PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Título

Mapeo sistemático sobre las arquitecturas de software en el desarrollo ágil

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INFORMÁTICA CON MENCIÓN EN INGENIERÍA DE SOFTWARE

AUTOR

Naldo Reupo-Musayón Gastulo

ASESOR

Mag. Dennis Stephen Cohn Muroy

Noviembre, 2020

DEDICATORIA

Al regalo más grande que me entregó la vida, mi hija Rubí y a mi abuela Violeta, aunque no pueda verla, vive en mis recuerdos y sé que me cuida.

AGRADECIMIENTOS

A mis padres Blanca y Naldo por su apoyo incondicional y constante.

A mi hija Rubí por darle sentido a mi vida y ser la causante de mis ganas de salir adelante.

A mis hermanas Isabel y Pierina, por creer en mí y siempre alentarme.

A mi asesor Dennis Cohn, por sus acertados y útiles consejos en la dirección de esta investigación.

RESUMEN

(ANTECEDENTES) El uso de *frameworks* y metodologías ágiles en el desarrollo de *software* es cada vez mayor, priorizando la entrega de valor al cliente, en este contexto las actividades de arquitectura de *software* son omitidas al no entregar un valor tangible, existiendo un aparente conflicto de perspectivas y no se tiene definido cuanto esfuerzo se debe invertir en el desarrollo de una arquitectura en proyectos ágiles.

(OBJETIVOS) El objetivo de este trabajo es consolidar las distintas investigaciones respecto al uso de arquitecturas de *software* en el desarrollo ágil, identificar patrones arquitectónicos, factores, beneficios, desafíos, y lecciones aprendidas con respecto a la combinación.

(MÉTODOS) Para este estudio se realizó un mapeo sistemático de la literatura en bases de datos digitales relevantes.

(RESULTADOS) Se seleccionaron 61 artículos publicados desde el año 2015 hasta el año 2020, el 54% fueron de aplicación industrial principalmente en el sector salud, aeroespacial y automotriz, se pudo identificar que en el año 2016 se publicaron el mayor número de artículos referente al tema de investigación, donde la conferencia es el tipo de publicación más utilizado y el evento *IEEE International Conference* es el mayor canal de distribución .Adicionalmente, se identificó que el estilo arquitectónico más empleado es SOA, la práctica ágil más referenciada es Scrum , el uso combinado del *framework* Scrum y el estilo SOA es el más usado, emplear el estilo SOA en el sector salud es el más citado en las publicaciones, la flexibilidad que brinda tener una arquitectura sólida es la mayor ventaja referenciada asimismo los conflictos de enfoques entre la agilidad y las actividades de arquitectura es identificado como el mayor inconveniente que se afronta ,y la comunicación es el factor que más influye en la adopción de arquitecturas de *software* en el desarrollo ágil.

Palabras clave

Arquitectura de *software*, Arquitectura de aplicaciones, Arquitectura ágil, Agilidad, desarrollo de *software*, ingeniería de *software*, diseño de *software*

ABSTRACT

(BACKGROUND) The use of agile frameworks and methodologies in software development is increasing, prioritizing the delivery of value to the client, in this context, software architecture activities are omitted by not delivering tangible value, with an apparent conflict of perspectives and it is not defined how much effort should be invested in the development of an architecture in agile projects.

(OBJECTIVES) The objective of this work is to consolidate the different investigations regarding the use of software architectures in agile development, to identify architectural patterns, factors, benefits, challenges, and lessons learned regarding the combination.

(METHODS) For this study, a systematic mapping of the literature in relevant digital databases was carried out.

(RESULTS) 61 articles published from 2015 to 2020 were selected, 54% were of industrial application mainly in the health, aerospace, and automotive sectors, it was possible to identify that in 2016 the largest number of articles were published on the subject of research, where the conference is the most used type of publication and the IEEE International Conference event is the largest distribution channel. Additionally, it was identified that the most used architectural style is SOA, the most referenced agile practice is Scrum, the combined use of Scrum framework and the SOA style is the most used, using the SOA style in the health sector is the most cited in publications, the flexibility provided by having a solid architecture is the greatest advantage referenced also the conflicts of approaches between agility and architectural activities is identified as the greatest inconvenience faced, and communication is the factor that most influences the adoption of software architectures in agile development.

Keywords

Software Architecture, Application Architecture, Agile Architecture, Agility, Software Development, Software Engineering, Software Design

CONTENIDO

ÍNDICE DE TABLA

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Contexto del problema

El uso de *frameworks* y metodologías ágiles en la construcción de *software* en diversas industrias ha comenzado a crecer sostenidamente en los últimos años. Por ejemplo, *CollabNet VersionOne* [1] en su más reciente encuesta de más de 1,300 empresas, el 97% indicó que usa *frameworks* ágiles en sus desarrollos; sin embargo, muchas organizaciones luchan con la toma de decisiones de arquitectura [2] dado que las actividades de arquitectura de *software* no se analizan en la mayoría de los métodos de desarrollo de *software* ágiles[3] y el tiempo invertido en arquitectura es cada vez más crucial [4]. Si el equipo pasa demasiado tiempo diseñando la arquitectura la entrega de valor al cliente se retrasa [5]. El diseño de una arquitectura es demasiado trabajo sin un beneficio claro según los agilistas [6]; generando una deuda técnica en los proyectos - que es uno de las principales problemas en los proyectos ágiles [7] - siendo percibida la elección de una arquitectura inadecuada como el principal factor en los proyectos de *software* [8].

No se tiene definido cuanto diseño es suficiente en el desarrollo ágil [9], esto genera un conflicto entre definir una arquitectura inicial completa o construir la arquitectura gradualmente durante desarrollo del producto [10]: adaptación versus anticipación [11] .

El problema central encontrado es que el desarrollo ágil es altamente vulnerable a deuda técnica por deficiencias de arquitectura [12].

Las causas que explican el problema planteado en la investigación son:

- Muchas organizaciones luchan con la toma de decisiones de arquitectura [13], las actividades de arquitectura de *software* no se analizan en la mayoría de los métodos de desarrollo de *software* ágiles[3].
- El tiempo invertido en arquitectura es cada vez más crucial [4] ;si el equipo pasa demasiado tiempo diseñando la arquitectura la entrega de valor al cliente se retrasa [5].
- El diseño de una arquitectura es demasiado trabajo sin un beneficio claro según los agilistas [6].

- Conflicto entre definir una arquitectura inicial completa o construir la arquitectura gradualmente durante desarrollo del producto [10]: adaptación versus anticipación[14].

Los efectos que derivan del problema central son:

- La deuda técnica es uno de las principales problemas en los proyectos ágiles[7], las inadecuadas decisiones sobre la arquitectura de diseño son el principal factor que genera deuda técnica[15].
- No se tiene definido cuanto diseño es suficiente en el desarrollo ágil [9].

A partir del problema planteado, las causas y efectos descritos anteriormente se propone el árbol de problemas mostrado en la Ilustración 1.

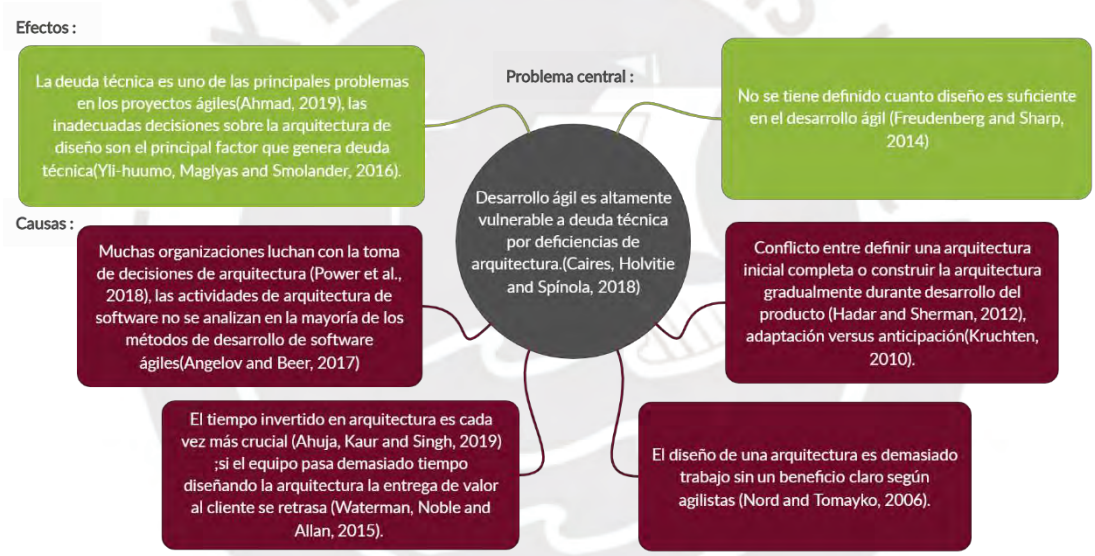

Ilustración 1 Árbol de problemas

1.2. Justificación

El desarrollo ágil enfrenta el dilema de determinar cuánto esfuerzo invierte en la arquitectura de *software* [16] , por eso surge la necesidad de un estudio que permita explorar y analizar los artículos en relación con el uso de arquitecturas en el desarrollo ágil para identificar los patrones y estilos arquitectónicos de *software* compatibles con prácticas ágiles, describir las relaciones, factores, beneficios e inconvenientes en el uso de arquitecturas y *frameworks* ágiles.

1.3. Objetivos del trabajo

El objetivo general es realizar un mapeo sistemático que consolide las distintas investigaciones respecto al uso de arquitecturas de *software* en el desarrollo ágil, identificar patrones arquitectónicos, factores, beneficios, desafíos, y lecciones aprendidas con respecto a la combinación.

Los objetivos específicos para esta investigación son:

- OE1: Identificar los patrones arquitectónicos compatibles con los *frameworks* y metodologías ágiles.
- OE2: Identificar *frameworks* y metodologías ágiles compatibles con arquitecturas de *software*.
- OE3: Identificar los ámbitos de las industrias que utilizan la combinación de arquitectura de *software* y *frameworks*/metodologías ágiles.
- OE4: Describir las ventajas y beneficios en la adopción de una arquitectura de *software* en el desarrollo ágil.
- OE5: Describir las desventajas e inconvenientes en la adopción de una arquitectura de *software* en el desarrollo ágil.
- OE6: Determinar los factores claves que influyen en la adopción de una arquitectura de *software* en el desarrollo ágil.
- OE7: Determinar las relaciones entre los patrones de arquitectura y los *frameworks* ágiles.

Los resultados que se esperan para esta investigación son:

- R1: Listado de arquitecturas compatibles con el desarrollo ágil (OE1).
- R2: Listado de *frameworks* y metodologías ágiles compatibles con arquitecturas de *software* (OE2).
- R3: Estadísticas sobre las industrias que utilizan la combinación de arquitectura de *software* y el desarrollo ágil (OE3).
- R4: Listado de ventajas y beneficios en la adopción de una arquitectura de *software* en el desarrollo ágil (OE4).

- R5: Listado de desventajas e inconvenientes en la adopción de una arquitectura de *software* en el desarrollo ágil (OE5).
- R6: Listado de factores claves que influyen en la adopción de una arquitectura de *software* en el desarrollo ágil (OE6).
- R7: Estadísticas sobre las relaciones entre los patrones de arquitectura y el desarrollo ágil (OE7).

A partir de los objetivos y resultados descritos anteriormente se elabora el árbol de soluciones que se muestra en la Ilustración 2.

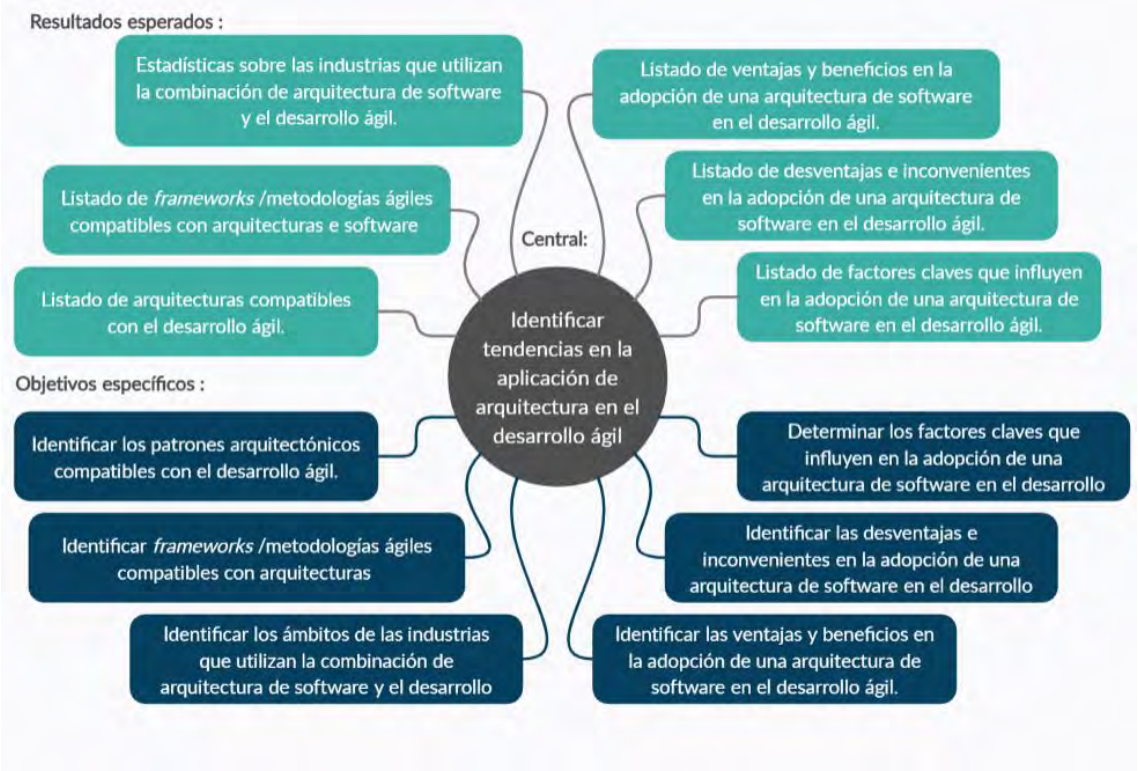

Ilustración 2 Árbol de soluciones

El actual estudio se encuentra constituido de la siguiente manera:

El capítulo 1 indica el contexto del problema, la justificación de la investigación y los objetivos del trabajo. En el capítulo 2 se presenta el marco teórico donde se describen las arquitecturas de *software* y *frameworks*/metodologías usadas en la investigación. En el capítulo 3 se muestra el estado del arte donde se analizan las 5 publicaciones relacionadas con el presente trabajo. En el capítulo 4 se

detalla la planificación del mapeo sistemático, se define la metodología, se plantean las preguntas de bibliometría e investigación, se define la estrategia de búsqueda y las cadenas de búsqueda, finalmente se plantean los criterios de inclusión y exclusión para la de selección de estudios. En el capítulo 5 se documenta la conducción del estudio, se presenta el detalle de la selección de estudios, se aplican los criterios de evaluación de calidad, finalmente se ejecuta la extracción de los datos. En el capítulo 6 se hace el análisis de resultados, se responden las preguntas bibliométricas y de investigación. En el capítulo 7 se analizan las amenazas a la validez interna, validez externa, validez del constructo y validez las conclusiones. En el capítulo 8 conclusiones y trabajo futuro. Finalmente, en el capítulo 9 se muestran las referencias usadas en la investigación.

