PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Revisión Sistemática de la Literatura: Prácticas de desarrollo de software que mejoran el tiempo de entrega de Aplicaciones con Arquitectura Monolítica

Trabajo de investigación para obtener el grado académico de Maestro en Informática con mención en Ingeniería de Software que presenta:

Christian Genaro Rojas Ramos

Asesora:

Natalí Flores Lafosse

Lima, 2025

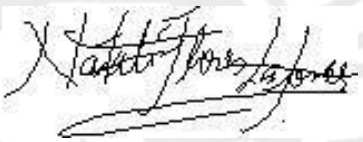
Informe de Similitud

Yo, Natalí Flores Lafosse, docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesora de la tesis titulada Revisión Sistemática de la Literatura: Prácticas de desarrollo de software que mejoran el tiempo de entrega de Aplicaciones con Arquitectura Monolítica, del autor Christian Genaro Rojas Ramos, dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 10% Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 30/10/2025.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Investigación, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha:

Lima, 30 de octubre del 2025.

<table>
<tr><td colspan="2">Apellidos y nombres del asesor / de la asesora:
Flores Lafosse, Natalí</td></tr>
<tr><td>DNI:
42324700</td><td rowspan="2">Firma
  </td></tr>
<tr><td>ORCID:
0000-0002-2611-9371</td></tr>
</table>

DEDICATORIA

Dedicado a Camila e Isabella, mis hijas, quienes siempre me brindan fortaleza con sus sonrisas y sus logros; a mi esposa Luisa, por apoyarme incondicionalmente en mis proyectos profesionales y académicos; y a mi madre, por todo el esfuerzo que hizo para educarme y formarme con principios y valores comprometidos con el bien común.

AGRADECIMIENTOS

A toda mi familia, por todo el apoyo brindado durante este tiempo.

A mi asesora, Natalí Flores, por su tiempo, exigencia y experiencia, fundamentales para guiarme en este trabajo académico.

A los profesores de la maestría, por compartir sus conocimientos y demostrar un alto compromiso con la calidad educativa, la cual me ayudó a crecer como profesional.

RESUMEN

(ANTECEDENTES) En la actualidad, las arquitecturas monolíticas están siendo progresivamente reemplazadas por enfoques más flexibles, escalables e independientes, como la arquitectura de microservicios. Este cambio se debe principalmente a la creciente complejidad que adquieren los sistemas monolíticos con el tiempo, así como al alto grado de acoplamiento entre sus componentes, lo que impacta negativamente en los tiempos de entrega del software. No obstante, este tipo de arquitectura sigue vigente en diversas industrias. Un ejemplo notable es el caso de Amazon Prime, que migró de una arquitectura basada en microservicios a una monolítica, motivado principalmente por la reducción de costos, sin comprometer la eficiencia en la implementación de cambios.
(OBJETIVOS) El objetivo de este estudio es analizar cómo diversas prácticas aplicadas al desarrollo de software con arquitectura monolítica permiten mitigar o resolver los problemas inherentes a este tipo de diseño, y cómo estas prácticas contribuyen a mejorar los tiempos de entrega.
(METODOS) Para alcanzar este objetivo, se llevará a cabo una **revisión sistemática de la literatura**, utilizando bases de datos académicas reconocidas y relevantes en el ámbito del desarrollo de software.
(RESULTADOS) Se filtraron 765 artículos a través del protocolo definido para la revisión sistemática, quedando 80 estudios primarios. Estos estudios permitieron identificar un conjunto de prácticas que abordan diversas dificultades asociadas a las arquitecturas monolíticas, contribuyendo a la reducción de los tiempos de entrega. Las prácticas encontradas se relacionan con el diseño arquitectónico, el desarrollo de software, el uso de herramientas tecnológicas, el mantenimiento y la cultura organizacional. Asimismo, se identificaron consideraciones clave para una implementación efectiva dentro de las organizaciones.
(CONCLUSIONES) Las arquitecturas monolíticas siguen siendo relevantes en la industria del software. Por ello, resulta fundamental implementar mejoras dentro del ciclo de vida del software que ayude a los productos a ser más competitivos y que optimicen los tiempos de entrega hacia el usuario final.

Palabras clave
ingeniería de software, Arquitectura de software, Arquitectura monolítica, Microservicios, Ciclo de vida del software, Tiempo de entrega

ABSTRACT

(BACKGROUND) Monolithic architectures are increasingly being replaced by more flexible, scalable, and independent approaches such as microservices architecture. This shift is primarily driven by the growing complexity of monolithic systems over time and the high degree of coupling between their components, which negatively impacts software delivery times. However, monolithic architectures remain relevant in various industries. A notable example is Amazon Prime, which transitioned from a microservices-based architecture back to a monolithic one, primarily to reduce costs without compromising the efficiency of implementing changes.
(OBJECTIVE) This study aims to analyze how various practices applied to the development of software with a monolithic architecture can mitigate or resolve the inherent challenges of this design and how these practices contribute to improving delivery times.
(METHODS) To achieve this objective, a systematic literature review was conducted using well-established academic databases relevant to the field of software development.
(RESULTS) A total of 765 articles were initially identified and filtered through the defined systematic review protocol, resulting in 80 primary studies. These studies revealed a range of practices that help address the challenges associated with monolithic architectures, thereby contributing to reduced delivery times. The identified practices relate to architectural design, software development, technological tools, maintenance, and organizational culture. Additionally, key considerations for successful implementation within organizations were identified.
(CONCLUSIONS) Monolithic architectures continue to play a significant role in the software industry. Therefore, it is essential to implement improvements throughout the software lifecycle to enhance product competitiveness and optimize delivery times to end users.

Keywords
Software engineering, Software architecture, Monolithic architecture, Microservices, Software lifecycle, Delivery time

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

En los últimos años, las arquitecturas monolíticas han sido progresivamente reemplazadas por soluciones basadas en microservicios. Diversos estudios identificados en la revisión primaria destacan esta transición como una respuesta a las limitaciones inherentes de las arquitecturas monolíticas. La investigación titulada *"Paradigm Shift From Monolithic to Microservices"* señala, entre los principales desafíos, el alto nivel de acoplamiento entre componentes, lo que obliga a realizar modificaciones sobre toda la solución incluso cuando los cambios afectan solo a una parte específica. Además, el proceso de despliegue resulta complejo, ya que requiere actualizar todos los componentes y realizar pruebas integrales para garantizar que las modificaciones no afecten funcionalidades no relacionadas. El uso compartido de recursos entre componentes también impacta negativamente en el rendimiento, y la complejidad del código dificulta su análisis, lo que incrementa el riesgo de errores y retrasa la entrega de nuevas funcionalidades al mercado (Saxena & Bhowmik, 2023).

Por su parte, el artículo *"Migrating to a Microservice Architecture: Benefits and Challenges"*, de Salii, Ajdari y Zenuni, identifica otras problemáticas asociadas a las arquitecturas monolíticas, como los altos costos de mantenimiento y los prolongados tiempos requeridos para aplicar correcciones. Además, los autores argumentan que este tipo de arquitectura no favorece la adopción de prácticas ágiles, lo que limita la capacidad de las organizaciones para mejorar sus procesos de entrega continua. En contraste, los microservicios promueven una mayor modularidad, lo que facilita el mantenimiento y permite una implementación más eficiente de metodologías ágiles orientadas a la entrega temprana de valor (2023).

Sin embargo, no todos los casos siguen esta tendencia. Un ejemplo notable es el de Amazon Prime Video, que migró parte de su sistema —específicamente el servicio de monitoreo de audio y video— de una arquitectura de microservicios a una monolítica. Esta decisión, poco común en la actualidad, se justificó por la necesidad de reducir costos en servicios en la nube, logrando ahorros de hasta un 90%, y por la mejora significativa en la escalabilidad del sistema. Esta reestructuración permitió manejar un mayor volumen de transacciones simultáneas, mejorando la experiencia del usuario en términos de calidad de audio y video. A pesar de la naturaleza monolítica de la nueva solución, el equipo no reportó dificultades en cuanto a la capacidad de realizar cambios futuros ni en los tiempos de entrega (Kolny, 2023).

