establecer las cadenas de búsqueda que serán utilizadas para realizar las búsquedas en las bases de datos académicas.

Las cadenas de búsqueda deben estar formadas por palabras clave que puedan arrojar resultados que satisfagan las preguntas propuestas en el paso 1 de la técnica de mapeo sistemático de la información. Se debe tener en cuenta que las cadenas de búsqueda definidas podrán combinarse a criterio del autor utilizando los conectores lógicos *AND* y *OR* que se encuentran disponibles en las bases de datos para discriminar búsquedas de información.

El autor de la técnica deja a criterio del investigador la manera de combinar las distintas palabras clave definidas a partir de las preguntas para búsqueda de información por lo que las palabras clave y las cadenas de búsqueda no necesariamente tienen que coincidir.

Las palabras clave que se utilizaron para la investigación realizada están resumidas en la ilustración 13.

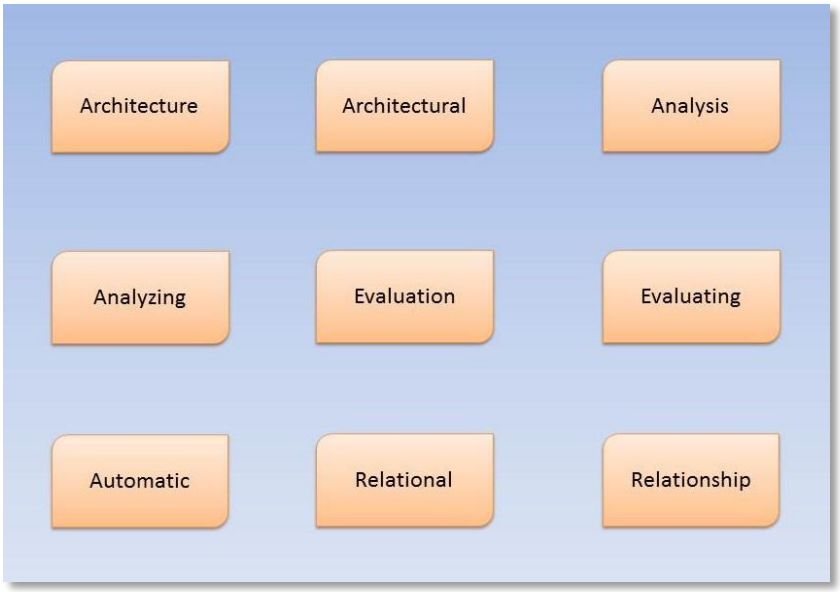

Ilustración 13. Palabras clave utilizadas para la construcción de cadenas de búsqueda (Elaboración propia)

Para el proyecto realizado y a partir de las preguntas definidas en el paso 1, se utilizaron las cadenas de búsqueda referenciadas en la tabla 2.

Tabla 2. Cadenas de búsqueda para mapeo sistemático de información

No.	Cadena de búsqueda
<cadena 1>	"architecture" OR "architectural"
<cadena 2>	"analysis" OR "evaluation"
<cadena 3>	"analyzing" OR "evaluating"

Cadena general: (C1 and C2) or (C3 and C1)

Paso 3: Definición De Sitios De Búsqueda y Mapeo Sistemático

El paso 3 es el más importante dentro del proceso de mapeo sistemático. En este paso se debe tener especial cuidado porque se definen los sitios en los cuales se realizarán las exploraciones de información haciendo uso de las cadenas de búsqueda definidas. El autor de la técnica recomienda la clasificación de artículos científicos encontrados mediante el uso de tablas o diagramas.

Las bases de datos científicas consultadas durante la investigación se encuentran referenciadas en la tabla 3.

Tabla 3. Bases de datos consultadas

Base De Datos	Dirección URL
ACM Digital Library	http://portal.acm.org
IEEE Computer Society	http://www.computer.org
ScienceDirect	http://www.sciencedirect.com
SpringerLink	http://www.springerlink.com
Scopus	http://www.scopus.com
Wiley Online Library	http://onlinelibrary.wiley.com
Citeseer	http://www.citeseerx.ist.psu.edu

Se debe agregar que solo las 4 primeras bases de datos referenciadas en la tabla 3 fueron las consultadas puesto que debido a la no disponibilidad de recursos económicos para la ejecución del proyecto solo se pudo hacer uso de las bases de datos con las que la Universidad De Cartagena tiene suscripciones activas.

Posterior a la definición de sitios a utilizar como fuentes de información se debe realizar la búsqueda de resultados con las cadenas de búsqueda definidas y la combinación de estas según el criterio del investigador.

En la tabla 4 se encuentra el resumen de las cadenas de búsqueda aplicadas a determinada base de datos y el número de resultados encontrados.

Tabla 4. Cadenas de búsqueda utilizadas, sitios consultados y resultados encontrados

Cadena De Búsqueda	Sitio Consultado	No. De Resultados
(("architecture" OR "architectural") AND ("analysis" OR "evaluation")) OR (("analyzing" OR "evaluating") AND ("architecture" OR "architectural"))	ACM	15
(("architecture" OR "architectural") AND ("analysis" OR "evaluation")) OR (("analyzing" OR "evaluating") AND ("architecture" OR "architectural"))	ScienceDirect	10
(("architecture" OR "architectural") AND ("analysis" OR "evaluation")) OR (("analyzing" OR "evaluating") AND ("architecture" OR "architectural"))	Springer	5
(("architecture" OR "architectural") AND ("analysis" OR "evaluation")) OR (("analyzing" OR "evaluating") AND ("architecture" OR "architectural"))OR (“Architecture” or “architectural”) and (“automatic”) and (“analysis” or “analyzing” or “evaluation” or “evaluating”)	IEE	19

Como se puede apreciar en la ilustración 14, las cadenas de búsqueda aplicadas no fueron igualmente efectivas en todas las bases de datos. Este tipo de comportamiento se debe a la naturaleza misma de cada una de las bases de datos puesto que aunque las 4 fuentes

consultadas están relacionadas con las ciencias de la computación, únicamente ACM e IEEE son exclusivamente enfocadas al área de la Ingeniería De Sistemas y lógicamente fueron las que más resultados arrojaron.

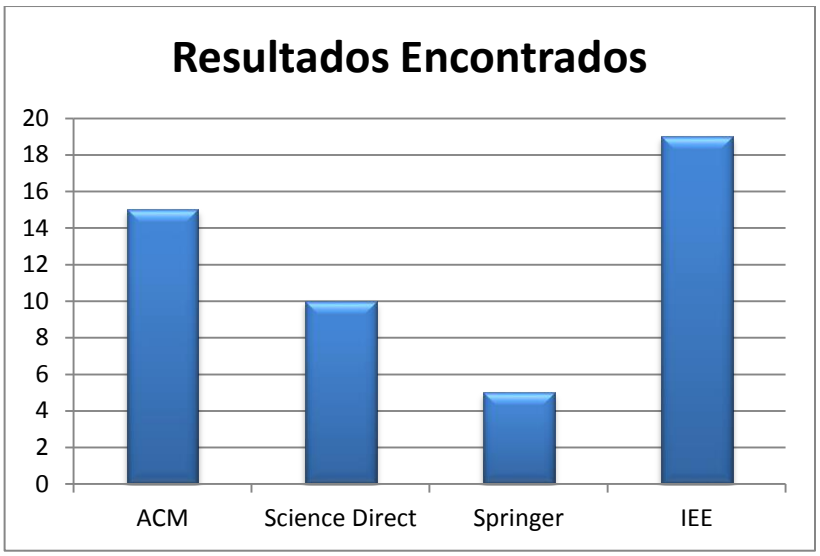

Ilustración 14. Resultados encontrados para cada base de datos consultada (Elaboración propia)

Paso 4: Interpretación De Resultados

Finalmente en la técnica de mapeo sistemático de información se debe realizar una interpretación de los resultados obtenidos. En primer lugar se definieron 2 criterios para discriminar la información recolectada: un criterio de inclusión y un criterio de exclusión, siendo el primero utilizado para determinar que artículos se iban a tener en cuenta para la investigación y que artículos no. Gracias al uso de los criterios definidos se pudo reducir sustancialmente el volumen de la información recolectada.

La descripción de los criterios utilizados es la siguiente:

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

- **Exclusión:** artículos publicados en revistas o actas de conferencias no arbitradas. Documentos con más de cinco años de publicación y sin haber sido citados. El artículo debe tenerse en cuenta sino es posible conocer el número de citaciones (algunas de las bases de datos no lo proporcionan al usuario proporcionado por la universidad), siempre y cuando sea de una conferencia reconocida y/o arbitrada.

Luego de haber aplicado los criterios de inclusión y exclusión definidos se obtuvo la clasificación de un conjunto de artículos científicos que fundamentaron la gran mayoría de la investigación realizada durante la ejecución del proyecto y que aportaron los aspectos más relevantes a los resultados obtenidos. Cabe anotar que durante la investigación se utilizaron más referentes de los descritos en la tabla 5 pero que no son mencionados pues no fueron obtenidos como resultado del mapeo sistemático realizado sino que fueron incluidos debido a que se consultaron a medida que se encontraron referencias en los resultados de la revisión de la literatura[3].

Posteriormente y teniendo disposición de los resultados obtenidos se procedió a utilizar toda la información recolectada para dar cumplimiento a los objetivos del proyecto constituidos por el estado del arte realizado –capítulo 3.2-, clasificación de necesidades encontradas, definición de método de análisis y finalmente la validación de los resultados obtenidos –capítulo 5-.

Tabla 5. Interpretación y clasificación de los resultados obtenidos

Título	Autor(es)	Aportes útiles para la investigación	Trabajos futuros
New Frontiers of Reverse Engineering	Gerardo Canfora, Massimiliano Di Penta	-Contextualización de la actualidad de la ingeniería	-Análisis de código.

[3] En el archivo anexo al documento Artículos_Obtenidos.xlsx se proporciona al lector la recopilación de todos los artículos involucrados en el proceso de mapeo sistemático.

		inversa como proceso importante en la ingeniería de software. -Descripción detallada de los procesos de análisis que se realizan durante un procedimiento que implique ingeniería inversa.	-Análisis de diseños. -Análisis arquitectural.
Achievements and challenges in software reverse engineering	Gerardo Canfora, Massimiliano Di Penta, Luigi Cerulo	-Este trabajo es una síntesis concreta de los aspectos propuestos en el artículo anteriormente referenciado.	-Además de los trabajos futuros del artículo anteriormente referenciado, es importante señalar el compromiso con el medio ambiente que propone el autor Cerulo basado en su concepto de "ingeniería verde".
Towards a Precise Description of Reverse Engineering Methods and Tools	Stan Jarzabek and Irene Woon	-Este trabajo se concentra en obtener aspectos abstractos de la solución analizada, por lo tanto resalta la importancia del uso de los artefactos de alto nivel (arquitectura). -El artículo describe prototipos de análisis relacional de los componentes obtenidos a partir del análisis realizado al código.	-Los autores resaltan la importancia de llegar a un lenguaje estandarizado de consultas a arquitecturas, aspecto que sólo podría ser posible si antes se definen parámetros o pasos a seguir en un método para realizar dicho análisis.

		-El artículo describe prototipos de lenguajes extendidos a partir del álgebra relacional, para representar relaciones entre las entidades de un sistema.	
Software Architecture Recovery and Modelling	Arie van Deursen	-El trabajo resalta la importancia de la recuperación de arquitecturas de software para poder entender un sistema a partir del análisis de la misma.	-El autor plantea la inquietud de analizar la arquitectura desde el punto de vista del usuario, para lo cual sería útil un método de análisis basado en UML o en representaciones gráficas.
Architecture Recovery of Web Applications	Ahmed E. Hassan, Richard C. Holt	-El trabajo se centra en la recuperación de arquitecturas y análisis de las mismas pero en entorno web.	-La necesidad de realizar análisis a arquitecturas de sistemas web se debe soportar con un método que permita estandarizar el proceso, pues las estructuras de las páginas web cambian con mucha dinámica.
Towards a Common Query Language for Reverse Engineering	Jingwei Wu, Richard C. Holt, Andreas Winter	-La investigación presenta un prototipo de lenguaje estándar de consultas para realizar análisis a sistemas software obtenidos partir de ingeniería inversa	-Los autores presentan una serie de requerimientos que consideran, debería abordar el lenguaje propuesto o que se busca proponer para realizar análisis a los sistemas software y sus arquitecturas. Dichos requerimientos pueden

			comenzar a ser analizados mediante la definición de un método de análisis a arquitecturas utilizando álgebra relacional.
Visual Comparison of Software Architectures	Fabian Beck, Stephan Diehl	-El punto de interés de la investigación radica en el análisis arquitectural que los autores proponen desde el punto de vista gráfico, pudiendo estudiar entidades, componentes y relaciones con ayuda de elementos visuales.	-Los autores concluyen en que es conveniente la materialización de métodos de análisis arquitectural "no visual" que puedan fundamentar una estandarización de las herramientas de análisis gráficas.
Reverse Engineering Architectural Feature Models	Mathieu Acher, Anthony Cleve, Philippe Collet, Philippe Merle, Laurence Duchien, Philippe Lahire	-El principal aporte del artículo al trabajo de investigación radica en la forma propuesta por los autores para obtener modelos característicos de arquitecturas de software a partir de análisis basado en teoría de conjuntos, lo que demuestra la posibilidad de abordar la temática de análisis de arquitecturas de software desde el punto de vista matemático.	-Los autores plantean la importancia de imprimir mayor interés en el campo de la arquitectura de software con miras a buscar la estandarización de métodos de análisis arquitectural que permitan manipular directamente las entidades del sistema para obtener una mayor comprensión de los mismos.
Towards and extended relational algebra for software architecture	Zude Li, Mechelle Gittens, Syed Sariyar Murtaza, Nazim Madhavji	-Este artículo es el más importante para la investigación pues comprende toda la fundamentación matemática a aplicar en el	-Los autores proponen la búsqueda por parte de los interesados de nuevos modelos matemáticos relacionales para manipulación de

		método a definir, propone formas de relacionar las entidades de una arquitectura de software que pueden desembocar en propuestas para análisis estructural, de relaciones entre entidades y mediciones de impacto.	arquitecturas de software, dejando a la libre elección de los investigadores este punto.
Towards the Unified Recovery Architecture for Reverse Engineering	Thomas Panas, Welf Löwe, Uwe Assmann	-La búsqueda del entendimiento de un sistema software a través de la recuperación y análisis de su arquitectura es el principal aporte de este trabajo, puesto que fundamenta de manera científica el propósito de la investigación en curso. Los autores proponen una metodología para obtener arquitecturas más comprensibles para los interesados.	-Los autores plantean la importancia de la necesidad de un estándar para realizar análisis arquitectural, en el que puedan confluir todos los postulados realizados relacionados con el tema. El objetivo del trabajo de grado puede sentar un precedente en este punto.
Caracterización de Herramientas de Ingeniería Inversa	Martín E. Monroy, José L. Arciniegas, Julio C. Rodríguez	-El artículo representa importancia para el aspecto de la construcción del estado del arte de la investigación, gracias a una clasificación detallada de herramientas de ingeniería inversa y sus características. Representa un artículo a nivel regional, lo que reafirma	-Desarrollo de herramientas de ingeniería inversa para aspectos emergentes. Estas herramientas pueden ser fundamentadas en la metodología objetivo del trabajo de grado.

