Desde RNFs a Componentes a través de Tácticas de Arquitectura: Una Revisión Sistemática de la Literatura

■ Gastón Márquez
email: gaston.marquez@sansano.usm.cl
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile

■ Hernán Astudillo
email: hernan@inf.utfsm.cl
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile

Fecha de Recepción: 15 de Agosto 2016
Fecha de Aceptación: 26 de Septiembre de 2016

RESUMEN

La arquitectura de un sistema de software es resultado de numerosas decisiones de diseño sobre los compo- nentes y estructura adecuados para proveer las funcionalidades y propiedades requeridas. Las tácticas de arquitectura son una forma destacada de abordar requisitos no-funcionales (RNFs), pero parece existir poco trabajo metodológico que formalice el uso de tácticas para combinar componentes para abordar RNFs específicos. Este artículo describe una revisión sistemática de la literatura (RSL) sobre el uso metodológico combinado de RNFs, tácticas de arquitectura,

y componentes de software. Se definió preguntas de investigación sobre (1) selección de componentes utilizando RNFs, (2) selección de componentes utilizando la relación entre RNFs y arquitectura, y (3) relación entre RNFs y tácticas. Se encontró 1964 artículos en directorios conocidos, luego filtrados a 59 artículos directamente relevantes al tema de estudio. Un examen detallado de estos artículos ilustra que las tácticas son útiles como concepto intermedio para identificar componentes desde RNFs, pero hasta ahora hay sólo un grupo pequeño de propuestas específicas.

Palabras Clave — Revisión sistemática de literatura, com- ponentes de software, requisitos no funcionales, tácticas de arquitectura.

I. INTRODUCIÓN

La arquitectura de software es un enlace entre los objetivos definidos en el negocio por los stakeholders y el resultado final del sistema, el cual en la mayoría de los casos, los requisitos y los intereses de los stakeholders se ven reflejado en los Requisitos No Funcionales (RNFs). Según [1], la complejidad de un sistema de software está determinada en parte por su funcionalidad y por su la globalidad de los requisitos. Estos RNFs juegan un rol crítico durante el desarrollo del sistema, sirviendo como criterio de selección para alternativas de decisiones de diseño hasta la implementación. Es en este punto donde Chen et al. [2] ha puesto en debate el concepto de requisitos de arquitectura significativos, con el objetivo de analizar si realmente todos los RNFs son realmente útiles al momento de tomar decisiones de arquitectura. Hasta el momento, una de las mejores maneras utilizadas para la toma de decisiones de diseño en sistemas de software son las tácticas de arquitectura. Una táctica de arquitectura es una decisión de diseño que influencia el cumplimiento de satisfacer los atributos de calidad obtenidos desde RNFs [3].

Por otro lado, la ingeniería de software basada en com- ponentes (CBSE, siglas en inglés) promueve el desarrollo y construcción de sistemas de software desde componentes de software ya existentes. El desarrollo de componentes se puede ver como una entidad reusable que va acorde a la evolución del sistema y adaptable a las necesidades de los stakeholders. La motivación del uso de la ingeniería de software basada en componentes nace desde la naturaleza del negocio con el objetivo de incrementar la eficencia y efectividad, el tiempo de salida de productos de software al mercado, reducir costos, entre otros [4].

Dicho lo anterior, nos hemos percatado que para los ar- quitectos de software y los stakeholders las decisiones que se toman en un sistema de software son muy críticas. Por un lado, el arquitecto de software se preocupa que cualquier decisión de diseño que se tome no afecte las funcionalidades del sistema, pero por otro lado, los stakeholders ponen más énfasis en que las necesidades del negocio se lleven a cabo los más pronto posible. Es aquí donde la correcta selección de componentes debe ser un factor relevante a la hora de considerar decisiones de diseño a nivel de arquitectura de software, satisfaciendo las necesidades de los stakeholders y lograr los atributos de calidad definidos en los requisitos.

Con el objetivo de abordar esta interrogante, este artículo describe una Revisión Sistemática de la Literatura (RSL) usando las directrices de Kitchenham et al. [5] para conocer los trabajos, perspectivas e iniciativas en la comunidad donde se aborde la selección de componentes desde RNFs mediante el uso de tácticas de arquitectura. Hemos desarrollado un protocolo de revisión obteniendo como resultado final 59 artículos que nos han ayudado a comprender el

contexto actual de nuestra interrogante.

El resto del artículo se organiza de la siguiente manera: la sección II describe aquellos trabajos que han abordado nuestra interrogante, desde cualquier punto de vista, mediante una RSL realizada previamente; la sección III explica la meto- dología usada en nuestra revisión describiendo las preguntas de investigación, el proceso de búsqueda, los criterios de inclusión/exclusión y las estrategias de extracción y síntesis de datos; la sección IV expone análisis y resultados de la ejecución del protocolo respondiendo a la preguntas de investigación; la sección V detalla las amenazas de validez; y la sección VI presenta conclusiones y trabajo futuro.

II. TRABAJOS RELACIONADOS

En esta sección discutiremos sobre los trabajos que existen en la comunidad que se relacionan con búsquedas sistemáticas de literatura sobre la obtención de componentes de software desde RNFs mediante el uso de tácticas de arquitectura. En primera instancia, no hemos podido encontrar algún trabajo que proponga una revisión sistemática sobre nuestro objeto de estudio. No obstante, si dividimos los temas, encontramos el trabajo de Vale et al. [4], donde realiza un estudio muy profundo sobre la evolución de CBSE. Como conclusiones del estudio, los autores se percatan que el interés en la comunidad sobre el reuso de componentes ha ido en aumento debido a las nuevas exigencias del mercado en relación al desarrollo de aplicaciones en el corto tiempo. Por otro lado, un estudio realizado por Morisio et al. [6] refleja, desde la práctica, 15 proyectos en donde se han utilizado el enfoque de componentes de software, llegando a la conclusión que los componentes tienen un gran impacto en definiciones de alto de nivel, específicamente en la integración y testing. Por último, hemos analizado el trabajo de Breivold et al. [7] donde se aborda la evolución del software. Para lo anterior, los autores realizan una revisión sistemática de la literatura para ver cómo afectan los requisitos en la evolución de la arquitectura de software en distinto niveles.

En resumen, en la literatura hemos encontrado trabajos que abordan desde diferentes puntos vista la interrogante de componentes y RNFs, pero la inclusión de tácticas de arquitectura es un tema que no ha sido abordado aún. Lo anterior nos motiva a realizar nuestra RSL con el objetivo de ver qué propuestas existen en la literatura que respondan en un cierto grado a nuestra interrogante de investigación.

III. METODOLGÍA

En esta sección vamos a describir el enfoque metodológico utilizado para realizar la RSL. Como resultado final, hemos obtenido un número significativo de artículos relevantes, los cuales serán resumidos en la siguiente sección.

Para realizar la revisión, hemos adoptado las directrices definidas por Kitchenham et al. [5], para conducir RSL en ingeniería de software. Estas directrices son utilizadas para identificar, evaluar e interpretar todos los trabajos relevantes en base a una pregunta de investigación, área de interés o un fenómeno de interés. Siguiendo las directrices, se debe definir un protocolo de revisión para la RSL y éste se debe seguir durante la revisión. El protocolo que nosotros hemos definido contiene la creación de la preguntas de investigación, el proceso de búsqueda, definición de criterios de inclusión y exclusión, estrategia de extracción y síntesis de datos y la ejecución. La Figura 1 resume lo anteriormente mencionado. La ejecución del protocolo fue realizada entre las fechas 13 de

Enero y 20 de Julio del 2016 en las siguientes librerías digitales: Scopus, IEEE Xplorer, SciendeDirect y ACM. La razón por la cual se seleccionaron estas librerías es debido a su gran potencial y relevancia en la comunidad en relación a las conferencias y revistas que poseen.