2. MARCO DE TEÓRICO

En esta sección se presenta la definición de arquitectura de *software* y los patrones utilizados; así como la definición de metodología ágil y las metodologías usadas en la presente investigación.

2.1. La arquitectura de *software*

La arquitectura de *software* es el conjunto de estructuras necesarias para que funcione el sistema, comprende los elementos del *software*, propiedades y las relaciones entre ellos [17], es el diseño de más alto nivel en un *software* y debe describir los diferentes aspectos del sistema.

Existen diversos paradigmas y diseños arquitectónicos. Adoptar una arquitectura dependerá de la necesidad y recursos con los que cuente en proyecto. Los patrones de arquitectura referenciados en la investigación son:

2.1.1. Arquitectura Orientada a Servicios

Es un estilo arquitectónico que modularizó el sistema de información en servicios. Con SOA, estos importantes programas se convierten en servicios comerciales. Con un único servicio comercial para una función determinada que se utiliza en todas partes de la organización. Cuando se necesita cambiar la política de negocios, se puede cambiar en un lugar y debido a que el mismo servicio se usa en todas partes, la consistencia se mantendrá en toda la organización. SOA permite a las empresas tomar decisiones comerciales respaldadas por la tecnología en lugar de tomar decisiones comerciales determinadas o limitadas por la tecnología[18].

La agilidad, adaptación e independencia de plataforma son las principales ventajas de esta arquitectura.

2.1.2. N-Capas

Separa los componentes de su solución en capas. Los componentes de cada capa deben ser cohesivo y tener aproximadamente el mismo nivel de abstracción [19].

Las capas son agrupaciones horizontales lógicas de componentes de software que forman la aplicación o el servicio. Nos ayudan a diferenciar entre los

diferentes tipos de tareas a ser realizadas por los componentes, ofreciendo un diseño que maximiza la reutilización y, especialmente, la mantenibilidad [20].

Estructura el sistema en un número apropiado de capas y las coloca una encima de otra, comienza en el nivel más bajo de abstracción, llamada capa 1, esta es la base del sistema, trabaja la abstracción poniendo la capa J en la parte superior de la capa de J-1 hasta llegar al nivel superior de la funcionalidad, llamado capa N ; la principal característica estructural del patrón de capas es que los servicios de la capa J sólo son utilizados por la capa J+1, no hay más dependencias directas entre capas , la responsabilidad de la capa J es proveer servicios usados por la capa J+1 y delegar subtareas a la capa J-1 [21].

El diseño del patrón se muestra en la Ilustración 3.

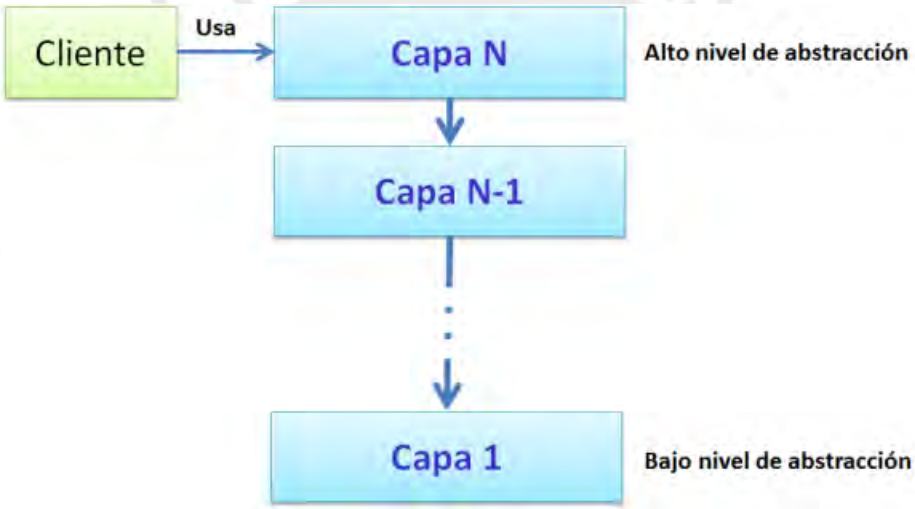

Ilustración 3 Patrón N-Capas

2.1.3. MVC

El patrón Modelo-Vista-Controlador MVC proporciona un nivel de separación entre la lógica empresarial y de presentación, trata de separar negocios lógica de la lógica de presentación [22].

El modelo se encarga de gestionar los datos de la aplicación, recibe información del usuario del controlador mientras la vista significa presentación del modelo en un formato particular asimismo el controlador responde a la entrada de datos del usuario y realiza interacciones en los objetos del modelo de datos , luego el

controlador recibe la entrada, opcionalmente la valida y luego pasa la entrada al modelo [23].

Steve Smith [24] describe en la ilustración 4 las interacciones entre los componentes MVC.

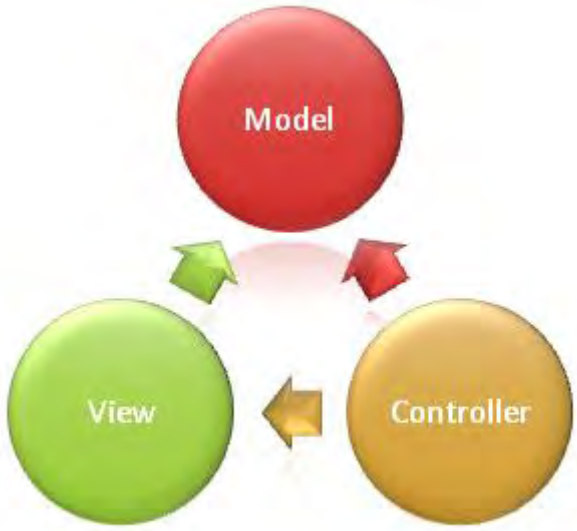

Ilustración 4 Patrón MVC

2.1.4. Microservicios

Según Derek et al. [25] una arquitectura de microservicios consta de una colección de pequeños servicios autónomos. Cada servicio es autónomo y debe implementar una única capacidad empresarial.

Los microservicios son pequeños, independientes y poco acoplados. Un solo equipo pequeño de desarrolladores puede escribir y mantener un servicio.

Cada servicio es una base de código separada, que puede ser administrada por un pequeño equipo de desarrollo.

Los servicios se pueden implementar de forma independiente. Un equipo puede actualizar un servicio existente sin reconstruir y volver a implementar toda la aplicación.

Es un patrón *stateful* y necesitaremos mantener el estado persistente. Esto difiere del modelo tradicional, donde una capa de datos separada maneja la persistencia de los datos.

La comunicación entre los componentes se realiza mediante APIs bien definidas. Los detalles de implementación interna de cada servicio están ocultos para otros servicios, no necesitan compartir la misma pila de tecnología, bibliotecas o marcos.

Derek et al. [25] en la Ilustración 5 muestra un diagrama de los componentes y sus interacciones en un estilo de microservicios.

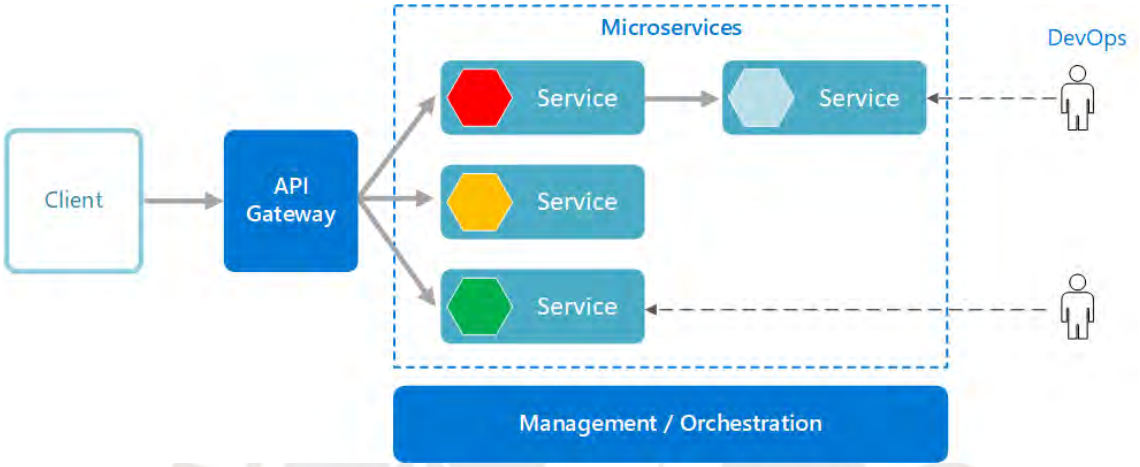

Ilustración 5 Microservicios

Los principales beneficios que aporta esta arquitectura son agilidad, equipos pequeños y enfocados, base de código pequeña, combinación de tecnologías, aislamiento de fallas, escalabilidad, aislamiento de datos.

2.1.5. Arquitectura orientada a Aspectos

Amparo Navasa et al. [26] definen a la programación orientada a aspectos (AOP) como un estilo que permite al diseñador de *software* especificar la estructura del sistema en términos de componentes y conectores. Los componentes especifican la funcionalidad del sistema mientras los conectores determinan la interacción entre los componentes. En estos términos, los arquitectos de *software* pueden concentrarse en las propiedades estructurales de los sistemas evitando los detalles de implementación. Esto hace posible afrontar sistemas complejos reduciendo costes y tiempo de desarrollo.

2.1.6. Basado en componentes

Arquitectura enfatiza en la separación de preocupaciones con respecto a la amplia funcionalidad disponible en un sistema de *software* dado [27]. Esta separación nos permite manejar problemas de arquitectura tanto a nivel de

definición como restricciones (qué tipos de componentes y conectores pueden conectarse entre sí.) y en nivel de instancia como multiplicidad (un servidor puede estar conectado por 0 ... n clientes) y dinámico.

Un componente puede tener su propia arquitectura interior. Tales componentes se llaman componentes compuestos. Con este concepto, podemos refinar la arquitectura gradualmente y hacer que el diseño sea más controlable. Además, el componente compuesto puede ser reutilizados y compuestos también: podemos reutilizar y componer artefactos de diseño a alto nivel.

2.1.7. DCI

Data, context, and interaction (DCI) enfatiza los modelos de objetos y la interacción entre objetos en lugar de clases, en complementario al patrón MVC [28]. DCI descompone el *software* en los componentes humanos más profundo que la programación orientada a clases y otros paradigmas. Además, DCI soporta explícitamente la agilidad, con soporte para:

• Usuarios finales y programadores como seres humanos con modelos mentales ricos;

• Código legible para lograr que el *software* funcione más fácilmente

• Crear un hogar para los compromisos del cliente de análisis de dominio y casos de uso;

• Evolución limpia a lo largo de los dominios dominantes de cambio en el *software*.

2.2. Metodologías y *frameworks* ágiles

Las *frameworks* ágiles surgen en la década de los noventa como respuesta a las metodologías tradicionales, siendo las primeras flexibles y modificables, dependiendo de la realidad de cada proyecto o equipo de trabajo. Los proyectos ágiles se subdividen en otros más pequeños a través de una lista ordenada de características; así mismo, cada proyecto es tratado de forma independiente y desarrolla un subconjunto de particularidades durante un periodo de tiempo corto, entre dos a seis semanas.

Es necesaria una comunicación constante con el cliente, además los proyectos son en gran parte colaborativos y se adaptan mejor a los cambios, siendo estos

una característica esperada y deseada, así como las entregas continuas al cliente y retroalimentación por parte de él.

Entre los *frameworks* ágiles referenciados en los artículos del mapeo tenemos:

2.2.1. SAFe

Scaled Agile Framework permite a las organizaciones complejas lograr los beneficios del desarrollo de *software* y sistemas *Lean-Agile* a escala. SAFe está diseñado para ayudar a las empresas a entregar valor de manera continua y más eficiente en un horario regular y predecible. Proporciona una base de conocimiento de principios y prácticas integradas y comprobadas para respaldar la agilidad empresarial [29].

2.2.2. Scrum

Scrum es un marco dentro del cual las personas pueden abordar problemas adaptativos complejos, al tiempo que ofrecen productos productivos y creativos del mayor valor posible.

El *framework* en sí mismo es un marco simple para la colaboración efectiva del equipo en productos complejos.

Las principales ventajas y desventajas de utilizar este marco de referencia son : ser ligero, simple de entender y difícil de dominar [30].

2.2.3. *Extreme Programming*

La programación extrema, conocida familiarmente como XP, es una disciplina del negocio de desarrollo de *software* que enfoca a todo el equipo en común, accesibles metas. Utilizando los valores y principios de XP, los equipos aplican prácticas de XP apropiadas en su propio contexto. Las prácticas de XP se eligen por su estímulo a creatividad humana y su aceptación de la fragilidad humana. Los equipos XP producen *software* de calidad a un ritmo sostenible [31].

2.2.4. Lean software development

Lean software development (LSD) es una metodología de desarrollo que aplica lo conceptos de lean directamente al desarrollo de *software* para mejorar en gran medida la entrega de valor a sus clientes. Concéntrese en los siete principios de *lean*: eliminar desperdicio, construir calidad, crear conocimiento, aplazar el compromiso, entregar rápido, respetar a las personas y optimizar el total [32].

3. ESTADO DEL ARTE

Se encontraron publicaciones de revisión de la literatura relacionadas al tema principal de la investigación. C. Yang et al. [33] presentan un mapeo sistemático sobre la combinación de actividades de arquitectura de *software* y desarrollo ágil, cubre 54 artículos desde junio 2001 hasta enero del 2014 en el que identifica y analiza los enfoques y las actividades de arquitectura, análisis, síntesis, evaluación, implementación y mantenimiento en el desarrollo ágil, sin embargo no presentan las arquitecturas de *software* utilizadas en las publicaciones. Por otro lado, Z. Dragičević y S. Bošnjak [34] desarrollan un mapeo sistemático en el cual seleccionan 61 artículos desde el año 2001 hasta el 2017 donde analizan las tendencias , desafíos y factores de éxito de las arquitecturas ágiles ,los desafíos claves encontrados en la arquitectura ágil en era digital son: la aplicación de microservicios, equilibrar la agilidad y la arquitectura, respecto a los factores del éxito son: comprender el contexto, elegir una estrategia de implementación ,no obstante no detallan las metodologías ágiles ni arquitecturas de *software* usadas en la investigación. Gilberto Borrego y et al. [35] presentan un mapeo sistemático donde 42 artículos son seleccionados desde el año 2002 al 2017 en el cual se identifican y describen enfoques para administrar conocimiento arquitectónico en los equipos de desarrollo de *software* ágil y global. Estos enfoques se pueden agrupar como documentación basado en artefactos, basado en la comunicación y basado en metodología, abordan la combinación sobre arquitectura de *software* y desarrollo ágil, pero se enfocan en las formas de administrar el conocimiento diferente enfoque al planteado en la investigación. Vinod Menon et al. [36] muestran un mapeo sistemático en el cual se seleccionan 14 artículos desde el año 2006 al 2015 donde identifica los desafíos a considerar en la arquitectura de *software* al migrar de proyectos tradicionales a entornos ágiles, abordando los factores humanos, de proceso y tecnología aunque no profundiza en la interacción de metodología o *frameworks* ágiles con arquitecturas de *software*.