		el interés científico en la temática de la ingeniería inversa.	
Layered Semantic analysis of Software Architecture	Liao Lijun, Li Changyun, Ma Gexin, Wang Zhibing	-Propone un aspecto de análisis arquitectural por capas, de tipo jerárquico. Esta caracterización puede ser complementada con el álgebra relacional y sus postulados, además de reafirmar científicamente la misma con ayuda del trabajo de Zude Li.	-Ninguno evidenciado por los autores.
Experimental Analysis of Textual and Graphical Representations for Software Architecture Design	Werner Heijstek, Thomas Kuhne, Michel R. V.Chaudron	-El trabajo muestra los resultados de un experimento realizado con personas a partir del análisis arquitectural realizado a un sistema utilizando como herramientas documentos y gráficos generados por una arquitectura de software.	-El experimento tuvo sus aspectos más complicados a la hora de unificar conceptos entre los participantes en el análisis, pues existen diferentes formas dependientes de cada interesado para realizar análisis. Se plantea la necesidad de un método de análisis estándar.
Flexible Analysis Software for Emerging Architectures	Kenneth Moreland, Brad Kingy, Robert Maynardy, and Kwan-Liu Ma	-La investigación plantea un tipo de análisis para arquitecturas emergentes, incluyendo no solo elementos software sino elementos físicos (hardware) donde el software funciona. Es importante porque reafirma el trabajo que se	-Ninguno evidenciado por los autores.

		viene realizando en análisis arquitectural.	
SAAM: A Method for Analyzing the Properties of Software Architectures	Rick Kazman, Len Bass, Gregory Abowd, Mike Webb	-El artículo presenta un completo trabajo de estado del arte del método de análisis arquitectural SAAM. Sin embargo, SAAM solo se enfoca en analizar los atributos de calidad de la arquitectura. Este método puede trabajar de la mano en un futuro con el método que se busca definir en el proyecto.	-Los autores plantean la necesidad de estandarizar métricas de evaluación para las arquitecturas de software.

5. RESULTADOS

5.1. NECESIDADES ENCONTRADAS EN EL CAMPO DEL ANÁLISIS ARQUITECTURAL: ESTUDIO DE LAS FALENCIAS HALLADAS EN EL ESTADO DEL ARTE

Una de las principales características encontradas en los trabajos analizados a lo largo del capítulo anterior del presente documento ha sido la recurrencia por parte de los autores a fomentar trabajos futuros. Ya sean documentales, investigativos o simplemente recomendaciones al lector, los distintos artículos científicos analizados plantean necesidades que pueden suponer nuevas iniciativas investigativas que de ser abordadas, pueden contribuir a alimentar todo el conocimiento generado en el campo de la arquitectura de software, de la ingeniería inversa y del análisis arquitectural.

En primer lugar, se debe partir del hecho de demarcar y contextualizar la definición del término *necesidad.* Para la real academia de la lengua española, RAE, el término necesidad se define como *"(...) Carencia de las cosas que son menester para la conservación de la vida."*, la palabra carencia resulta clave en el alcance y contexto que los diferentes autores analizados en el estado del arte le dan al término necesidad: las propuestas de trabajos futuros buscan suplir la carencia de herramientas y artefactos en el campo específico de la arquitectura de software. Teniendo en cuenta esta definición, se puede comprender de mejor manera lo que representará a lo largo del presente trabajo el término necesidad y todos sus elementos relacionados.

Las necesidades encontradas a lo largo del análisis de literatura realizado no difieren entre sí, si se toma como base un tópico general: el análisis arquitectural. Algunos autores apuntan a estandarizar lenguajes de consulta (Wu & Winter, 2002) (Ducasse & Pollet, 2009), otro autores por su parte proponen métodos de análisis utilizando diferentes enfoques pudiendo ser visuales o matemáticos (Jarzabek & Woon, 2007) (Beck & Diehl,

2010) (Li, Gittens, Shariyar Murtaza, & Madhavji, 2010) y por otra parte, algunos autores innovan con nuevas tendencias y contextos alejados del software pero más familiarizados con el hardware[4] (Heijstek, Kühne, & Chaudron, 2011) (Moreland, King, Maynard, & Ma, 2013).

Teniendo en cuenta el párrafo anterior, se propone la clasificación de las necesidades encontradas a lo largo del estado del arte en tres grandes grupos: el primero comprendido por necesidades de tipo **investigativo**, que buscan resolver temáticas que se encuentran aún difusas y sin escenarios concretos de aplicación, necesidades de tipo **implementación** cuyo objetivo es la construcción de herramientas software que aplicando los conocimientos generados gracias a las necesidades investigativas satisfechas, proporcione facilidades a las personas interesadas en el análisis arquitectural para realizar sus procesos. Por último se proponen las necesidades de tipo **propositivo** que simplemente sugieren trabajos futuros.

A continuación se hace un recorrido por cada uno de los grupos de necesidades propuestos.

5.1.1. Necesidades de tipo investigativo: fijando el rumbo

En el campo del análisis de arquitecturas, de la ingeniería inversa y de la ingeniería de software en general, el conocimiento está en constante generación. Constantemente surgen nuevas inquietudes e iniciativas que permiten a los autores iniciar investigaciones que no tengan aplicación alguna hasta el momento en que surge la necesidad de abordarlas.

En el recorrido realizado en el capítulo del estado del arte se presentan varias necesidades de este tipo. En primer lugar, se plantea la necesidad de iniciar nuevas investigaciones en el campo de análisis de arquitecturas (Canfora, Di Penta, & Cerulo, 2011) como punto de partida a nivel macro para dar lugar a todas las iniciativas relacionadas con este campo. El

[4] En el contexto de análisis arquitectural, el hardware se puede tomar como componente de la arquitectura, pues la vista de despliegue de una aplicación software mezcla elementos de abstracción de alto nivel y de bajo nivel (Larman, 2003).

hecho de que el análisis de arquitecturas haya sido muy poco abordado por los científicos (Moreland, King, Maynard, & Ma, 2013) plantea una necesidad investigativa bastante compleja, puesto que no van a existir muchos referentes que permitan encontrar un norte para las propuestas que se planteen y de esta manera, finalizar exitosamente el trabajo realizado.

Continuando con el plano de necesidades investigativas, en un punto más particular aparecen necesidades un poco más concretas y demarcadas que igualmente carecen de un rumbo delimitado, por lo que se hace necesario realizar revisiones exploratorias en primer lugar para luego continuar con el trabajo investigativo. Tal es el caso de las necesidades que buscan satisfacer las necesidades de metodologías y/o métodos de análisis, como las propuestas desde el enfoque matemático de teoría de conjuntos (Acher, Cleve, Collet, Merle, Duchien, & Lahire, 2011) y del álgebra relacional –que constituye el enfoque de la presente investigación- (Li, Gittens, Shariyar Murtaza, & Madhavji, 2010). Este tipo de necesidades constituye una de las principales falencias en el campo, puesto que constituyen la base para poder implementar herramientas de software que faciliten los procesos de análisis arquitectural y de recuperación de arquitecturas a través de ingeniería inversa.

No se hará un análisis profundo y caracterizado de las necesidades de tipo investigativo, pues sería realizar nuevamente un recorrido generalizado por el estado del arte detallado en el capítulo anterior[5].

A partir de los párrafos anteriores se hace notoria una tendencia: las falencias investigativas que se presentan en el campo del análisis arquitectural parten de un marco general a un marco particular. Se inició con el requerimiento de abordar el tópico de análisis y se desembocó en la necesidad puntual de definir métodos de análisis que permitan obtener herramientas. Este flujo permite realizar la conexión con el siguiente tipo de necesidades propuestas en el presente documento, las necesidades de tipo implementación.

[5] Se recomienda al lector realizar una lectura detallada de la sección Marco De Referencia si se requiere ahondar más en las necesidades investigativas propuestas en la presente sección.

5.1.2. Necesidades de tipo implementación: conocimiento aplicado

En el campo de implementación, el análisis arquitectural presenta cierto avance comparado con el poco conocimiento generado en el campo de falencias investigativas. Este punto resulta curioso, pues se esperaría que si hay numeroso material a nivel de implementación de herramientas, sería debido a numerosas investigaciones que sienten las bases de los mismos; sin embargo, esta tendencia se presenta porque la mayoría de implementaciones se dan bajo los mismos conceptos abordados que son pocos. A pesar de la idea expresada en este párrafo, el campo implementación en análisis arquitectural no está exento de insolvencias.

En primer lugar la principal falencia que se presenta en este tipo de necesidades, es la ausencia de variedad de herramientas software que permitan a las personas interesadas en el proceso, realizar análisis arquitectural (Jarzabek & Woon, 2007). La falta de herramientas de este tipo no permite que el proceso sea más eficiente, puesto que con ayuda de soluciones software los tiempos de trabajo se ven reducidos y los resultados se obtienen de forma más precisa sin mucho margen de error (Störmer, 2007).

La necesidad expuesta en el párrafo anterior puede mitigarse de manera eficaz si se abordan las necesidades de tipo investigativo expuestas con anterioridad, puesto que se tendrían las bases y generalizaciones que permitirían implementar algoritmos que al ser codificados ayudarían a facilitar los procedimientos de análisis arquitectural.

Otra falencia que se encuentra en el campo de las necesidades de implementación, es la poca accesibilidad por parte de los usuarios a las herramientas que actualmente existen para análisis arquitectural (Kazman, Clements, & Klain, 2005). Este aspecto se centra principalmente en el costo de las mismas, que suele ser bastante elevado para su adquisición por parte de usuarios independientes o particulares, limitando su mercado a universidades que tengan buenos recursos económicos o a empresas que en determinado

momento tengan la necesidad de uso de este tipo de software en sus departamentos de investigación y desarrollo.
El campo de la implementación representa una oportunidad de mercado amplia (Moreland, King, Maynard, & Ma, 2013) puesto que la oferta es casi nula. Sin embargo, la demanda no es para nada elevada.

Contextualizando los aspectos encontrados en el campo de investigación, las dos principales falencias que caracterizan este punto son la poca accesibilidad a las herramientas y el elevado costo de las mismas –aspecto que podría tomarse como desencadenante del primer punto-. Debido a esta problemática, muchos de los autores incluidos en la revisión bibliográfica realizada, conscientes de la situación, enfatizan en otro enfoque: el enfoque propositivo, que igual contiene falencias que serán repasadas a continuación.

5.1.3. Necesidades de tipo propositivo

En este grupo de falencias, se concentran las secciones propuestas en la mayoría de artículos consultados, denominadas "trabajos futuros". El principal inconveniente de este grupo de necesidades es que el autor deja los tópicos abiertos a consideración del autor, lo que desembocaría en retomar nuevamente las necesidades de tipo investigativo, mencionadas al inicio de la presente sección.

El propósito de los trabajos futuros es que el lector indague y aborde sus requerimientos a partir de los trabajos estudiados, sin embargo este propósito se convierte en una falencia porque para el caso específico del análisis arquitectural, es necesario fijar un rumbo investigativo que permita basar las hipótesis que se quieran probar (Van Deursen, 2001).

Concluyendo, es beneficioso tener propuestas investigativas en el campo del análisis arquitectural, pero dichas propuestas deben estar correctamente fundamentadas. A partir de todo el recorrido realizado en esta sección, se ratifica la relación que existe entre las

necesidades de tipo investigativo, implementación y propositivo, puesto que la solución a las mismas permitiría concretar beneficios que cada una en particular puede proveer al experto en software.

Para la presente investigación, el trabajo se centrará en abordar la sección de necesidades investigativas y propositivas, mediante la definición de un método de análisis arquitectural basado en el álgebra relacional que en un futuro permita la implementación de una herramienta software, solventando de esta manera requerimientos difusos en los tres grupos de falencias propuestos.