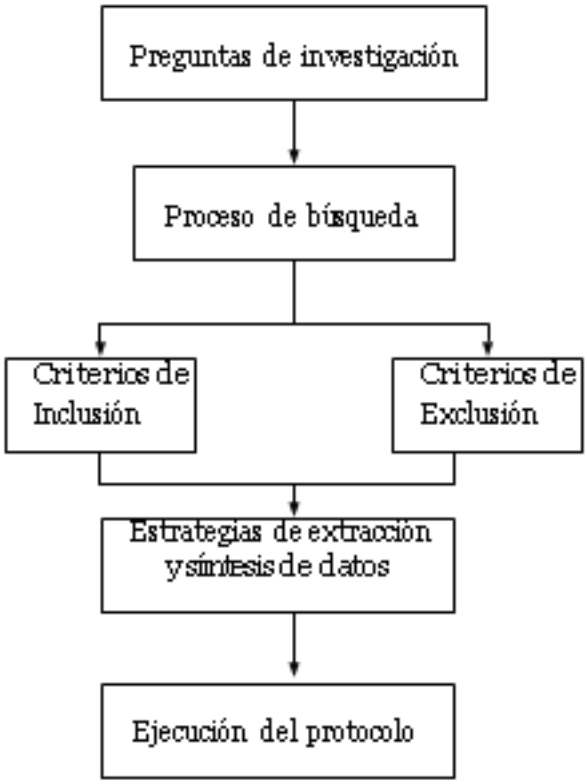

Figura 1: Protocolo de búsqueda

III-A. Preguntas de investigación

Para formular las preguntas de investigación, hemos confor- mado reuniones previas con expertos en donde se debatieron ideas bajo la técnica de "Lluvia de ideas" (Brainstorming) bajo en contexto de RNFs, tácticas y componentes. La noción de Lluvia de Ideas se basa en el concepto de ampliar la participación, democratizarla, a todos los presentes en el espacio en el cual la reunión o encuentro se lleva a cabo. Cada integrante contribuye a la generación de ideas y de posibles enfoques. La Lluvia de Ideas comienza con la definición de un tema o quizás también con el establecimiento de un problema o conflicto a resolver. Luego, se invita a que los miembros o los presenten propongan ideas, conceptos, posibles soluciones, formas de actuar, respecto de ese tema planteado [8]. Nosotros realizamos varios encuentros y debates con expertos en el tema y, como consecuencias de esto, hemos formulado las siguientes preguntas de investigación (PI):

- PI-1: ¿Qué iniciativas se han llevado a cabo para selec- cionar componentes de software mediante RNFs?
- PI-2: ¿Es posible que la relación entre requisitos y arqui- tectura de software permita seleccionar componentes?
- PI-3: ¿En que forma pueden los RNFs relacionarse con las tácticas de arquitectura?

Las preguntas de investigación que hemos creados tienen como objetivo observar los trabajos que aborden los temas de interés de esta RSL. Como objetivo final, queremos saber si es posible que existan propuestas que muestren iniciativas sobre la selección de componentes mediante RNFs usando las tácticas de arquitectura como medio para lograr la obtención.

III-B. Proceso de búsqueda

Una vez definidas las preguntas de investigación, se procede a realizar la búsqueda de información. Para proceder con lo anterior, es necesario definir en primer lugar una cadena de palabras para poder consular en cada librería digital. La cadena que hemos creado es la siguiente:

TABLA I. CADENA DE BÚSQUEDA

("non functional requirement" OR "architecture tactics" OR "tactic") AND	1)
("non functional requirement" OR "software component") AND	2)
("non functional requirement" OR "component based software engineering" OR "CBSE"))	3)

En la parte 1) de la cadena nos hemos enfocado en buscar aquellos trabajos que hablen sobre RNFs y tácticas de arqui- tectura. No nos hemos limitado a restringir la cadena con el objetivo de tener un universo más amplio de opciones. La parte 2) de la cadena busca lo mismo que la parte 1). No obstante, en la parte 3) nosotros hemos querido extender la consulta a la disciplina CBSE, debido a que es un enfoque de desarrollo de software que se basa en el reuso de software y que en este último tiempo ha emergido debido a los problemas que ha causado el desarrollo orientado a objetos con respecto al reuso [9]. Por lo tanto, nosotros también queremos averiguar si existen iniciativas que hablen sobre la relación entre RNFs y CBSE.

Con la cadena de consulta lista, procedemos a realizar la búsqueda. Como explicamos anteriormente, nosotros realiza- mos la búsqueda durante un rango de tiempo. Como resultado final, hemos obtenido en total 1964 artículos de las librerías digitales. A continuación, procederemos a realizar los criterios de inclusión y exclusión con el objetivo de filtrar los trabajos más relevantes.

III-C. Criterios de inclusión y exclusión

Para crear los criterios de inclusión y exclusión, nos hemos basado en la propuesta de Biolchini et al. [10]. A continuación, se resumen los criterios de exclusión más relevantes:

1. Trabajos que no se relacionen directamente con el campo de la ingeniería de software, por ejemplo: máquinas de aprendizaje, comercio electrónico, entre otros.
2. Trabajos en donde no muestren algún tipo de evidencia en sus propuestas.
3. Trabajos que no estén relacionados a alguna conferencia, workshop o revista.
4. Trabajos que no posean estudios primarios.

Para los criterios de exclusión 2. y 3. hemos querido ser enfáticos en que los trabajos que vamos a seleccionar tengan evidencia de su investigación y que haya sido demostrada en algún evento o revista. Nos hemos percatado que existen una cantidad de artículo en donde se exponen ideas que son muy subjetivas. A pesar de que en algunos artículos las ideas son interesantes, no presentan un respaldo que apoye la idea principal. Por otro lado, el punto 4. es un criterio de importancia, pues las directrices de [5] enfatizan que las RSL deben tener este

filtro. Hemos aplicado otro criterio de exclusión que no hemos querido mencionar en el listado el cual es que los trabajos que sean seleccionados deben estar escritos en inglés, debido a que la mayoría de los congresos y revistas están escritos en este idioma.

Por otro lado, como criterios de inclusión hemos determi- nado los siguientes:

1. Trabajos que se relacionen con el área de ingeniería de software
2. Trabajos en donde exista evidencia en las propuestas
3. Trabajos que se encuentren en eventos y revistas de relevancia
4. Trabajos que tengan estudio primarios
5. Trabajos que estén en el idioma inglés

Hemos decido agregar como criterio de inclusión trabajos que nos recomienden aquellos interesados que han participado en esta RSL. Este tipo de trabajos nos permiten conocer un gama más amplia que quizás el protocolo de la RSL no aborda.

La accesibilidad del artículo es muy importante para poder estudiarlo. Del universo de artículos obtenidos, hubo algunos en los cuales no se tuvo acceso, por lo tanto, se tuvieron que descartar de esta revisión.

III-D. Estrategia de extracción y síntesis de datos

Al aplicar los filtros de exclusión e inclusión, la cantidad de artículos se redujo a 68. Con este nuevo universo de trabajo, se procedió a realizar la lectura detallada de los siguientes campos con el objetivo de obtener un conjunto de artículos que se relacione directamente con los objetivos de la RSL. Los campos son:

- Título
- Resumen
- Palabras claves

Los campos mencionados anteriormente, nos permiten visua- lizar de manera general que los artículos obtenidos sean cohe- rentes con lo que deseamos investigar. Durante el proceso de lectura, nos percatamos que algunos resumenes no representan necesariamente lo que el título describe. Para aquellos casos (9 artículos), se leyó la introducción y conclusión de manera excepcional, con el objetivo de saber si se debía descartar o no el artículo. Una vez terminado este proceso de lectura, el universo de artículos quedó en 59 trabajos, los cuales fueron analizados en profundidad. Como resultado final, la extracción de datos de cada artículo se basó en los siguientes campos:

- Digital Object Identifier (DOI)
- Año
- Autores
- Título
- Resumen
- Palabras claves
- URL
- Asunto (Conferencia, Revista, Workshop, entre otros)

Al momento de extraer los datos para los campos descritos anteriormente, cabe destacar que hemos encontrado 3 artículos sin DOI, 1 artículo sin URL y 6 artículos sin asunto esta- blecido. A pesar de que tentativamente,

los artículos que no cumplen con la información solicitada deberían ser eliminados de a RSL, no hemos querido descartar estos artículos, debido a que el tema que proponen fueron discutidos en eventos universitarios que no necesariamente fueron expuestos en congresos o revistas. Estos son casos particulares, que fueron analizado en detalle en las reuniones de Lluvia de ideas. Una vez leído completamente los artículos, a continuación resumiremos qué iniciativas consideramos en el análisis de cada artículo:

a) Metodología
b) Estudio de caso
c) Framework
d) Survey
e) Trabajo experimental
f) Trabajo empírico
g) Ejemplo
h) Herramienta
i) Prototipo
j) Enfoque
k) Mecanismo
l) Investigación
m) Proceso
n) Taxonomía

Todas las iniciativas anteriormente descritas fueron estudia- das en cada artículo. La Figura 2 resume los pasos ejecutados para obtener los 59 artículos. Como consecuencia, en la siguiente sección, detallaremos los resultados obtenidos de la ejecución del protocolo.