4. PLANIFICACIÓN

4.1. Metodología

Los estudios de mapeo sistemáticos o estudios de alcance están diseñados para dar una visión general de un área de investigación a través de la clasificación y contando contribuciones en relación con las categorías de esa clasificación.

Petersen et al. [37] también definieron cinco etapas para realizar el mapeo sistemático: Definir preguntas de investigación, realizar la búsqueda literaria, seleccionar estudios, clasificar artículos y extraer y realizar la agregación de datos

Basado en la guía presentada por Petersen et al. [37] se desarrolló Diagrama de flujo prsentado en la Ilustración 6.

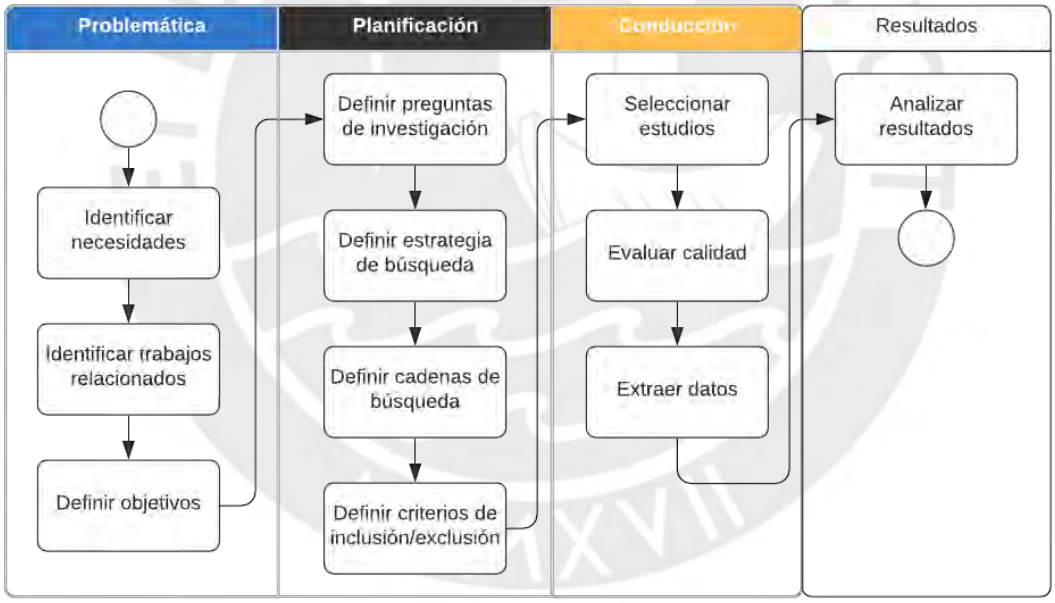

Ilustración 6 Diagrama de flujo proceso de mapeo sistemático.

4.2. Preguntas de bibliometría

Para la investigación, se han propuesto las siguientes peguntas de bibliometría:

PB-1 ¿Qué proporción de las investigaciones son académicas y de aplicación industrial, en que sectores industriales se investiga y/o aplica?

El objetivo es identificar el sector de origen de las publicaciones existentes.

PB-2 ¿Cómo ha ido evolucionando el número de publicaciones relacionados al tema de investigación?

El objetivo es establecer la evolución de los estudios relacionados a las arquitecturas de *software* en el desarrollo ágil.

PB-3 ¿Qué tipos de estudios se realizan con mayor frecuencia con relación al tema trazado?

El objetivo es identificar el tipo de estudios que se realizan.

PB-4 ¿Qué canales de publicación (revistas y/o eventos) son los principales objetivos para las investigaciones relacionados al tema de investigación?

El objetivo es identificar los canales de publicación.

4.3. Preguntas de investigación

Las preguntas formuladas para la presente investigación son:

PI-1 ¿Cuáles son las arquitecturas de *software* que pueden utilizarse en el desarrollo ágil?

El objetivo es identificar las arquitecturas compatibles con el desarrollo ágil.

PI-2 ¿Cuáles son los *frameworks*/metodologías ágiles usados?

El objetivo es identificar las *frameworks*/metodologías ágiles usados con arquitecturas de *software*.

PI-3 ¿En qué ámbitos de la industria tienden a utilizar en combinación arquitecturas de *software* y *frameworks* ágiles, considerando proyectos ágiles?

El objetivo es verificar los sectores de la industria que utilizan en combinación arquitecturas de *software* y desarrollo ágil.

PI-4 ¿Cuáles son las ventajas y beneficios en la adopción de una arquitectura de *software* en el desarrollo ágil?

El objetivo es analizar el beneficio del uso combinado de arquitecturas de *software* y el desarrollo ágil.

PI-5 ¿Cuáles son las desventajas e inconvenientes en la adopción de una arquitectura de *software* en el desarrollo ágil?

El objetivo es analizar las dificultades del uso combinado de arquitecturas de *software* y el desarrollo ágil.

PI-6 ¿Cuáles son los factores que influyen adopción de una arquitectura de *software* en el desarrollo ágil?

El objetivo es determinar los factores claves en el uso combinado de arquitectura de *software* y prácticas ágiles.

PI-7 ¿Cuáles son las relaciones entre arquitecturas de *software* y *frameworks*/metodologías ágiles?

El objetivo es explorar las combinaciones usadas entre arquitectura de *software* y *frameworks*/metodologías ágiles.

PI-8 ¿Cuáles son las relaciones entre las arquitecturas de *software* y las industrias?

El objetivo es identificar las arquitecturas de *software* en las diversas industrias.

PI-9 ¿Cómo han evolucionado las principales arquitecturas de *software* en el desarrollo ágil a través de los años?

Se pretender identificar las variaciones del uso de arquitecturas de *software* a través de los años.

PI-10 ¿Cómo ha evolucionado los principales *frameworks*/metodologías usados a través de los años?

Se pretender identificar las variaciones del uso de los principales *frameworks* y/o metodologías de *software* a través de los años.

4.4. Estrategia de búsqueda

Las preguntas de investigación planteadas en la sección 4.3 para este mapeo sistemático se han constituido con la ayuda de los criterios PICo (Población , interés, Comparación) ,el detalle se muestra en la Tabla 1.

Concepto	Términos
Población	Arquitectura de *software*, arquitectura de aplicaciones, arquitectura ágil
Interés	Ágil, agilidad, programación Extrema, XP, desarrollo basado en funcionalidades, FDD, Scrum, cristal, programación en parejas, *lean software development, lean development*, LSD
Comparación	Computación en la nube, transformación digital, marco ágil escalado (SAFe), desarrollo de *software*, ingeniería de software, diseño de *software*

Tabla 1 PICo

4.5. Cadenas de búsqueda

Las bases de datos especializadas electrónicas de artículos científicos seleccionados para este estudio son: SCOPUS (www.scopus.com), IEEE (https://ieeexplore.ieee.org), *Web of Science* (www.webofknowledge.com), *ACM Digital* (dl.acm.org). Las cadenas de búsqueda se detallan en la Tabla 2.

Fuente	Cadena de búsqueda
Scopus	*TITLE-ABS-KEY(("Software architecture" OR "Application architecture" OR "Agile architecture") AND ("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "lean software development" OR "lean development" OR "LSD") AND ("Cloud computing" OR "Digital Transformation" OR "Scaled Agile Framework" OR "software development" OR "software engineering")) AND ((LIMIT-TO(LANGUAGE"English") OR LIMIT-TO(LANGUAGE"Spanish")))*
IEEE	*(("All Metadata": "Software architecture" OR "All Metadata": "Application architecture" OR "All Metadata": "Agile architecture") AND ("All Metadata": "agile" OR "All Metadata": "agility" OR "All Metadata": "extreme programming" OR "All Metadata": "XP" OR "All Metadata": "feature driven development" OR "All Metadata": "FDD" OR "All Metadata": "scrum" OR "All Metadata": "crystal" OR "All Metadata": "pair programming" OR "All Metadata": "lean software development" OR "All Metadata": "lean development" OR "All Metadata": "LSD") AND ("All Metadata": "Cloud computing" OR "All Metadata": "Digital Transformation" OR "All Metadata": "Scaled Agile Framework" OR "All Metadata": "software development" OR "All Metadata": "software engineering" OR "All Metadata": "Software design")*

ACM	*(Abstract:("Software architecture" OR "Application architecture" OR "Agile architecture")) AND (Abstract:("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "lean software development" OR "lean development" OR "LSD")) AND (Abstract:("Cloud computing" OR "Digital Transformation" OR "Scaled Agile Framework" OR "software development" OR "software engineering" OR "Software design"))*
Web of Science	*TS=(("Software architecture" OR "Application architecture" OR "Agile architecture") AND ("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "lean software development" OR "lean development" OR "LSD") AND ("Cloud computing" OR "Digital Transformation" OR "Scaled Agile Framework" OR "software development" OR "software engineering" OR "Software design"))*

Tabla 2 Cadenas de búsqueda

4.6. Criterios de selección de estudios

Los criterios de inclusión/exclusión se definieron para seleccionar las publicaciones que se usarán en la investigación, los siguientes criterios fueron considerados para la investigación:

- **Criterios de Inclusión**

Se aplicaron los siguientes criterios de inclusión a los títulos y resúmenes que se detallan en la Tabla 3.

Id	Criterios de inclusión
CI-1	Artículos académicos con sustentación metodológica (Experimentos Controlados, Estudios de Caso, Revisiones Sistemáticas, Mapeos Sistemáticos)
CI-2	Artículos extraídos de las bases de datos digitales (*Web of Science*,Scopus,IEEE,ACM), conferencias y bibliotecas establecidas.
CI-3	Publicaciones y artículos escritos en idioma inglés y español.

CI-4	Se aceptarán solamente artículos que en su contenido referencien el uso combinado de prácticas ágiles y arquitectura de software.

Tabla 3 Criterios de Inclusión

- **Criterios de Exclusión**

Los criterios de exclusión aplicados a los títulos y resúmenes se detallan en la tabla 4.

Id	Criterios de exclusión
CE-1	No se consideran artículos que no formen parte de los siguientes tipos de publicaciones: *Journals*, Conferencias, Revistas Indexadas, Bases de Datos digitales.
CE-2	No serán considerados los estudios terciarios ni los resúmenes de conferencias
CE-3	Serán excluidos los estudios cuyo contenido sea irrelevante o esté fuera del contexto de la MSL
CE-4	Se excluyen artículos que indiquen únicamente la aplicación de una arquitectura de software sin considerar el desarrollo ágil.
CE-5	Se excluyen artículos que indiquen únicamente la aplicación de una práctica ágil sin considerar el contexto de arquitecturas de software
CE-6	Publicaciones a partir del año 2015

Tabla 4 Criterios de Exclusión

5. CONDUCCIÓN

5.1. Selección de estudios

En mayo del 2020 se hizo una búsqueda inicial en la base de datos Scopus para detectar si existen investigaciones que tratan sobre la combinación de arquitecturas de software en el desarrollo ágil, no se encontró mapeos sistemáticos que aborden directamente el tema y objetivos planteados en la presente investigación, la cadena de búsqueda ejecutada fue

(TITLE-ABS-KEY ("literature Review" OR "systematic mapping") AND TITLE-ABS-KEY ("agile" OR "agility") AND TITLE-ABS-KEY ("Software architecture" OR "Application architecture" OR "Agile architecture"))

En junio del 2020 se ejecutaron las cadenas de búsqueda definidas en las Base de datos SCOPUS (www.scopus.com), IEEE (https://ieeexplore.ieee.org), *Web of Science* (www.webofknowledge.com) y ACM Digital (dl.acm.org) siguiendo el protocolo de Petersen [37]. Se descartaron los artículos duplicados y, luego de aplicar los criterios de exclusión e inclusión, se obtuvieron 121 artículos. Se realizó una lectura rápida de estos artículos, obteniendo un total de 68 artículos preseleccionados, el detalle se presenta en la Tabla 5.

Fuente	Original	Duplicados	CE1	CE2	CE3	CE4	CE5	CE6	Sub Total	Lectura rápida	Total
Scopus	396	(5)	(203)	(2)	(19)	(57)	(18)	(8)	**84**	(38)	46
IEEE	306	(60)	(185)	(6)	-	(15)	(9)	(5)	**26**	(10)	16
ACM	164	(44)	(100)	-	(1)	(8)	(3)	-	**8**	(2)	6
WoS	32	(17)	(7)	-	(1)	(4)	-	-	**3**	(3)	-
TOTAL	**898**	**(126)**	**(495)**	**(8)**	**(21)**	**(84)**	**(30)**	**(13)**	**121**	**(53)**	**68**

Tabla 5 Criterios de selección

5.2. Evaluación de calidad

Según el protocolo recomendado por Petersen [37] se definió una lista de preguntas de comprobación basadas en la lista de preguntas propuestas por

Kitchenham [38] de 3 criterios para evaluar la calidad de los estudios: Sí cumple (S) = 2, Cumple parcialmente (P) = 1, No cumple (N) = 0; obteniéndose de esa forma una calificación entre 0 y 6 por cada estudio. Se considerarán válidos los artículos con un puntaje superior a 3. Se detallan los criterios de calidad en la Tabla 6.

Id	Criterio
QA1	¿La motivación del estudio está claramente fijado?
QA2	¿El proceso de investigación está claramente definido?
QA3	¿L os hallazgos contribuyen a dar respuesta a las preguntas de investigación?

Tabla 6 Criterios de calidad

En la tabla 7 se muestran los puntajes obtenido luego de aplicar los criterios de calidad.