Estos antecedentes evidencian que, si bien las arquitecturas monolíticas presentan limitaciones ampliamente documentadas, aún existen escenarios en los que pueden ser viables y eficientes. En este contexto, la presente investigación tiene como objetivo identificar prácticas que contribuyan a mejorar los tiempos de entrega en aplicaciones con arquitectura monolítica, permitiendo así que las organizaciones que aún operan bajo este modelo puedan optimizar sus procesos sin necesidad de migrar hacia arquitecturas más complejas.

1.1. Objetivo del estudio

El objetivo de este estudio es analizar cómo diversas prácticas aplicadas al desarrollo de software con arquitectura monolítica permiten mitigar o resolver los problemas inherentes a este tipo de diseño, y cómo estas prácticas contribuyen a mejorar los tiempos de entrega.

Para poder lograr este objetivo necesitamos completar los siguientes objetivos específicos:

- OE1: Identificar las principales problemáticas que afectan el tiempo de entrega en aplicaciones desarrolladas con arquitectura monolítica.
- OE2: Analizar las prácticas que permiten mitigar o resolver dichas problemáticas, contribuyendo a la mejora de los tiempos de entrega.
- OE3: Identificar los resultados cuantitativos obtenidos de la aplicación de las prácticas que impacten sobre los tiempos de entrega.
- OE4: Determinar las recomendaciones clave para asegurar una adopción e implementación exitosa de las prácticas identificadas.

1.2. Estructura de la tesis

La presente investigación está organizada de la siguiente forma:

- Capítulo 2: Presenta el marco conceptual de los principales conceptos que se trataran en el presente estudio.
- Capítulo 3: Presenta el estado del arte donde se revisa las revisiones sistemáticas que podrían tener alguna similitud este trabajo académico.
- Capítulo 4: Desarrolla el protocolo de la revisión sistemática de la literatura que se utilizará para filtrar los artículos encontrados en las diferentes bases de datos académicas.
- Capítulo 5: Resultados tras la ejecución del protocolo definido para la revisión sistemática de la literatura.
- Capítulo 6: Responde a las preguntas bibliográficas y de investigación definidas en el protocolo de revisión.
- Capítulo 7: Presenta las conclusiones a las que se llegaron por las respuestas obtenidas a las preguntas de investigación y las recomendaciones de futuros trabajos.

CAPÍTULO 2. MARCO TEÓRICO

2.1. Ciclo de vida del Software

Todas las soluciones desarrolladas con una arquitectura monolítica están inmersas en el **ciclo de vida del software**, el cual se define como el período que abarca desde la concepción del producto hasta el momento en que deja de estar disponible o en uso (Manju Khari, 2016). A continuación, se describen las etapas que conforman este ciclo.

- Esto inicia con la **planificación**, el cual, establece la forma en cómo se va a implementar una solución de software. En ella se definirá el objetivo y el alcance. Además, se evaluará la viabilidad técnica y los recursos necesarios para cumplir con las expectativas de los interesados (Raghi, Sudha, Sreeram, & Steve Joshua, 2024).
- En la etapa de **definición de requerimientos** se definen las necesidades del cliente y se realizan especificaciones más detalladas del sistema (Storm, et al., 2021).
- Conocida las necesidades se inicia con la etapa de **diseño**, la cual hace referencia a la elaboración o modificación de la arquitectura de software en donde se especificarán los componentes de la solución (Mittal & Chopra, 2011).
- La **implementación** es el momento en el que el equipo construye los diferentes componentes de la solución (Raghi, Sudha, Sreeram, & Steve Joshua, 2024) y asegura la calidad a través de los diferentes tipos de pruebas como funcionales, integración, performance, seguridad, etc. (Aishwarya, et al., 2023).
- Finalizada la validación y verificación de calidad del software iniciamos con la etapa de **despliegue**, en la cual ponemos en producción el producto software y puede ser utilizado por el usuario final (Yu & Le, 2012).
- Cuando el software se encuentra en uso inicia la etapa de **mantenimiento**, el cual implica la corrección de errores, actualizaciones y mejoras de la solución tanto a nivel de software como de hardware (Klespitz, Bíró, & Kovács, 2015).
- Al finalizar su vida útil el software pasa por la etapa de **retiro o cierre** en la cual el producto es retirado, y en muchos casos, es reemplazado por otra solución (Manju Khari, 2016).

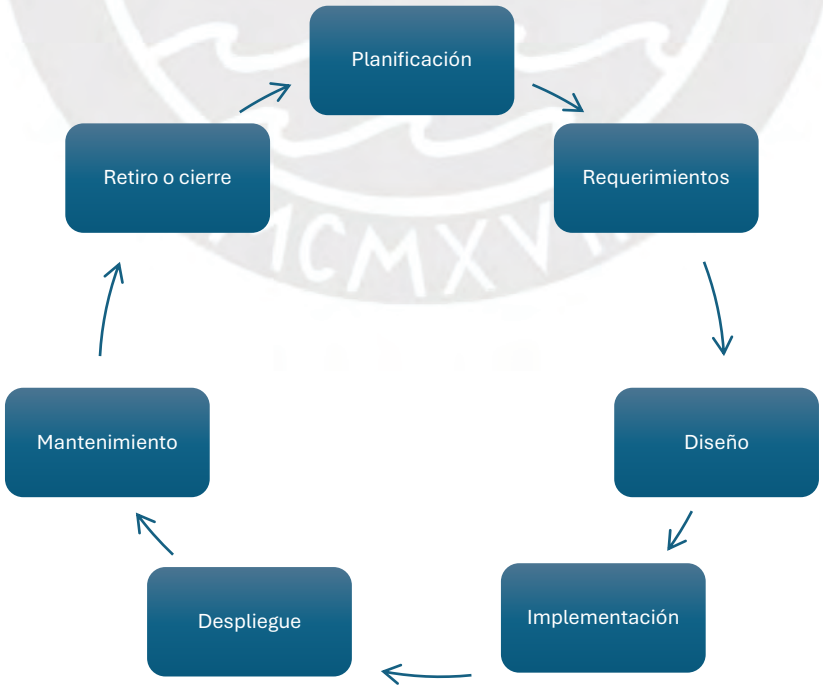


Figura 1. Ciclo de vida del software.
Fuente: Elaboración propia.

2.2. Arquitectura de software

2.2.1. Arquitectura monolítica

En esta revisión sistemática nos centramos en la arquitectura monolítica, también conocida como “monolito”. Este estilo representa un enfoque tradicional en el diseño de software, caracterizado por no priorizar la modularidad como principio fundamental (Kyryk, Pleskanka, Tymchenko, & Pleskanka, 2022).

Esta arquitectura se utiliza comúnmente para describir aplicaciones en las que todos los componentes están integrados en un único sistema. Durante las etapas iniciales del desarrollo, este tipo de diseño puede resultar útil, ya que permite a equipos pequeños construir y desplegar soluciones de manera rápida y sencilla. Sin embargo, con el paso del tiempo y el crecimiento de la aplicación, tiende a volverse difícil de mantener debido a la acumulación de funcionalidades y a la creciente complejidad del sistema (IEEE152, 1).

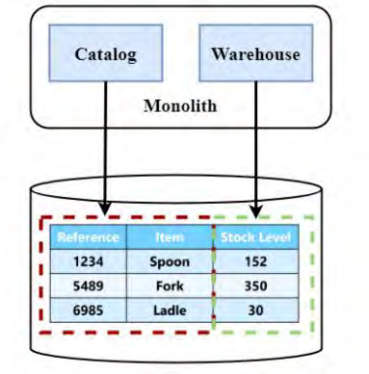


Figura 2. Ejemplo de una arquitectura monolítica.
(Farsi, Allaki, En-nouaary, & Dahchour, 2023, p. 5)

2.2.2. Arquitectura de microservicios

La arquitectura de microservicios está definida como un enfoque para desarrollar una aplicación como un conjunto de pequeños servicios, con una función específica, que se pueden implementar cada uno de manera independiente y se comunican a través de protocolos ligeros (Bajaj, Goel, & Gupta, 2022, p. 67008; Bataieneh, Ziadeh, & Al-Qora'n, 2024, p. 1). Entre sus principales características tenemos:

- **Escalabilidad:** Permite adaptar la infraestructura del sistema en función de la demanda, asegurando un rendimiento eficiente en distintos niveles de carga (Bajaj, Goel, & Gupta, 2022, p. 67008)
- **Flexibilidad:** Facilita la elección de tecnologías y lenguajes de programación adecuados según el contexto y los requisitos específicos del proyecto (Munjal, Gangodkar, & Lohumi, 2023, p. 4).
- **Independencia:** Permite desplegar componentes de forma aislada, sin afectar el funcionamiento del sistema (Munjal, Gangodkar, & Lohumi, 2023, p. 1).
- **Mantenibilidad:** Simplifica las tareas de mantenimiento y actualización, minimizando el impacto en el sistema (Bataieneh, Ziadeh, & Al-Qora'n, 2024, p. 3).
- **Resiliencia:** Mejora la tolerancia a fallos, evitando que errores en un componente afecten al resto del sistema (Bataieneh, Ziadeh, & Al-Qora'n, 2024, p. 3).