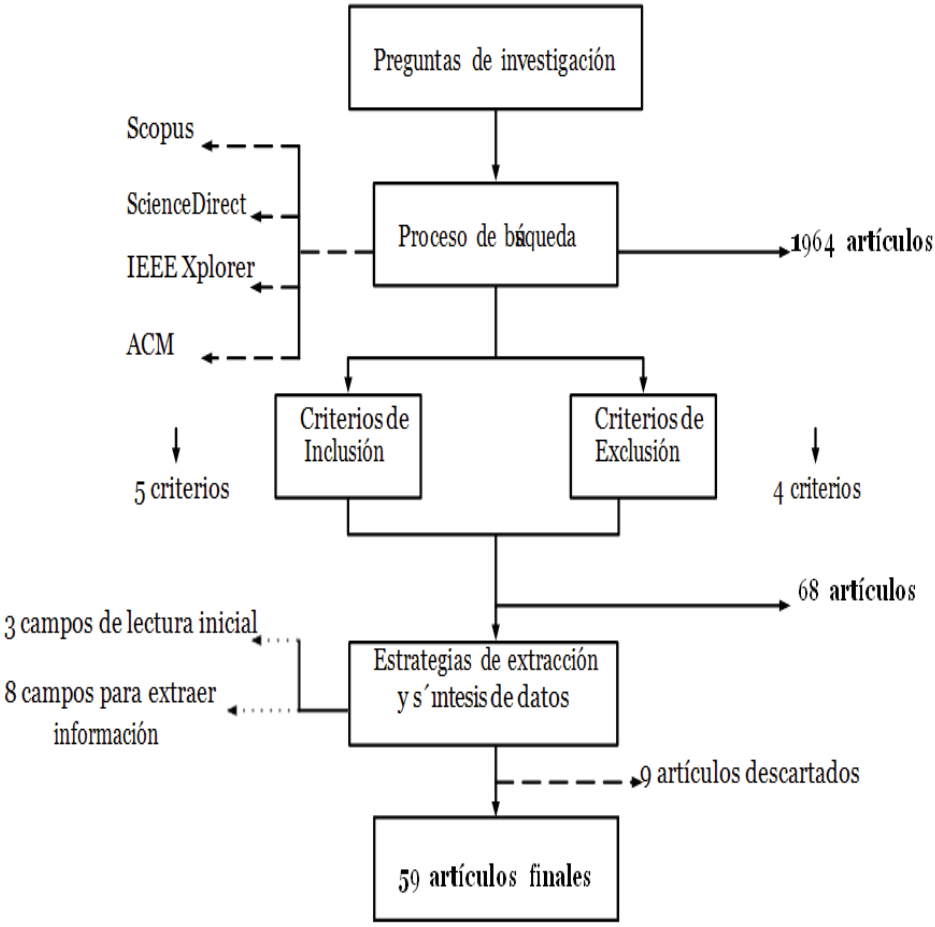

Figura 2: Pasos ejecutados en el protocolo junto a sus resul- tados temporales

IV. ANÁLISIS Y RESULTADOS

En esta sección vamos a describir los resultados logrados de nuestro estudio. Vamos a discutir separadamente cada pregunta de investigación en base a la evidencia encontrada en la literatura. A continuación, detallaremos los trabajos relevantes detallando un identificador, los autores, el nombre del artículo y cantidad de citas obtenidas en Scopus.

•A.1: Feng Chen, From architecture to

requirements: Relating requirements and architecture for better Requi- rements Engineering [11] (Citas: 2)

•A.2: David Ameller et al., Non-functional requirements in architectural decision making [12] (Citas: 30)

•A.3: Vibhu Saujanya Sharma et al., A Framework for Identifying and Analyzing Non-functional Requirements from Text Categories and Subject Descriptors [13] (Citas: 1)

•A.4: David Ameller et al., Non-functional requirements in software architecture practice [14] (Citas: 4)

•A.5: Mehdi Mirakhorli et al., A Tactic-Centric Approach for Automating Traceability of Quality Concerns [15] (Citas: 43)

•A.6: Remco C. de Boer et al., On the similarity between requirements and architecture [16] (Citas: 52)

•A.7: Matthias Galster et al., Transition from Require- ments to Architecture: A Review and Future Perspective [17] (Citas: 24)

•A.8: Remo Ferrari et al., Architecting-problems rooted in requirements [18] (Citas: 20)

•A.9: Preethu Rose Anish et al., What You Ask Is What You Get: Understanding Architecturally Significant Fun- ctional Requirements [19] (Citas: 0)

•A.10: Jonas Eckhardt et al., Are "Non-functional" Re- quirements really Non-functional? [20] (Citas: 0)

•A.11: Matthias Galster et al., Comparing Methodologies for the Transition between Software Requirements and Architectures [21] (Citas: 5)

•A.12: Jane Cleland-Huang et al., A Persona-Based Ap- proach for Exploring Architecturally Significant Requi- rements in Agile Projects [22] (Citas: 12)

•A.13: Remo Ferrari et al., Requirements and Systems Ar- chitecture Interaction in a Prototypical Project: Emerging Results [23] (Citas: 7)

•A.14: Claudia López et al., Explicit Architectural Policies
to Satisfy NFRs Using COTS [24] (Citas: 10)

•A.15: Hernán Astudillo et al., Identifying "Interesting" Component Assemblies for NFRs Using Imperfect Information [25] (Citas: 7)

•A.16: Preethu Rose Anish et al., Identifying Architec- turally Significant Functional Requirements [26] (Citas: 2)

•A.17: Preethu Rose Anish et al., A Knowledge-Assisted Framework to Bridge Functional and Architecturally Sig- nificant Requirements [27] (Citas: 1)

•A.18: Jane Cleland-Huang et al., Dynamically Tracing Non-Functional Requirements through Design Pattern Invariants [28] (Citas: 49)

•A.19: Suntae Kim, A quantitative and knowledge-based approach to choosing security architectural tactics [29] (Citas: 2)

•A.20: Mehdi Mirakhorli et al., A Domain-Centric Ap- proach for Recommending Architectural Tactics to Sa- tisfy Quality Concerns [30] (Citas: 3)

•A.21: Jungwoo Ryoo et al., Revising a Security Tactics Hierarchy through Decomposition, Reclassification, and Derivation [31] (Citas: 5)

•A.22: Neil B. Harrison et al., How do architecture pat- terns and tactics interact? A model and annotation [32] (Citas: 49)

•A.23: Gilberto Pedraza-Garcia et al., A Methodological Approach to Apply Security Tactics in Software Archi- tecture Design [33] (Citas: 1)

•A.24: Mehdi Mirakhorli et al., A Pattern System for Tracing Architectural Concerns [34] (Citas: 7)

•A.25: Mehdi Mirakhorli et al., Mining Big Data for Detecting, Extracting and Recommending Architectural Design Concepts

[35] (Citas: 0)

•A.26: Koen Yskout et al., Change patterns Co-evolving requirements and architecture [36] (Citas: 3)

•A.27: René Nöel et al., Exploratory Comparison of Security Patterns and Tactics to Harden Systems [37] (Citas: 2)

•A.28: Mehdi Mirakhorli et al., Detecting, Tracing, and Monitoring Architectural Tactics in Code [38] (Citas: 4)

•A.29: Jungwoo Ryoo et al., A Methodology for Mining Security Tactics from Security Patterns [39] (Citas: 7)

•A.30: Mohamad Kassab et al., A Quantitative Evalua- tion of the Impact of Architectural Patterns on Quality Requirements [40] (Citas: 8)

•A.31: Anton Uzunov et al., Decomposing Distributed Software Architectures for the Determination and Incor- poration of Security and Other Non-functional Require- ments [41] (Citas: 8)

•A.32: Yi Liu et al., An Approach to Integrating Non- functional Requirements into UML Design Models Based on NFR-Specific Patterns [42] (Citas: 2)

•A.33: Alejandro Sanchez et al., Analysing Tactics in Architectural Patterns [43] (Citas: 2)

•A.34: Alireza Parvizi-Mosaed et al., Towards a Tactic- Based Evaluation of Self-Adaptive Software Architecture Availability [44] (Citas: 1)

•A.35: Gilberto Pedraza-Garcia et al., A methodological approach to apply security tactics in software architecture design [33] (Citas: 1)

•A.36: Azadeh Alebrahim et al., Towards systematic se- lection of architectural patterns with respect to quality requirements [45] (Citas: 0)

•A.37: Mohamad Kassab et al., Towards Quantifying Quality, Tactics and Architectural Patterns Interactions [46] (Citas: 0)

•A.38: Eduardo B. Fernandez et al., Revisiting Architec- tural Tactics for Security [47] (Citas: 0)

•A.39: Azadeh Alebrahim et al., Towards a reliable mapping between performance and security tactics, and architectural patterns [48] (Citas: 0)

•A.40: Jungwoo Ryoo et al., The Use of Security Tactics in Open Source Software Projects [49] (Citas: 0)

•A.41: Ahmed Sabry, Decision Model for Software Ar- chitectural Tactics Selection based on Quality Attributes Requirements [50] (Citas: 1)

•A.42: Romina Torres et al., From early architectural decisions to self-discovery components [51] (Citas: 0)

•A.43: Claudia López et al., Multidimensional Catalogs for Systematic Exploration of Component-Based Design Spaces [52] (Citas: 7)

•A.44: Lawrence Chung et al., Matching, Ranking, and Selecting Components: A COTS-Aware Requirements Engineering and Software Architecting Approach [53] (Citas: 22)

•A.45: Marcel Ibe et al., CREATE: A Co-Modeling Approach for Scenario-based Requirements and Component-based Architectures - A Detailed View [54] (Citas: 2)

•A.46: Andreea Vescan et al., A Fuzzy-Based Approach for the Multilevel Component Selection Problem [55] (Citas: 0)

•A.47: Tegegne Marew et al., Tactics based approach for integrating non-functional requirements in object- oriented analysis and design [56] (Citas: 21)

•A.48: Muhammad Ali Khan et al., A graph based re- quirements clustering approach for component selection [57] (Citas: 12)