5.2. DE LA NECESIDAD A LA SOLUCIÓN: APROXIMACIÓN AL ÁLGEBRA RELACIONAL COMO HERRAMIENTA DE ANÁLISIS

En la sección anterior se analizaron las distintas necesidades encontradas durante la revisión realizada a la literatura y enmarcadas dentro del capítulo del estado del arte. Se agruparon las distintas problemáticas encontradas teniendo en cuenta la afinidad que tuvieran entre sí y se realizó una clasificación en varios grupos que ayudan a demarcar diferentes contextos pertenecientes a la problemática macro de realizar análisis a arquitecturas de software. Para poder solventar las necesidades mencionadas, es necesario encontrar una solución.

En el mundo del software se pueden proponer múltiples soluciones, un ingeniero de software es capaz de construir desde un sencillo aplicativo que sirva como reloj, hasta complejos sistemas multicapa que controlan operaciones en tiempo real dentro de plantas de construcción; sin embargo, para pasar de la necesidad a la solución materializada y funcional se debe seguir un largo proceso cuyo objetivo es buscar la fundamentación teórica de la solución que se busca construir, pues no puede ser posible resolver una hipótesis espontáneamente y obtener resultados de calidad (López Cano, 1984).

Para la presente investigación, se ha definido con anterioridad una hipótesis concreta resumida en la pregunta de investigación: ¿Cómo realizar análisis a arquitecturas de software obtenidas como resultado de un proceso de ingeniería inversa?. La revisión a la literatura muestra aproximaciones a soluciones que buscan resolver la problemática desde diferentes perspectivas ya sea utilizando análisis visual, análisis de patrones de colores, análisis de capas de la arquitectura e incluso análisis matemático. Dentro del análisis matemático se presentan diferentes características que han sido definidas por los distintos autores consultados dependiendo del enfoque, características y necesidades puntuales que estos quisieran solventar: han logrado la definición de modelos conceptualizados en métodos. Es así como utilizando la teoría de conjuntos se han logrado definir métodos de análisis arquitectural con lógica de predicados por ejemplo (Acher, Cleve, Collet, Merle, Duchien, & Lahire, 2011), utilizando teoría de grafos se han definido métodos de análisis arquitectural a través de capas (Lijun, Changyun, Gexin, & Zhibing, 2012) y también utilizando principios de álgebra relacional se han logrado conceptualizar modelos relacionales que pueden permitir la definición de métodos de análisis arquitectural basados en álgebra relacional (Li, Gittens, Shariyar Murtaza, & Madhavji, 2010).

El álgebra relacional es una vertiente matemática basada en la definición de relaciones entre elementos y la computación de relaciones entre estos para la obtención de resultados (Date, 2013). La afinidad que presenta el álgebra relacional con el mundo del software es bastante grande: el modelo de bases de datos que actualmente se utiliza, el modelo relacional, está basado enteramente en el álgebra relacional y sus postulados. El álgebra relacional permite operar con grupos de elementos y a través de las relaciones entre estos, obtener información relevante que puede ser tratado como resultado de realizar análisis. En el caso concreto de una arquitectura de software puede permitir al interesado en los resultados, manipular las entidades que componen la arquitectura a través de las relaciones que existan entre estas y obtener resultados que le permitan aplicar su trabajo de la mejor manera.

En las necesidades de tipo investigativo descritas en los capítulos anteriores del presente documento, se hace evidente la necesidad de un mecanismo de análisis que actúe sobre la arquitectura de un determinado software como tal, involucrando sus componentes y proporcionando resultados. Los modelos matemáticos propuestos para este punto en específico pueden llegar a ser efectivos, pero distan de ser sencillos y su complejidad hace muy difícil la implementación de soluciones software basadas en estos; sin embargo, dentro del grupo de necesidades de tipo propositivo anteriormente clasificadas se deja la puerta abierta a propuestas puntuales basadas en los métodos descritos como investigaciones. El álgebra relacional cumple perfectamente el papel de ser encajado como herramienta de solución para las necesidades investigativas y propositivas dispuestas en el presente trabajo, puesto que aunque su manejo matemático es complejo, es una temática que ya tiene suficientes demostraciones soportadas en distintos autores y además proporciona la ventaja de que su validez ha sido comprobada durante años a partir del uso de bases de datos relacionales en el mundo de la computación.

Las bases de datos relacionales fundamentadas en el modelo relacional, se han constituido como potentes herramientas de análisis, pudiendo contener millones de datos que pueden ser operados a través de consultas que no son más que la implementación computacional de las operaciones definidas para el álgebra relacional. A través de la operación de selección se puede obtener un compendio de datos disponible, mediante la operación de proyección se pueden seleccionar datos con características especificadas, gracias al producto cartesiano se pueden realizar consultas cruzadas a distintas tablas y muchas otras operaciones permiten realizar diferentes manipulaciones a los datos disponibles. Se recomienda al lector la consulta a los referentes bibliográficos disponibles al final del presente documento, concernientes al álgebra relacional para poder profundizar en el tema.

El álgebra relacional se ha propuesto con anterioridad como método de análisis arquitectural. La autora Zude Li, en su trabajo –mencionado en la sección estado del arte- realizó la proposición y demostración matemática de un álgebra relacional extendida aplicable a arquitecturas de software. El modelo descrito en el trabajo de Zude Li es

estrictamente matemático y no está acompañado de un método que permita su aplicación en el campo del análisis arquitectural, pero sin embargo presenta la ventaja de estar demostrado matemáticamente por autores con los debidos conocimientos y la fundamentación teórica necesaria para realizar publicaciones académicas relacionadas con la temática. El álgebra relacional juega un papel importante en el trabajo propuesto por la autora Li, debido a que su principal elemento de manipulación son las relaciones que pueden existir entre distintos componentes de una arquitectura de software.

Tomando como un compendio los estudios publicados por la autora Zude Li, además de las diferentes herramientas que ofrece el álgebra relacional se puede definir un método que se convierta en una solución de análisis arquitectural en el mundo del software, puesto que se podrán manipular todas las entidades y relaciones que son la columna vertebral de una arquitectura en cualquier software debidamente modelado y construido. A partir de las relaciones se podrá obtener información de tipo estructural y de tipo medición de impactos, especialmente útiles para solventar las necesidades de tipo investigativo y propositivo dispuestas en el capítulo anterior del presente documento, puesto que a partir de la información estructural que se obtenga a través del análisis se podrá determinar que componentes se encuentran relacionados entre sí y a través de información relacionada con medición de impacto se puede fácilmente determinar que componente de la arquitectura analizada se vería afectado si se realizaran modificaciones en otro, gracias a que en el análisis estructural se pueden definir las relaciones existentes entre las entidades de una arquitectura de software.

El álgebra relacional se muestra entonces como una potente herramienta de análisis y manipulación de datos, que puede ser utilizada beneficiosamente en el desarrollo del método que se busca definir.

5.3. MÉTODO DE ANÁLISIS ARQUITECTURAL BASADO EN ÁLGEBRA RELACIONAL

A lo largo del presente documento se ha realizado un recorrido a través de un compendio de antecedentes, características y soluciones encaminadas al análisis de arquitecturas de software, propuestas por múltiples autores. Gracias a las conclusiones y recomendaciones encontradas en los diferentes trabajos estudiados durante la investigación, se tienen herramientas suficientes para proceder a la definición de un método propio, basado en álgebra relacional para realizar análisis a arquitecturas de software obtenidas –o no- a través de ingeniería inversa, aspecto que constituye el objetivo más importante del presente trabajo de grado.

En primer lugar es pertinente realizar una breve reseña acerca de los aspectos generales que tiene el método a definir, aspectos que pueden ser consultados en detalle en la sección de alcance incluida en el presente documento. Un método, está conformado por una serie de pasos encaminados a lograr una solución, a través de la manipulación de ciertos elementos que pueden representar cualquier entidad que esté en condiciones de ser abstraída (Sampieri Hernández, Fernández, & Baptista, 1996), pudiendo ser aplicados estos pasos en situaciones determinadas dependiendo de las necesidades que se presenten en un momento específico. El método a definir, como resultado de la investigación realizada, estará conformado por una serie de pasos que permitirán a un experto interesado en análisis arquitectural, realizar manipulación de entidades pertenecientes a una arquitectura de software obteniendo información útil para examinar aspectos estructurales y de medición de impacto en la arquitectura seleccionada. Se busca que el experto en software sea capaz de poder aplicar el método a distintas arquitecturas de software obteniendo resultados de calidad.

El método será descrito paso por paso, detalladamente, realizando las aclaraciones respectivas en el campo del álgebra relacional que previamente se definirá y delimitará con ayuda de uno de los trabajos estudiados y analizados durante la elaboración del capítulo del

estado del arte de la presente investigación. Gracias a esta demarcación del contexto matemático de la propuesta, los pasos del método podrán ser enfocados exclusivamente al campo de la teoría relacional mencionada en el capítulo inmediatamente anterior y que ha sido aplicada con éxito en el campo de las bases de datos relacionales.

La validación del método obtenido se realizó mediante la aplicación de los pasos descritos en un escenario real, conformado por una arquitectura de software perteneciente a un sistema. Se han realizado modificaciones a la arquitectura para aumentar su complejidad y poder abordarla de una manera más completa a través del método descrito en el presente trabajo. Los resultados obtenidos para la validación del método se presentan en un capítulo exclusivo para este aspecto.

5.3.1. Fundamentación matemática del método a partir del álgebra relacional propuesta en el trabajo de Zude Li y de la teoría relacional

En la sección de estado del arte, dentro de los distintos trabajos estudiados durante el proceso de revisión literaria, se encontró un artículo especialmente relacionado con la investigación en curso y muy beneficioso para esta, pues define un álgebra relacional extendida[6] completamente aplicable a las arquitecturas de software. La autora Zude Li es experta en el área de las matemáticas, por lo que su teoría se encuentra debidamente demostrada desde el punto de vista numérico, aspecto que se convierte en una ventaja para la presente iniciativa, pues proporciona bases sólidas y reales para fundamentar el método a definir.

En primer lugar la autora Li representa las arquitecturas de software como un grafo, siendo sus nodos la caracterización de las entidades que conforman una arquitectura de un sistema y sus aristas, las relaciones entre estas como se puede apreciar en la ilustración 6. La representación a través de grafos, es uno de los grandes aciertos que tiene el trabajo

[6] Es una variación del álgebra relacional con características especiales agregadas, aplicable a un contexto determinado, en este caso al análisis arquitectural.

realizado por la autora Zude Li puesto que permite expresar algebraicamente los componentes ilustrados, que representarían una arquitectura de software de un sistema.

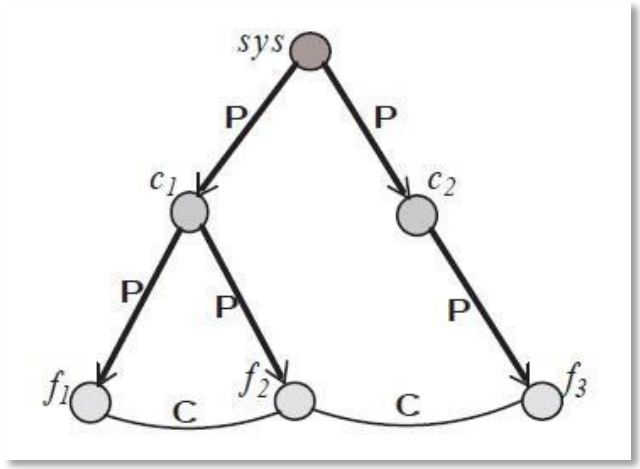

Ilustración 15. Representación gráfica para arquitecturas de software, propuesta por Zude Li. (Li, Gittens, Shariyar Murtaza, & Madhavji, 2010)

En la ilustración 15 se pueden apreciar diferentes elementos. En primer lugar, los nodos tienen diferentes denominaciones representadas como se observa en la tabla 6:

Tabla 6. Descripción de los nodos dispuestos en el grafo mostrado en la ilustración 6

Nomenclatura del nodo	Descripción
sys	Nodo principal, ubicado en la parte superior de la jerarquía, representa el sistema analizado.
c1, c2	Nodos intermedios, representan entidades de la arquitectura del sistema analizado que pueden tener o no, relaciones entre sí.
f1, f2, f3	Nodos inferiores, representan entidades de la arquitectura del sistema analizado que pueden tener o no, relaciones entre sí. Estructuralmente dependen de sus nodos padres.

La disposición jerárquica de los elementos dispuestos en el grafo propuesto, es especialmente útil para la representación de las relaciones entre las entidades. Las

relaciones mostradas constituyen el aspecto estructural de una arquitectura de software y son descritas en la tabla 7:

Tabla 7. Descripción de las aristas dispuestas en el grafo mostrado en la ilustración 15.

Nomenclatura de la arista (relación)	Descripción
P	Relaciones de herencia. Son representadas con la letra P en el estudio realizado por la autora Zude Li.
C	Relaciones de asociación. Son representadas con la letra C en el estudio realizado por la autora Zude Li. Pueden constituir relaciones de agregación, de composición o de dependencia.

Uno de los aspectos relevantes en las relaciones representadas en la ilustración 15 es la generalización de las distintas relaciones no jerárquicas en un solo tipo de relación denominado con la letra C. Esta representación se da porque en la abstracción de alto nivel, el nivel de detalle con miras a implementación no es relevante, como si lo es la estructura y disposición de los elementos (Larman, 2003).