Autores	Titulo	QA1	QA2	QA3	To-tal
Capilla R., Jansen A., Tang A., Avgeriou P., Babar M.A.	*10 years of software architecture knowledge management: Practice and future*	2	1	2	5
Schlichthaerle S., Becker K., Sperber S.	*A domain-specific language based architecture modeling approach for safety critical automotive software systems*	2	2	2	6
J. Hernández-Reveles; G. Sobrevilla-Dominguez; P. Velasco-Elizondo; S. Soriano-Grande	*Adding agile architecture practices to a Cyber-Physial System devel-opment*	2	2	2	6

N. Santos; R. J. Machado; N. Ferreira	*Adopting Logical Architectures within Agile Projects*	2	1	1	4
Usländer T., Batz T.	*Agile service engineering in the Industrial Internet of Things*	2	2	2	6
Durisic D., Berenyi A.	*Agile System Architecture in Large Organizations: An Experience Report from Volvo Cars*	2	2	2	6
Waterman M.	*Agility, risk, and uncertainty, part 2: How risk impacts agile architecture*	1	0	1	2
Woods E.	*Aligning Architecture Work with Agile Teams*	1	0	1	2
Pang, Chung-Yeung	*An Agile Architecture for a Legacy Enterprise IT System*	1	1	2	4
Rix M., Kujat B., Meisen T., Jeschke S.	*An Agile Information Processing Framework for High Pressure Die Casting Applications in Modern Manufacturing Systems*	2	1	2	5
Posadas J.V.	*Application of mixed distributed software architectures for social-productive projects management in peru*	2	1	2	5
Zimmermann O., Wegmann L., Koziolek H., Goldschmidt T.	*Architectural Decision Guidance Across Projects-Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge*	2	2	2	6
Valente P., Silva T., Winckler M., Nunes N.	*Bridging enterprise and software engineering through an user-centered design perspective*	2	2	2	6
Gamal M.M., Ramadan T., El Adawy H.	*Case study applying agile service-oriented modeling and architecture approach for better business-services alignment*	1	1	2	4
Tharayil R.	*Change vector tracking in emergent design*	2	2	2	6
A. Hindle	*Complexity: Let's Not Make This Complicated*	1	0	2	3

V. Stirbu; T. Mikkonen	*CompliancePal: A Tool for Supporting Practical Agile and Regulatory-Compliant Development of Medical Software*	2	2	2	6
Maric M., Matkovic P., Tumbas P., Pavlicevic V.	*Contextual factors of architectural strategy for complex systems*	2	2	2	6
O. Schmidts; B. Kraft; M. Schreiber; A. Zündorf	*Continuously Evaluated Research Projects in Collaborative Decoupled Environments*	2	2	2	6
Lescano N.L., Mamani S.E., Illatopa J.G.	*Cultiventura software architecture tool supporting the learning of the Moche culture: Videogames and augmented reality*	2	2	2	6
S. Filiposka; A. Mishev; F. Wein; J. Sobieski	*Customer-Centric Service Provider Architecture for the R&E Community*	1	1	2	4
Angelov S., de Beer P.	*Designing and applying an approach to software architecting in agile projects in education*	2	2	2	6
Cervantes, Humberto; Kazman, Rick	*Designing Software Architectures: A Practical Approach*	2	2	2	6
Tipaldi M., Legendre C., Koopmann O., Ferraguto M., Wenker R., D'Angelo G.	*Development strategies for the satellite flight software on-board Meteosat Third Generation*	1	2	1	4
Raghunathan V.S., Dinesh Kumar S., Thamaraiselvi G.	*E-governance service delivery platform – Platform to optimize SDLC, re-engineering application architecture and elimination of processes*	2	2	1	5
De Lange P., Nicolaescu	*Enhancing MDWE with collaborative live coding*	1	1	2	4

P., Winkler T., Klamma R.					
J. Werewka; A. Spiechowicz	*Enterprise architecture approach to SCRUM processes, sprint retrospective example*	2	2	2	6
Bluemke I., Stepień A.	*Experiences with DCI pattern*	2	1	1	4
Fazio M., Celesti A., Marquez F.G., Glikson A., Villari M.	*Exploiting the FIWARE cloud platform to develop a remote patient monitoring system*	2	2	2	6
Dingsøyr T., Moe N.B., Fægri T.E., Seim E.A.	*Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation*	2	2	2	6
R. O'Connor; P. Elger; P. M. Clarke	*Exploring the Impact of Situational Context — A Case Study of a Software Development Process for a Microservices Architecture*	2	2	2	6
Gultureanu D., Kerns K., Henthorn T., Quach J., Kleen M.	*Flight software development and validation workflow management system*	2	2	2	6
J. A. Diaz-Pace; A. J. Bianchi	*High-Level Design Stories in Architecture-Centric Agile Development*	2	1	1	4
Pedoto R.W., Albers C.M., Benjamin D.N., Reynolds J.F.	*Innovative development of a cross-center timeline planning tool*	2	2	2	6
Taibi D., Lenarduzzi V., Pahl C., Janes A.	*Microservices in agile software development: A workshop-based study into issues, advantages, and disadvantages*	2	2	2	6
C. Pautasso; O. Zimmermann; M. Amundsen; J.	*Microservices in Practice, Part 1: Reality Check and Service Design*	1	0	2	3

Lewis; N. Josuttis					
Zimmermann O.	*Microservices tenets: Agile approach to service development and deployment*	2	2	2	6
Essebaa I., Chantit S.	*Model driven architecture and agile methodologies: Reflexion and discussion of their combination*	2	2	2	6
Ihor B., Oleksii D., Aleksandr K., Nataliia K., Oleksandr M., Volodymyr P.	*Multicriteria Choice of Software Architecture Using Dynamic Correction of Quality Attributes*	2	2	2	6
Atanasovski B., Bogdanovic M., Velinov G., Stoimenov L., Dimovski A.S., Koteska B., Jankovic D., Skrceska I., Kon-Popovska M., Jakimovski B.	*On defining a model driven architecture for an enterprise e-health system*	2	2	2	6
Wohlrab R., Pelliccione P., Knauss E., Heldal R.	*On interfaces to support agile architecting in automotive: An exploratory case study*	2	2	2	6
Costa, Pedro Henrique Teixeira; Canedo, Edna Dias; Bonif'{a}cio, Rodrigo	*On the Use of Metaprogramming and Domain Specific Languages: An Experience Report in the Logistics Domain*	2	2	1	5
Wirfs-Brock, Rebecca; Yoder, Joseph; Guerra, Eduardo	*Patterns to Develop and Evolve Architecture during an Agile Software Project*	2	2	2	6

Borrego G., Moran A.L., Palacio R.	*Preliminary evaluation of a tag-based knowledge condensation tool in agile and distributed teams*	2	2	2	6
Carver J.C., De Almeida E.S., Capilla R., Minku L.L., Muccini H., Penzenstadler B.	*Product Lines, Energy Conservation, Use Cases, Agile Development, and Infotainment*	1	2	1	4
Bosnic I., Cavrak I.	*Project Work Division in Agile Distributed Student Teams - Who Develops What?*	2	2	2	6
L. Prechelt; H. Schmeisky; F. Zieris	*Quality Experience: A Grounded Theory of Successful Agile Projects without Dedicated Testers*	2	2	2	6
Galster M., Angelov S., Martínez-Fernández S., Tofan D.	*Reference architectures and scrum: Friends or foes?*	2	2	2	6
S. Ma; Y. Chuang; C. Lan; H. Chen; C. Huang; C. Li	*Scenario-Based Microservice Retrieval Using Word2Vec*	2	2	2	6
Carrillo M.H., Franky C., Páez P.S., Pedraza A.F.	*SCLOUDPY: Multi-tenant and cloud web information system for orders processing in SMEs [SCLOUDPY: Sistema Informático Web de Multi-Tenencia para el Procesamiento en la Nube de Pedidos de PYMES]*	2	2	2	6
Essebaa I., Chantit S.	*Scrum and V lifecycle combined with model-based testing and model driven architecture to deal with evolutionary system issues*	2	2	2	6
Sievi-Korte O.,	*Software architecture design in global software development: An empirical study*	2	2	2	6

Richardson I., Beecham S.					
Misra R., Panigrahi C.R., Panda B., Pati B.	*Software design*	0	0	0	0
Arabnia, Hamid R.; Deligiannidis, Leonidas; Jandieri, George; Solo, Ashu M. G.; Tinetti, Fernando G.	*Software Engineering Research and Practice*	0	0	0	0
Martini A., Bosch J.	*The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles*	2	2	2	6
P. Kruchten	*The End of Agile as We Know It*	2	2	2	6
M. Kersten	*The End of the Manufacturing-Line Analogy*	1	1	1	3
Valente P., Silva T., Winckler M., Nunes N.	*The goals approach: Agile enterprise driven software development*	2	2	2	6
Alsahli A., Khan H., Alyahya S.	*Toward an Agile Approach to Managing the Effect of Requirements on Software Architecture during Global Software Development*	2	2	2	6
Martin K., Omer U., Florian M.	*Towards a continuous feedback loop for service-oriented environments*	2	2	2	6
Fischinger M., Egger N., Binder C., Neureiter C.	*Towards a Model-centric Approach for Developing Dependable Smart Grid Applications*	2	2	2	6
Santos N., Ferreira N., Machado R.J.	*Towards agile architecting: Proposing an architectural pathway within an industry 4.0 project*	2	2	2	6

O. Alam	*Towards an Agile Concern-Driven Development Process*	2	2	1	5
Martini A., Pareto L., Bosch J.	*Towards introducing agile architecting in large companies: The CAFFEA framework*	2	2	2	6
Gilson F., Englebert V.	*Transformation-wise design of software architectures*	2	2	2	6
Shumaiev, Klym; Bhat, Manoj; Klymenko, Oleksandra; Biesdorf, Andreas; Hohenstein, Uwe; Matthes, Florian	*Uncertainty Expressions in Software Architecture Group Decision Making: Explorative Study*	2	2	2	6
Uludağ Ö., Kleehaus M., Erçelik S., Matthes F.	*Using social network analysis to investigate the collaboration between architects and agile teams: A case study of a large-scale agile development program in a german consumer electronics company*	2	2	1	5
T. Ueda; T. Nakaike; M. Ohara	*Workload characterization for microservices*	2	2	1	5

Tabla 7 Aplicación de criterios de calidad

Luego de aplicar los criterios de calidad definidos, de descartan 7 artículos que no superaron el puntaje definido, quedando finalmente 61 artículos para la presente investigación, el detalle se presenta en la tabla 8.

Fuente	Original	Duplicado	Criterios inclusión exclusión	Lectura rápida	Sub Total	Criterios de calidad	Total
Scopus	396	(5)	(307)	(38)	46	(3)	43
IEEE	306	(60)	(220)	(10)	16	(3)	13
ACM	164	(44)	(112)	(2)	6	(1)	5
WoS	32	(17)	(12)	(3)	-	-	-
TOTAL	**898**	**(126)**	**(651)**	**(53)**	**68**	**(7)**	**61**

Tabla 8 Selección de artículos

Basado en el gráfico propuesto por Petersen et al. [37] se elaboró la Ilustración 7 donde se muestra el flujo de selección de artículos.

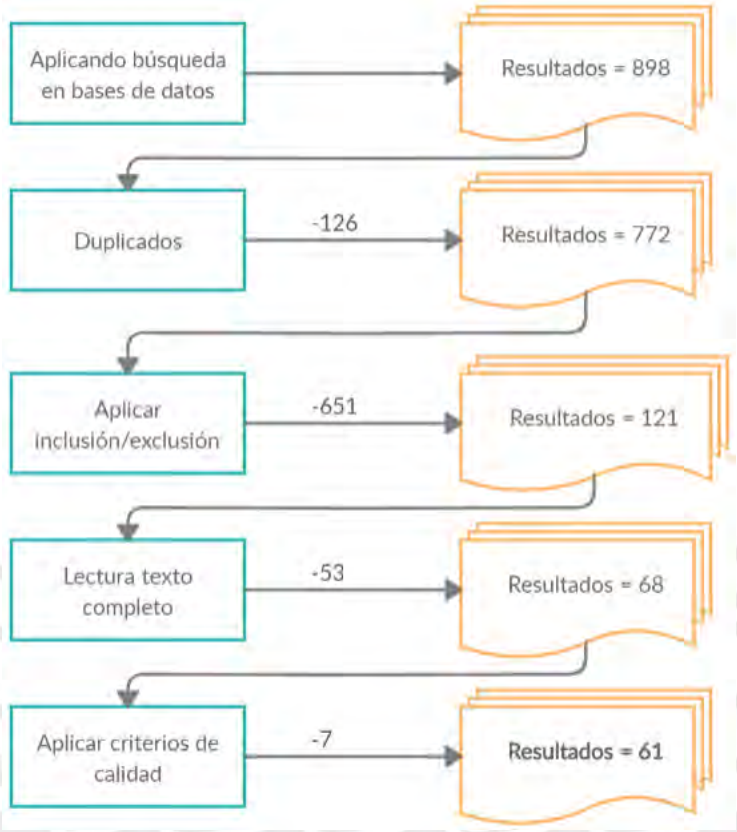

Ilustración 7 Selección de artículos

5.3. Extracción de los datos

Con el objetivo de responder a las preguntas planteadas se procede a la extracción de datos de los 61 artículos seleccionados se procederá a la, los siguientes datos fueron obtenidos de los estudios:

- **Id**: Identificador
- **Autores**
- **Título**
- **Origen de publicación:** Identificar el sector de origen si es académico o aplicación industrial, responde a la pregunta PB-1.
- **Fecha de publicación:** Responde la pregunta PB-2.
- **Tipo de estudio**: Si es un artículo de revista o un artículo de conferencia, responde a la pregunta PB-3.

- **Canal de Publicación**: Nombre de la conferencia o revista, responde la pregunta PB-4.
- **Arquitecturas de software:** Identificar los patrones de arquitectura usados, responde a la pregunta PI-1.
- ***Frameworks* y/o metodologías ágiles**: Identificar los frameworks y/o metodologías ágiles usados, responde a la pregunta PI-2
- **Industria:** identificar el sector industrial donde se aplicó el estudio si es el caso, responde a la pregunta PI-3.
- **Ventajas y beneficios**: Obtener las ventajas y beneficios en la adopción de arquitecturas de software en entornos ágiles, responde a la pregunta PI-4.
- **Desventajas e inconvenientes**: Determinar las desventajas e inconvenientes en la adopción de arquitecturas de software en entornos ágiles, responde a la pregunta PI-5.
- **Factores**: Factores claves para la adopción de arquitecturas de software en entornos ágiles, responde a la pregunta PI-6.

Los artículos seleccionados para la extracción de datos se detallan en la tabla 9.