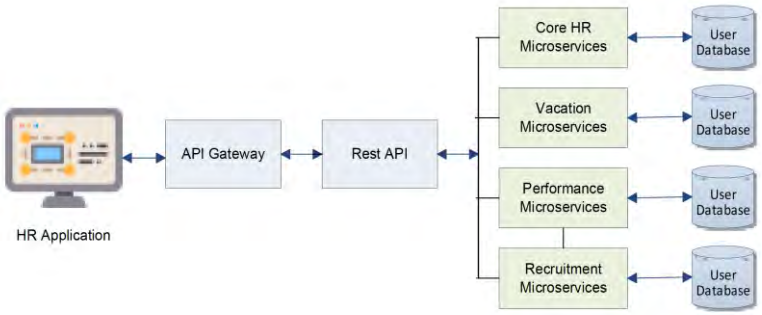


Figura 3. Ejemplo de una arquitectura de microservicios.
Fuente: Shabani, Hiseni, Hyseni, & Çiço (2024).

2.3. Tiempo de entrega o Lead Time

En las organizaciones modernas, la capacidad de adaptación al cambio es un factor clave para aprovechar las oportunidades del mercado. En este contexto, el tiempo que se requiere para incorporar nuevas necesidades del mercado en los productos de software puede marcar la diferencia entre el éxito y el fracaso de un negocio.

Una métrica fundamental en este proceso es el Lead Time o tiempo de entrega, que se define como el período total que transcurre desde la identificación de una necesidad o requisito hasta la entrega de la solución (Poppendieck & Poppendieck, 2003).

Uno de los principios esenciales del Lean Software Development es *"entregar rápido"*, con el objetivo de acortar los ciclos de desarrollo y obtener retroalimentación temprana y continua (Poppendieck & Poppendieck, 2003). Esto se logra mediante tiempos de entrega reducidos. Sin embargo, en el caso de las arquitecturas monolíticas, cumplir con este principio representa un desafío significativo, especialmente cuando el software ha crecido en complejidad y funcionalidades a lo largo del tiempo.

Por esta razón, es crucial considerar el Tiempo de Entrega o Lead Time como una métrica de referencia para la mejora continua de los procesos dentro del ciclo de vida del software.

CAPÍTULO 3. ESTADO DEL ARTE

Dentro de la búsqueda inicial de artículos primarios, se encontraron algunos estudios relacionados con revisiones sistemáticas de la literatura; sin embargo, estos no guardan similitud con la propuesta del presente estudio. Uno de los artículos más relacionados aborda los efectos del uso de AUTOSAR, un estándar abierto de arquitectura de software automotriz, que permite reducir los tiempos de entrega de software para automóviles, los cuales son considerados monolitos (Dersten, Axelsson, & Froberg, 2011).

Otra de las revisiones sistemáticas encontradas tiene como objetivo la gestión de resiliencia y confiabilidad en entornos distribuidos de soluciones que involucran el Internet de las Cosas (IoT). La resiliencia se entiende como la capacidad de los sistemas para mantener un alto nivel de servicio a pesar de las dificultades que pueda enfrentar la infraestructura. Por otro lado, la confiabilidad se refiere a la capacidad de los sistemas para proporcionar servicios seguros y adaptables (Amiri, Heidari, & Navimipour, 2022). Si bien ambos aspectos son importantes, la investigación no se relaciona con las arquitecturas monolíticas.

Por último, se encontró un artículo que revisa y analiza la información sobre Cloud Brokers, los cuales son utilizados para gestionar, integrar y seleccionar los servicios en la nube para una solución de software (Khan, Habaebi, & Islam, 2024); lo cual puede ser considerado una buena práctica, pero también se encuentra fuera del dominio de esta investigación.

Teniendo en cuenta la experiencia de Amazon Prime Video y los artículos académicos referenciados en este capítulo, las problemáticas que tienen las arquitecturas monolíticas se pueden solucionar a través de la migración hacia microservicios; sin embargo, las arquitecturas monolíticas han demostrado tener grandes beneficios a niveles de costo sin necesariamente incrementar de manera significativa los tiempos de entrega. Adicionalmente, en la búsqueda de artículos similares a esta investigación, no se ha encontrado en las fuentes académicas elegidas ninguna publicación similar. Por ello, esta revisión sistemática de la literatura es relevante para la comunidad académica, dado que busca identificar las mejores prácticas durante el ciclo de vida del software, que permitan a las arquitecturas monolíticas mejorar sus tiempos de entrega de las soluciones y así ser una referencia para futuros artículos que quieran profundizar en los resultados que tienen las prácticas sobre este tipo de arquitecturas.

CAPÍTULO 4. REVISIÓN SISTEMÁTICA DE LA LITERATURA

4.1. Metodología

Para cumplir el objetivo del presente estudio realizaremos una revisión sistemática de la literatura, la cual según Kitchenham y Charters lo definen como un método para identificar, evaluar e interpretar toda investigación disponible y que sea relevante para una pregunta de investigación especifica, un área temática o un fenómeno de interés (2007, pág. 3).

4.2. Proceso de revisión

Para la presente revisión sistemática utilizaremos el proceso recomendado por Kitchenham, quien lo resume en 3 etapas y un conjunto de actividades discretas tal como se muestra en la Figura 4.

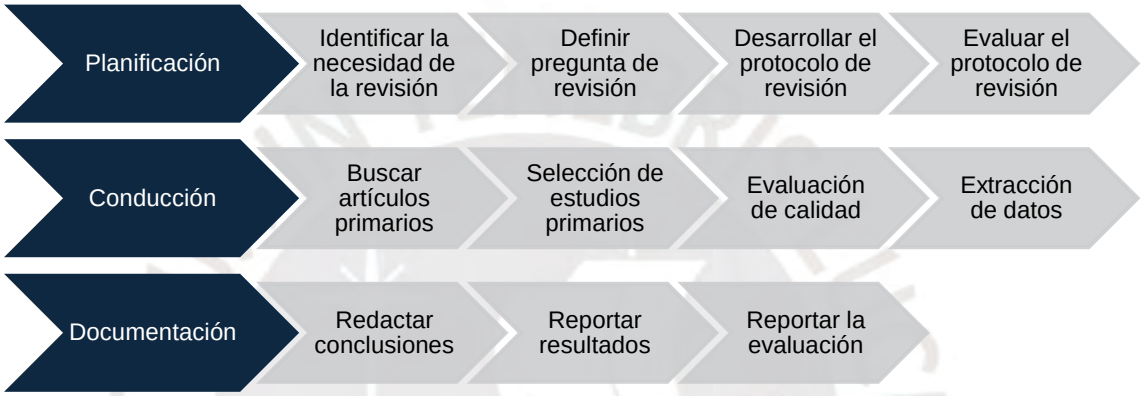


Figura 4. Proceso de revisión recomendado por Kitchenham.
Fuente: Elaboración propia.

4.3. Necesidad de la revisión

Para identificar las diversas prácticas y sus resultados en arquitecturas monolíticas, es esencial revisar estudios relacionados de manera ordenada y sistemática. Esto no solo garantiza una comprensión exhaustiva del tema, sino que también minimiza el sesgo de los investigadores, lo cual es crucial para asegurar la relevancia y validez de la investigación.