La representación matemática de las relaciones entre aristas y nodos propuestos por Zude Li se realiza mediante el uso de tuplas, uno de los conceptos pertenecientes al álgebra relacional, que permite la representación de elementos relacionados entre sí especificando además el tipo de asociación que existe entre estos. Una tupla puede estar constituida por 1 o *n* elementos, permitiendo esta representación matemática la abstracción de todos los elementos pertenecientes a una arquitectura de software. Para el caso específico de la ilustración 15, se muestran a continuación el conjunto de elementos disponible para ser manipulados y las tuplas que pueden ser obtenidas a partir de estos:

Conjunto de entidades = *{sys, c1, c2, f1, f2, f3}*

Conjunto de tuplas para relaciones de tipo jerárquico "P" = *{<sys, c1>, <sys, c2>, <c1, f1>, <c1, f2>, <c2, f3>}*

Conjunto de tuplas para relaciones de tipo asociativo "C" = *{<f1, f2>, <f2, f3>}*

La representación mediante conjuntos de tuplas propuestos por Zude Li, resulta bastante sencilla de definir en un conjunto de tablas relacionales, manipulables a través de operaciones del álgebra relacional, aspecto que no es abordado en la investigación de la autora mencionada. Para el tema específico del caso diagramado en la ilustración 15, se propone la siguiente estructura:

Tabla 8. Descripción de las entidades que intervienen en la arquitectura.

Entidad	Descripción
sys	Entidad que representa el sistema estudiado
c1	Entidad *c1*
c2	Entidad *c2*
f1	Entidad *f1*
f2	Entidad *f2*
f3	Entidad *f3*

Tabla 9. Descripción de las relaciones que intervienen en la arquitectura.

Relación	Descripción
P	Relación de tipo jerárquico o hereditaria
C	Relación de tipo asociativo

Tabla 10. Tabla relacional para las entidades y sus relaciones, abstraídas de la representación matemática realizada por Zude Li.

Entidad 1	Entidad 2	Relación
sys	*c1*	P
sys	*c2*	P
c1	*f1*	P
c1	*f2*	P
c2	*f3*	P
f1	*f2*	C
f2	*f3*	C

En la tabla 8 se propone una estructura que representa las entidades pertenecientes al sistema abstraídas del grafo que constituye la arquitectura, en este caso el que se muestra en la ilustración 15; a su vez, en la tabla 9 se propone la estructuración relacional de una tabla para la descripción de las relaciones que pueden existir en la arquitectura estudiada. Este aspecto es una de las características extensibles del álgebra relacional, puesto que en el sistema pueden coexistir de 1 a n tipos de relaciones para las entidades estudiadas. Finalmente en la tabla 10, haciendo uso de una tabla relacional, se dispone la estructura para la representación tabular de las distintas entidades y las relaciones que intervengan con estas.

En el campo del álgebra relacional, la estructura relacional primeramente diagramada con ayuda de un grafo y luego dispuesta en tablas, puede manipularse a través de una operación perteneciente a la aritmética relacional, denominada operación de selección simbolizada a través de la letra griega "σ". La operación de selección dentro del álgebra relacional presenta la estructura mostrada en la siguiente ecuación:

$$\sigma_p(R)$$

De donde σ representa la selección de un determinado subconjunto de tuplas de una relación (R) –o tabla dentro de la teoría relacional- que satisfagan la condición *P*. Un ejemplo de aplicación del operador de selección para la relación estructurada en la tabla 10 es el siguiente:

$$\sigma_{Relación=P}(Tabla\ 10)$$

La anterior operación de selección permite obtener todas las tuplas que satisfagan la condición *Relación = P*, es decir, todas las tuplas en las que intervengan relaciones de tipo jerárquico. A su vez, también pueden especificarse relaciones de tipo asociativo representadas con la letra C en la tabla 10 para el operador de selección y en general, se pueden especificar condiciones determinadas por el experto en software, de acuerdo a la necesidad específica que tenga para la obtención de información.

Adicionalmente se pueden proyectar solo columnas específicas de la relación estudiada, gracias a la operación de proyección del álgebra relacional, representada a través del símbolo "∏" de la siguiente manera:

$$\prod_{Entidad\ 2} (\sigma_{Relación=P}(Tabla\ 10))$$

La ecuación anterior proporciona todos los registros que cumplen la condición "Relación = P" existentes en la tabla 10, pero esta vez solo mostrando los valores contenidos en la columna Entidad 2. De esta forma se puede simplificar la información obtenida como resultado de una operación de selección por ejemplo, anidándola con una operación de proyección.

El álgebra relacional también permite utilizar operaciones de unión y de intersección de forma similar al trabajo realizado en el campo de la teoría de conjuntos: es posible obtener registros cuyas condiciones sean satisfechas en una de las relaciones implicadas o en ambas (unión) o por el contrario, exclusivamente en ambas (intersección). Un breve ejemplo es mostrado en las siguientes dos ecuaciones:

$$\sigma_{Entidad\ 1=c1\ \wedge\ Entidad\ 2=f2}(Tabla\ 10) \cup\ \sigma_{Relación=P}(Tabla\ 10)$$

$$\sigma_{Entidad\ 1=sys}(Tabla\ 10) \cap\ \prod_{Relación}(\sigma_{Entidad\ 2\ =c1}(Tabla\ 10))$$

La primera ecuación es una operación de unión que permite obtener todos los registros que satisfagan cualquiera de las condiciones mostradas en la operación, es decir tanto los que cumplan el hecho de tener como *entidad 1* a *c1* y como *entidad 2* a *f2* además de todos los registros cuya columna *Relación* sea igual a *p*. Este tipo de operación se caracteriza por arrojar muchos registros como resultado debido a que no existe restricción para que las consultas relacionadas cumplan ambas –o todas en caso de que existan más de dos- las condiciones definidas.

La segunda ecuación por su parte es una operación de intersección. Solo va arrojar los registros que cumplan ambas condiciones, siendo la primera todos los que cumplan el

hecho de tener como *Entidad 1* al elemento *sys* y la segunda los que tengan como *Entidad 2* al elemento *c1*. Adicionalmente se puede observar una operación de proyección que anida la segunda selección para devolver únicamente la columna relación.

Otra de las posibilidades que brinda el álgebra relacional surge a través de la operación producto cartesiano del álgebra relacional. En un primer momento la operación no representa utilidad pues solo devuelve todas las concatenaciones posibles entre dos relaciones, esto es, si se poseen dos relaciones A y B el producto devolverá cada uno de los elementos de A concatenados con cada uno de los elementos de B. Esta operación se representa de la siguiente manera:

$$A \; X \; B$$

La utilidad real de la operación producto cartesiano del álgebra relacional se evidencia en una operación adicional que surge a partir de una extensión del producto algebraico que complementa el producto cartesiano, denominada unión natural o join. Este tipo de operación realiza el producto cartesiano –ya lo lleva implícito- de dos relaciones pero solo devolviendo los registros que cumplan una condición establecida en una operación de selección, que a su vez puede estar anidada bajo una operación de proyección, reduciendo de esta manera el volumen de registros devuelto como resultado de la operación y la pertinencia de la información obtenida. Un ejemplo de uso de la operación join es el siguiente:

$$\prod_{Tabla\ 10.Entidad\ 2, Tabla\ 9.Descripción} \left(\sigma_{Tabla\ 10.Entidad\ 2 = c2\ \vee\ Tabla\ 9.Relación = P}(Tabla\ 10\ |X|\ Tabla\ 9)\right)$$

En la anterior ecuación, se realiza el producto join entre la tabla 10 y la tabla 9 –su símbolo es $|X|$ para obtener las concatenaciones que satisfagan la condición establecida en la operación de selección que denota todos los registros cuya *Entidad 2* sea igual a *c2* dentro de la tabla 10 ó los registros cuya columna *Relación* sea igual a *P* dentro de la tabla 9. De todos los resultados obtenidos solo se proyectarán finalmente las columnas *Entidad 2* y *Descripción.*

Finalmente, dentro del álgebra relacional la operación de renombramiento representa gran utilidad y permite como lo indica su nombre, cambiar el nombre de una relación u operación para abreviar la escritura y hacer más entendibles las ecuaciones. Por ejemplo:

$$\rho_s(\sigma_{Tabla\ 10.Entidad\ 2=c2\ \vee\ Tabla\ 9.Relación=P}(Tabla\ 10\ |X|\ Tabla\ 9))$$

En la ecuación anterior, se renombra toda la operación de unión natural realizada a una variable denominada *s*. Esta variable tendrá exactamente el mismo valor y contenido que la operación del álgebra manipulada. El renombramiento en álgebra relacional se representa mediante el símbolo ρ.

Para el presente trabajo las operaciones de selección, proyección, renombramiento, unión, intersección y producto cartesiano a través de unión natural o join, así como el uso anidado de las mismas, serán las utilizadas al momento de realizar análisis matemático a las relaciones obtenidas a partir de la representación de la arquitectura de software analizada. Las operaciones de división, sustracción y agrupación también pueden ser utilizadas cuando se desee, pero para el escenario de validación de la investigación realizada no fueron tenidas en cuenta debido a que a partir de las características de la arquitectura seleccionada (Date, 2013) no proveían información relevante para los resultados. En la tabla 11 se resumen las operaciones, junto con su utilidad para el análisis arquitectural.

Tabla 11. Resumen de operaciones de álgebra relacional e información que proveen en el análisis arquitectural

Operación	Signo	Utilidad
Selección	σ	Permite obtener entidades de la arquitectura, que cumplan una condición.
Proyección	$\prod$	Permite seleccionar información particular del resultado de una operación de selección.
Unión	$\cup$	Agrupa los resultados de operaciones realizadas sobre arquitecturas.
Intersección	$\cap$	Provee elementos comunes entre arquitecturas y también entre entidades.
Sustracción	-	Obtiene entidades de una arquitectura luego de haber sustraído elementos comunes con otra.
División	/	Obtiene entidades de una arquitectura siempre que exista el mismo dominio en común con otra.

Agrupación	Ģ	Permite agrupar entidades en función de otra.
Producto Cartesiano	X	Permite obtener la concatenación de todos los elementos y relaciones de una arquitectura.
Unión Natural	$\|X\|$	Redefine el producto cartesiano, permite obtener la concatenación entre elementos definidos de la arquitectura, cumpliendo una o más condiciones de selección.
Renombramiento	ρ	Permite abreviar ecuaciones aplicadas a la arquitectura analizada.

Gracias a la teoría de álgebra relacional propuesta por la autora Zude Li se pueden representar arquitecturas de software en grafos. Dichos grafos pueden ser abstraídos en tablas denominadas relaciones utilizando los principios de la teoría relacional que actualmente opera en las bases de datos relacionales y posteriormente dichas relaciones pueden ser manipuladas mediante la operación de selección perteneciente a la aritmética del álgebra relacional para obtener información relevante que satisfaga una determinada condición como se ha mostrado en los anteriores párrafos. Toda la estructura matemática anteriormente descrita provee suficientes herramientas para proceder a la definición del método de análisis a arquitecturas de software que será descrito a continuación.

5.3.2. Descripción del método

A partir de la caracterización realizada en la sección anterior resulta más sencillo realizar la descripción de los pasos a seguir para el método a definir. Cada uno de los pasos tiene componentes pertenecientes a los descritos en la fundamentación matemática del método anteriormente analizada y son explicados con la mayor claridad y detalle posibles. Se recomienda al lector realizar un repaso a las secciones de marco teórico y fundamentación matemática del método disponibles en el presente documento para contextualizar las ideas previas a la lectura de forma correcta.

Paso 1: selección de la arquitectura del sistema y representación mediante grafos.

El paso inicial del método radica en la escogencia de la arquitectura a trabajar y su representación mediante nodos y aristas, abstrayendo un grafo. Este paso es de suma

importancia debido a que el éxito en la estructuración de las tablas relacionales que se realiza posteriormente, está ligado a un buen diseño del grafo generado a partir de la arquitectura.

El sistema escogido para análisis, debe tener una arquitectura perfectamente definida. Para el caso de las arquitecturas obtenidas a partir de ingeniería inversa, se debe haber realizado un proceso exitoso de obtención de vistas arquitectónicas mediante herramientas CASE para trabajos de ingeniería inversa. Se ilustra este paso y los siguientes con ayuda de la arquitectura de prueba, mostrada en la ilustración 16.

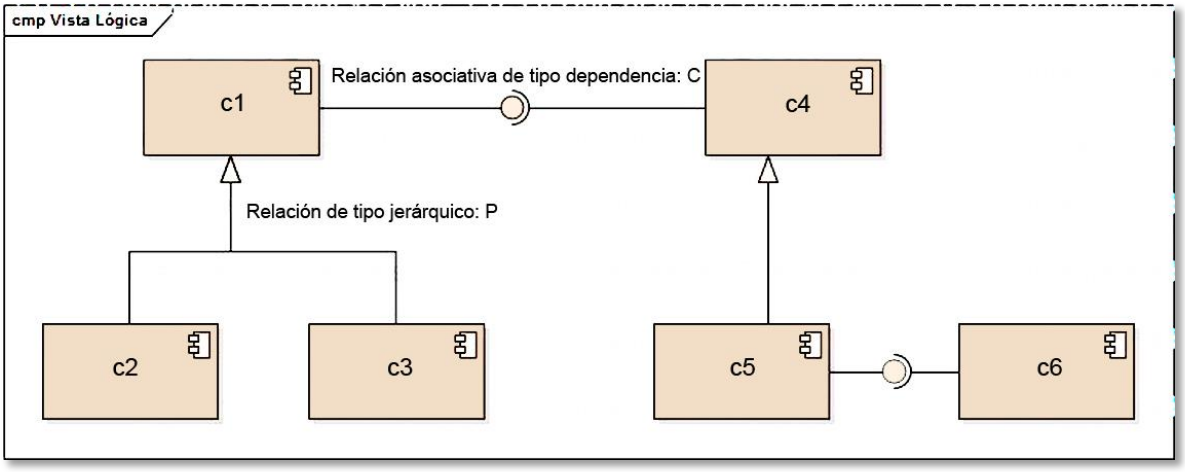

Ilustración 16. Arquitectura de software de prueba para realizar la descripción de los pasos del método (Elaboración propia)

Para la arquitectura mostrada en la ilustración 16 se obtienen los siguientes elementos:

- Nodo *sys*, que representa todo el sistema, como se propone en el modelo de Zude Li.
- Nodos *c1, c2, c3, c4, c5* y *c6* correspondientes a las entidades presentes en la arquitectura analizada.
- Relaciones de tipo jerárquico –herencia- representadas por la letra P y de tipo asociativo –dependencia- representadas por la letra C.