•A.49: Pasqualina Potena, Composition and tradeoff of non-functional attributes in software systems: research directions [58]

(Citas: 8)

•A.50: Liming Zhu et al., UML Profiles for Design Decisions and Non-Functional Requirements [59] (Citas: 60)

•A.51: Chouki Tibermacine et al., Component-based spe- cification of software architecture constraints [60] (Citas: 17)

•A.52: Anas Mahmoud et al., Detecting, classifying, and tracing non-functional software requirements [61] (Citas: 0)

•A.53: Nida Afreen et al., A Taxonomy of Software's Non-functional Requirements [62] (Citas: 0)

•A.54: Mebarka Yahlali et al., Towards a software com- ponent assembly evaluation [63] (Citas: 0)

•A.55: Fábio Silva et al., STREAM-AP: A process to systematize architectural patterns choice based on NFR [64] (Citas: 2)

•A.56: Tomás Ruiz-López et al., A pattern approach to dealing with NFRs in ubiquitous system [65] (Citas: 0)

•A.57: David Ameller et al., ArchiTech: Tool support for NFR-guided architectural decision-making [66] (Citas: 8)

•A.58: David Ameller et al., How do software architects consider non-functional requirements: An exploratory study [67] (Citas: 44)

•A.59: Rahma Bouaziz et al., Applying Security Patterns for Component Based Applications Using UML Profile [68] (Citas: 10)

A partir de la lista anterior, nos hemos percatado que los años de mayor actividad de los artículos se encuentra entre los años 2011 hasta la fecha (ver Figura 3).

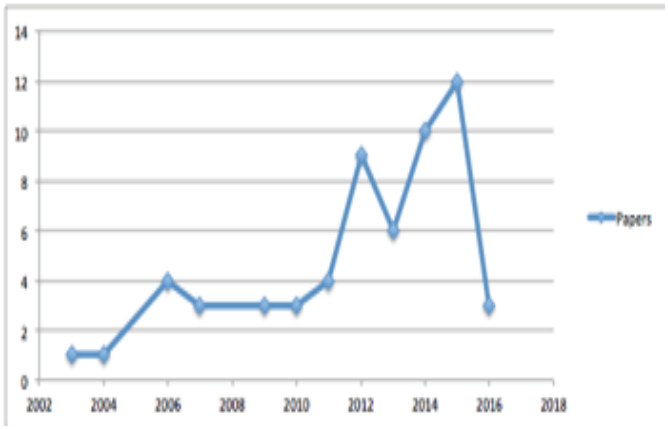

Figura 3: Gráfica cantidad de artículos versus años de publicación

Por ejemplo, para los años 2014 y 2015 hemos encontrado 22 artículos que exponen temas ligados a nuestras preguntas de investigación. Dado lo anterior, hemos descubierto que el vínculo entre RNFs, tácticas y componentes son temas que están interesando en la comunidad científica. Dentro del universo de artículos obtenidos, el 47 % corresponden a conferencias, 22 % a revistas, 19 % a workshop y un 12 % a otro tipos de eventos (ver Figura 4).

Figura 4: Gráfico circular de trabajos en conferencias, revistas, workshop y otros

Algunos de los eventos y revistas en donde se han presen- tados los trabajos ligados a nuestra RSL son:

- Conferencias

- IEEE International Requirements Engineering Con- ference
- International Conference on Software Engineering
- International Conference on Software Enginee- ring, Artificial Intelligence, Networking, and Para- llel/Distributed Computing
- IEEE International Conference on Systems, Man and Cybernetics
- International Working Conference on Requirements Engineering: Foundation for Software Quality
- International Conference on Model Driven Enginee- ring Languages and Systems
- International Conference on Software Security and Reliability Companion
- Conference on Pattern Languages of Programs
- International Workshop on Big Data Software Engi- neering
- International Conference on System Sciences
- International Conference on Software Engineering, Research, Management
- Revistas
- IEEE software
- Journal of Systems and Software
- Information and Software Technology
- International Journal of Ad Hoc and Ubiquitous Computing
- IEEE Transactions on Software Engineering
- Workshop
- International Workshop on Twin Peaks of Require- ments and Architecture
- European Workshop on Software Architecture
- Workshop on Traceability in Emerging Forms of Software Engineering
- Workshop on Experimental Software Engineering

En la estrategia de extracción y síntesis de datos hemos decidido en primer lugar leer los títulos, resúmenes y palabras claves de cada artículo. Dado lo anterior, realizamos un estudio sobre qué palabras y conceptos son los que más se repiten en los 3 campos mencionados anteriormente. Como resultado final, la Figura 5 nos muestra que las palabras y conceptos que más se mencionan son arquitectura, no-funcionales, requisitos, software, tácticas y patrones. Esta última palabra nos ha lla- mado la atención, pues nos hemos percatado que los patrones de arquitectura aparecen ligados a tácticas en algunos de los trabajos obtenidos de la RSL.

Un patrón de arquitectura expre- sa un esquema de organización para un sistema de software, que consta de subsistemas, responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones de arquitectura tienen un nivel de abstracción mayor. Esto tiene sentido con el concepto de tácticas de arquitectura, pues éstas representan decisiones de diseño en sistema de software.

Figura 5: Nube de palabras y conceptos claves

En las siguientes Tablas II y III detallaremos las propues- tas que ofrecen cada uno de los artículos. Como se puede observar en Tabla II, la mayor

cantidad de propuestas de los artículos son enfoques, seguido de procesos e investigación. Para nosotros tiene sentido que la mayoría de los artículos sean enfoques, pues la mayor parte de los autores proponen ciertos enfoques que tratan sobre el vínculo entre RNFs y componentes exponiéndolos en casos de estudio aplicados en la industria (la mayoría) obteniendo resultados interesantes. En la Tabla III resumiremos las propuestas por cada artículo, donde se verá que gran parte de las propuestas mezcla varias iniciativas.

En este punto del análisis de los resultados, procederemos a contestar las preguntas de investigación que nos hemos planteado al inicio de nuestra RSL.

IV-A. PI-1: ¿Qué iniciativas se han llevado a cabo para seleccionar componentes de software mediante RNFs?

Esta es la pregunta inicial que ha motivado nuestra RSL. El objetivo principal de esta pregunta es conocer qué trabajos se han expuesto en la comunidad que sean relevantes de ser estudiados. Con el universo obtenido de la RSL, nos hemos percatado que existen propuestas interesantes que abordan el tema. El artículo A.4 realiza un estudio completo sobre aquellos trabajos que han propuesto validación, elicitación, documentación sobre los RNFs. Como resultado final, se ha realizado un estudio empírico basado en entrevistas a expertos en el área de arquitectura de software donde se menciona que la generación de información de componentes desde los RNFs parecer ser un desafío complejo, pues la base de qué es un RNF para lo expertos no está construida sólidamente aún. Siguiendo la misma línea empírica, A.13 se plantea la interrogante, desde el punto de vista práctico, sobre cuál es rol de un arquitecto de software en la decisión de diseño de los requisitos. En este trabajo se demuestra que las decisiones de diseño están ligadas al proceso de desarrollo de software y el impacto que éstas puede producir en el producto mismo. Lo anterior se ratifica con las consecuencias de una decisión en base a componentes. Los autores del artículo dejan abierta la interrogante sobre las implicancia de los componentes de software en área sobre planificación del proyecto, riesgo, ingeniería de requisitos, entro otros. Por otro lado, los trabajos A.14, A.15 y A.42 proponen un enfoque más detallado relacio- nado con una técnica automatizada que ayuda a los arquitectos de software en la generación de ensamblados de componentes dado un conjunto de requisitos de arquitectura usando una analogía de mercado de componentes en base a un framework llamado Azimut. Esta idea fue aplicada a un estudio de caso con resultados interesantes, los cuales se pueden resumir en que este enfoque permitió a los arquitectos de software no solamente a generar potenciales soluciones basadas en requisitos, sino también soluciones basadas en restricciones de diseño tempranas. Siguiendo la misma línea que el trabajo A.42, la propuesta descrita en A.43 establece las bases de una idea compuesta de políticas de arquitectura. El punto principal de los autores es que estas políticas de arquitectura se pueden refinar en mecanismos y componentes que se pueden implementar con caracterizaciones multi-dimensionales. Una vez más, Azimut juega un rol importante en esta propuesta, ya que este framework es capaz de soportar la trazabilidad del arquitectura mediante el razonamiento arquitectónico en distinto niveles, permitiendo al arquitecto de software una exploración sistemática del espacio de diseño. Como com- plemento a lo anterior, A.54 propone una evaluación cuali- tativa del ensamblado de componentes. Como alternativa a la

evaluación del ensamblaje, A.44 propone ir más allá, en el sentido de proponer un enfoque iterativo llamado CARE/SA que compara, ordena y selecciona componentes en base a RFs y RNFs.