4.4. Preguntas de investigación

La presente investigación se enfoca en la realización de una revisión sistemática de la literatura. Para la formulación de las preguntas de investigación, se emplearán los criterios PICOC (Población, Intervención, Comparación, Resultado y Contexto), propuestos originalmente por Petticrew y Roberts para estudios en el ámbito médico (Kitchenham & Charters, 2007, págs. 9-12). No obstante, se tomará en cuenta la adaptación realizada por Kitchenham, quien ajustó estos criterios para su aplicación en investigaciones relacionadas con la ingeniería de software, como es el caso del presente estudio. La Tabla 1 presenta una descripción detallada del uso de cada uno de estos criterios.

Criterio	Descripción
Población	Grupos específicos como roles, categorías, áreas de conocimiento, industrias, etc.
Intervención	Metodología, herramienta, tecnología o procedimiento que aborda un problema específico.

Comparación	Metodología, herramienta, tecnología o procedimiento que se comparara con la intervención.
Resultados	Factores importantes para los profesionales. Suelen ser métricas que manifiestan un estado positivo o negativo.
Contexto	Contexto en donde se realiza la comparación. Puede ser industrial o académico, por ejemplo.

Tabla 1. Descripción del uso de los criterios PICOC
Fuente: (Kitchenham & Charters, 2007)

A partir de los criterios PICOC, se identificaron los términos clave necesarios para formular tanto las preguntas bibliográficas como las preguntas de investigación. Estas preguntas permitirán recopilar la información requerida para analizar la evolución de las investigaciones a lo largo del tiempo. Asimismo, facilitarán la identificación de prácticas aplicables en las distintas etapas del ciclo de vida del software, con el objetivo de optimizar la entrega de aplicaciones basadas en una arquitectura monolítica.

Criterio	Descripción
Población	Arquitecturas de software, Arquitectura de sistemas, Arquitectura de aplicaciones
Intervención	Arquitectura monolítica, Monolito tradicional, Monolito en capas, Monolito modular, Microkernel, Arquitectura Plugin, Arquitectura de capas y Arquitectura de Pipes & Filters
Comparación	
Resultados	
Contexto	Tiempo de entrega, Ciclo de vida del desarrollo de software, requerimiento, diseño, implementación, pruebas, instalación

Tabla 2. PICOC para PB1 al PB3
Fuente: Elaboración propia

PB1: ¿Cuál es la distribución de los artículos encontrados en las fuentes de investigación que están relacionadas con el tema?

La respuesta a esta pregunta permitirá estimar el volumen de artículos identificados inicialmente en las distintas fuentes consultadas, así como determinar cuál de ellas aporta la mayor cantidad de información relevante para la revisión.

PB2: ¿Cuál es la distribución de los artículos que tienen una relación directa con el tema de investigación contra los que no?

El objetivo de esta pregunta es identificar cuántos artículos están directamente relacionados con la mejora del tiempo de entrega, en contraste con aquellos que abordan el tema de manera secundaria, sin que constituya el eje central del estudio.

PB3: ¿Cuál es la distribución de los artículos seleccionados con respecto al país de origen?

Para esta investigación, resulta relevante identificar los países de origen de las publicaciones, ya que esta información puede ofrecer indicios sobre los contextos geográficos en los que los hallazgos podrían tener mayor pertinencia o aplicabilidad.

PI1: ¿Cómo las dificultades o problemáticas que tienen arquitecturas monolíticas durante el ciclo de vida de software afectan al tiempo de entrega de las aplicaciones?

La pregunta de investigación PI1 fue formulada a partir de los términos identificados en la Tabla 3. Su propósito es analizar cómo las dificultades asociadas a las aplicaciones con arquitectura monolítica afectan los tiempos de entrega de cambios o nuevas funcionalidades.

Criterio	Descripción
Población	Arquitecturas de software, Arquitectura de sistemas, Arquitectura de aplicaciones
Intervención	Arquitectura monolítica, Monolito tradicional, Monolito en capas, Monolito modular, Microkernel, Arquitectura Plugin, Arquitectura de capas y Arquitectura de Pipes & Filters
Comparación	No aplica
Resultados	Dificultade(s), problema(s), incidente(s)
Contexto	Ciclo de vida del desarrollo de software, requerimiento, diseño, implementación, pruebas, instalación

Tabla 3. PICOC de la PI1
Fuente: Elaboración propia.

PI2: ¿Cómo las prácticas, métodos o herramientas identificados en el ciclo de vida del software facilitan el delivery de aplicaciones con arquitectura monolítica?

Esta pregunta, formulada utilizando los criterios PICOC presentados en la Tabla 4, tiene como objetivo identificar, en la literatura académica, cómo la aplicación de diversas prácticas contribuye a mejorar los tiempos de entrega en aplicaciones con arquitectura monolítica.

Criterio	Descripción
Población	Arquitecturas de software, Arquitectura de sistemas, Arquitectura de aplicaciones
Intervención	Arquitectura monolítica, Monolito tradicional, Monolito en capas, Monolito modular, Microkernel, Arquitectura Plugin, Arquitectura de capas y Arquitectura de Pipes & Filters
Comparación	No aplica
Resultados	Práctica(s), guía(s), estándar(es), método(s), técnica(s), procedimiento(s) de desarrollo de software
Contexto	Ciclo de vida del desarrollo de software, requerimientos, diseño, implementación, pruebas, instalación

Tabla 4. PICOC de la PI2
Fuente: Elaboración propia.

PI3: ¿Qué resultados han dado la aplicación de las prácticas o métodos encontrados en esta investigación sobre los tiempos de entrega?

Para considerar una práctica o método como eficiente en la entrega de software, es necesario evaluar su impacto en la reducción del tiempo de entrega. El objetivo de esta pregunta es identificar tanto las prácticas que ya han sido validadas empíricamente como aquellas que aún requieren una validación experimental. En la Tabla 5 se presentan los términos utilizados para el diseño de la pregunta PI3.

Criterio	Descripción
Población	Arquitecturas de software, Arquitectura de sistemas, Arquitectura de aplicaciones
Intervención	Arquitectura monolítica, Monolito tradicional, Monolito en capas, Monolito modular, Microkernel, Arquitectura Plugin, Arquitectura de capas y Arquitectura de Pipes & Filters
Comparación	No aplica
Resultados	Tiempo de entrega
Contexto	Ciclo de vida del desarrollo de software, requerimiento, diseño, implementación, pruebas, instalación

Tabla 5. PICOC de la PI3
Fuente: Elaboración propia.

PI4: ¿Qué consideraciones debemos tomar para implementar las prácticas identificadas de manera exitosa?

Conocer las consideraciones necesarias para aplicar las distintas prácticas identificadas es relevante para este estudio, ya que permite delimitar las actividades clave para su adecuada implementación, comunicación y mantenimiento en el tiempo. En la Tabla 6 se presentan los términos de búsqueda definidos según los criterios PICOC.

Criterio	Descripción
Población	Arquitecturas de software, Arquitectura de sistemas, Arquitectura de aplicaciones
Intervención	Arquitectura monolítica, Monolito tradicional, Monolito en capas, Monolito modular, Microkernel, Arquitectura Plugin, Arquitectura de capas y Arquitectura de Pipes & Filters
Comparación	No aplica
Resultados	Recomendaciones, sugerencias, indicaciones, propuesta
Contexto	Ciclo de vida del desarrollo de software, requerimiento, diseño, implementación, pruebas, instalación, tiempo de entrega

Tabla 6. PICOC de la PI4
Fuente: Elaboración propia.

4.5. Protocolo de revisión

4.5.1. Definición de términos y cadena de búsqueda

Para llevar a cabo la búsqueda de los artículos primarios, es fundamental definir una estrategia que garantice la relevancia de los documentos recuperados en las bases de datos académicas seleccionadas (IEEE Xplorer, Scopus, ACM Digital Library y SpringerLink). Cabe señalar que estas fuentes fueron elegidas por su reconocida importancia en el campo de la ingeniería de software.

La cadena de búsqueda fue construida a partir de los términos definidos según los criterios PICOC, presentados en las Tablas 2 a 6. Estos términos fueron traducidos al inglés e incluyen sus respectivos sinónimos, cuando corresponde. El resultado de esta construcción se presenta en la Tabla 7.