Llevando los anteriores elementos a la representación gráfica mediante nodos y aristas, se obtiene el siguiente el grafo propuesto en la ilustración 17.

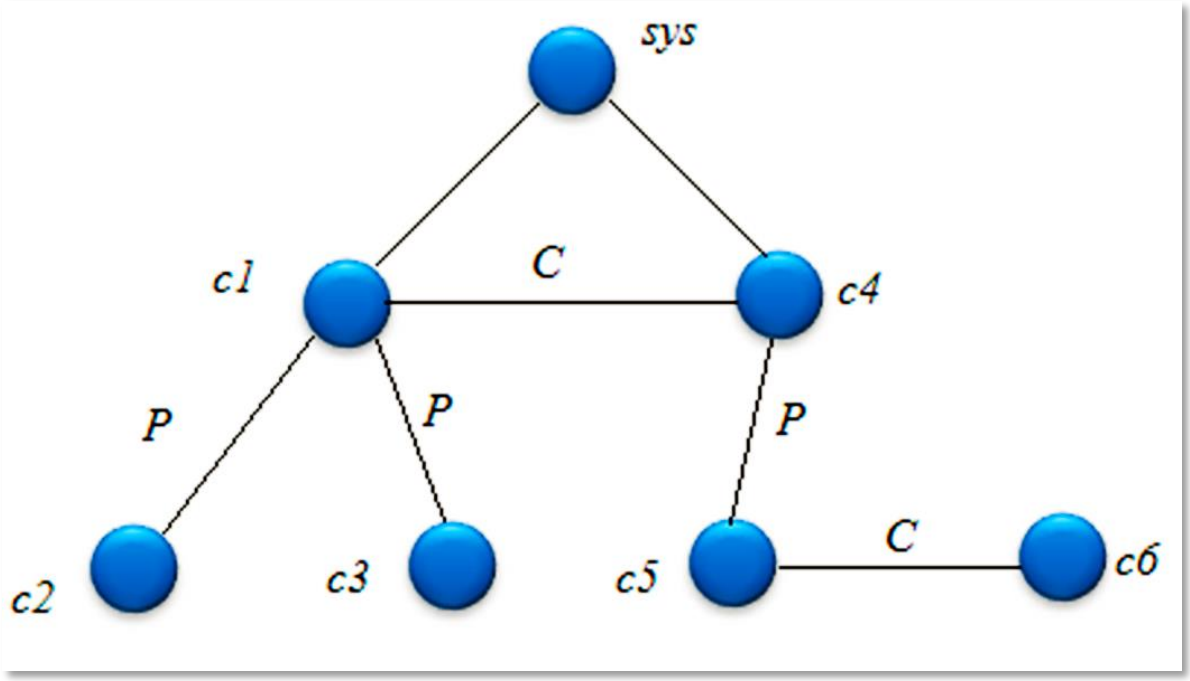

Ilustración 17. Representación de arquitectura de software mediante un grafo. (Elaboración propia)

Se debe realizar una verificación minuciosa para constatar que los elementos representados en el grafo tienen su correspondencia con cada una de las entidades existentes en la arquitectura analizada.

Es importante destacar que pueden darse casos en los que existan "arquitecturas dentro de arquitecturas", es decir, que un componente tenga incluidos dentro de sí varios nodos que puedan ser considerados como una arquitectura independiente. En este tipo de situaciones bastará con tratar como una arquitectura independiente los nodos que se consideren relevantes para el análisis realizado y podrá aplicarse el método propuesto de la misma manera que a cualquier otra arquitectura de software siempre y cuando dichos componentes tengan relaciones entre sí.

Al tener el grafo debidamente construido, se considerará el paso como exitoso y se puede continuar trabajando con el siguiente paso.

Paso 2: representación algebraica de relaciones entre entidades mediante tuplas.

La finalidad de este paso es la construcción de las tuplas con la estructura de las relaciones entre los distintos elementos que componen la arquitectura analizada. Las tuplas obtenidas serán de utilidad en el momento de la construcción de la tabla relacional que conjuga las entidades y sus respectivas relaciones, que será abordada en otro paso posterior. Es importante anotar que la entidad que representa el sistema, denominada *sys* no se tiene en cuenta en este paso, pues es utilizada únicamente para facilitar la disposición de la arquitectura del sistema en el grafo obtenido en el paso anterior.

A partir del grafo representado en la ilustración 17 se obtienen las siguientes relaciones:

- Existe una relación jerárquica entre los nodos *c1* y los nodos *c2* y *c3.*
- Existe una relación jerárquica entre los nodos *c4* y *c5.*
- Existe una relación asociativa –en este caso de dependencia- entre los nodos *c1* y *c4.*
- Existe una relación asociativa –en este caso de dependencia- entre los nodos *c5* y *c6.*

Representando cada una de las relaciones anteriores en tuplas se obtienen los siguientes conjuntos:

- Para las relaciones de tipo jerárquico, representadas con la letra P:
    - *P = {<c1, c2>, <c2, c3>, <c4, c5>}*
- Para las relaciones de tipo asociativo, representadas con la letra C:
    - *C = {<c1, c4>, <c5, c6>}*

Paso 3: construcción de tablas relacionales.

En este paso se construyen las tablas relaciones propuestas con anterioridad en la sección de fundamentación matemática del método. Se deben obtener tres tablas: la primera con la definición de cada una de las entidades –que en el grafo se han representado como nodos- y una breve descripción de las mismas; la segunda con la definición de las relaciones existentes en la arquitectura analizada y una breve descripción de estas y finalmente la tabla relacional construida a partir de la tabla que define las entidades y la tabla que define las relaciones.

Continuando con el ejemplo trabajado, se construye entonces la tabla que define las entidades como se puede apreciar en la tabla 12.

Tabla 12. Tabla para definición de entidades

Entidad	Descripción
sys	Entidad que representa el sistema
c1	Descripción para la entidad *c1*
c2	Descripción para la entidad *c2*
c3	Descripción para la entidad *c3*
c4	Descripción para la entidad *c4*
c5	Descripción para la entidad *c5*
c6	Descripción para la entidad *c6*

Posteriormente se debe continuar con la construcción de la tabla que define las relaciones existentes en el sistema. Los resultados han sido consignados en la tabla 13.

Tabla 13. Tabla para definición de relaciones

Relación	Descripción
P	Representación para las relaciones de tipo jerárquico o de herencia.
C	Representación para las relaciones de tipo asociativo,

	pudiendo ser dependencias, asociaciones, agregaciones simples o agregaciones por composición.

Finalmente se construye la tabla relacional que conjuga las entidades y las relaciones obtenidas gracias al manejo aplicado al grafo obtenido, utilizando principios de álgebra relacional. La tabla 14 muestra los resultados para esta sección.

Tabla 14. Tabla relacional que conjuga las entidades existentes en el sistema y sus relaciones

Entidad 1	Entidad 2	Relación
c1	*c2*	*P*
c2	*c3*	*P*
c4	*c5*	*P*
c1	*c4*	*C*
c5	*c6*	*C*

Paso 4: análisis de los datos mediante la operación de selección del álgebra relacional.

En este paso del método se realiza el análisis a los datos obtenidos y consignados anteriormente en tablas. Aplicando la operación de selección, perteneciente a la aritmética del álgebra relacional, se obtienen resultados que satisfacen condiciones establecidas por el experto en software que se encuentra llevando a cabo el proceso de análisis arquitectural.

Para el ejemplo que se viene trabajando se aplicarán 3 operaciones de selección que permitirán obtener datos concretos. Las dos primeras operaciones permiten obtener información de tipo estructural, pues seleccionan las entidades que se encuentran correspondidas mediante relaciones de tipo jerárquico y asociativo –dependencia- respectivamente y, la última operación, permite obtener información aplicable a análisis de medición de impacto pues selecciona las relaciones que involucren a la entidad *c1*.

En primer lugar, se realiza la operación de selección para obtener todas las relaciones de tipo jerárquico. Se procede a construir la siguiente ecuación:

$$\sigma_{Relación=P}(Tabla\ 14)$$

De donde *Relación=P* constituye la condición de solo seleccionar aquellas tuplas de datos en las cuales intervenga una relación de tipo jerárquico, existentes en la tabla –o relación, según la teoría relacional- 14. Aplicando dicha operación se obtienen los siguientes resultados, consignados en la tabla 15.

Tabla 15. Resultados de la operación de selección del álgebra relacional para relaciones de tipo jerárquico

Entidad 1	Entidad 2	Relación
c1	*c2*	*P*
c2	*c3*	*P*
c4	*c5*	*P*

Se continúa realizando la misma operación anterior, esta vez para las relaciones de tipo *C* mediante la variación de la condición que interviene en la operación, se anida bajo una operación de proyección que solo arroje como resultado la columna Entidad 1 para la información obtenida, lo que permitirá a la persona que realiza el análisis conocer la primera entidad involucrada para cada una de las relaciones de tipo asociativo. Adicionalmente se concatena con una operación de unión hacia una selección que permite obtener todos los registros cuya Entidad 2 sea *c4*. Se plantea entonces la siguiente ecuación:

$$\prod_{Entidad\ 1}\left(\sigma_{Relación=c}(Tabla\ 14)\right) \cup\ \sigma_{Entidad\ 2=c4}(Tabla\ 14)$$

De donde *Relación=C* constituye la condición de solo seleccionar aquellas tuplas de datos en las cuales intervenga una relación de tipo asociativo, existentes en la tabla –o relación,

según la teoría relacional- 14, mostrando en el resultado final solo los valores contenidos en la columna Entidad 1. Adicionalmente se anexarán todos los registros, con todas sus columnas, cuya entidad 2 sea c4 mostrando un ejemplo de unión. Los resultados obtenidos han sido consignados en la tabla 16.

Tabla 16. Resultados de la operación de selección del álgebra relacional para relaciones de tipo asociativo

Entidad 1	Entidad 2	Relación
c1		
c5		
c1	*c4*	*C*

Finalmente se aplica la última operación propuesta para el ejemplo que se ha venido trabajando, que permite obtener las tuplas en las que intervenga la entidad *c1*. Este tipo de selección permite realizar análisis de tipo medición de impacto, debido a que todos las entidades que se obtengan van a estar relacionadas con el elemento *c1* lo que significa que los cambios realizados en dichas entidades pueden afectar al elemento *c1* o viceversa, cambios realizados a la entidad *c1* pueden reflejarse en los elementos relacionados con dicha entidad. Adicionalmente se realizará una operación de intersección con la tabla 16 para determinar si la entidad *c1* solo tiene relaciones de tipo asociativo *C.*

Se construye entonces la operación de selección deseada. En primer lugar se utiliza la operación de renombrar para abreviar la escritura de la operación que produjo los resultados consignados en la tabla 16 en la variable *S*

$$\rho_S(\prod_{Entidad\ 1}(\sigma_{Relación=c}(Tabla\ 13)) \cup\ \sigma_{Entidad\ 2=c4}(Tabla\ 13))$$

Luego se realiza la operación de intersección deseada

$$\sigma_{Entidad\ 1=c1\ \vee\ Entidad\ 2=c2}(Tabla\ 14) \cap S$$

De donde *Entidad 1 = c1* ∨ *Entidad 2 = c2* es la condición establecida para seleccionar todas aquellas tuplas en las que intervenga la entidad *c1*, de las existentes en la tabla 14. Al aplicar esta operación, se obtienen los resultados descritos en la tabla 17.

Tabla 17. Resultados para la operación de selección con intervención de la entidad *c1*

Entidad 1	Entidad 2	Relación
c1	*c4*	*C*

El experto en software podrá aplicar todas las operaciones del álgebra relacional que considere necesarias a la tabla relacional que conjuga las entidades y sus relaciones, pudiendo obtener datos específicos que puedan satisfacer necesidades puntuales. Este aspecto es posible debido a que la condición existente en el operador de selección del álgebra relacional puede variar sin ninguna restricción y, al estar bien estructurada y manteniendo coherencia con los datos existentes, arrojará resultados beneficiosos para el proceso de análisis, adicionalmente se pueden anidar múltiples operaciones no solo de selección sino también de proyección, unión natural, renombramiento, producto cartesiano, unión e intersección para aumentar la pertinencia de la información.

Luego de haber realizado todas las operaciones de selección necesarias, se puede proceder con el paso final del método, que constituye la interpretación de los resultados.

Paso 5: interpretación de los resultados obtenidos

En este paso, los datos obtenidos con anterioridad son sometidos a análisis y el interesado en software que ha aplicado el método elabora sus conclusiones y determina el curso de acción a seguir. Este paso no está atado a ninguna estructura especial, puesto que el análisis realizado por una persona va a estar sujeto a muchas características propias del trabajo de esta, así como a necesidades puntuales.