Id	Autores	Título
P1	Capilla R., Jansen A., Tang A., Avgeriou P., Babar M.A.	*10 years of software architecture knowledge management: Practice and future*
P2	Schlichthaerle S., Becker K., Sperber S.	*A domain-specific language based architecture modeling approach for safety critical automotive software systems*
P3	J. Hernández-Reveles; G. Sobrevilla-Dominguez; P. Velasco-Elizondo; S. Soriano-Grande	*Adding agile architecture practices to a Cyber-Physial System development*

P4	N. Santos; R. J. Machado; N. Ferreira	*Adopting Logical Architectures within Agile Projects*
P5	Usländer T., Batz T.	*Agile service engineering in the Industrial Internet of Things*
P6	Durisic D., Berenyi A.	*Agile System Architecture in Large Organizations: An Experience Report from Volvo Cars*
P7	Pang, Chung-Yeung	*An Agile Architecture for a Legacy Enterprise IT System*
P8	Rix M., Kujat B., Meisen T., Jeschke S.	*An Agile Information Processing Framework for High Pressure Die Casting Applications in Modern Manufacturing Systems*
P9	Posadas J.V.	*Application of mixed distributed software architectures for social-productive projects management in peru*
P10	Zimmermann O., Wegmann L., Koziolek H., Goldschmidt T.	*Architectural Decision Guidance Across Projects-Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge*
P11	Valente P., Silva T., Winckler M., Nunes N.	*Bridging enterprise and software engineering through an user-centered design perspective*
P12	Gamal M.M., Ramadan T., El Adawy H.	*Case study applying agile service-oriented modeling and architecture approach for better business-services alignment*
P13	Tharayil R.	*Change vector tracking in emergent design*
P14	V. Stirbu; T. Mikkonen	*CompliancePal: A Tool for Supporting Practical Agile and Regulatory-Compliant Development of Medical Software*

P15	Maric M., Matkovic P., Tumbas P., Pavlicevic V.	*Contextual factors of architectural strategy for complex systems*
P16	O. Schmidts; B. Kraft; M. Schreiber; A. Zündorf	*Continuously Evaluated Research Projects in Collaborative Decoupled Environments*
P17	Lescano N.L., Mamani S.E., Illatopa J.G.	*Cultiventura software architecture tool supporting the learning of the Moche culture: Videogames and augmented reality*
P18	S. Filiposka; A. Mishev; F. Wein; J. Sobieski	*Customer-Centric Service Provider Architecture for the R&E Community*
P19	Angelov S., de Beer P.	*Designing and applying an approach to software architecting in agile projects in education*
P20	Cervantes, Humberto; Kazman, Rick	*Designing Software Architectures: A Practical Approach*
P21	Tipaldi M., Legendre C., Koopmann O., Ferraguto M., Wenker R., D'Angelo G.	*Development strategies for the satellite flight software on-board Meteosat Third Generation*
P22	Raghunathan V.S., Dinesh Kumar S., Thamaraiselvi G.	*E-governance service delivery platform – Platform to optimize SDLC, re-engineering application architecture and elimination of processes*
P23	De Lange P., Nicolaescu P., Winkler T., Klamma R.	*Enhancing MDWE with collaborative live coding*
P24	J. Werewka; A. Spiechowicz	*Enterprise architecture approach to SCRUM processes, sprint retrospective example*
P25	Bluemke I., Stepień A.	*Experiences with DCI pattern*

P26	Fazio M., Celesti A., Marquez F.G., Glikson A., Villari M.	*Exploiting the FIWARE cloud platform to develop a remote patient monitoring system*
P27	Dingsøyr T., Moe N.B., Fægri T.E., Seim E.A.	*Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation*
P28	R. O'Connor; P. Elger; P. M. Clarke	*Exploring the Impact of Situational Context — A Case Study of a Software Development Process for a Microservices Architecture*
P29	Gultureanu D., Kerns K., Henthorn T., Quach J., Kleen M.	*Flight software development and validation workflow management system*
P30	J. A. Diaz-Pace; A. J. Bianchi	*High-Level Design Stories in Architecture-Centric Agile Development*
P31	Pedoto R.W., Albers C.M., Benjamin D.N., Reynolds J.F.	*Innovative development of a cross-center timeline planning tool*
P32	Taibi D., Lenarduzzi V., Pahl C., Janes A.	*Microservices in agile software development: A workshop-based study into issues, advantages, and disadvantages*
P33	Zimmermann O.	*Microservices tenets: Agile approach to service development and deployment*
P34	Essebaa I., Chantit S.	*Model driven architecture and agile methodologies: Reflexion and discussion of their combination*
P35	Ihor B., Oleksii D., Aleksandr K., Nataliia K.,	*Multicriteria Choice of Software Architecture Using Dynamic Correction of Quality Attributes*

	Oleksandr M., Volodymyr P.	
P36	Atanasovski B., Bogdanovic M., Velinov G., Stoimenov L., Dimovski A.S., Koteska B., Jankovic D., Skrceska I., Kon-Popovska M., Jakimovski B.	*On defining a model driven architecture for an enterprise e-health system*
P37	Wohlrab R., Pelliccione P., Knauss E., Heldal R.	*On interfaces to support agile architecting in automotive: An exploratory case study*
P38	Costa, Pedro Henrique Teixeira; Canedo, Edna Dias; Bonif'{a}cio, Rodrigo	*On the Use of Metaprogramming and Domain Specific Languages: An Experience Report in the Logistics Domain*
P39	Wirfs-Brock, Rebecca; Yoder, Joseph; Guerra, Eduardo	*Patterns to Develop and Evolve Architecture during an Agile Software Project*
P40	Borrego G., Moran A.L., Palacio R.	*Preliminary evaluation of a tag-based knowledge condensation tool in agile and distributed teams*
P41	Carver J.C., De Almeida E.S., Capilla R., Minku L.L., Muccini H., Penzenstadler B.	*Product Lines, Energy Conservation, Use Cases, Agile Development, and Infotainment*
P42	Bosnic I., Cavrak I.	*Project Work Division in Agile Distributed Student Teams - Who Develops What?*
P43	L. Prechelt; H. Schmeisky; F. Zieris	*Quality Experience: A Grounded Theory of Successful Agile Projects without Dedicated Testers*

P44	Galster M., Angelov S., Martínez-Fernández S., Tofan D.	*Reference architectures and scrum: Friends or foes?*
P45	S. Ma; Y. Chuang; C. Lan; H. Chen; C. Huang; C. Li	*Scenario-Based Microservice Retrieval Using Word2Vec*
P46	Carrillo M.H., Franky C., Páez P.S., Pedraza A.F.	*SCLOUDPY: Multi-tenant and cloud web information system for orders processing in SMEs [SCLOUDPY: Sistema Informático Web de Multi-Tenencia para el Procesamiento en la Nube de Pedidos de PYMES]*
P47	Essebaa I., Chantit S.	*Scrum and V lifecycle combined with model-based testing and model driven architecture to deal with evolutionary system issues*
P48	Sievi-Korte O., Richardson I., Beecham S.	*Software architecture design in global software development: An empirical study*
P49	Martini A., Bosch J.	*The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles*
P50	P. Kruchten	*The End of Agile as We Know It*
P51	Valente P., Silva T., Winckler M., Nunes N.	*The goals approach: Agile enterprise driven software development*
P52	Alsahli A., Khan H., Alyahya S.	*Toward an Agile Approach to Managing the Effect of Requirements on Software Architecture during Global Software Development*
P53	Martin K., Omer U., Florian M.	*Towards a continuous feedback loop for service-oriented environments*

P54	Fischinger M., Egger N., Binder C., Neureiter C.	*Towards a Model-centric Approach for Developing Dependable Smart Grid Applications*
P55	Santos N., Ferreira N., Machado R.J.	*Towards agile architecting: Proposing an architectural pathway within an industry 4.0 project*
P56	O. Alam	*Towards an Agile Concern-Driven Development Process*
P57	Martini A., Pareto L., Bosch J.	*Towards introducing agile architecting in large companies: The CAFFEA framework*
P58	Gilson F., Englebert V.	*Transformation-wise design of software architectures*
P59	Shumaiev, Klym; Bhat, Manoj; Klymenko, Oleksandra; Biesdorf, Andreas; Hohenstein, Uwe; Matthes, Florian	*Uncertainty Expressions in Software Architecture Group Decision Making: Explorative Study*
P60	Uludağ Ö., Kleehaus M., Erçelik S., Matthes F.	*Using social network analysis to investigate the collaboration between architects and agile teams: A case study of a large-scale agile development program in a german consumer electronics company*
P61	T. Ueda; T. Nakaike; M. Ohara	*Workload characterization for microservices*

Tabla 9 Estudios seleccionados

6. ANÁLISIS DE RESULTADOS

Siguiendo el formulario de extracción se procedió a recopilar la información para responder las preguntas bibliométricas y de investigación planteadas.

6.1. Preguntas Bibliométricas

PB-1 ¿Qué proporción de las investigaciones son académicas y de aplicación industrial, en que sectores industriales se investiga y/o aplica?

De los 61 *papers* finales ,33 son investigaciones aplicadas en la industria (54%) y 28 investigaciones (46%) en el ámbito académico, en la Tabla 10 se detallan las investigaciones.

Sector (Total)	Artículos
Industria (33)	[P2], [P3], [P6], [P8], [P9], [P11], [P12], [P13], [P14], [P17], [P19], [P21], [P22], [P26], [P27], [P28], [P29], [P30], [P31], [P34], [P35], [P36], [P37], [P38], [P41], [P42], [P44], [P46], [P47], [P52], [P54], [P59], [P60]
Académico (28)	[P1], [P4], [P5], [P7], [P10], [P15], [P16], [P18], [P20], [P23], [P24], [P25], [P32], [P33], [P39], [P40], [P43], [P45], [P48], [P49], [P50], [P51], [P53], [P55], [P56], [P57], [P58], [P61]

Tabla 10 Artículos clasificados por sector

En la Ilustración 8 se muestra la proporción de investigaciones en el ámbito académico e industrial.

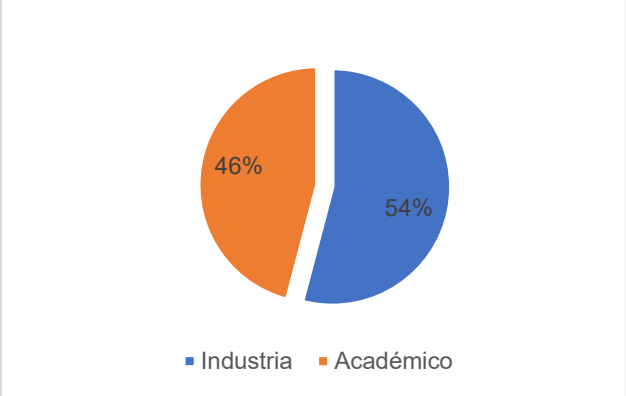

Ilustración 8 Artículos clasificados por sector

PB-2 ¿Cómo ha ido evolucionando el número de publicaciones relacionados al tema de investigación?

En el año 2016 se realizaron el mayor número de investigaciones que abordan el tema a tratar con 16 investigaciones, 2018 con 15 documentos, 2017 y 2019 con 10 investigaciones. en la Tabla 11 se detalla las investigaciones por año.

Año (Total)	**Artículos**
2015 (7)	[P10], [P22], [P25], [P39], [P49], [P57], [P58]
2016 (16)	[P1], [P3], [P4], [P7], [P8], [P11], [P12], [P18], [P20], [P26], [P28], [P41], [P43], [P46], [P52], [P61]
2017 (10)	[P9], [P15], [P17], [P19], [P24], [P32], [P33], [P40], [P44], [P51]
2018 (15)	[P5], [P13], [P16], [P21], [P23], [P27], [P29], [P31], [P34], [P36], [P38], [P45], [P47], [P53], [P59]
2019 (10)	[P6], [P30], [P37], [P42], [P48], [P50], [P54], [P55], [P56], [P60]
2020 (3)	[P2], [P14], [P35]

Tabla 11 Artículos clasificados por año

En la Ilustración 9 se muestra la evolución de investigación a lo largo de los años.

Ilustración 9 Artículos clasificados por año

PB-3 ¿Qué tipos de estudios se realizan con mayor frecuencia con relación al tema trazado?

Las conferencias son el tipo de estudios que se realiza con mayor frecuencia, con 45 investigaciones (73.77%), Jornal con 14 investigaciones (22.95%) y capítulos de libre con2 investigaciones (3.28%), en la tabla 12 se detalla el conteo.

Tipo de estudio (Total)	Artículos
Book Chapter (2)	[P20], [P51]
Conference (45)	[P2], [P3], [P4], [P6], [P8], [P9], [P10], [P11], [P12], [P13], [P14], [P15], [P16], [P17], [P18], [P23], [P24], [P25], [P26], [P28], [P30], [P31], [P32], [P34], [P35], [P37], [P38], [P39], [P40], [P42], [P43], [P44], [P45], [P47], [P49], [P50], [P53], [P54], [P55], [P56], [P57], [P58], [P59], [P60], [P61]
Journal (14)	[P1], [P5], [P7], [P19], [P21], [P22], [P27], [P29], [P33], [P36], [P41], [P46], [P48], [P52]

Tabla 12 Artículos clasificados por tipo de estudio

En la Ilustración 10 se muestra la proporción por tipo de estudios.

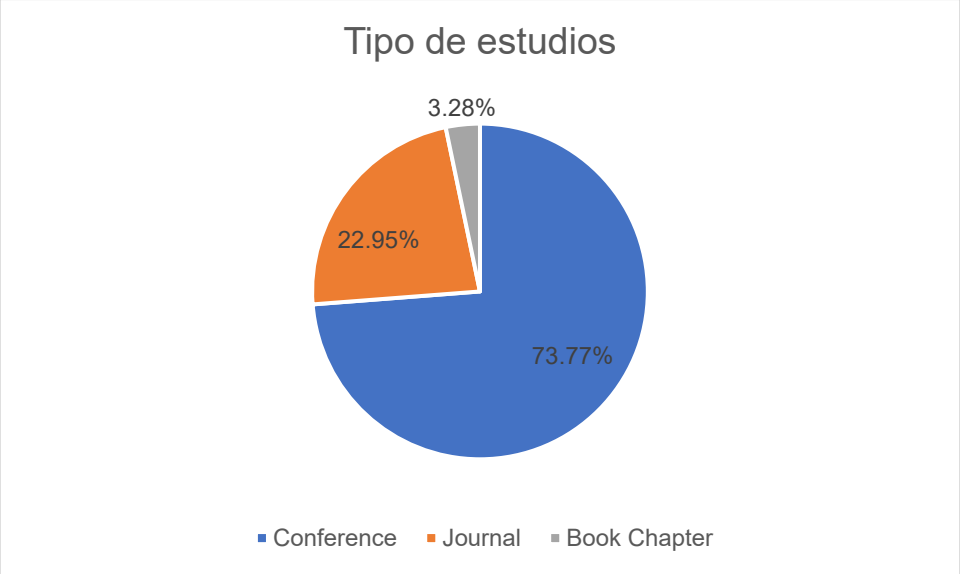

Ilustración 10 Artículos clasificados por tipo de estudio

PB-4 ¿Qué canales de publicación (revistas y/o eventos) son los principales objetivos para las investigaciones relacionados al tema de investigación?

IEEE International Conference es el principal canal de publicación, con 12 publicaciones, los canales de publicación se detallan en la tabla 13.

Revisto/evento	**Artículos**	**Con-teo**
IEEE International Conference	[P6], [P14], [P18], [P28], [P30], [P37], [P40], [P42], [P43], [P45], [P50], [P56]	12
Lecture Notes in Business Information Processing	[P15], [P55], [P57], [P60]	4
ACM International Conference Proceeding Series	[P12], [P13], [P32]	3
Journal of Systems and Software	[P1], [P19], [P48]	3
Advances in Intelligent Systems and Computing	[P25], [P35]	2

IEEE International Congress on Electronics	[P9], [P17]	2
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	[P11], [P47]	2
Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015	[P10], [P49]	2
15th International Conference on Space Operations 2018	[P31]	1
2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)	[P4]	1
2016 IEEE International Symposium on Workload Characterization (IISWC)	[P61]	1
2016 International Conference on Software Process Improvement (CIMPS)	[P3]	1
2017 Federated Conference on Computer Science and Information Systems (FedCSIS)	[P24]	1
2019 4th International Conference on System Reliability and Safety ICSRS 2019	[P54]	1
Acta Astronautica	[P21]	1
Addison-Wesley Professional	[P20]	1

CEUR Workshop Proceedings	[P2]	1
Communications in Computer and Information Science	[P58]	1
Computer Science - Research and Development	[P33]	1
Empirical Software Engineering	[P27]	1
Enterprise Information Systems	[P36]	1
European Conference on Software Architecture	[P59]	1
Future Internet	[P5]	1
IEEE Software	[P41]	1
IEEE/ACM International Workshop	[P16]	1
INCAS Bulletin	[P29]	1
Informacion Tecnologica	[P46]	1
International Journal of Applied Engineering Research	[P22]	1
International Journal of Organizational and Collective Intelligence	[P7]	1
Lecture Notes in Informatics (LNI) Proceedings - Series of the Gesellschaft fur Informatik (GI)	[P23]	1
Lecture Notes in Information Systems and Organisation	[P51]	1
Procedia CIRP	[P8]	1
Proceedings - 2018 International Conference on the Quality of	[P53]	1

Information and Communications Technology QUATIC 2018		
Proceedings - IEEE Symposium on Computers and Communications	[P26]	1
Proceedings of the 2018 Federated Conference on Computer Science and Information Systems FedCSIS 2018	[P34]	1
Proceedings of the 22nd Conference on Pattern Languages of Programs	[P39]	1
Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering	[P44]	1
Proceedings of the VII Brazilian Symposium on Software Compo-nents Architectures and Reuse	[P38]	1
Scientific Programming	[P52]	1

Tabla 13 Artículos clasificados por revistas y/o evento

6.2. Preguntas de investigación

PI-1 ¿Cuáles son las arquitecturas de *software* que pueden utilizarse en el desarrollo ágil?