	PB1-PB3	PI1	PI2	PI3	PI4
Población	("Software Architecture" OR "System Architecture" OR "Application Architecture")				
Intervención	("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture"))				
Resultados		(Difficult* OR Problem* OR Incident*)	(Practice* OR Standard* OR Guideline* OR Method* OR Technique* OR Procedure*)		(Recommend* OR Suggest* OR Indicate* OR Propose*)
Contexto	("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")				

Tabla 7. Términos de búsqueda derivados de PICOC

Fuente: Elaboración propia.

De acuerdo con lo definido en la Tabla 7, las cadenas de búsqueda definidas, según las recomendaciones de Ahmad, Jamshidi y Pahl, para cada una de las preguntas de investigación son las siguientes:

- Cadena 1: ("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND ("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
- Cadena 2: Cadena 1 AND (Difficult* OR Problem* OR Incident*)
- Cadena 3: Cadena 1 AND (Practice* OR Standard* OR Guideline* OR Method* OR Technique* OR Procedure*)
- Cadena 4: Cadena 1 AND (Recommend* OR Suggest* OR Indicate* OR Propose*)

La relación de las cadenas con respecto a las preguntas bibliográficas y de investigación son las siguientes:

Cadena de búsqueda	Preguntas
Cadena 1 (Principal)	PI1, PI2, PI3, PB3
Cadena 2	PB1
Cadena 3	PB2
Cadena 4	PB4

Tabla 8. Relación de las cadenas de búsqueda con las preguntas.
Fuente: Elaboración propia.

4.5.2. Selección de artículos

Para seleccionar los artículos relevantes para la investigación, es fundamental definir claramente los criterios de inclusión y exclusión. Estos criterios, detallados en las Tablas 9 y 10, garantizan que la selección de artículos se realice de manera objetiva y libre de la influencia del investigador.

Código	Descripción
CI01	Artículos extraídos desde una base de datos indexada
CI02	Estudios primarios (estudios de casos, experimentación, lecciones aprendidas, encuestas, etc.)
CI03	Artículos de investigación que aborden de manera parcial o total el estudio de arquitecturas monolíticas.
CI04	Artículos relacionados a prácticas de desarrollo de software aplicadas en arquitecturas monolíticas.

Tabla 9. Criterios de inclusión
Fuente: Elaboración propia.

Código	Descripción
CE01	Artículos que sea revisiones sistemáticas de la literatura o mapeos conceptuales
CE02	Estudios que no han sido escritos en inglés y español
CE03	Artículos de opinión o filosóficos
CE04	Publicación anterior al 2001
CE05	Acceso al artículo a través de una cuenta académica
CE06	Artículos duplicados

Tabla 10. Criterios de exclusión
Fuente: Elaboración propia.

Con respecto al criterio de exclusión CE04, se descartaron las publicaciones anteriores al año 2001. Este criterio se estableció considerando que, a partir de ese año, cobró mayor relevancia la entrega temprana de software, en línea con el principio ágil de “software funcionando sobre documentación extensiva”, establecido en el Manifiesto Ágil.

4.5.3. Criterios de selección

Una vez definidos los criterios de inclusión y exclusión, cada uno de los artículos resultantes de la búsqueda deberá pasar por las siguientes etapas de selección:

Etapa 1: Exclusión por acceso restringido
Se excluirán los artículos que no puedan ser accedidos mediante una cuenta académica institucional, ya que esta limitación impide la revisión del documento. Esta etapa responde al criterio de exclusión CE05.

Etapa 2: Selección según características formales
En esta etapa se evaluarán las características formales de los artículos, conforme a los criterios CI01, CI02, CE01, CE02, CE03 y CE04. Estos criterios permiten filtrar publicaciones que no cumplen con los requisitos básicos, como artículos de opinión, documentos en idiomas no dominados por los investigadores o publicaciones cuya antigüedad las hace poco relevantes para el estudio.

Etapa 3: Revisión de duplicados y relevancia temática
Primero, se eliminarán los artículos duplicados conforme al criterio CE06. Luego, se evaluará la relevancia temática de los artículos mediante la revisión del título, resumen y conclusiones, de acuerdo con los criterios CI03 y CI04. Esta revisión tiene como finalidad determinar si el artículo está relacionado con alguna de las preguntas de investigación definidas en la sección 4.1. Para garantizar la objetividad del proceso, se llevará un registro en una bitácora donde se documentará la justificación del investigador y la validación del asesor.

Etapa 4: Evaluación del contenido completo
Finalmente, se realizará una revisión detallada del contenido completo de los artículos seleccionados, aplicando nuevamente los criterios CI03 y CI04. Esta etapa permitirá una evaluación más precisa y acotada de la relevancia de cada artículo. Al igual que en la etapa anterior, se documentarán en la bitácora las razones de exclusión, en caso corresponda.

4.5.4. Criterios de evaluación de calidad

Además de los criterios de inclusión y exclusión, la evaluación de calidad permitirá determinar la solidez de las inferencias presentadas en los estudios seleccionados. Para ello, se empleará la lista de verificación propuesta por Zarour et al. (2015), la cual consta de cinco preguntas orientadas a evaluar la calidad metodológica de los artículos. Estas preguntas se presentan en la Tabla 11, junto con su respectivo objetivo.

Cada pregunta será calificada según el grado de cumplimiento: se asignará 1 punto si la respuesta cumple completamente, 0.5 puntos si el cumplimiento es parcial y 0 puntos si no se cumple. El puntaje total que puede obtener un artículo varía entre 0 y 5. Para considerar un estudio como de calidad aceptable, se establecerá un umbral mínimo de 2.5 puntos.

Código	Pregunta	Objetivo pregunta
Q01	¿Se explica suficientemente el objetivo de la investigación?	Conocer si se tienen los objetivos claros.
Q02	¿Se explica claramente la idea o el enfoque presentado?	Conocer si proporciona antecedentes.
Q03	¿Se tiene en cuenta las amenazas a la validez del estudio?	Validar la validez de la investigación.
Q04	¿Se describe claramente el contexto en el que se llevó a cabo la investigación?	El contexto de la investigación es claro.
Q05	¿Las conclusiones del artículo están debidamente sustentados en los resultados de la investigación?	Documentación de los hallazgos.

Tabla 11. Criterios de evaluación de calidad
Fuente: Elaboración propia.

4.6. Estrategia de extracción de datos

Para poder realizar la extracción de datos y mapear los artículos se diseñó un formulario que nos permita recopilar la información de la investigación según las preguntas de revisión y los criterios de selección. Esto tiene como finalidad reducir el riesgo de selección de documentos por algún sesgo del investigador. El formulario diseñado tiene debe pasar por la conformidad de una segunda revisión, en este caso del asesor académico. En Tabla 5 se detalla cada uno de los campos del formulario de extracción de datos.

Campo	Descripción	RQ
Código del artículo	Código interno para esta revisión sistemática	
Base de conocimiento	Nombre de la fuente de donde fue extraído el artículo	PB2
Año de publicación	Año en el que el artículo fue publicado en la base de conocimiento	PB3
País	País donde fue elaborado el estudio	General
Idioma	Idioma en el que es publicado el artículo	CE02
DOI	Código único de la investigación	General
Título	Título completo del artículo en su idioma original	General
Resumen	Resumen del artículo	PB1
Autores	Nombres de los autores del artículo	General
Palabras claves	Palabras claves del artículo	General
Enlace web	Enlace donde se encontró el artículo	General
Tipo de investigación	Clasificación que se le da en la fuente académica	General
Problemática identificada	Problemas que pueda identificar el artículo de investigación	PI1
Prácticas de desarrollo de software	Listar todas las prácticas que consideren que ayudan a mejorar el ciclo de vida del software en cualquiera de sus etapas	PI2
Resultados de la aplicación	Resultados de la aplicación de las prácticas identificadas	PI3
Consideraciones para realizar las prácticas	Recomendaciones o actividades que la investigación utiliza para aplicar las prácticas identificadas	PI4

Tabla 12. Campos del formulario de extracción de datos
Fuente: Elaboración propia.

CAPÍTULO 5. EJECUCIÓN DEL PROTOCOLO

5.1. Estudios relacionados

La búsqueda de estudios relacionados se inició en noviembre de 2024, utilizando las bases de datos académicas IEEE y ACM, con el objetivo de identificar investigaciones previas con objetivos similares. Para esta búsqueda, se emplearon los mismos términos definidos mediante la técnica PICOC, a los cuales se añadió el término “revisión sistemática de la literatura” y su abreviatura en inglés.