Para ilustrar este paso con el ejemplo trabajado, se analizarán los resultados reflejados en las tablas 15, 16 y 17. En primer lugar las tablas 15 y 16 muestran que existe mayor número de relaciones de tipo jerárquico que de tipo asociativo –en este caso de dependencia-, lo que se puede traducir como una tendencia a mayores niveles de bajo acoplamiento en la arquitectura analizada, pues entre menor número de relaciones de dependencia el acoplamiento entre componentes disminuye (Gorton, 2006). La tabla 17 muestra que el componente *c1* tiene una relación de asociación con el componente *c4*, este aspecto es crítico, puesto que todos los cambios realizados en el componente *c1* van a afectar el funcionamiento del componente *c4* pues este utiliza servicios proveídos por *c1*. La operación de intersección aplicada permitió obtener pocos registros que satisfacen la información que realmente se necesitaba conocer.

En el párrafo anterior se refleja un análisis realizado a los resultados obtenidos luego del proceso de aplicación de los pasos del método definido. El análisis consignado ha sido realizado a manera de ilustración por el autor, basado en necesidades de ejemplo para mostrar la pertinencia del paso de interpretación de resultados. Se recomienda al lector la aplicación de preguntas de análisis distintas a las consignadas en el documento, pues no son las únicas. Este aspecto depende exclusivamente de las necesidades de análisis que se tengan en un determinado momento, de la construcción adecuada de la condición para la operación de selección y de la creatividad que tenga el experto en software para analizar la arquitectura.

En el siguiente capítulo, se realizará la validación del método propuesto, aplicándolo a un escenario real para determinar su pertinencia científica, dependiendo de los resultados obtenidos.

5.4. VALIDACIÓN DEL MÉTODO PROPUESTO

5.4.1. Descripción del caso de prueba

Durante el primer período académico del año 2011, se realizó un proyecto de aula perteneciente a la asignatura de Ingeniería del Software impartida en el programa de Ingeniería De Sistemas de la Universidad De Cartagena titulado ***Detección y Notificación de Eventos Fortuitos a través de Dispositivos Móviles*** bajo la dirección del docente de planta de la Universidad De Cartagena Ing. Msc. Martín Emilio Monroy Ríos, asesoramiento del también docente de planta de la Universidad De Cartagena Dr. Julio César Rodríguez Ribón y desarrollado por el grupo de trabajo conformado por los estudiantes del programa de Ingeniería De Sistemas de la Universidad De Cartagena Arturo Verbel De León, Luis Miguel Mejía y el autor del presente documento Jorge Osorio Romero.

La funcionalidad del proyecto consistió en el envío de coordenadas geográficas mediante un dispositivo móvil que reaccionaba a movimientos bruscos, ubicables en un mapa gracias a un sistema de información web. El software tiene múltiples escenarios de ejecución debido a la intervención de distintos elementos hardware tales como dispositivos móviles y computadoras, además de servidores de aplicación y de bases de datos, lo que trae como resultado una arquitectura de software rica en componentes y entidades.

Durante el diseño de la solución se optó por la construcción de una arquitectura orientada a servicios, puesto que este tipo de estructura permite la convivencia de tecnologías y herramientas de desarrollo que difieren entre sí y que se ejecutan en plataformas distintas en despliegue interno y funcionamiento. Las arquitecturas orientadas a servicios están generalmente conformadas por múltiples componentes que atienden los requerimientos funcionales de la solución, agrupados en paquetes y representados mediante un diagrama de componentes pertenecientes al lenguaje unificado de modelado –UML-, conteniendo la representación de relaciones entre cada uno de los elementos de la arquitectura.

Para la aplicación del método definido en la presente investigación se optó por trabajar con la arquitectura del sistema previamente descrito, puesto que pertenece a una solución funcional cuyo funcionamiento fue probado a nivel de despliegue en escenarios de prueba del mundo real, aspecto que permitió la participación de los autores como ponentes de póster en el ***Congreso Nacional De Ingeniería De Sistemas*** realizado en el mes de Octubre del año 2011 en la ciudad de Barranquilla, Colombia.

El proceso a seguir durante la validación del método, constó de los siguientes pasos:

- Descripción de la arquitectura de software seleccionada.
- Aplicación del método de análisis definido en el presente trabajo, obtenido como fruto de la investigación realizada, incluyendo manipulación de la información obtenida a través de operaciones del álgebra relacional tales como selección, proyección, intersección, unión, unión natural o join, producto cartesiano y renombramiento.
- Interpretación de resultados.

Es importante resaltar, que la arquitectura del sistema en su momento fue obtenida a través de un proceso de ingeniería inversa, realizado a la solución final a través de una herramienta de diseño asistido por computadora –CASE- llamada Enterprise Architect[7] que facilitó la obtención de los distintos componentes a nivel de diseño e implementación, mediante los cuales fue posible el diagramado final de los componentes pertenecientes a la arquitectura previamente definida que aún no habían sido evidenciados en los diagramas pertinentes.

[7] Herramienta CASE perteneciente a la compañía Sparx Systems. Permite la manipulación de diagramas UML. Al ser un producto costoso, se optó por el uso de una versión de prueba.

5.4.2. Descripción de la arquitectura seleccionada

En la ilustración 18, se muestra el diagrama de componentes que representa la arquitectura de software del sistema escogido como caso de prueba para la validación del método definido en el presente proyecto de grado.

Al ser una arquitectura orientada a servicios, los componentes se encuentran agrupados en paquetes que contienen funcionalidades orientadas a la satisfacción de requerimientos funcionalmente parecidos entre sí. En primer lugar se observa un paquete denominado *vista* encargado de mantener las interfaces gráficas del sistema analizado, el cual contiene cuatro componentes funcionalmente independientes entre sí –aspecto que garantiza el cumplimiento del patrón bajo acoplamiento[8]- sin la existencia de relaciones entre estos. Para efectos de mayor facilidad y eficiencia a la hora de aplicar el método, este paquete ha sido tomado como una sola entidad *c1* puesto que la ausencia de relaciones internas entre sus componentes hace indiferente el manejo de los mismos como una unidad o independientemente unos de otros.

Posteriormente se observa un paquete denominado *controlador* que contiene dos componentes de la arquitectura del sistema. El paquete controlador como unidad, tiene una relación de tipo asociativo –en este caso, de dependencia- con la entidad *c1* definida anteriormente, a su vez sus componentes internos tienen relaciones de dependencia con otras entidades de la arquitectura. En este caso, el paquete *controlador* se tomó de forma mixta proporcionando la existencia de las entidades *c2* que representa la relación de dependencia con la entidad *c1* y la entidad *c3* que tiene relaciones de dependencia con otros elementos de la arquitectura. El paquete *modelo* de la arquitectura ilustrada solo tiene una entidad que se denominó *c4* y no representó mayor dificultad para su abstracción como entidad.

[8] Patrón de diseño de software perteneciente al conjunto de patrones GRASP que busca el mínimo grado de acoplamiento y dependencia entre entidades del software.

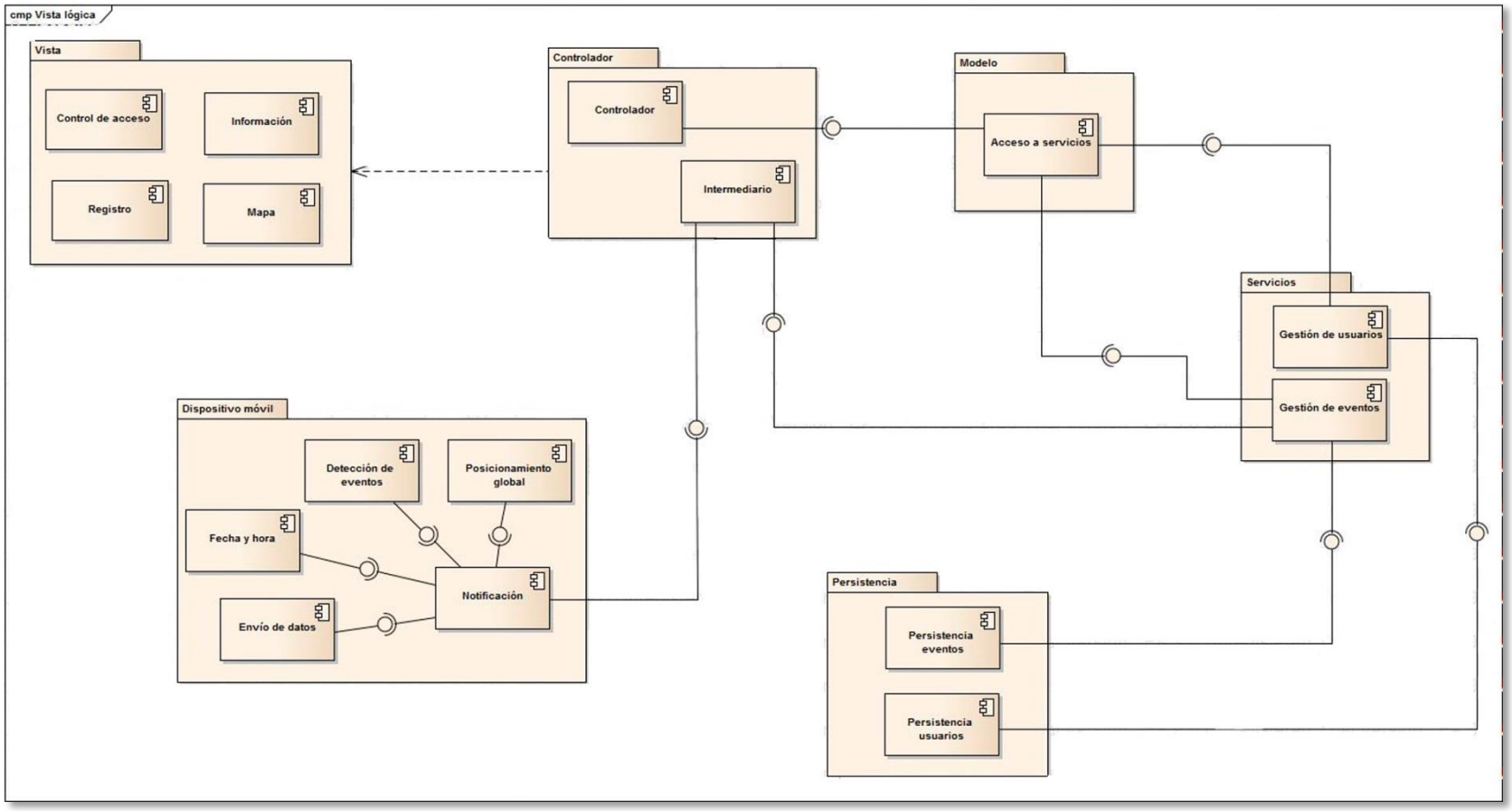

Ilustración 18. Vista lógica que representa la arquitectura del sistema escogido para validar el método propuesto. (Elaboración propia)

Adicionalmente el paquete *servicios* posee dos entidades que han sido denominadas *c5* y *c6*, al igual que el paquete *persistencia* que posee dos entidades que en el caso de prueba se denominaron *c7* y *c8.* Finalmente el paquete *dispositivo móvil* posee cinco entidades, a saber, nombradas *c9, c10, c11, c12* y *c13.*

El resumen de la abstracción de entidades realizada a la arquitectura se encuentra consignado en la tabla 18.

Tabla 18. Resumen de las entidades encontradas en la arquitectura analizada y su respectiva abstracción

Nombre en el diagrama	Descripción	Denominación luego de abstracción
Vista	Mantiene las interfaces gráficas de usuario. Contiene cuatro componentes y es tomado como una unidad dado que los nodos internos (Control de acceso, información, registro y mapa) no resultan relevantes para el estudio puesto que no tienen relaciones entre sí, optándose por no tomarlos como una arquitectura independiente.	*c1*
Controlador	Responde a los eventos acontecidos desde la interfaz gráfica. Se maneja de modo mixto, atendiendo el componente *controlador* la dependencia del paquete del mismo nombre como una unidad y el paquete *intermediario* de manera independiente.	*c2*, para el componente *controlador*. *c3*, para el componente *intermediario*.
Modelo	Contiene los requerimientos de acceso a servicios web. Está conformado por un único componente.	*c4*, para el componente *acceso a servicios*.

Servicios	Tiene dos componentes encargados de la interacción con los servicios web del sistema.	*c5*, para el componente *gestión de usuarios*. *c6*, para el componente *gestión de eventos*.
Persistencia	Entidad encargada del manejo de datos. Contiene dos componentes.	*c7*, para el componente *persistencia eventos.* *c8*, para el componente *persistencia usuarios.*
Dispositivo móvil	Se encarga de las funcionalidades ejecutadas en el dispositivo móvil que interviene en el sistema.	*c9*, para el componente *notificación.* *c10*, para el componente *posicionamiento global.* *c11*, para el componente *detección de eventos.* *c12*, para el componente *fecha y hora.* *c13*, para el componente *envío de datos.*

5.4.3. Aplicación del método de análisis arquitectural

Paso 1: selección de la arquitectura del sistema y representación mediante grafos.

La selección de la arquitectura del sistema, descripción de la misma e incluso abstracción de las entidades que existen en esta ha sido realizada con anterioridad en el ítem 6.4.2 del presente documento. El paso faltante es la representación mediante un grafo, para lo cual se incluye la ilustración 19 en la cual se ha representado mediante nodos y aristas la arquitectura seleccionada como caso de prueba.