TABLA II. PORCENTAJES DE INICIATIVAS VERSUS TOTAL DE ARTÍCULOS

	Iniciativa	
a)	Metodología	8 %
b)	Caso de estudio	15 %
c)	Framework	19 %
d)	Survey	3 %
e)	Trabajo experimental	10 %
f)	Trabajo empírico	8 %
g)	Ejemplo	17 %
h)	Herramienta	14 %
i)	Prototipo	2 %
j)	Enfoque	58 %
k)	Mecanismo	12 %
l)	Investigación	27 %
m)	Proceso	34 %
n)	Taxonomia	3 %

Cambiando el paradigma de propuestas, A.46, A.48 y A.51
proponen distintos puntos de vista. El trabajo descrito en
A.46 describe un enfoque difuso para seleccionar componentes usando métricas, logrando resultados interesantes. Como otra manera de resolver el problema de selección de componentes,

A.48 propone un proceso basado en un grafo de interdepen- dencias donde los objetivos son clusters. A pesar de que la propuesta puede ser un poco compleja, los autores presentan su proceso en dos estudios de casos donde

los resultados obtenidos son prometedores, no obstante como trabajo futuro detallado por ellos, desean validar empíricamente su propuesta. Finalmente, A.51 propone trabajar con ADL (Architecture Description Language)

TABLA III. DETALLE DE INICIATIVAS POR CADA ARTÍCULO

Id	*a)*	*b)*	*c)*	*d)*	*e)*	*f)*	*g)*	*h)*	*i)*	*j)*	*k)*	*l)*	*m)*	*n)*
A.1												X		
A.2				X										
A.3			X					X	X	X				
A.4						X						X		
A.5			X							X				
A.6												X		
A.7	X			X								X	X	
A.8		X				X						X	X	
A.9											X			
A.10						X						X		
A.11	X									X		X		
A.12										X				
A.13		X											X	
A.14			X				X			X	X			
A.15							X			X				
A.16								X						
A.17										X				
A.18										X				
A.19										X			X	
A.20					X		X			X				
A.21	X									X				X
A.22							X							
A.23		X								X				
A.24													X	
A.25												X	X	
A.26		X	X		X	X	X	X		X			X	
A.27					X					X		X		
A.28	X													
A.29										X			X	
A.30			X				X			X			X	
A.31										X			X	
A.32			X							X				
A.33		X	X							X				
A.34		X								X				
A.35													X	
A.36										X			X	
A.37												X		
A.38					X							X		
A.39												X		
A.40			X					X			X	X		
A.41		X	X							X		X		
A.42			X				X			X	X			
A.43	X									X		X		
A.44							X			X				
A.45		X								X				
A.46										X	X	X		
A.47								X		X			X	
A.48			X					X					X	
A.49										X	X			
A.50							X	X						
A.51					X					X				
A.52														X
A.53										X			X	
A.54										X			X	
A.55							X			X				
A.56								X				X	X	
A.57						X						X		
A.58		X									X		X	

para la selección de componentes.
IV-A1. Discusión: De los trabajos que son capaces de responder la pregunta de investigación, nos hemos percatado que la mayoría de ellos abordan el problema de selección de componentes por distintos lados. Vemos que existen propues- tas por el lado de RNFs y por el lado de componentes, pero hay poco trabajos que abordan el camino completo. No obstante, A.15, A.42 y A.43 se acercan a proponer distintas formas para conectar RNFs con la selección de componentes, pero con resultados distintos.

IV-B. PI-2: ¿Es posible que la relación entre requisitos y arquitectura de software permita seleccionar componentes?

Esta pregunta de investigación nos ha demostrado que el tema de saber si existe alguna relación entre requisitos y arquitectura es un foco de interés en la comunidad. A partir de los artículos obtenidos en la RSL, nos encontrado de que existen 3 perspectivas, la perspectiva de requisitos, arquitectura de software y la cohesión entre ambos. No obstante, para iniciar la posible respuesta a nuestra pregunta de investigación, el artículo presentado en A.58 es una buena forma de entender la controversia que tiene los arquitectos de software con los requisitos. Este trabajo presenta un estudio empírico realizado a 13 arquitectos de software, en donde se les realiza una entrevista sobre quién o quienes deciden los RNFs, qué tipos de RNFs les interesa a los arquitectos de software, entre otras interrogantes.

Si tomamos la última perspectiva, los trabajo presentados en A.6 y A.7 son unos de los primeros que da inicio a la discusión entre si existe relación en estos dos mundos opuestos, requisitos y arquitectura de software. Este trabajo muestra las inquietudes que existen en la comunidad sobre si se puede enfatizar la decisiones de diseño como primera clase. Desde este punto de vista emergente, los autores del artículo argumentan que no existe mucha diferencia entre decisiones de arquitectura y requisitos de arquitectura significativos. Lo anterior resulta ser interesante, pues en este artículo se argumenta que no existen diferencias entre ambos tipos de requisitos, pero en la práctica, falta argumentar qué se entiende como requisitos de arquitectura significativos.

Siguiendo la misma línea, A.1 trabaja la misma interrogante sobre requisitos y arquitectura de software, pero enfoca la atención hacia la ingeniería de requisitos. El autor del artículo propone abordar el problema desde el proceso de ingeniería de requisitos y, en el mismo proceso, adaptar las decisiones de arquitectura. Lo anterior se discute de la misma manera en A.2 con resultados similares.

Los trabajos presentados en el párrafo anterior plantean la perspectiva de requisitos y arquitectura de software de manera teórica, pero también existen artículos en donde se ha abordado esta perspectiva desde el punto de vista práctico. Un artículo interesante en esta perspectiva práctica es la comparación que se realiza en A.11, ya que en este trabajo se establecer que la metodología de transición entre requisitos y arquitectura de software es la que mejor alcanza los objetivos para lograr mejores arquitecturas. Dicho lo anterior, los autores de este trabajo presentan 14 metodologías actuales que abordan este problema con resultados interesantes, desde el punto de vista práctico. En esta misma línea, el trabajo presentado en A.8 enfoca el debate a los problemas orientados a objetivos, demostrando en un estudio exploratorio realizado a 16 equipos de arquitectos de software que este tipo de problemas refleja diferencias en la opinión entre los expertos,

desde el punto de vista de la ingeniería de requisitos. Para llevar a cabo la interrogante sobre requisitos y arquitectura de software, el trabajo presentado en A.57 describe una herramienta llamada ArchiTech, la cual ofrece una ayuda a los arquitectos de software sugiriendo alternativas de decisiones de diseño con el objetivo de mejorar algunos tipos de RNFs para proyectos par- ticulares y facilitar el reuso del conocimiento de arquitectura basados en proyectos.

Relacionado con la última perspectiva, nos hemos percatado que algunos trabajos abordan la relación entre requisitos y arquitectura de software mediante patrones de arquitectura. No quisimos descartar estos trabajos de nuestra RSL, pues la pregunta de investigación la dejamos para que sea amplia, en el sentido de que si la relación entre requisitos y arquitectura (de cualquier forma) permite la selección de componentes. Trabajos tales como: A.25 que habla sobre extracción de datos en Big Data para decisiones de arquitectura, A.26 que habla sobre la evolución de patrones entre requisitos y arquitectura de software, A.31 que trata sobre descomposición de arquitectura distribuidas, A.36 que trata sobre la selección sistemática de patrones en función a requisitos de calidad y A.55 que presenta un proceso para la selección de patrones de arquitectura desde RNFs, muestran que los patrones de arquitectura juegan un rol importante al momento de derivar decisiones de arquitectura desde RNFs. Algunos de los trabajos evidenciados en este párrafo muestran casos prácticos, pero mencionan de forma discreta los componentes de software, como es el caso de A.56.

Además, hemos encontrado trabajos que abordan la pregun- ta de investigación desde el enfoque de entender en primer lugar, qué son para los arquitectos de software, los RNFs. En este punto nace un concepto que hemos encontrado, el cual es requisitos de arquitectura significativos (ya lo hemos mencionado anteriormente). Este tipo de requisitos no están considerados como RNFs, no obstante, son parte de ellos. El trabajo descrito en A.9 presenta un estudio cuantitativo en donde se caracterizan requisitos de arquitectura significativos desde varios dominios de negocio. Como resultado final, los autores obtienen 15 categorías de requisitos de arquitectura significativos a partir de entrevistas realizadas a 14 arquitectos de software. Como consecuencia del estudio anterior, en A.16 se propone una identificación automática de requisitos de arquitectura significativos. Pero, desde el punto de vista netamente de RNFs, A.10 debate sobre si lo que la comunidad entiende como RNFs son realmente RNFs. Como resultado final del estudio, los autores concluyen que la mayoría de los RNFs que conocemos describen más el comportamiento del sistema. Siguiendo la misma línea, los estudios descritos en A.3, A.12, A.17, A.32, A.49, A.52, y A.53 proponen opciones de rescatar información relevante de arquitectura de software desde RNFs con el objetivo de crear una base de conocimiento y entender qué es lo que realmente buscan los arquitectos de software en los RNFs.

Finalmente, desde la perspectiva de arquitectura de softwa- re, el trabajo A.30 ofrece un debate interesante sobre la evaluación cualitativa del impacto de los patrones de arquitectura en los requisitos de calidad, proponiendo un enfoque para medir dicho impacto. Además, se describen dos propuestas (A.50 y A.59) desde el punto de vista de UML para representar patrones de arquitectura y RNFs, permitiendo la representación de decisiones de arquitectura bajo un lenguaje de modelado.