La cadena de búsqueda final en inglés fue la siguiente:
(“Systematic Literature Review” OR “Systematic Review” OR “Systematic Mapping” OR “Systematic Evidence” OR “Study Synthesis” OR “SRL”) AND (“Software Architecture” OR “System Architecture” OR “Application Architecture”) AND (“Monolithic Architecture” OR (“Traditional Monolith” OR “Layer Monolith” OR “Modular Monolith” OR Microkernel OR “Plugin Architecture” OR “Pipes and Filters Architecture” OR “Layered Architecture”)) AND (Practice* OR Standard* OR Guideline* OR Method* OR Technique* OR Procedure*) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")

El resultado de esta búsqueda nos mostró 1 artículo en IEEE y 14 en ACM; sin embargo, ninguno de los artículos encontrados aborda el objetivo de esta investigación.

5.2. Búsqueda de estudios primarios

En diciembre de 2024 se ejecutó la cadena de búsqueda utilizando los términos establecidos en la Tabla 3. En esta etapa aún no se aplicaron los criterios de exclusión. La búsqueda se llevó a cabo siguiendo las recomendaciones metodológicas de Ahmad, Jamshidi y Pahl (2012), quienes proponen un proceso de cinco pasos para la construcción de cadenas de búsqueda. Estos pasos se detallan en la Tabla 12.

Paso de búsqueda primaria	Descripción
Derivar cadena de búsqueda	Términos PICOC de la Tabla 7.
Considerar sinónimos y alternativas	Se identificaron términos sinónimos y acrónimos.
Combinar términos de búsqueda	Utilizar operadores lógicos para unir términos principales (AND) y sinónimos y acrónimos (OR)
Dividir cadena de búsqueda	Se divide la cadena para realizar la búsqueda en las diferentes bases de conocimientos
Gestionar referencias	Se utilizaron citas con notas finales

Tabla 13. Cinco pasos para definir la cadena de búsqueda
Fuente: Elaboración propia.

Las cadenas de búsqueda generadas mediante este procedimiento se presentan en el Anexo A. Los resultados obtenidos al ejecutar cada una de estas cadenas en las bases de datos seleccionadas fueron los siguientes:

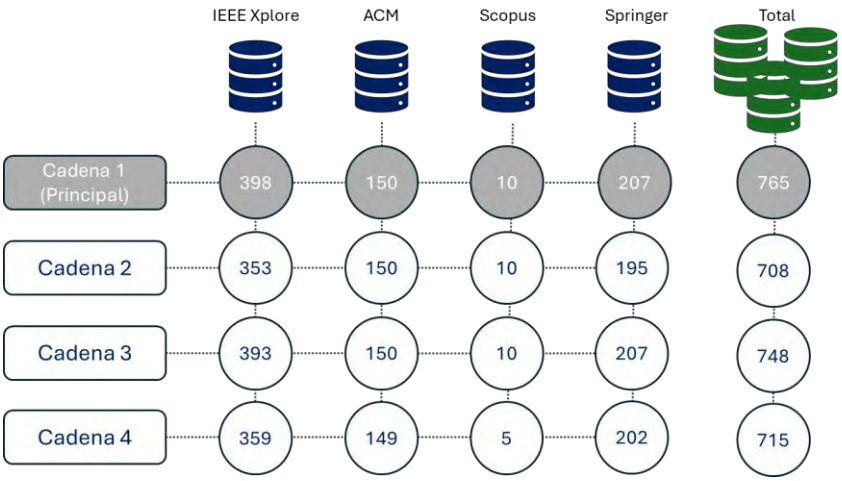


Figura 5. Resultados de la búsqueda primaria.
Fuente: Elaboración propia.

En esta etapa no se aplicó ningún filtro ni se eliminaron artículos del conjunto de resultados. La cadena principal fue la que generó la mayor cantidad de artículos, con un total de 765 registros obtenidos en las cuatro bases de datos consultadas. Por esta razón, los resultados derivados de su ejecución fueron considerados como base para continuar con el proceso de selección de estudios primarios.

5.3. Selección de estudios primarios

A partir de los 765 artículos obtenidos en la búsqueda primaria, se procedió a la selección de documentos siguiendo las cuatro etapas descritas en el apartado 4.6. Durante este proceso, se aplicaron los criterios de inclusión y exclusión previamente definidos, con el fin de identificar los estudios más relevantes para la investigación. Los resultados de esta selección se presentan en la Tabla 14.

Base de datos	Inicio	Etapa 1	Etapa 2	Etapa 3	Etapa 4
IEEE Xplore	398	294	253	204	58
ACM Digital Library	150	150	55	47	11
Scopus	10	8	4	4	1
Springer Link	207	67	65	23	10
Total	**765**	**519**	**377**	**278**	**80**

Tabla 14. Resultados de la selección primaria
Fuente: Elaboración propia.

Finalmente, se seleccionaron 80 artículos académicos, lo que representa el 10,6 % del total de documentos obtenidos en la búsqueda inicial. En el Anexo B se presenta un cuadro con los resultados del proceso de filtrado, únicamente de los artículos que llegaron a la Etapa 3 en adelante, aplicado mediante los criterios de inclusión y exclusión, con el propósito de transparentar las razones por las cuales ciertos estudios fueron descartados. Asimismo, en la Tabla C.1 del Anexo C se incluye el listado completo de los artículos seleccionados.

5.4. Evaluación de calidad

La evaluación de calidad se aplicó a los 80 estudios primarios seleccionados en la etapa anterior. Las preguntas utilizadas para esta evaluación se encuentran detalladas en la sección 4.7, específicamente en la Tabla 11. Los resultados completos de dicha evaluación se presentan en el Anexo D.

Como se observa en la Tabla 15, los estudios muestran un alto nivel de cumplimiento en las preguntas Q01, Q02, Q04 y Q05. Esto indica que, en general, los artículos analizados formulan con claridad sus objetivos, proporcionan antecedentes relevantes, describen adecuadamente el contexto de la investigación y documentan sus hallazgos de manera apropiada.

No obstante, en la pregunta Q03, relacionada con el análisis de posibles amenazas a la validez de los resultados, se evidencia un 42.5 % de incumplimiento total. Esto sugiere que la mayoría de los estudios no aborda este aspecto de forma explícita. Sin embargo, el 47,5 % lo trata parcialmente y solo el 10% lo aborda de manera completa.

ID	Si	Parcial	No
Q01	80 (100%)	0 (0.0%)	0 (0.0%)
Q02	80 (100%)	0 (0.0%)	0 (0.0%)
Q03	8 (10.0%)	38 (47.5%)	34 (42.5%)
Q04	78 (97.5%)	0 (0.0%)	2 (2.5%)
Q05	69 (86.3%)	0 (0.0%)	11 (13.8%)

Tabla 15. Resumen de los resultados de calidad por pregunta.
Fuente: Elaboración propia.

La distribución de resultados de cada una de las investigaciones primarias seleccionadas la podemos observar en la Figura 6. De esta, podemos concluir que ningún artículo ha tenido puntaje menor a 2.5, lo cual haría que tengamos que descartarlo. Por lo tanto, los 80 artículos elegidos en la búsqueda selección primaria son considerados dentro de la muestra para responder las preguntas bibliográficas y de investigación.

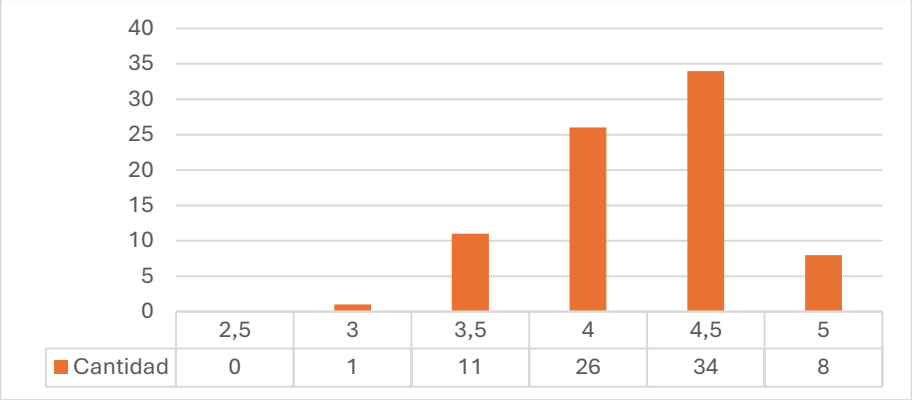


Figura 6. Distribución de estudios según el resultado de calidad
Fuente: Elaboración propia.