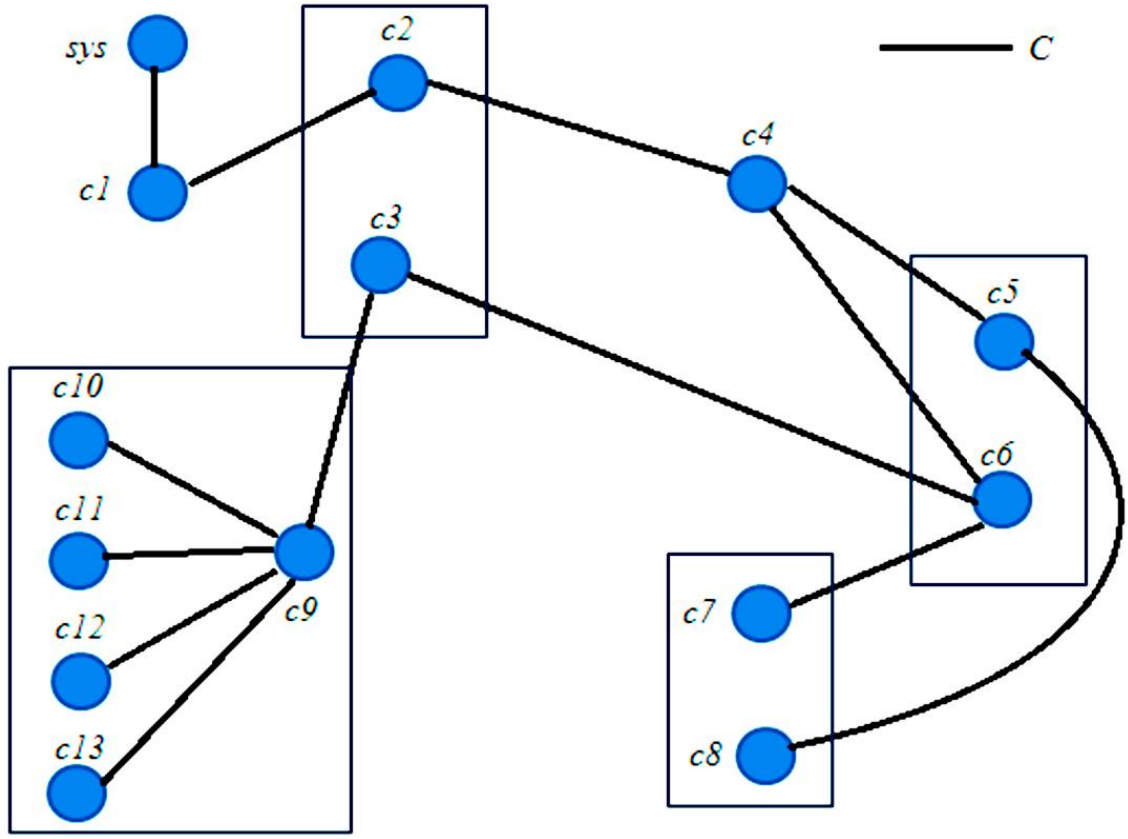

Ilustración 19. Representación de la arquitectura de software del sistema caso de prueba en un grafo (Elaboración propia)

El aspecto más notorio que difiere con el ejemplo trabajado durante la descripción del método, es que en la arquitectura de software analizada hay ausencia de relaciones de tipo jerárquico denominadas con la letra *P* siendo entonces todas las relaciones entre entidades de tipo asociativo –en este caso, dependencia- representadas con la letra *C.* Las relaciones de tipo *P* son muy poco comunes a nivel de arquitecturas de software (Gorton, 2006), siendo más comunes a nivel de diseño e implementación y se utilizan mayormente para relaciones entre clases y no entre componentes de una arquitectura. Para el escenario de prueba validado no se incluyen relaciones de este tipo puesto que se estaría incurriendo en modificaciones a la arquitectura previamente desplegada en el mundo real, sin haberla puesto a prueba.

Paso 2: representación algebraica de relaciones entre entidades mediante tuplas.

Habiendo realizado la abstracción a la arquitectura, representando sus entidades y relaciones en el grafo mostrado en la ilustración 19 se procedió a representar matemáticamente, utilizando notación propia del álgebra relacional extendida propuesta por la autora Zude Li, la entidades y sus relaciones.
En primer lugar se tiene el conjunto de entidades que conforman en el sistema, incluyendo por defecto el elemento *sys* que representa el software como un todo.

$$Entidades = \{sys, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10.\ c11, c12, c13\}$$

Posteriormente, se representan las relaciones existentes en el sistema. Como se mencionó anteriormente, en este caso particular solo existen relaciones de dependencia que son representadas por el carácter *C*. Por lo tanto el conjunto de relaciones es unitario y está representado de la siguiente manera:

$$Relaciones = \{C\}$$

Finalmente se representan mediante tuplas las entidades que tienen relación entre sí, obteniendo el siguiente conjunto:

$$Relaciones\ tipo\ C = \{<c1, c2>, <c2, c4>, <c3, c9>, <c3, c6>, <c4, c5>, <c4, c6>, <c5, c8>, <c6, c7>, <c9, c10>, <c9, c11>, <c9, c12>, <c9, c13>,\}$$

Paso 3: construcción de tablas relacionales.

Se continuó con la construcción de las tablas relacionales, a partir de la abstracción algebraico-relacional realizada a la arquitectura analizada. Los resultados de este paso han sido consignados en las tablas 19, 20 y 21 para la definición de entidades, relaciones existentes en el sistema y relaciones entre componentes respectivamente.

Tabla 19. Definición de entidades para la arquitectura del caso de prueba

Entidad	Descripción
sys	Entidad que representa el sistema.
c1	Mantiene las interfaces gráficas.
c2	Responde a los eventos de la interfaz gráfica.
c3	Intermediario entre eventos y acceso a servicios.
c4	Componente de acceso a servicios.
c5	Gestión de usuarios.
c6	Gestión de eventos.
c7	Componente de gestión de persistencia para eventos.
c8	Componente de gestión de persistencia para usuarios.
c9	Componente de notificación.
c10	Posicionamiento global.
c11	Detección de eventos.
c12	Fecha y hora.
c13	Envío de datos.

Tabla 20. Relaciones existentes en el sistema

Relación	Descripción
C	Representación para las relaciones de tipo asociativo, pudiendo ser dependencias, asociaciones, agregaciones simples o agregaciones por composición.

Tabla 21. Relaciones entre componentes dentro del sistema

Entidad 1	Entidad 2	Relación
c1	*c2*	*C*
c2	*c4*	*C*
c3	*c9*	*C*
c3	*c6*	*C*
c4	*c5*	*C*

c4	*c6*	*C*
c5	*c8*	*C*
c6	*c7*	*C*
c9	*c10*	*C*
c9	*c11*	*C*
c9	*c12*	*C*
c9	*c13*	*C*

Paso 4: análisis de los datos mediante el uso del álgebra relacional.

En el caso de prueba analizado, se plantearon 3 preguntas que surgieron a partir de dudas generadas en el proceso de construcción del proyecto desarrollado en la asignatura de ingeniería de software.

La primera inquietud que se tenía era determinar qué entidad o entidades de la arquitectura interactuaban directamente con el dispositivo móvil para realizar modificaciones a la forma como se enviaban los datos entre las plataformas operativas con miras a agilizar los procesos; la segunda inquietud radicaba en el manejo de fecha y hora dentro del dispositivo con miras a determinar qué entidad interactuaba con el componente encargado de este punto apuntando a una posible eliminación del mismo y finalmente la última inquietud radicó en determinar si la persistencia estaba siendo manipulada a través de componentes de servicios o no. Esta información se obtuvo mediante la aplicación de operaciones de selección, haciendo uso de la aritmética del álgebra relacional.

Para la primera inquietud, se tiene un componente *c9* que proporciona acceso a todas las entidades del dispositivo móvil ilustradas en la arquitectura del sistema, por lo tanto todas las tuplas que contengan a *c9* estarán interactuando con el dispositivo móvil; sin embargo la operación debe anidarse bajo una consulta de proyección para solo obtener los valores de las entidades puesto que la arquitectura solo maneja relaciones de tipo *C* por lo tanto esa

información será la misma para todos los registros. Adicionalmente se realiza una intersección con el resultado de la unión natural del resultado obtenido con la tabla 18 – tabla de entidades- para obtener la descripción de la función de las entidades afectadas. La operación de selección quedó conformada de la siguiente manera, arrojando los resultados consignados en la tabla 22:

$$\rho_S\{(\prod_{Entidad\ 1,Entidad\ 2,Descripción}(\sigma_{Entidad\ 1=c9}(Tabla\ 21))$$
$$\sigma_{Tabla\ 20.Relacion=C\ \wedge\ Tabla\ 19.Relacion=c}(Tabla\ 21\ |X|Tabla\ 20))$$
$$\cup$$
$$(\prod_{Entidad\ 1,Entidad\ 2,Descripción}(\sigma_{Entidad\ 2=c9}(Tabla\ 21))$$
$$\sigma_{Tabla\ 20.Relacion=C\ \wedge\ Tabla\ 19.Relacion=c}(Tabla\ 21\ |X|Tabla\ 20))\}$$
$$\cap \prod_{Entidad,Descripción}$$
$$(\sigma_{\substack{(S.Entidad\ 1=c9\ V\ S.Entidad\ 2=c9)\\ \wedge(Tabla\ 18.Entidad=S.Entidad\ 1\ \vee\ Tabla\ 18.Entidad=S.Entidad\ 2)}}(Tabla\ 19\ |X|S))$$

Tabla 22. Resultados para la operación realizada con condición c9

Entidad	Descripción
c9	*Componente de notificación*
c10	*Posicionamiento global*
c11	*Detección de eventos*
c12	*Fecha y hora*
c13	*Envío de datos*

Para la segunda inquietud planteada, se tiene la intervención del componente de fecha y hora denominado como *c12.* La operación de selección quedó conformada como se muestra a continuación y los resultados de la aplicación de la misma han sido consignados en la tabla 23.

$$\rho_S(\prod_{Entidad\ 1,Entidad\ 2}(\sigma_{Entidad\ 1=c12\ \vee\ Entidad\ 2=c12}(Tabla\ 21)))$$

$$\prod_{Entidad,Descripción}\{\sigma \quad (S.Entidad\ 1{=}C12 \vee S.Entidad\ 2 = C12) \wedge (S.Entidad\ 1{=}Tabla\ 19.Entidad\ 1 \vee S.Entidad\ 2 = Tabla\ 19.Entidad\ 2) (Tabla\ 19\ |X|S)\}$$

Tabla 23. Resultados para la operación de selección con condición *c12*

Entidad	Descripción
c9	*Componente de notificación*

Para la última inquietud se debía determinar si la manipulación de la persistencia se estaba realizando de forma correcta, es decir, mediante entidades pertenecientes al paquete de servicios. Estas entidades son respectivamente *c5* y *c6* y las entidades de persistencia son respectivamente *c7* y *c8.* La operación de selección se dividió en dos estructurada como se muestra a continuación y los resultados de las mismas fueron consignados en las tablas 24 y 25 respectivamente.

Para la entidad de persistencia *c7*, solo se debería tener acceso a través de *c5* o de *c6:*

$$\rho_S\{(Entidad\ 1 = c5\ \vee\ Entidad\ 2 = c5\ )\ \vee\ (Entidad\ 1 = c6\ \vee\ Entidad\ 2 = c6\ )\}$$

$$\prod_{Entidad\ 1,Entidad\ 2,Relación} (\sigma_{(Entidad\ 1{=}c7 \vee Entidad\ 2{=}c7) \wedge S}\ (Tabla\ 21))$$

Tabla 24. Resultados para la operación de selección con condición *c7*

Entidad 1	Entidad 2	Relación
c6	*c7*	*C*

Para la entidad de persistencia *c8*, solo se debería tener acceso a través de *c5* o de *c6:*

$$\rho_S\{(Entidad\ 1 = c5\ \vee\ Entidad\ 2 = c5\ )\ \vee\ (Entidad\ 1 = c6\ \vee\ Entidad\ 2 = c6\ )\}$$

$$\prod_{Entidad\ 1,Entidad\ 2,Relación} (\sigma_{(Entidad\ 1{=}c8 \vee Entidad\ 2{=}c8) \wedge S}\ (Tabla\ 21))$$

Tabla 25. Resultados para la operación de selección con condición *c8*

Entidad 1	Entidad 2	Relación
c5	*c8*	*C*

Paso 5: interpretación de los datos obtenidos.

Gracias a la aplicación del método de análisis arquitectural definido a lo largo de la presente investigación, se obtuvieron datos relevantes para las 3 preguntas que se plantearon antes de someter a análisis la arquitectura de software del sistema seleccionado para el caso de prueba.

En primer lugar, se pudo determinar con facilidad mediante los resultados consignados en la tabla 22 las entidades que interactúan con el componente del dispositivo móvil existente en el sistema. Gracias a la aritmética relacional se obtuvieron los componentes relacionados en las tablas construidas con la entidad que permite el acceso al dispositivo móvil. A partir de las entidades encontradas y utilizando la información disponible en la tabla 19 que describe los componentes pertenecientes a la arquitectura se puede efectuar un curso a seguir directamente sobre las entidades de software que interactúan con el dispositivo móvil, teniendo seguridad de que las modificaciones realizadas solo afectarán a los componentes relacionados entre sí.

De la misma manera descrita en el párrafo anterior, se obtuvieron resultados similares que muestran los componentes relacionados con la entidad de manejo de fecha y hora, mostrados en la tabla 23.

Finalmente en las tablas 24 y 25, se muestran las entidades que interactúan con los componentes de persistencia. Estas entidades permiten determinar si el sistema está construido de una manera adecuada, puesto que solo a través de las entidades de acceso a datos se deberían manipular los componentes de persistencia. Los resultados obtenidos

arrojan un balance positivo, puesto que las entidades obtenidas son en efecto las destinadas a comportarse como componentes de acceso a datos.

Adicionalmente conociendo las entidades particulares requeridas y las interacciones que tienen con otros componentes, se pudieron establecer cursos de acción a seguir basados en medición de impacto; es decir, los costos en cuanto a tiempo y funcionalidad que tendrían los cambios aplicados en determinadas entidades, reflejándose en componentes relacionados.