Al momento de analizar si la relación entre requisitos y arquitectura de software da nociones sobre el uso de componentes de software, nos percatamos que los trabajos descritos no detallan en mayor grado el uso de com- ponentes de software. Hemos visto que existen dos mundos, el de requisitos por un lado y el de arquitectura de software por otro.

Los trabajos que tratan sobre la unión de ambos mundo se centran más en el conocimiento de arquitectura que un ejemplo concreto. Dado lo anterior, los artículos seleccionados para responder esta pregunta de investigación no detallan algún mecanismo para seleccionar componentes desde requisitos y arquitectura de software, por lo que deja abierta la pregunta de investigación.

IV-C. PI-3: ¿En que forma pueden los RNFs relacionarse con las tácticas de arquitectura?

En esta última pregunta de investigación, hemos querido resumir en qué forma se pueden relacionar los RNFs con las tácticas de arquitectura. Dentro de las propuestas, logramos concluir que existen propuestas como máquinas de aprendizaje (A.5, A.34), trazabilidad (A.18, A.24), procesos analíticos (A.19), recomendaciones (A.20, A.28), estudios empíricos (A.27, A.39, A.40, A.41), metodologías (A.29, A.33, A.35,
A. 7, A.38, A.47) y notaciones formales (A.22) que abordan la problemática sobre la relación entre RNFs y tácticas de arquitectura.

IV-C1. Discusión:

Como se puede observar, encontramos varios caminos que pueden responder de alguna forma la pregunta de investigación. Lo interesante de esto es que los trabajos expuestos nos da a entender que conocer si existe algún vínculo entre RNFs y tácticas de arquitectura es un foco de interés en la comunidad. Dado lo anterior, para proponer una nueva propuesta de relación entre RNFs y tácticas, debe ser una forma que sea respaldada por una validación sólida. La mayoría de los trabajos mencionados en esta pregunta de investigación proponen iniciativas enfocadas a estudios de casos, pero pocos validan sus propuestas de manera robusta, como lo es una validación empírica o experimental.

V. AMENAZAS DE VALIDACIÓN

En esta sección, utilizaremos parte del listado de amenazas definidas en [69] para realizar estudios. El análisis de amenazas que realizamos fueron en base a la validez interna y externa.

V-A. Validez interna

La validez interna está relacionada a la posible mal inter- pretación de la conclusión obtenida sobre la relación entre tratamiento y salida [69]. En el caso de nuestra RSL, la validez interna describe la correcta interpretación de los resultados obtenidos a partir de lo que dice la literatura. Por lo tanto, el objetivo principal de la RSL es minimizar las amenazas de la validez interna. Para realizar lo anterior, el equipo de investigación que realizó la RSL descrita en este artículo, discutió cada artículo encontrado en base a un esquema de resumen de artículos, el cual posee la siguiente estructura:

- Identificación del artículo
- Resumen completo del artículo
- Resumen de la introducción
- Propuestas descritas
- Breve descripción de las secciones del

artículo

- Observaciones y comentarios

Por cada uno de los 59 artículo generamos un documento con la estructura anterior y analizamos que dichos artículos no se escaparan de los objetivos de la RSL. De esta forma pudimos minimizar el riesgo de amenazas de mal interpretar la opinión expresada en la literatura.

V-B. Validez externa

La validez externa describe la habilidad de generalizar los resultados de la experimentación a la práctica [69]. Para nuestra RSL, la validez externa depende de los trabajos que hemos podido identificar, por lo tanto, nosotros hemos descartado aquellos trabajos que no han podido validar de manera concreta su idea. En ese sentido fuimos rigurosos, debido a que la comunidad científica es categórica en solicitar evidencia de las propuestas descritas en los artículos.

VI. CONCLUSIONES

En el presente artículo se ha llevado a cabo una RSL sobre los estudios que abordan la obtención de componentes de software a través de RNFs y tácticas de arquitectura. Para lo anterior, se ha definido un protocolo de búsqueda el cual consiste en la creación de la preguntas de investigación, definición de un proceso de búsqueda, establecer criterios de inclusión y exclusión, estrategia de extracción y síntesis de datos y la ejecución. Como resultado final, hemos obtenido un universo de 59 artículos que pueden ser capaces de responder las preguntas de investigación que nos hemos planteado. A partir de lo anterior, hemos encontrado resultados interesantes desde el punto de vista de requisitos y arquitectura de software. Nos hemos dado cuenta que la interrogante ligada a saber si se puede establecer un “puente” entre requisitos y arquitectura de software es un foco de interés en la comunidad. No obstante, las propuestas actuales para responder a la interrogante se centran o por el lado de requisitos o por el lado de arquitectura de software. Existe un numero reducido de propuestas que abordan este “puente” con resultados interesantes.

Otro punto que nos llamó la atención es que también existe interés en la comunidad en saber cómo obtener componentes de software a partir del conocimiento de arquitectura. Especí- ficamente, las decisiones de diseño juegan un rol importante al momento de decidir qué componentes son los adecuado para tomar decisiones en proyectos. Lo anterior tiene sentido, pues los trabajos obtenidos de la RSL describen que el conocimiento de arquitectura se obtiene a partir de estudios empíricos realizado con arquitectos de software.

Como conclusión final, la RSL reflejó la mirada general de la comunidad científica sobre la incógnita de obtener componentes desde RNFs y tácticas. Desde esta mirada, nos hemos percatado que las tácticas de arquitectura son útiles como medio de obtención de componentes de software a través de RNFs. En otras palabras, las tácticas de arquitectura son decisiones de diseño que se rescatan desde atributos de calidad y éstos se pueden derivar desde los RNFs. A su vez, desde la descripción de las tácticas de arquitecturas se pueden crear mecanismos o métodos de búsqueda para encontrar compo- nentes de software. Por lo tanto, como resultado derivado de esta RSL, nosotros hemos encontrado un foco de investigación que ha sido abordado discretamente en la comunidad, el cual consiste en la obtención de componentes de software desde RNFs a través de tácticas de arquitectura. Lo anterior

puede se usado como propuesta de un punto de investigación para nuevos investigadores y también como motivación para nuevos desarrollos en este campo de investigación.

VI-A. COMPACT framework

A partir de los resultados obtenidos de nuestra RSL, hemos estado trabajando en un framework para poder abordar el vínculo entre RNFs y selección de componentes a través de tácticas de arquitectura. Nuestro trabajo se llama COMPACT, el cual es un framework para obtener componentes de software a través de RNFs. Este framework se compone de 5 elementos que lo conforman, los cuales son: RNFs, un esquema de tácticas versus escenarios, un catalogo de componentes, una búsqueda componentes, un ensamblado de componentes y posibles soluciones. Cuando nos referimos a RNFs, éstos se obtienen desde un análisis de requisitos definidos por los stakeholders. Una vez que los RNFs son establecidos, el equipo que utiliza COMPACT debe asociar atributos de calidad a cada RNF encontrado. Esto se debe a que las tácticas de arquitectura se derivan desde atributos de calidad. Una vez que lo anterior esté terminado, se utiliza un esquema predeterminado de tácticas y escenarios. Para construir este esquema, se han utilizados técnicas de construcción de escenarios desde la definición de tácticas con el objetivo de que estos escenarios sean priorizados por los stakeholders. La importancia de la priorización de los stakeholders es que las tácticas seleccio- nadas por COMPACT satisfacen las necesidades tanto de los stakeholders como de los objetivos del negocio. Una vez que los escenarios están priorizados, se selecciona el escenario con mayor priorización. Cada escenario tiene un conjunto de tácticas de arquitectura que lo componen. Entonces, al momento de seleccionar el escenario, COMPACT ofrece un conjunto de tácticas las cuales deben ser seleccionadas por el equipo de trabajo que realizará el proyecto. Con las tácticas ya seleccionadas, se extraen palabras claves que se asocian a las tácticas y se unen con palabras claves que definen los stakeholders. Finalmente, con este conjunto de palabras claves, se realiza una búsqueda colaborativa en un catalogo de componentes de software, el cual está conformado por el nombre del componente y una descripción. Al finalizar la búsqueda, se obtiene un ensamblado de componentes, los cuales serán de gran ayuda para el arquitecto de software al momento de decidir qué alternativas de diseño deberá considerar para cumplir con los RNFs definidos por los stakeholders. Hemos probado COMPACT en distintos proyectos de desarrollo de software obteniendo resultado prometedores. En la Figura 6, se describe una mirada general de COMPACT.