5.5. Extracción y síntesis de datos

Finalizado la evaluación de calidad, se extrajo los datos de cada uno de los artículos que han sido considerados para la investigación. La información de cada documento se encuentra en el Anexo C. A continuación, se presentan la síntesis de los datos de los estudios primarios.

Publicaciones por año

En la Figura 7 se muestra la cantidad de publicaciones por año, abarcando el periodo comprendido entre 2001 y 2024. El promedio anual de publicaciones relacionadas con esta revisión sistemática es de 3.3 artículos por año. Al observar esta tendencia, se destaca un aumento significativo en el número de publicaciones entre los años 2021 y 2024, lo que sugiere un creciente interés por parte de la comunidad académica en mejorar los tiempos de entrega de este tipo de arquitecturas.

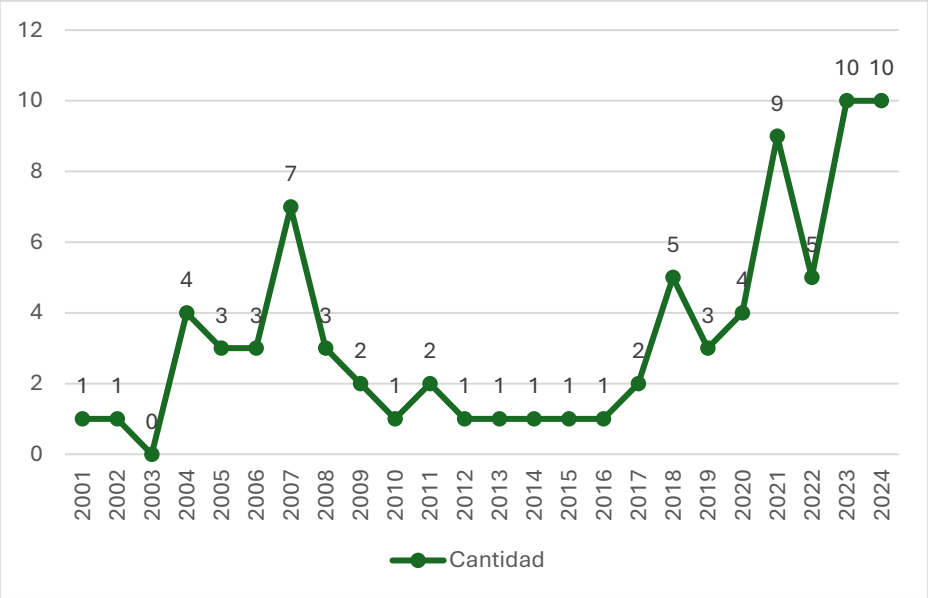


Figura 7. Cantidad de artículos publicados por año
Fuente: Elaboración propia.

Fuentes de publicación

Los 80 estudios seleccionados se agrupan en tres tipos de publicaciones: revistas científicas, que representan el 2.5%; artículos, que constituyen el 20% del total; y ponencias en congresos (conference papers), que conforman el 77.5% de la muestra. Esta distribución refleja una marcada preferencia de los investigadores por difundir sus trabajos en espacios académicos donde puedan ser discutidos, retroalimentados y enriquecidos por la comunidad científica, lo que contribuye al avance del conocimiento en torno a las problemáticas de la industria.

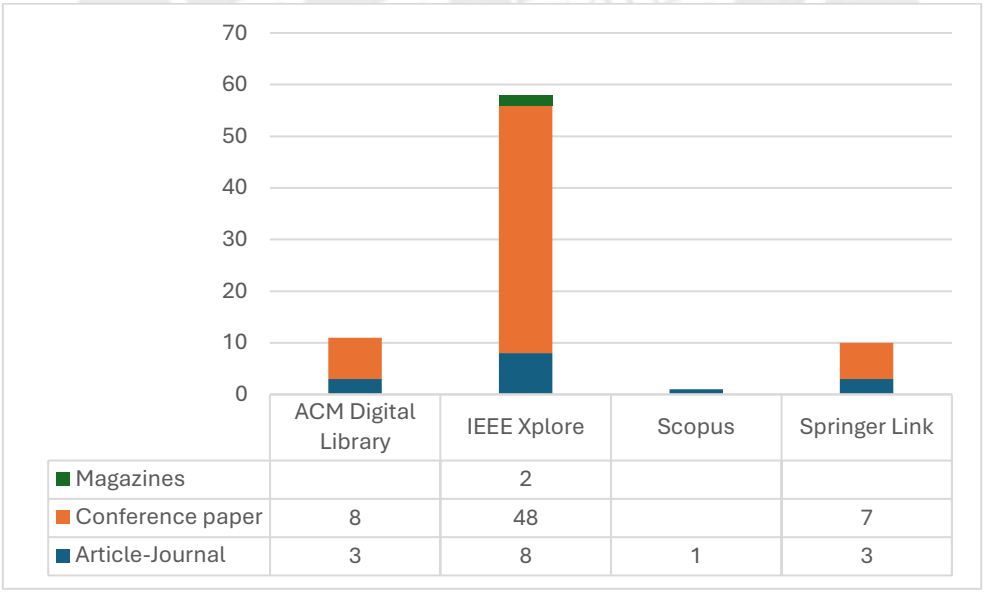


Figura 8. Distribución por tipo de publicación y fuente académica.
Fuente: Elaboración propia.

CAPÍTULO 6. RESULTADOS DE LA REVISIÓN SISTEMÁTICA

6.1. Preguntas bibliográficas

6.1.1. ¿Cuál es la distribución de los artículos encontrados en las fuentes de investigación que están relacionadas con el tema? (PB1)

La búsqueda inicial arrojó un total de 765 artículos en las cuatro fuentes seleccionadas. La mayor cantidad provino de IEEE Xplore, representando el 52% (398 documentos), de los cuales se seleccionaron 48 para esta investigación, lo que equivale al 12% de la muestra. Springer Link aportó 207 investigaciones inicialmente, representando el 27%, de las cuales se seleccionaron 12, equivalentes al 6% del total encontrado en esta fuente. En ACM se encontraron 150 artículos, que constituyen el 20% del total, y se seleccionaron 19 para la investigación, lo que representa el 13% del total de la base de datos. Finalmente, en Scopus se encontraron 10 artículos, que representan el 1% de la muestra inicial, y se seleccionó 1 de ellos para la investigación, lo que equivale al 10% de su aporte inicial. En la Figura 9 se puede observar mejor la distribución de cada fuente académica en la búsqueda inicial.

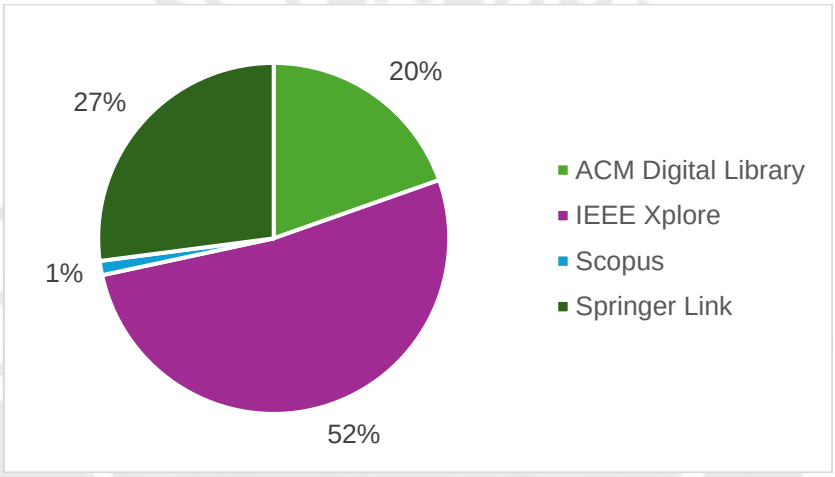


Figura 9. Distribución de los artículos encontrados por fuente.
Fuente: Elaboración propia.