El estilo de arquitectura SOA, con 27 publicaciones, el estilo N-Capa con 21 publicaciones y microservicios con 17 publicaciones; son los estilos arquitectónico más usados en la investigación. Las arquitecturas de *software* se detallan en la Tabla 14.

Arquitectura (Total)	**Artículos**
SOA (27)	[P1], [P2], [P5], [P7], [P8], [P9], [P10], [P12], [P13], [P14], [P18], [P20], [P22], [P26], [P27], [P30], [P31],

	[P34], [P36], [P38], [P40], [P42], [P43], [P48], [P53], [P58], [P59]
Capas (21)	[P7], [P11], [P17], [P18], [P20], [P21], [P22], [P24], [P26], [P28], [P29], [P31], [P35], [P36], [P38], [P49], [P51], [P52], [P54], [P58], [P59]
Microservicios (17)	[P9], [P12], [P13], [P15], [P16], [P18], [P23], [P28], [P32], [P33], [P37], [P42], [P45], [P50], [P53], [P55], [P61]
MVC (6)	[P4], [P9], [P11], [P12], [P25], [P47]
Arquitectura ad-hoc (1)	[P3]
Arquitectura de multi-tenencia (1)	[P46]
Basado en componentes (1)	[P7]
DCI (1)	[P25]
IEEE *Software* (1)	[P41]

Tabla 14 Artículos clasificados por patrón de arquitectura

En la Ilustración 11 se gráfica la cantidad de veces que las arquitecturas de *software* has sido citados en los papers seleccionados.

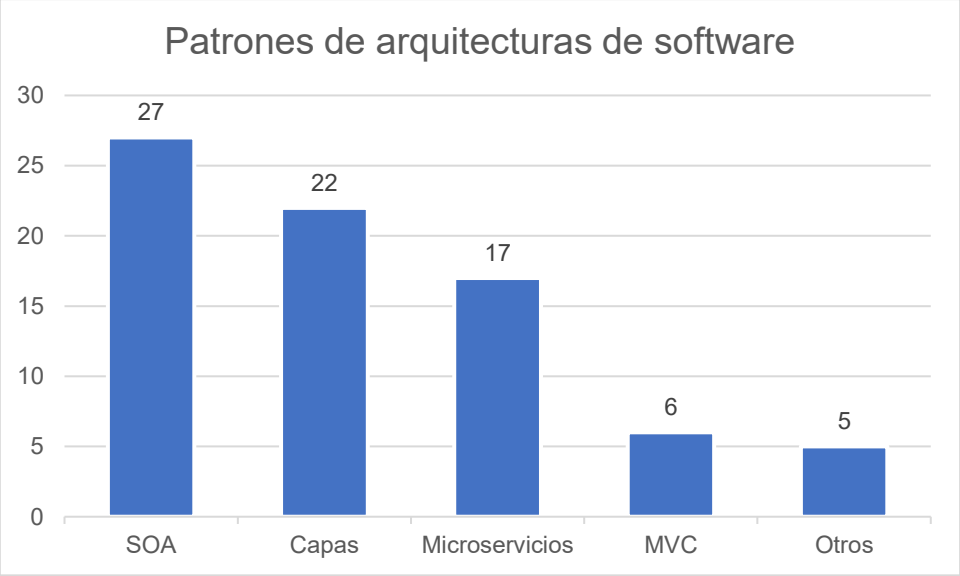

Ilustración 11 Artículos clasificados por patrón de arquitectura

PI-2 ¿Cuáles son los *frameworks*/metodologías ágiles?

Scrum es el *framework* más utilizado en la investigación con 30 publicaciones, seguido por la metodología XP con 8 publicaciones. En la Tabla 15 se detallan las publicaciones.

***Frameworks*/ Metodologías (Total)**	**Artículos**
Scrum (30)	[P1], [P2], [P3], [P4], [P5], [P7], [P12], [P14], [P15], [P16], [P17], [P24], [P27], [P29], [P30], [P31], [P33], [P34], [P35], [P41], [P42], [P43], [P44], [P47], [P48], [P49], [P50], [P51], [P56], [P60]
XP (8)	[P27], [P32], [P33], [P41], [P43], [P46], [P50], [P51]
SAFe (3)	[P6], [P37], [P60]
Agile SDLC (2)	[P22], [P55]

BDD (Behavior-Driven Development) (1)	[P45]
FDD (1)	[P1]
Kanban (1)	[P16]
LeSS -marco (1)	[P2]
MDWE (1)	[P23]
ScrumBut (1)	[P14]

Tabla 15 Artículos clasificados por *frameworks*/metodologías

En la Ilustración 12 se gráfica la cantidad de veces que han sido referenciados los *frameworks* o metodologías ágiles.

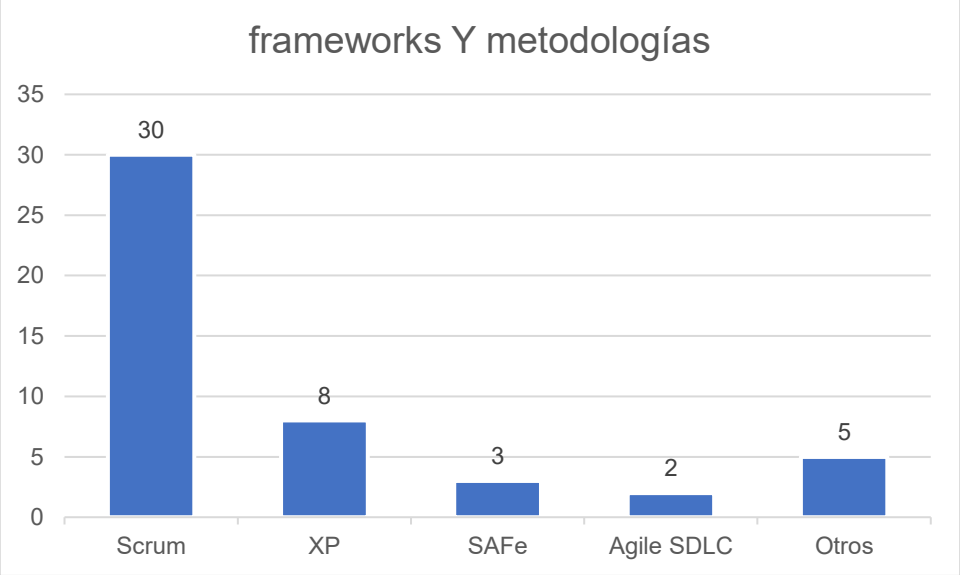

Ilustración 12 Artículos clasificados por *frameworks*/metodologías

PI-3 ¿En qué ámbitos de la industria tienden a utilizar en combinación arquitecturas de *software* y *frameworks* ágiles considerando proyectos ágiles?

Salud es el sector donde más se aplica arquitectura de *software* en ambientes ágiles con 6 investigaciones, seguido por el sector aeroespacial y automotriz con 4 investigaciones. En la Tabla 16 se detallan las publicaciones.

Industria (Total)	Artículos
Salud (6)	[12], [14], [26], [36], [44], [59]
Aeroespacial (4)	[11], [21], [29], [31]
Automotriz (4)	[2], [6], [8], [37]
Educación (3)	[17], [19], [42]
Gobierno (3)	[9], [22], [41]
Seguros (3)	[27], [30], [44]
Alquiler de autos (2)	[34], [47]
Logística (2)	[38], [46]
Telecomunicaciones (2)	[52], [59]
Automatización (1)	[44]
Consultoría (1)	[28]
Consultoría de *software* (1)	[35]
e-commerce: (1)	[13]
Energía renovable (1)	[54]
Minería (1)	[3]
Printing (1)	[44]
Retail (1)	[60]
Textil (1)	[44]

Tabla 16 Artículos clasificados por industria

En la Ilustración 13 se gráfica la cantidad de veces que los sectores indutriales que utilizan esta combinación son refrenciados.

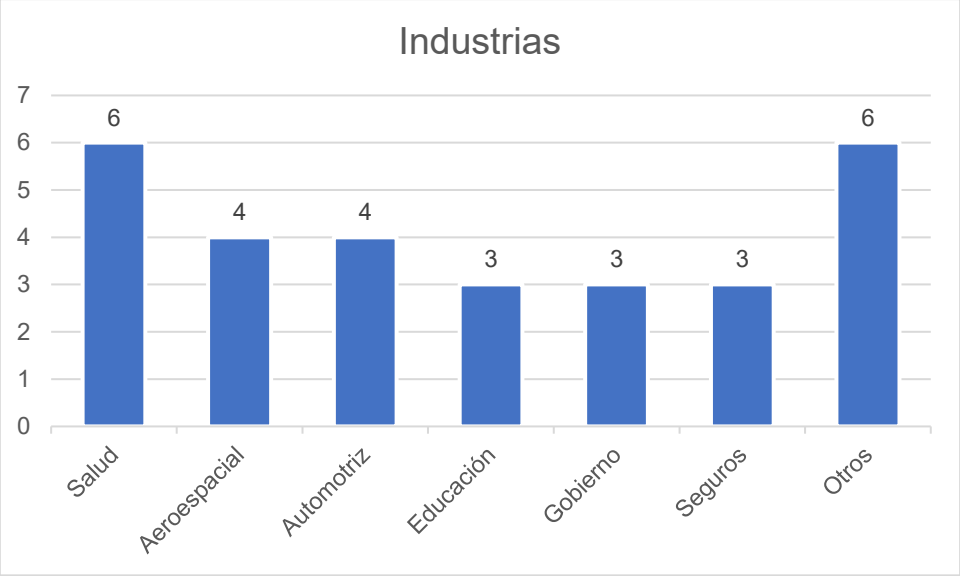

Ilustración 13 Artículos clasificados por industria

PI-4 ¿Cuáles son las ventajas y beneficios en la adopción de una arquitectura de *software* en el desarrollo ágil?

Flexibilidad es la ventaja más valora en las investigaciones con 30 publicaciones, luego mantenimiento 13 publicaciones. Las ventajas y beneficios se detallan en la Tabla 17.

Beneficios (Total)	**Artículos**
Flexibilidad (30)	[1], [2], [5], [8], [13], [15], [18], [20], [22], [23], [25], [26], [27], [28], [29], [33], [34], [35], [36], [37], [38], [41], [44], [45], [46], [47], [48], [52], [54], [56]
Mantenimiento (13)	[7], [12], [13], [17], [18], [22], [25], [27], [32], [37], [44], [47], [58]
Aseguramiento la calidad (7)	[6], [17], [20], [21], [29], [35], [36]
Reusabilidad (6)	[10], [25], [29], [44], [45], [56]
Costos (5)	[14], [20], [22], [56], [58]
Escalabilidad (5)	[26], [27], [32], [45], [46]

Entregas constantes (3)	[3], [53], [57]
Seguridad (3)	[26], [36], [46]
Facilita el desarrollo (2)	[7], [12]
gestionar esta complejidad (2)	[7], [12]
Acelera el desarrollo (1)	[61]
ciclo de desarrollo más corto (1)	[18]
Evolución (1)	[58]
Límites claros (1)	[32]
Mejorar el flujo e intercambio de información clave (1)	[9]
Modularidad (1)	[26]
Portabilidad (1)	[8]
Reduce riesgo (1)	[39]
Simplifica la trazabilidad entre los requisitos comerciales y la implementación del *software* (1)	[11]
Trabajo en equipo (1)	[42]
Trazabilidad (1)	[54]

Tabla 17 Artículos clasificados por ventajas y beneficios

En la Ilustración 14 se gráfica la cantidad de veces que son referenciadas las ventajas que utilizan esta combinación.

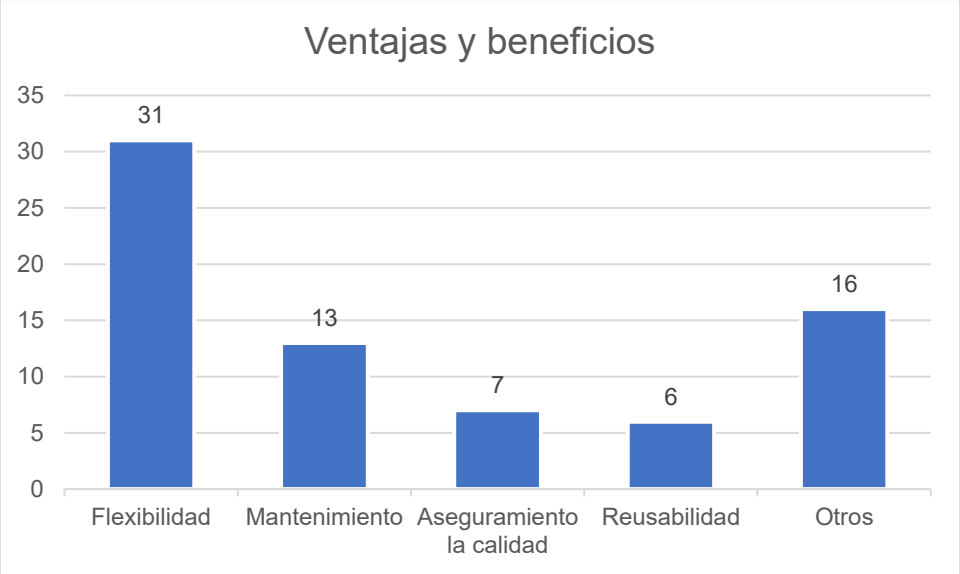

Ilustración 14 Artículos clasificados por ventajas y beneficios

PI-5 ¿Cuáles son las desventajas e inconvenientes en la adopción de una arquitectura de *software* en el desarrollo ágil?

Conflictos de enfoques es el inconveniente más referenciado en las investigaciones con 23 publicaciones. Las desventajas e inconvenientes se detallan en Ta tabla 18.

Desventajas (Total)	**Artículos**
Conflictos de enfoques (23)	[1], [2], [4], [12], [14], [15], [19], [20], [27], [29], [30], [32], [34], [37], [39], [40], [41], [44], [47], [48], [55], [57], [60]
Aumenta complejidad (3)	[32], [33], [36]
Requerimientos cambiantes (3)	[1], [11], [44]
Riesgo de cambio arquitectó-nico (2)	[57], [58]
Arquitectura incompleta (1)	[11]
Cambio de requisitos (1)	[35]
Consume más recursos (1)	[61]

Curva de aprendizaje (1)	[38]
falta de comunicación y transparencia (1)	[6]
Incertidumbre (1)	[59]
Pérdida del conocimiento técnico (1)	[40]
Requiere mucha atención (1)	[32]
Requisitos no claros (1)	[42]
Resistencia al cambio (1)	[28]
Sistemas legados y poco independientes (1)	[6]

Tabla 18 Artículos clasificados por desventajas e inconvenientes

En la Ilustración 15 se gráfica la cantidad de veces que son referenciadas las desventajas que utilizan esta combinación.