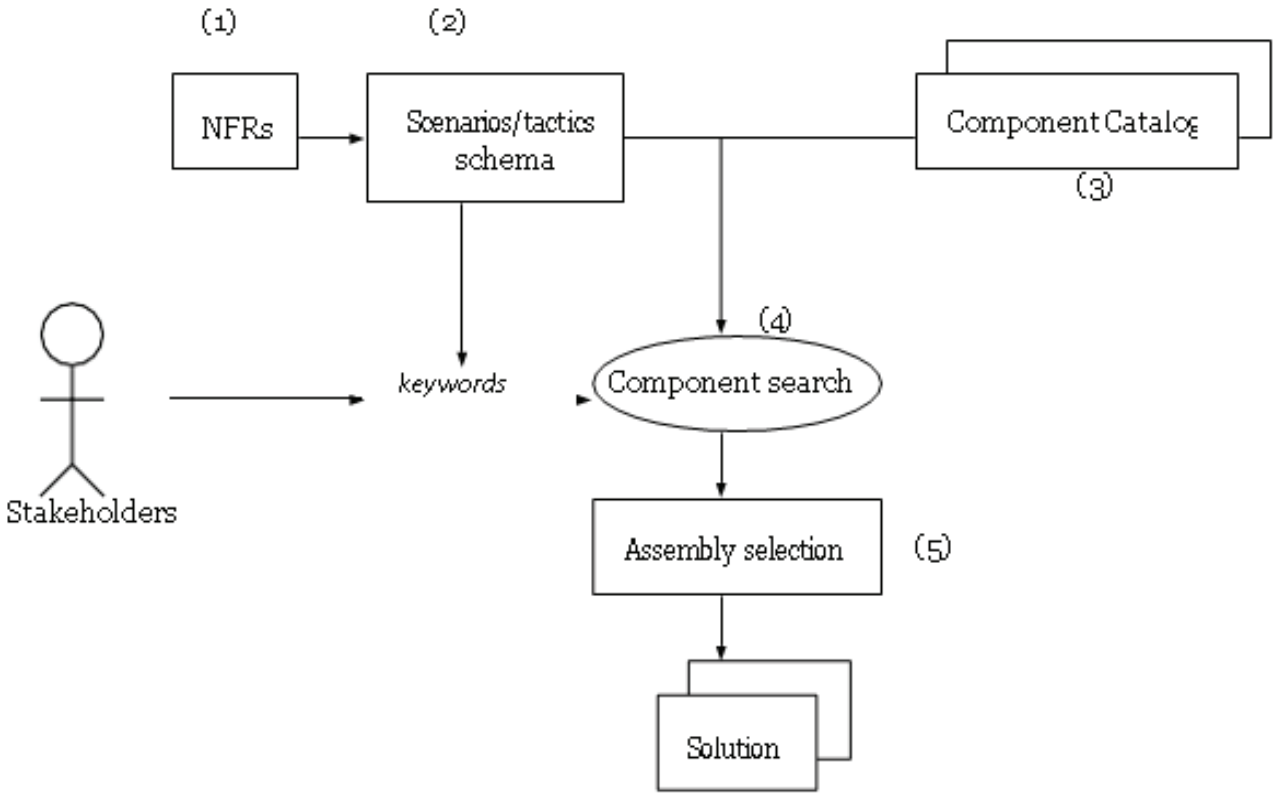

Figura 6: Mirada general de COMPACT

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por la Co- misión Nacional de Investigación Científica y Tecnológica, proyecto FONDECYT regular 1140408 y por el Programa de Incentivos a la Iniciación Científica de la Universidad Técnica Federico Santa María.

REFERENCIAS

[1]L. Chung and J. C. S. do Prado Leite, “On non-functional requirements in software engineering,” Conceptual modeling: Foundations and appli- cations, pp. 363–379, 2009.

[2]L. Chen, M. A. Babar, and B. Nuseibeh, “Characterizing architecturally significant requirements,” IEEE Software, vol. 30, no. 2, pp. 38–45, 2013.

[3]L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice (3rd Edition). SEI Series in Software Engineering, 2013.

[4]T. Vale, I. Crnkovic, E. S. De Almeida, P. A. D. M. Silveira Neto, Y. C. Cavalcanti, and S. R. D. L. Meira, “Twenty-eight years of component-based software engineering,” Journal of

Systems and Software, vol. 111, pp. 128 – 148, 2016.
[5]B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Engineering, vol. 2, p. 1051, 2007.
[6]M. Morisio, C. Seaman, V. Basili, A. Parra, and S. K. andS.E Condon, “Cots-based software development: Processes and open issues,” Journal of Systems and Software, vol. 61, no. 3, pp. 189 –199, 2002.
[7]H. P. Breivold, I. Crnkovic, and M. Larsson, “A systematic review of software architecture evolution research,” Information and Software Technology, vol. 54, no. 1, pp. 16 – 40, 2012.
[8]D. Mishra, A. Mishra, and A. Yazici, “Successful requirement elicitation by combining requirement engineering techniques,” Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the, no. 258-263, 2008.

[9]M. Gasparic and A. Janes, “What recommendation systems for software engineering recommend: A systematic literature review,” Journal of Systems and Software, vol. 113, pp. 101–113, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215002605
[10]J. Biolchini, P. G. Mian, A. Candida, and C. Natali, “Systematic Review in Software Engineering,” Engineering, vol. 679, no. May, pp. 165–176, 2005. [Online]. Available: http://www.cin.ufpe.br/in1037/leitura/systematicReviewSE-COPPE.pdf
[11]F. Chen, “From architecture to requirements: Relating requirements and architecture for better Requirements Engineering,” Requirements Engineering Conference (RE), 2014 IEEE 22nd International, pp. 451– 455, 2014.
[12]D. Ameller, C. Ayala, J. Cabot, and X. Franch, “Non-functional require- ments in architectural decision making,” IEEE Software, vol. 30, no. 2, pp. 61–67, 2013.
[13]V. S. Sharma, R. R. Ramnani, and S. Sengupta, “A Framework for Identifying and Analyzing Non-functional Requirements from Text Categories and Subject Descriptors,” pp. 1–8.
[14]D. Ameller, C. Ayala, J. Cabot, and X. Franch, “Non-Functional Requi- rements in Software Architecture Practice,” p. 20, 2012.
[15]M. Mirakhorli, Y. Shin, J. Cleland-Huang, and M. Cinar, “A tactic- centric approach for automating traceability of quality concerns,” Pro- ceedings of the 34th International Conference on Software Engineering ICSE 2012, June 2, 2012 - June 9, 2012, pp. 639–649, 2012.
[16]R. C. de Boer and H. van Vliet, “On the similarity between requirements and architecture,” Journal of Systems and Software, vol. 82, no. 3, pp. 544–550, mar 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0164121208002574
[17]M. Galster, A. Eberlein, and M. Moussavi, “Transition from require- ments to architecture: A review and future perspective,” Proc. - Seventh ACIS Int. Conf. on Software Eng., Artific. Intelligence, Netw., and Parallel/Distributed Comput., SNPD 2006, including Second ACIS Int. Worshop on SAWN 2006, vol. 2006, pp. 9–16, 2006.
[18]R. N. Ferrari and N. H. Madhavji, “Architecting-problems rooted in requirements,” Information and Software Technology, vol. 50, no. 1-2, pp. 53–66, jan 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S095058490700119X
[19]P. R. Anish, M. Daneva, J. Cleland-Huang,

ask is what you get: Understanding architecturally significant functional requirements," In 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp. 86–95, 2015.

[20]J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are "non-functional" requirements really non-functional?" 38th International Conference on Software Engineering, At Austin, Texas, 2016.

[21]M. Galster, A. Eberlein, and M. Moussavi, "Comparing methodologies for the transition between software requirements and architectures," Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on, no. October, pp. 2380–2385, 2009.

[22]J. Cleland-Huang, A. Czauderna, and E. Keenan, "A persona-based approach for exploring architecturally significant requirements in agile projects," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7830 LNCS, pp. 18–33, 2013.

[23]R. Ferrari, O. Sudmann, C. Henke, J. Geisler, W. Schafer, and N. Mad- havji, "Requirements and systems architecture interaction in a prototy- pical project: Emerging results," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6182 LNCS, pp. 23–29, 2010.

[24]C. Lopez and H. Astudillo, "Explicit Architectural Policies to Satisfy NFRs Using COTS," Satellite Events at the MoDELS 2005 Conference, pp. 227–236, 2005.

[25]H. Astudillo, J. Pereira, and C. López, "Identifying "interesting" compo- nent assemblies for nfrs using imperfect information," Third European Workshop, EWSA 2006, pp. 204–211, 2006.

[26]P. R. Anish, B. Balasubramaniam, J. Cleland-Huang, R. Wieringa, M. Daneva, and S. Ghaisas, "Identifying Architecturally Significant Functional Requirements," 2015 IEEE/ACM 5th International Workshop on the Twin Peaks of Requirements and Architecture, pp. 3–8, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7184705

[27]P. R. Anish and B.Balasubramaniam, "A knowledge- assisted framework to bridge functional and architecturally significant requirements," Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture - TwinPeaks 2014, pp. 14–17, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2593861.2593864

[28]J. Cleland-huang and D. Schmelzer, "Dynamically Tracing Non- Functional Requirements through Design Pattern Invariants," Procee- dings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering TEFSE, no. JANUARY 2003, 2003.