Toda la información obtenida pudo ser sometida a distintos tipos de análisis puntual, sin embargo la interpretación fue resumida a las tres preguntas puntuales que se plantearon como escenario de prueba.

5.5. ANÁLISIS DE RESULTADOS

Luego de haber realizado la presente investigación, se presenta la tabla 26 que conjuga las similitudes entre los resultados obtenidos y las propuestas existentes en la actualidad para el campo del análisis arquitectural.

Tabla 26. Comparativo entre resultados obtenidos y publicaciones existentes

Resultado Obtenido	¿Existen estudios con resultados similares?	Publicación de mayor similitud con el trabajo realizado
Método de análisis arquitectural basado en álgebra relacional	Sí. Adicionalmente no solo se centran en álgebra relacional sino en distintas ramas de la matemática	Li, Z et al (2010). Towards an Extended Relational Algebra for Software Architecture.
Estado del arte de análisis arquitectural	Sí. En este punto muchos trabajos están encaminados a contextos específicos y pocos son generalizados.	Panas, T et al (2003). Towards the Unified Recovery Architecture for Reverse Engineering.
Estudio de necesidades existentes en el campo del análisis	Aunque existen trabajos similares no se encontró en el mapeo	Acher, M. et al (2011). Reverse Engineering Architectural Feature

arquitectural.	sistemático de la información realizado, una publicación que se encargara de trabajar exactamente este punto.	Models.

La tabla 26 ha sido restringida a los 3 resultados del proyecto, posterior a su realización y las publicaciones que más afinidad presentan con estos. En la tabla 5, el lector podrá encontrar otras publicaciones que fueron utilizadas para el trabajo realizado y que fundamentaron la investigación.

Tal y como se ha recalcado a lo largo de todo el documento, al ser una investigación de tipo exploratoria debido a que la temática expuesta ha sido poco abordada en la actualidad, no se encontraron muchos referentes relacionados directamente con los resultados que se obtuvieron aunque si se encontraron publicaciones que presentan similitudes con el trabajo realizado. Este punto se justifica en que la búsqueda estuvo restringida a las bases de datos accesibles a través de la Universidad De Cartagena y no a otros recursos investigativos de carácter privado que muy seguramente hubieran ofrecido mayores resultados que hubieran permitido fundamentar aún más los resultados obtenidos a partir del trabajo realizado.

En términos generales se puede afirmar que los resultados coinciden con la literatura previa, tanto desde el punto de vista matemático como desde el punto de vista de análisis propuesto. Matemáticamente el álgebra relacional se encuentra presente en varias de las investigaciones utilizadas como referente y fue una herramienta fundamental para el cumplimiento de los objetivos de la investigación realizada; sin embargo, se debe recalcar que en la literatura previa los autores no solo se limitan a utilizar conceptos de álgebra relacional sino también de otros campos de las matemáticas.

A partir de la estructura general utilizada para publicación de artículos científicos, se obtuvieron buenos referentes en cuanto a estados del arte de la temática se refiere, lo que facilitó la redacción del resultado presentado como estado del arte en el presente

documento. Los puntos abordados por el mismo coinciden grandemente con el análisis de estado del arte realizado por autores consultados cuyas publicaciones existen en la literatura académica.

Finalmente fueron pocos los autores cuyos estudios coincidieron con el resultado de clasificación de necesidades encontradas en el campo de análisis arquitectural, obedeciendo esto a que a partir de los criterios del autor se pueden realizar muchos tipos de clasificaciones dependiendo del contexto estudiado y nivel académico adquirido a través de los años.

6. CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

El análisis arquitectural ha emergido como una de las principales temáticas a futuro que concentrará los esfuerzos científicos aplicados en la ingeniería de software, los múltiples resultados obtenidos durante toda la investigación realizada fundamentados en referentes bibliográficos aceptados por la comunidad científica así lo demuestran.

El enfoque matemático puede proporcionar una salida a la necesidad de análisis arquitectural en el campo de las arquitecturas de software pero antes de llegar a una solución definitiva, múltiples propuestas y enfoques serán abordados lo que enriquecerá el resultado final una vez se consume.

Para la investigación realizada se cumplieron los objetivos planteados. En primer lugar se obtuvo como resultado un estado del arte debidamente construido y organizado que sin duda servirá a futuros investigadores del área como punto de partida para el desarrollo de sus proyectos. Sumado a esto en el presente documento se tiene a disposición una relación de necesidades encontradas a partir de la revisión de la literatura realizada, con una clasificación de las mismas según su tipo lo que constituye el cumplimiento del segundo objetivo del presente proyecto. Finalmente se logró la definición de un método basado en el álgebra relacional para realizar análisis arquitectural, el cual fue validado en un escenario de prueba pertinente dando cumplimiento a los objetivos cuarto y quinto respectivamente de los planteados para la investigación.

Cabe mencionar que los resultados obtenidos presentan bastante afinidad con el trabajo científico de análisis relacional que habían realizado algunos autores, el método definido proporciona información proveniente del análisis arquitectural realizado que se complementa fácilmente con resultados encontrados en la revisión literaria coincidiendo de esta manera con estudios que existían previamente en el área de la arquitectura de software.

Es importante destacar que la investigación se vio limitada por varios factores. En primer lugar el factor económico restringió el acceso únicamente a bases de datos gratuitas proporcionadas por la Universidad De Cartagena, que si bien fueron muy útiles, seguramente no contienen trabajos más específicos y mejor realizados que pudieron haber sido incluidos en la investigación. En segundo lugar el factor tiempo fue otro limitante puesto que había un cronograma que cumplir y no se podían seguir prolongando las fechas de entrega. Finalmente el recurso humano fue otro de los limitantes de la investigación que solo recayó en el director del proyecto y el investigador principal debido a la falta de interés por parte de otros estudiantes del Programa De Ingeniería De Sistemas por abordarla.

6.2. RECOMENDACIONES

Para trabajos futuros se recomienda la indagación en dos puntos que pueden complementar y aumentar la calidad del resultado obtenido en el proyecto realizado.

En primer lugar, el término *"arquitectura dentro de arquitectura"* mencionado en el capítulo 5 puede constituir una interesante temática de investigación encaminada a determinar si pueden existir o no relaciones de inclusión entre arquitecturas de software y a su vez ser analizadas, formando parte del proceso que se aplica en el método descrito, lo que sin dudas robustecería los resultados obtenidos.

Por otra parte el uso de grafos en el análisis arquitectural puede ser más específico mediante el uso de grafos con aristas dirigidas, simulando las relaciones existentes en una arquitectura de software diagramadas en UML que van dirigidas de un nodo a otro e incluso de forma bidireccional.

A partir de los objetivos específicos se tienen varias recomendaciones. En primer lugar, construir un protocolo debidamente estructurado para mapeo sistemático de literatura al momento de realizar un estado del arte puesto que la herramienta agilizó el trabajo de manera bastante notoria. Por otro lado la definición de un caso de prueba es fundamental a la hora de proponer trabajos de investigación documentales, pues permite la validación de

los resultados obtenidos. El escenario de prueba debe ser previamente validado por el director del proyecto y tener su visto bueno para obtener los mejores resultados.

Adicionalmente, partiendo del objetivo específico de definición de un método basado en álgebra relacional, es muy importante buscar los fundamentos matemáticos necesarios para justificar la solución que se proponga pues los resultados de la investigación deben estar basados en autores académicos reconocidos con publicaciones indexadas en bases de datos, posiblemente del área de las matemáticas y ciencias exactas.

Una última recomendación fruto de la experiencia del autor es que en la medida de lo posible, la temática del análisis arquitectural no debería abordarse de manera individual. Sería beneficioso que al menos dos personas investigaran la misma temática para obtener puntos de vista diferentes y no supeditar los resultados a la apreciación de un único autor. De esta forma se enriquecerían los resultados obtenidos.

Finalmente el diseño y desarrollo de una herramienta de software que aplique el método obtenido como resultado de la investigación realizada constituiría un excelente valor agregado a los proyectos realizados dentro del área del análisis arquitectural y arquitecturas de software puesto que implicaría la aplicación de muchos de los conocimientos adquiridos durante la formación como Ingeniero De Sistemas en el Programa De Ingeniería De Sistemas de la Universidad De Cartagena.

7. BIBLIOGRAFÍA

Acher, M., Cleve, A., Collet, P., Merle, P., Duchien, L., & Lahire, P. (2011). Reverse Engineering Architectural Feature Models. *IEEE Computer Society*.

Achinstein, P. (2004). General Introduction to Science Rules: A Hitorical Introduction to Scientific Methods. Science Direct - Johns Hopkins University Press.

Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., & Wood, W. (2003). *Quality Attribute Workshops (QAWs), Third Edition.* Pittsburg, Pennsylvania: Carnegie Mellon University.

Beck, F., & Diehl, S. (2010). Visual Comparison of Software Architectures. *ACM - International symposium on Software visualization SOFTVIS*, 183-192.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language User Guide.* Reading, MA: Addison-Wesley.

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

Canfora, G., Di Penta, M., & Cerulo, L. (2011). Achievements and Challenges in Software Reverse Engineering. *Communications of the ACM*, 142-151.

Chifosky, E., & Cross, J. (1990). Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, 13-17.

Date, C. (2013). *Relational Theory for Computer Professionals.* United States: ACM - O'Reilly Media Inc.

Ducasse, S., & Pollet, D. (2009). Software Architecture Reconstruction: A Process Oriented Taxonomy. *IEEE Transactions On Software Engineering*, 573-590.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, MA: Addison-Wesley.

Gorton, I. (2006). Essential Software Architecture. *Springer-Verlag*.

Hassan, A. E., & Holt, R. C. (2002). Architecture recovery of web applications. *ICSE '02*, 19-25.

Heijstek, W., Kühne, T., & Chaudron, M. (2011). Experimental Analysis of Textual and Graphical Representations for Software Architecture Design. *ACM - International Symposium on Empirical Software Engineering and Measurement*.

Jarzabek, S., & Woon, I. (2007). Towards a Precise Description of Reverse Engineering Methods and Tools. *IEEE Computer Society*.

Kazman, R., & Bass, L. (2002). Making architecture reviews work in the real world. *IEEE Software*, 67-73.

Kazman, R., Bass, L., Abowd, G., & Webb, M. (2007). SAAM: A Method for Analyzing the Properties of Software Architectures. *Software Engineering Institute Papers*.

Kazman, R., Clements, P., & Klain, M. (2005). *Evaluating Software Architectures, Methods and Case Studies.* Adison-Wesley Professional.

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation.* Carnegie Mellon University.

Larman, C. (2003). *UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Madrid: Pearson Educación, S.A.

Li, Z., Gittens, M., Shariyar Murtaza, S., & Madhavji, N. (2010). Towards an Extended Relational Algebra for Software Architecture. *ACM - Software Engineering Notes SIGSOFT volume 35 Issue 3* , 1-4.

Lijun, L., Changyun, L., Gexin, M., & Zhibing, W. (2012). Layered Semantic Analysis of Software Architecture. *IEEE Computer Society*.

López Cano, J. L. (1984). *Métodos e Hipótesis Científicas.* Ciudad De México: Pearson Education.

Maturro, G., & Saavedra, J. (2012). *Gestión Del Conocimiento y La Experiencia en Ingeniería Del Software.* Montevideo, Uruguay: Lambert Academic Publishing GmbH & Co.

Monroy, M., Arciniegas, J., & Rodríguez, J. (2012). Caracterización de Herramientas de Ingeniería Inversa. *Información Tecnológica*, 31-42.

Moreland, K., King, B., Maynard, R., & Ma, K.-L. (2013). Flexible Analysis Software for Emerging Architectures. *IEEE Computer Society*.

Muller, H., Jahnke, J., Smith, D., Storey, M., Tilley, S., & Wong, K. (2000). Reverse Engineering: A roadmap. *ACM - Future of Software Engineering Track*, 47-60.

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

Podgurski, A. C. (2006). A formal model of program dependences and its implications for software testing, debugging, and maintenance. *IEEE Transaction on Software Engineering*, 965-979.

Pressman, R. (2009). *Ingeniería de software: un enfoque práctico.* Prentice-Hall.

Real Academia De La Lengua Española - RAE. (s.f.). *Diccionario De La Lengua Española*. Recuperado el 22 de Julio de 2013, de http://lema.rae.es/drae/

Ritchey, T. (1991). Analysis and Sythesis on Scientific Method, Based on a Study by Bernhard Riemann. *Systems Research*, 21-41.

Rojas, R. (2005). *Guía para realizar investigaciones sociales.* Ciudad de México: Plaza y Valdéz editores.

Sampieri Hernández, R., Fernández, C., & Baptista, P. (1996). *Metodología De La Investigación.* Bogotá: Mc Graw-Hill.

SEI. (2013). *Software Engineering Institute - Carnegie Mellon*. Recuperado el 26 de Mayo de 2013, de http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm

Sicilia, M. (2009). *Técnicas de Mantenimiento del Software*. Recuperado el 25 de Agosto de 2013, de http://cnx.org/content/m17431/1.4/

Störmer, C. (2007). *Software Quality Attribute Analysis by Architecture Reconstruction (Doctoral Thesis).* Vrije Universiteit.

Universidad De Sevilla. (Noviembre de 2012). *Introducción al Álgebra Relacional.* Recuperado el 26 de Mayo de 2013, de http://www.lsi.us.es/docencia/get.php?id=6364

Van Deursen, A. (2001). Software Architecture Recovery and Modelling. *Working Conference on Reverse Engineering*, (págs. 4-7).

Wu, J., & Winter, A. (2002). Towards a Common Query Language for Reverse Engineering. *IEEE Computer Society*.