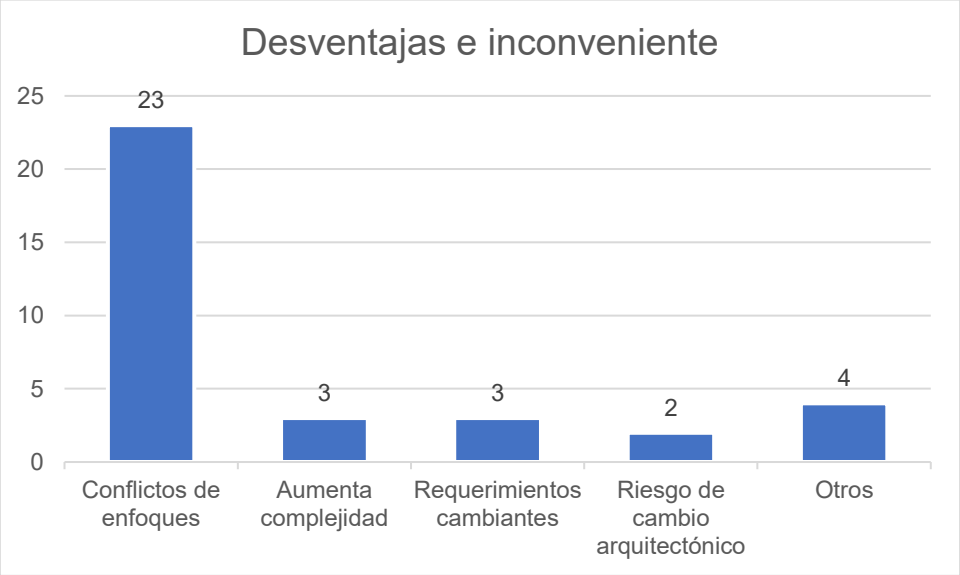

Ilustración 15 Artículos clasificados por desventajas e inconvenientes

PI-6 ¿Cuáles son los factores que influyen adopción de una arquitectura de *software* en el desarrollo ágil?

Comunicación es el factor más influyente en las investigaciones con 15 publicaciones, luego *Sprint* de arquitectura con 13 publicaciones y colaboración con 10 artículos. Los factores se detallan en la Tabla 19.

Factores (Total)	**Artículos**
Comunicación (15)	[1], [2], [3], [23], [27], [30], [37], [39], [40], [42], [44], [48], [57], [59], [60]
Sprint de arquitectura (13)	[2], [4], [17], [20], [30], [34], [37], [39], [44], [47], [48], [53], [55]
Colaboración (10)	[6], [18], [22], [23], [27], [31], [42], [45], [57], [60]
Herramientas adecuadas (4)	[14], [20], [28], [29]
Alinear el negocio y TI (2)	[11], [12]
Definir roles (2)	[20], [57]
Documentación adecuada (2)	[1], [58]
Monitoreo (2)	[1], [13]
Administración deuda técnica (1)	[39]
Administrar el cambio (1)	[52]
Combinar con metodologías tradicionales (1)	[27]
Entrenamiento (1)	[19]
Fases simultáneas (1)	[52]
Mathematical model (1)	[35]
Procesos definidos (1)	[59]

Repositorio de conocimientos (1)	[52]
Responsabilidad (1)	[6]

Tabla 19 Artículos clasificados por factores de adopción

En la Ilustración 16 se gráfica la cantidad de veces que son referenciadas los factores de adopción que utilizan esta combinación.

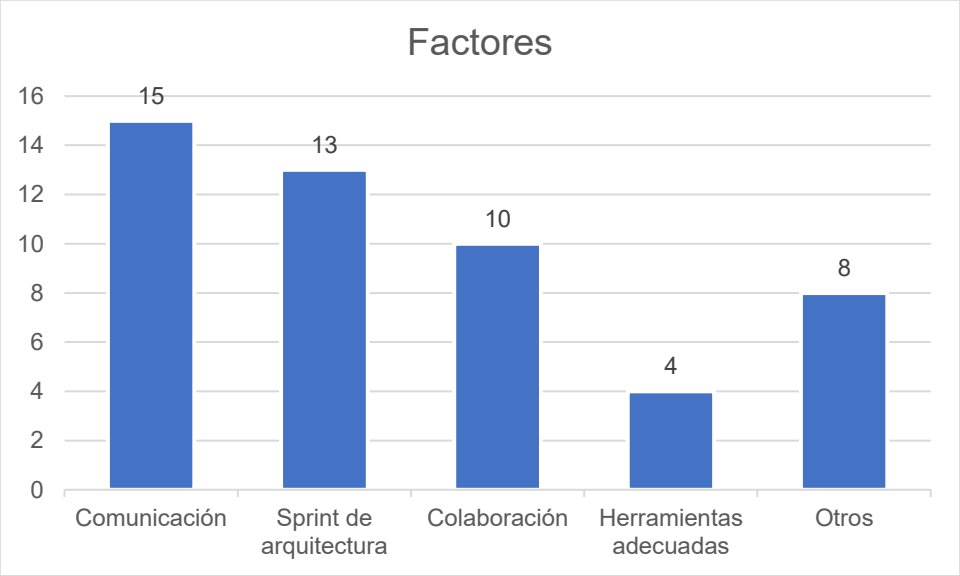

Ilustración 16 Artículos clasificados por factores de adopción

PI-7 ¿Cuáles son las relaciones entre arquitecturas de *software* y *frameworks*/metodologías ágiles?

La arquitectura de *software* SOA y el *framework* ágil Scrum es la combinación más usada en las investigaciones con 13 publicaciones, luego la arquitectura de *software* Capas y el *framework* ágil Scrum con 8 publicaciones y arquitectura de *software* Microservicios y el *framework* Scrum con 6 publicaciones. Todas las combinaciones se detallan en la Tabla 20.

Arquitectura-Metodología/Frame-work (Total)	Artículos
SOA - Scrum (13)	[P48], [P43], [P42], [P34], [P27], [P31], [P30], [P14], [P12], [P7], [P5], [P2], [P1]

Capas - Scrum (8)	[P51], [P49], [P35], [P7], [P31], [P29], [P24], [P17]
Microservicios - Scrum (6)	[P12], [P16], [P15], [P33], [P42], [P50]
Microservicios - XP (3)	[P50], [P32], [P33]
MVC - Scrum (3)	[P4], [P12], [P47]
SOA - XP (2)	[P27], [P43]
Arquitectura ad-hoc - Scrum (1)	[P3]
Arquitectura de multitenant - XP (1)	[P46]
Basado en componentes - SCRUM (1)	[P7]
Capas - Agile SDLC (1)	[P22]
Capas - XP (1)	[P51]
IEEE *Software* - Scrum (1)	[P41]
IEEE *Software* - XP (1)	[P41]
Microservicios - Agile SDLC (1)	[P55]
Microservicios - BDD (1)	[P45]
Microservicios - Kanban (1)	[P16]
Microservicios - MDWE (1)	[P23]
Microservicios - SAFe (1)	[P37]
SOA - *Agile* SDLC (1)	[P22]
SOA - FDD (1)	[P1]
SOA - LeSS -marco (1)	[P2]
SOA - ScrumBut (1)	[P14]

Tabla 20 Artículos clasificados por la combinación de arquitectura de *software* y frameworks/metodología ágil

En la Ilustración 17 se gráfica la proporción de las combinaciones de arquitecturas de *software* y *frameworks*/metodologías ágiles.

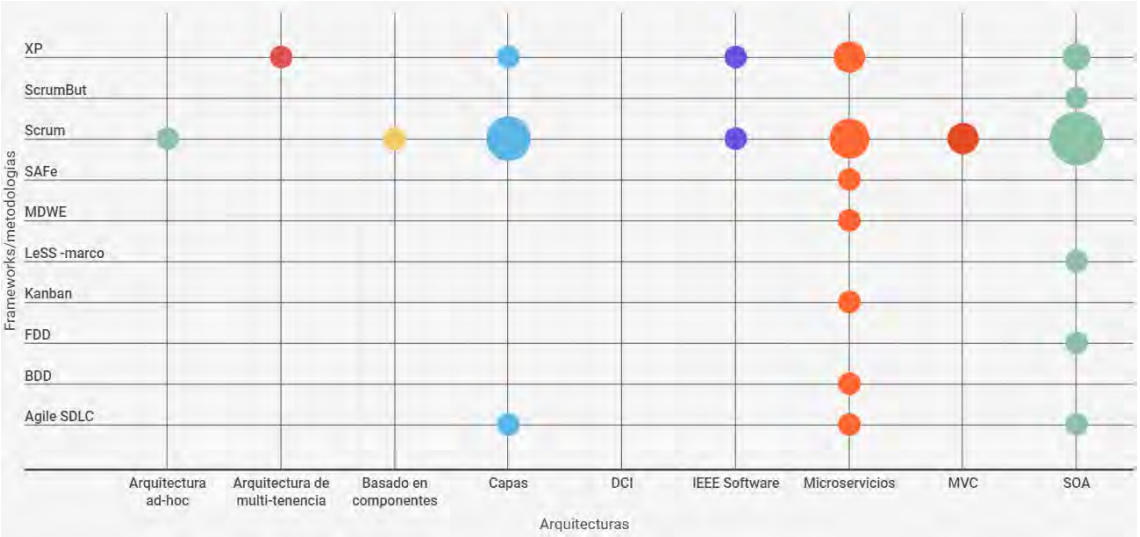

Ilustración 17 Artículos clasificados por la combinación de arquitectura de *software* y *frameworks*/metodología ágil

PI-8 ¿Cuáles son las relaciones entre las arquitecturas de *software* y las industrias?

La arquitectura de *software* SOA en el sector salud es la combinación más usada en las investigaciones con 5 publicaciones, luego la arquitectura en capas y el *framework* Scrum con 8 publicaciones. Todas las combinaciones se detallan en la Tabla 21.

Arquitectura-Sector (Total)	**Artículos**
SOA - Salud (5)	[P59], [P12], [P14], [P26], [P36]
Capas - Aeroespacial (4)	[P21], [P11], [P31], [P29]
Capas - Salud (3)	[P59], [P26], [P36]
Capas - Telecomunicaciones (2)	[P52], [P59]
SOA - Automotriz (2)	[P2], [P8]
SOA - Gobierno (2)	[P22], [P9]

SOA - Seguros (2)	[P30], [P27]
Arquitectura ad-hoc - Minería (1)	[P3]
Arquitectura de multi-tenencia - Logística (1)	[P46]
Capas - Consultoría (1)	[P28]
Capas - Consultoría de *software* (1)	[P35]
Capas - Educación (1)	[P17]
Capas - Energia renovable (1)	[P54]
Capas - Gobierno (1)	[P22]
Capas - Logística (1)	[P38]
IEEE *Software* - Gobierno (1)	[P41]
Microservicios - Automotriz (1)	[P37]
Microservicios - Consultoría (1)	[P28]
Microservicios - e-commerce: (1)	[P13]
Microservicios - Educación (1)	[P42]
Microservicios - Gobierno (1)	[P9]
Microservicios - Salud (1)	[P12]
MVC - Aeroespacial (1)	[P11]
MVC - Alquiler de autos (1)	[P47]
MVC - Gobierno (1)	[P9]
MVC - Salud (1)	[P12]
SOA - Aeroespacial (1)	[P31]
SOA - Alquiler de autos (1)	[P34]
SOA - e-commerce: (1)	[P13]

SOA - Educación (1)	[P42]
SOA - Logística (1)	[P38]
SOA - Telecomunicaciones (1)	[P59]

Tabla 21 Artículos clasificados por la combinación de arquitectura de *software* e industrias

En la Ilustración 18 se gráfica la proporción entre las arquitecturas de *software* y las industrias.

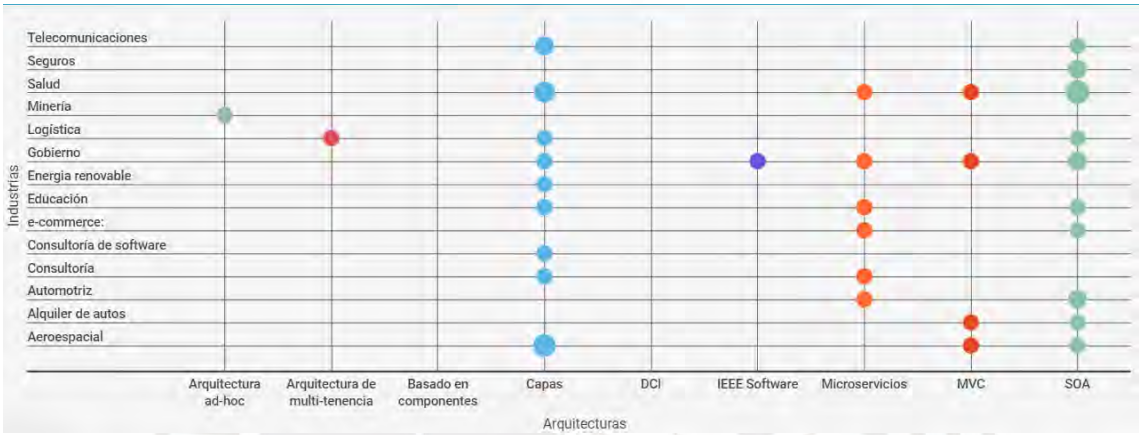

Ilustración 18 Artículos clasificados por la combinación de arquitectura de *software* e industrias

PI-9 ¿Cómo han evolucionado las principales arquitecturas de *software* en el desarrollo ágil a través de los años?

SOA, capas y microservicios son las arquitecturas más usadas en las publicaciones, en el año 2016 es cuando se realizan más publicaciones. En la Tabla 22 se detallan la relación de los años y las arquitecturas identificadas.

Año	SOA	Capas	Microservicios
2015	3	3	0
2016	8	7	4
2017	2	3	4
2018	9	6	5

2019	3	1	4

Tabla 22 Evolución de las principales arquitecturas de *software* en el desarrollo ágil a través de los años

En la Ilustración 19 se muestra la evolución de las arquitecturas en el desarrollo ágil a través de los años.