[29]S. Kim, "A quantitative and knowledge-based approach to choosing security architectural tactics," Ad Hoc and Ubiquitous Computing, vol. 18, no. 1/2, pp. 45–53, 2015.

[30]M. Mirakhorli, J. Carvalho, J. Cleland-Huang, and P. Mader, "A domain-centric approach for recommending architectural tactics to satisfy quality concerns," in TwinPeaks 2013 - 3rd International Workshop on the Twin Peaks of Requirements and Architecture (@ICSE 2013). Ieee, jul 2013, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6617352

[31]J. Ryoo, P. Laplante, and R. Kazman, "Revising a security tactics hierarchy through decomposition, reclassification, and derivation," Pro- ceedings of the 2012 IEEE 6th International Conference on Software Security and Reliability Companion, SERE-C 2012, pp.

85–91, 2012.
[32]N. B. Harrison and P. Avgeriou, “How do architecture patterns and tactics interact? A model and annotation,” Journal of Systems and Software, vol. 83, no. 10, pp. 1735–1758, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2010.04.067
[33]G. Pedraza-Garcia, H. Astudillo, and D. Correal, “A methodological approach to apply security tactics in software architecture design,” 2014 IEEE Colombian Conference on Communications and Computing, COLCOM 2014 - Conference Proceedings, 2014.
[34]M. Mirakhorli and J. Cleland-Huang, “A pattern system for tracing architectural concerns,” Proceedings of the 18th Conference on Pattern Languages of Programs - PLoP ’11, pp. 1–10, 2011. [Online].
Available: http://dl.acm.org/citation.cfm?doid=2578903.2579143 [35]M. Mirakhorli, H.-M. Chen, and R. Kazman, “Mining Big Data for Detecting, Extracting and Recommending Architectural Design Concepts,” 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering, pp. 15–18, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7166053
[36]K. Yskout and R. Scandariato, “Change Patterns : Co-evolving Requi- rements and Architecture Katholieke Universiteit Leuven Department of Computer Science Change Patterns : Co-evolving Requirements and Architecture,” no. August, 2010.
[37]R. Noel, G. Pedraza-García, and H. Astudillo, “An exploratory compa- rison of security patterns and tactics to harden systems,” Proceedings of the 11th Workshop on Experimental Software Engineering (ESELAW 2014), ser. CibSE, 2014.
[38]M. Mirakhorli and J. Cleland-Huang, “Detecting, Tracing, and Monitoring Architectural Tactics in Code,” IEEE Transactions on Software Engineering, vol. XX, no. X, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7270338
[39]J. Ryoo, P. Laplante, and R. Kazman, “A methodology for mining security tactics from security patterns,” Proceedings of the Annual Hawaii International Conference on System Sciences, pp. 1–5, 2010.
[40]M. Kassab, G. El-Boussaidi, and H. Mili, “A quantitative evaluation of the impact of architectural patterns on quality requirements,” Software Engineering Research,Management and Applications, vol. 377, pp. 173– 184, 2012.
[41]A. V. Uzunov, K. Falkner, and E. B. Fernandez, “Decomposing distri- buted software architectures for the determination and incorporation of security and other non-functional requirements,” 2013 22nd Australian Software Engineering Conference, pp. 30 – 39, 2013.
[42]Y. Liu, Z. Ma, R. Qiu, H. Chen, and W. Shao, “An approach to integrating non-functional requirements into uml design models based on nfr-specific patterns,” 2012 12th International Conference on Quality Software, pp. 132 – 135, 2012.
[43]A. Sanchez, A. Aguiar, L. S. Barbosa, and D. Riesco, “Analysing tactics in architectural patterns,” Software Engineering Workshop (SEW), 2012 35th Annual IEEE, pp. 32 – 41, 2012.
[44]A. Parvizi-Mosaed, S. Moaven, J. Habibi, and A. Heydarnoori, “Towards a tactic-based evaluation of self-adaptive software architecture availabi- lity,” 26th Software Engineering and Knowledge Engineering (SEKE), Vancouver, Canada, 2014.

[45]A. Alebrahim, S. Fassbender, M. Filipczyk, M. Goedicke, and M. Heisel, "Towards systematic selection of architectural patterns with respect to quality requirements," Proceedings of the 20th European Conference on Pattern Languages of Programs, pp. 40:1–40:20, 2015.
[46]M. Kassab and G. El-Boussaidi, "Towards quantifying quality, tactics and architectural patterns relations," The 25th International Conference on Software Engineering and Knowledge Engineering, 2013.
[47]E. B. Fernandez, H. Astudillo, and G. Pedraza-García, "Revisting architectural tactics for security," Software Architecture. Springer In- ternational Publishing, pp. 55–69, 2015.
[48]A. Alebrahim and M. Heisel, "Towards developing secure software using problem-oriented security patterns," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8708, pp. 45–62, 2014. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84921767147&partnerID=40&md5=150d6224e36164aa2e111145368942ef
[49]J. Ryoo, B. Malone, P. A. Laplante, and P. Anand, "The use of security tactics in open source software projects," IEEE Transactions on Reliability, vol. PP, no. 99, pp. 1 – 10, 2015.
[50]A. E. Sabry, "Decision model for software architectural tactics selection based on quality attributes requirements," International Conference on Communications, management, and Information technology (ICC- MIT'2015), 2015.
[51]R. Torres and H. Astudillo, "From early architectural decisions to self- discovery components," CIbSE, vol. 58, 2011.
[52]C. López and H. Astudillo, "Multidimensional catalogs for systematic exploration of component-based design spaces," Advanced Software Engineering: Expanding the Frontiers of Software Technology, pp. 32– 46, 2006.
[53]L. Chung and K. Cooper, "Matching , Ranking , and Selecting Com- ponents : A COTS-Aware Requirements Engineering and Software Ar- chitecting Approach," International Workshop on Models and Processes for the Evaluation of COTS Components at ICSE, pp. 41 – 44, 2004.
[54]M. Ibe, M. Vogel, B. Schindler, and A. Rausch, "Create: A co- modeling approach for scenario-based requierements and component- based architectures." International Conference on Software Engineering Advances, pp. 220 – 227, 2013.
[55]A. Vescan and C. S¸ erban, "A fuzzy-based approach for the multilevel component selection problem," Hybrid Artificial Intelligent Systems, pp. 463 – 474, 2016.
[56]T. Marewa, J.-S. Leeb, and D.-H. Baea, "Tactics based approach for integrating non-functional requirements in object-oriented analysis and design," Journal of Systems and Software, vol. 82, no. 10, pp. 1642– 1656, 2009.
[57]M. A. Khan and S. Mahmoodb, "A graph based requirements clustering approach for component selection," Advances in Engineering Software, vol. 54, pp. 1–16, 2012.
[58]P. Potena, "Composition and tradeoff of non-functional attributes in software systems: Research directions," Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, pp. 583–586, 2007.
[59]L. Zhu and I. Gorton, "Uml profiles for design decisions and non- functional requirements," Proceedings of the Second Workshop on SHA- ring and Reusing Architectural Knowledge Architecture,

Rationale, and Design Intent, 2007.
[60]C. Tibermacine, S. Sadou, C. Dony, and L. Fabresse, “Component- based specification of software architecture constraints,” Proceedings of the 14th International ACM Sigsoft Symposium on Component Based Software Engineering, pp. 31–40, 2011.
[61]A. Mahmoud and G. Williams, “Detecting, classifying, and tracing non- functional software requirements,” Requirements Engineering, vol. 21, no. 3, pp. 357–381, 2015.
[62]N. Afreen, A. Khatoon, and M. Sadiq, “A taxonomy of software’s non-functional requirements,” Proceedings of the Second International Conference on Computer and Communication Technologies, vol. 1, pp. 47–53, 2016.
[63]M. Yahlali and A. Chouarfia, “Towards a software component assembly evaluation,” IET Software, vol. 9, no. 1, pp. 1 – 6, 2015.
[64]F. Silva, M. Lucena, and L. Lucena, “STREAM-AP : A Process to Systematize Architectur- al Patterns Choice Based on NFR,” in TwinPeaks 2013 - 3rd International Workshop on the Twin Peaks of Requirements and Architecture, no. i, 2013, pp. 27–34.
[65]T. Ruiz-López, J. L. Garrido, S. Supakkul, and L. Chung, “A pattern approach to dealing with nfrs in ubiquitous systems,” 25th International Conference on Advanced Information Systems Engineering, CAiSE, vol. 998, 2013.
[66]D. Ameller, O. Collell, and X. Franch, “Architech: Tool support for nfr- guided architectural decision-making,” 2012 20th IEEE International Requirements Engineering Conference (RE), 2012.
[67]D. Ameller and X. Franch, “How do software architects consider non- functional requirements: A survey,” Conference: Requirements Engi- neering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, 2010.
[68]R. Bouaziz and B. Coulette, “Applying security patterns for component based applications using uml profile,” Computational Science and En- gineering (CSE), 2012 IEEE 15th International Conference on, pp. 186 – 193, 2012.
[69]C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in software engineering. Springer Science and Business Media, 2012.