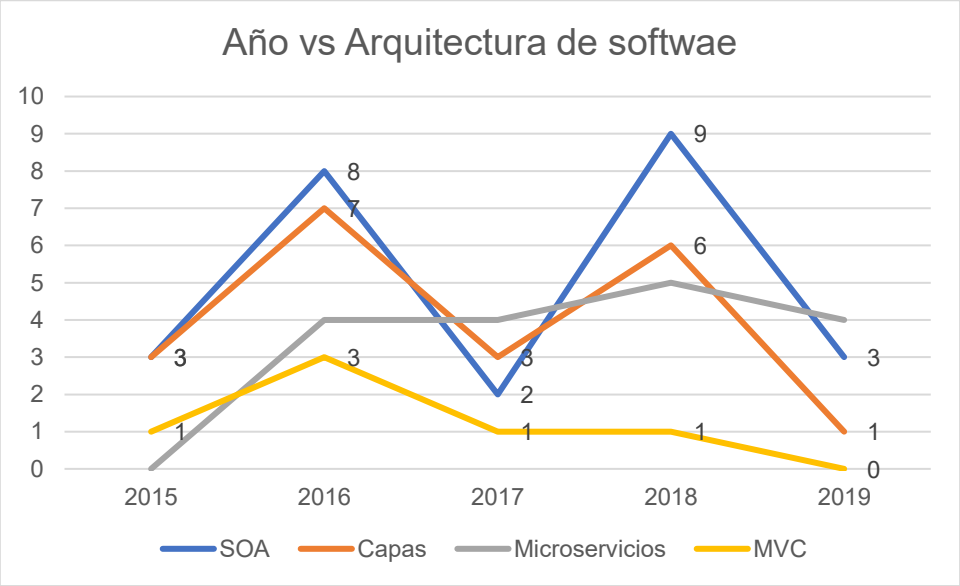

Ilustración 19 Evolución de las arquitecturas de *software* en el desarrollo ágil a través de los años

PI-10 ¿Cómo ha evolucionado los principales *frameworks*/metodologías usados a través de los años?

Scrum, XP, SAFe y *agile* SDLC son los *frameworks*/metodologías más usadas en las publicaciones, en el año 2016 y 2018 cuando se realizaron más publicaciones, siendo Scrum el *framework* más utilizado en todos los años. En la tabla 23 se detallan años y los *frameworks*/metodologías usados.

Año	Scrum	XP	SAFe	Agile SDLC
2015	1	0	0	1
2016	7	3	0	0

2017	6	3	0	0
2018	7	1	0	0
2019	6	1	3	1

Tabla 23 Evolución de los principales *frameworks*/metodologías usados a través de los años

En la Ilustración 20 se muestra la evolución de los principales *frameworks*/metodologías usados en la investigación a través de los años.

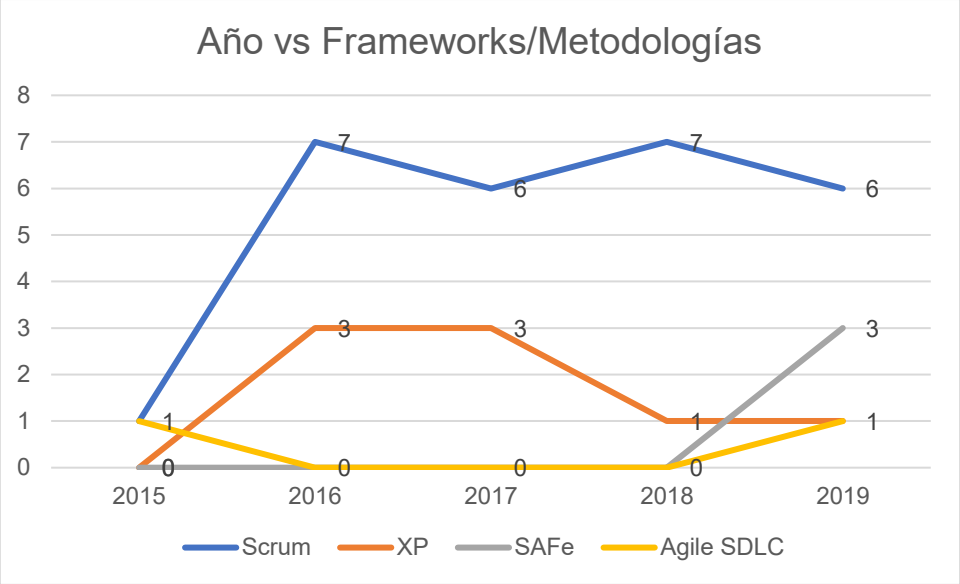

Ilustración 20 Evolución de los principales *frameworks*/metodologías usados a través de los años

7. AMENAZAS A LA VALIDEZ

Wohlin et al. [39] proponen cuatro amenazas a la validez en experimentos de ingeniería de *software* ,que son analizadas a continuación:

7.1. Validez interna

La validez interna es la medida en que el investigador puede afirmar que ninguna otra variable excepto la que está estudiando influyó el resultado. Para mitigar esta amena se siguió un protocolo definido con criterios de inclusión, exclusión y criterios de calidad.

7.2. Validez de las conclusiones

La validez de conclusión se refiere a que la conclusión a la que se está alcanzando desde el conjunto de datos obtenido es realmente correcta y justificada. Una posible amenaza es que no se hayan elegido algunos estudios relevantes durante el proceso de investigación, mitigamos esta amenaza extrayendo artículos de 4 bases de datos distintas para tener un universo de investigación más amplio.

7.3. Validez del constructo

La validez del constructo verifica que la construcción de la investigación esté relacionada con los objetivos planteados y que las fuentes seleccionadas sean relevantes. Una posible amenaza a la validez del constructo puede ser la experiencia del investigador en una determinada arquitectura, framework ágil o metodología que influya en el resultado. Para atenuar esta amenaza respecto a la búsqueda de información o la obtención en la cadena de búsqueda, se tuvo la colaboración oportuna del asesor .

7.4. Validez externa

La validez externa se trata de la generalización de los resultados. Para este estudio se siguió el protocolo definido, lo que permite generalizar los resultados dentro del dominio de la investigación.

8. CONCLUSIONES Y TRABAJO FUTURO

En esta sección se analizarán los resultados del mapeo y se muestran opciones de trabajo futuro para continuidad del proyecto.

8.1. Conclusiones

Se presentaron los resultados del mapeo sistemático respecto al uso de arquitecturas de *software* en el desarrollo ágil, identificar patrones arquitectónicos, industrias, factores, beneficios y desafíos con respecto a la combinación de patrones de arquitectura de *software* y *frameworks*/metodologías ágiles.

De los 61 estudios seleccionados para este estudio, se detectó que el 54% son aplicaciones en la industria, siendo el sector salud, aeroespacial y automotriz los sectores donde más se investiga esta combinación. En el año 2016 se realizaron el mayor número de estudios que abordan el tema con 16 investigaciones; siendo las conferencias el tipo de estudios que se realiza con mayor frecuencia con el 73.77% de las publicaciones. De éstas, el principal canal de publicación es *IEEE International Conference*.

Asimismo, el estilo de arquitectura SOA, el *framework* ágil Scrum y el uso combinado de ambos son los más referidos en las publicaciones; además usar SOA en el sector salud es la combinación más referenciada.

Finalmente, se encontró que la flexibilidad es el mayor beneficio que aporta usar arquitecturas de *software* en el desarrollo ágil; siendo el principal inconveniente el conflicto entre las perspectivas de agilidad y los procesos de arquitectura. Para superar esto, se reconoce a la comunicación como el factor más importante a considerar.

8.2. Trabajo futuro

Como parte del trabajo futuro, se puede realizar una comparativa de las arquitecturas de *software* encontradas en el mapeo para estudiar sus discrepancias y semejanzas en entornos de desarrollo ágil.

Otra sugerencia de trabajo futuro es comparar los *frameworks* y metodologías agiles hallados para identificar ventajas y debilidades con el uso combinado de arquitecturas de *software*.

Finalmente, se sugiere profundizar en las combinaciones de arquitecturas y *frameworks*/metodologías ágiles con a sus ventajas, desventajas y factores de adopción utilizados para identificar patrones de comportamiento y buenas prácticas.

Se busca extender el alcance de la investigación para los interesados en el tema.

9. REFERENCIAS

[1] CollabNet VersionOne, “The 13th Annual State of Agile Report - 2019,” *State Agil.*, vol. 13, p. 16, 2019, [Online]. Available: https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508.

[2] K. Power and R. Wirfs-Brock, “Understanding architecture decisions in context: An industry case study of architects’ decision-making context,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11048 LNCS, no. October, pp. 284–299, 2018, doi: 10.1007/978-3-030-00761-4_19.

[3] S. Angelov and P. De Beer, “The Journal of Systems and Software Designing and applying an approach to software architecting in agile projects in education,” *J. Syst. Softw.*, vol. 127, pp. 78–90, 2017, doi: 10.1016/j.jss.2017.01.029.

[4] C. Ahuja, P. Kaur, and H. Singh, *Software architecture evaluation in Agile environment*, vol. 731. Springer Singapore, 2019.

[5] M. Waterman, J. Noble, and G. Allan, “How much up-front? A grounded theory of agile architecture,” *Proc. - Int. Conf. Softw. Eng.*, vol. 1, pp. 347–357, 2015, doi: 10.1109/ICSE.2015.54.

[6] R. L. Nord and J. E. Tomayko, “Software architecture-centric methods and agile development,” *IEEE Softw.*, vol. 23, no. 2, pp. 47–53, 2006, doi: 10.1109/MS.2006.54.

[7] M. O. Ahmad, “Preliminary Citation and Topic Analysis of International Conference on Agile Software Development Papers (2002-2018),” vol. 18, pp. 803–812, 2019, doi: 10.15439/2019F114.

[8] J. Holvitie *et al.*, “Technical debt and agile software development practices and processes : An industry practitioner survey,” *Inf. Softw. Technol.*, vol. 96, no. December 2017, pp. 141–160, 2018, doi: 10.1016/j.infsof.2017.11.015.

[9] S. Freudenberg and H. Sharp, “The Top 10 Burning Research Questions from Practitioners,” no. November 2010, 2014, doi:

10.1109/MS.2010.129.

[10] I. Hadar and S. Sherman, “Agile vs . Plan-Driven Perceptions of Software Architecture,” pp. 50–55, 2012.

[11] P. Abrahamsson, M. A. Babar, and P. Kruchten, “Agility and architecture: Can they coexist?,” *IEEE Softw.*, vol. 27, no. 2, pp. 16–22, 2010, doi: 10.1109/MS.2010.36.

[12] V. C. Caires, J. Holvitie, and R. O. Spínola, “Investigating the Effects of Agile Practices and Processes on Technical Debt - The Viewpoint of the Brazilian Software Industry,” no. April 2019, 2018, doi: 10.18293/SEKE2018-131.

[13] K. Power, R. Wirfs-brock, I. Researcher, and L. Agile, “Understanding Architecture Decisions in Context : An industry case study of architects ’ decision-making context Understanding Architecture Decisions in Context An industry case study of architects ’ decision-making context Architecture decisions can sign,” no. October, 2018, doi: 10.1007/978-3-030-00761-4.

[14] P. Kruchten, “Software Architecture and Agile Software Development — A Clash of Two Cultures ?,” pp. 497–498, 2010.

[15] J. Yli-huumo, A. Maglyas, and K. Smolander, “How do software development teams manage technical debt ? – An empirical study,” *J. Syst. Softw.*, vol. 0, pp. 1–24, 2016, doi: 10.1016/j.jss.2016.05.018.

[16] M. Waterman, “Agility , Risk , and Uncertainty , Part 1 : Designing an Agile Architecture,” pp. 99–101, 2018.

[17] P. Clements, D. Garlan, R. Little, R. Nord, and J. Stafford, “Documenting Software Architectures : Views and Beyond Documenting Software Architectures : Views and Beyond,” no. June, pp. 2002–2004, 2002.

[18] F. Mccabe, “Service Oriented Architecture Reference Architecture,” *Management*, no. April, pp. 1–104, 2008.

[19] W. Cunningham, “Enterprise Solution Patterns Using Microsoft . NET Enterprise Solution Patterns Using Microsoft . NET.”

[20] M. Kimberly *et al.*, *Arquitectura Macro de N-Capas*, no. 1. 2014.

[21] L. CÃ¡rdenas Escalante, “El patrÃ³n de arquitectura n-capas con orientaciÃ³n al dominio como soluciÃ³n en el diseÃ±o de aplicaciones empresariales.,” *Rev. Tecnol. Desarro.*, vol. 11, no. 1, pp. 59–66, 2016, doi: 10.18050/td.v11i1.679.

[22] H. Eiiti, “MVC dance: Connecting software development and corporeality from agile process and pattern language perspectives,” *Proc. - Second Int. Conf. Creat. Connect. Collab. Through Comput.*, p. 174, 2004, doi: 10.1109/C5.2004.1314388.

[23] “¿Cuál es la diferencia entre MVC y MVVM?” https://qastack.mx/programming/667781/what-is-the-difference-between-mvc-and-mvvm.

[24] Steve Smith, “Información general de ASP.NET Core MVC | Microsoft Docs,” *הנוטע עלון*, vol. 66, pp. 37–39, 2020, [Online]. Available: https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-5.0.

[25] Derek, “Microservices architecture style,” 2012. https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices.

[26] A. Navasa and J. M. Murillo, “Aspect Oriented Software Architecture : a Structural Perspective Aspect Oriented Software Architecture : a Structural Perspective 1,” no. January 2013, 2002.

[27] F. Chen, Q. Wang, H. Mei, and F. Yang, “An architecture-based approach for component-oriented development,” *Proc. - IEEE Comput. Soc. Int. Comput. Softw. Appl. Conf.*, no. February 2002, pp. 450–455, 2002, doi: 10.1109/cmpsac.2002.1045042.

[28] J. O. Coplien, “1 . The DCI Paradigm : Taking Object Orientation Into the Architecture World.”

[29] I. Scaled Agile, “What is safe?,” *Nutrition Bulletin*, 2020. https://www.scaledagile.com/enterprise-solutions/what-is-safe/.

[30] A. J. Nathan and A. Scobell, “What is SCRUM?,” *Foreign Affairs*, 2012. .

[31] K. Beck, *Extreme Programming Explained , Second Edition*. 2004.

[32] M. Poppendieck and T. Poppendieck, “Implementing Lean Software Development: From Concept to Cash,” *Addison-Wesley Signat. Ser.*, p. 304, 2006.

[33] C. Yang, P. Liang, and P. Avgeriou, “A systematic mapping study on the combination of software architecture and agile development,” *J. Syst. Softw.*, vol. 111, pp. 157–184, 2016, doi: 10.1016/j.jss.2015.09.028.

[34] Z. Dragičević and S. Bošnjak, “Agile architecture in the digital era: Trends and practices,” *Strateg. Manag.*, vol. 24, no. 2, pp. 12–33, 2019, doi: 10.5937/straman1902011d.

[35] G. Borrego, A. L. Morán, R. R. P. Cinco, O. M. Rodríguez-Elias, and E. García-Canseco, “Review of approaches to manage architectural knowledge in Agile Global Software Development,” *IET Softw.*, vol. 11, no. 3, pp. 77–88, 2017, doi: 10.1049/iet-sen.2016.0197.

[36] V. Menon, R. Sinha, and S. MacDonell, “Architectural challenges in migrating plan-driven projects to agile,” *ENASE 2015 - Proc. 10th Int. Conf. Eval. Nov. Approaches to Softw. Eng.*, pp. 223–228, 2015, doi: 10.5220/0005383502230228.

[37] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Inf. Softw. Technol.*, vol. 64, pp. 1–18, 2015, doi: 10.1016/j.infsof.2015.03.007.

[38] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in SE,” *Guidel. Perform. Syst. Lit. Rev. SE*, pp. 1–44, 2007, [Online]. Available: https://userpages.uni-koblenz.de/%7B~%7Dlaemmel/esecourse/slides/slr.pdf.

[39] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, “Experimentation in software engineering,” *Exp. Softw. Eng.*, vol. 9783642290, pp. 1–236, 2012, doi: 10.1007/978-3-642-29044-2.

ET LUX IN TENEBRIS LUCET
MCMXVII