6.1.2. ¿Cuál es la distribución de los artículos que tienen una relación directa con el tema de investigación contra los que no? (PB2)

Según los datos presentados en la PB1, se identificaron un total de 765 artículos. Sin embargo, como se muestra en la Figura 10, solo 80 cumplieron con los criterios de inclusión y exclusión establecidos, lo que representa aproximadamente el 10 % del total. Este porcentaje sirve como referencia común en investigaciones de este tipo. Por otro lado, se observa que la base de datos IEEE fue la que más aportes realizó, con 60 artículos, y concentra la mayor cantidad de publicaciones relacionadas con la ingeniería de software en comparación con las demás fuentes consultadas.

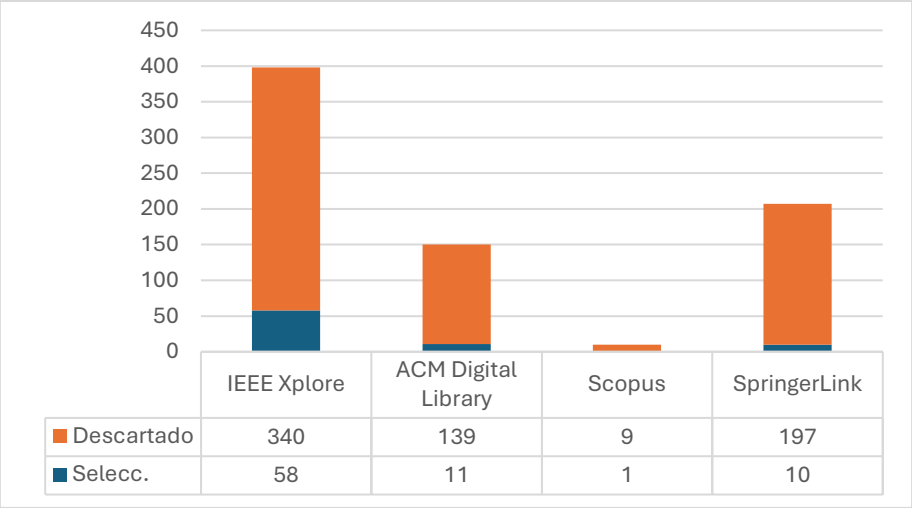


Figura 10. Distribución diferenciada de artículos encontrados por fuente.
Fuente: Elaboración propia.

6.1.3. *¿Cuál es la distribución de los artículos seleccionados con respecto al país de origen? (PB3)*

En la Figura 11 se observa que Estados Unidos es el país con mayor número de publicaciones, con un total de 8 artículos, lo cual resulta comprensible dado que su mercado tecnológico se encuentra entre los más relevantes a nivel global. En segundo lugar, se encuentra India y Alemania con 7 artículos, países que lidera industria del software en sus respectivas regiones (statista.com, 2025).

A continuación, se destacan países como Brasil, Suecia, Países Bajos, Francia y Finlandia, cuyos aportes oscilan entre 6 y 4 artículos. El resto de los países presenta una contribución menor, con entre 3 y 1 artículos, por lo que han sido agrupados por región para facilitar el análisis. Este agrupamiento revela que tanto Europa como Asia concentran un volumen significativo de publicaciones académicas en el área. En cuanto a Sudamérica, además de Brasil, se destacan una contribución proveniente de Chile.

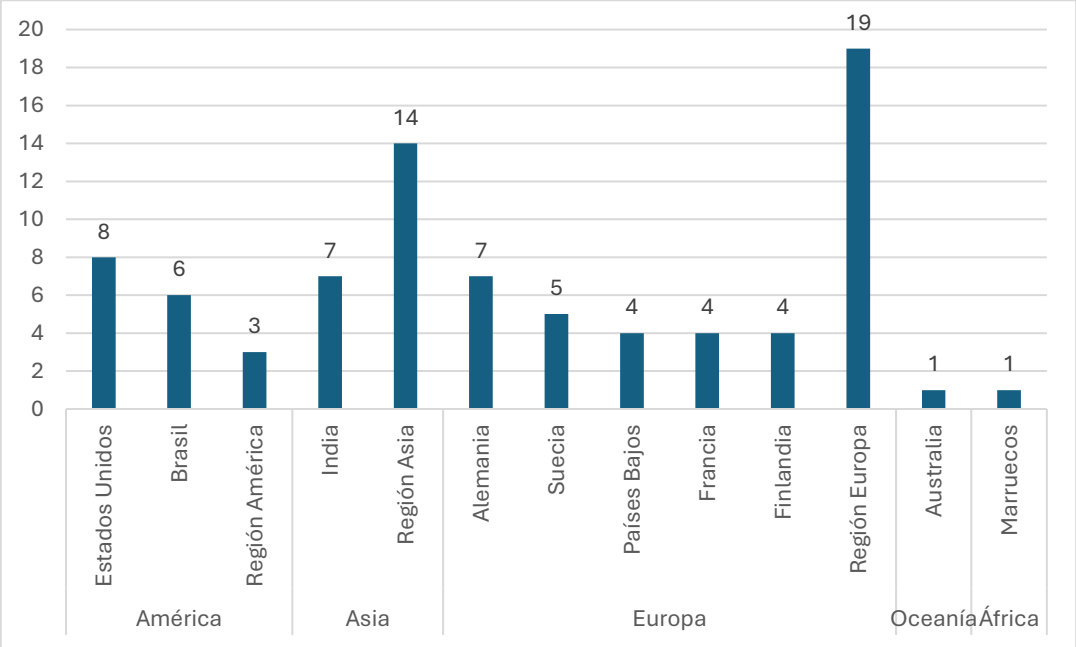


Figura 11. Distribución de artículos seleccionados por país o región de origen.
Fuente: Elaboración propia.

6.2. Preguntas de investigación

6.2.1. ¿Cómo las dificultades o problemáticas que tienen arquitecturas monolíticas durante el ciclo de vida de software afectan al tiempo de entrega de las aplicaciones? (PI1)

Las arquitecturas monolíticas al inicio de su implementación pueden ser una solución que genere resultados rápidos, sin embargo, conforme va creciendo se puede volver más compleja y traer diferentes problemas que afectan los tiempos de entrega (Rocha & Teixeira, 2024, p. 469). A continuación, se describen como diferentes dificultades o problemáticas asociadas a las características de este tipo de arquitectura afectan los tiempos de entrega en este tipo de arquitectura:

- El **alto acoplamiento** característico de las arquitecturas monolíticas implica un esfuerzo significativo para ejecutar pruebas y desplegar cambios, incluso cuando estos son menores (Gouigoux & Tamzalit, 2017, p. 62). Esta rigidez dificulta tanto la modificación del código fuente, debido a su elevada complejidad, como la migración hacia tecnologías modernas como la computación en la nube (Farsi, Allaki, En-nouaary, & Dahchour, 2023, p. 2; Menychtas, et al., 2013, p. 427). Asimismo, la transición hacia arquitecturas basadas en microservicios se ve limitada, ya que resulta más complejo descomponer el sistema en servicios independientes y garantizar la consistencia de los datos (Ayas, Leitner, & Hebig, 2023, p. 2; Mallidi, Sharma, & Singh, 2021, p. 164).
- La **escasa flexibilidad** para la escalabilidad horizontal representa otra limitación importante, ya que dificulta la optimización independiente de los componentes durante el despliegue (Hawilo, Jammal, & Shami, 2019, p. 202). Además, la integración de nuevos elementos se ve obstaculizada por la obsolescencia tecnológica, lo que requiere esfuerzos adicionales para asegurar un rendimiento adecuado de la solución y puede extender los tiempos de entrega (Mparmpoutis & Kakarontzas, 2023, p. 1; Nam & Hung, 2021).
- La **falta de modularidad** incrementa las dificultades para integrar nuevos componentes, generando esfuerzos adicionales para lograr su adaptación y una comunicación efectiva (Ayas, Leitner, & Hebig, 2023, p. 27). Esta rigidez también restringe la posibilidad de realizar cambios de manera incremental, lo que limita la evolución progresiva del sistema (Christian, Steven, Kurniawan, & Anggreainy, 2023, p. 67). A su vez, la interdependencia entre componentes puede prolongar el tiempo requerido para las pruebas (Sotomayor, Allala, Santiago, King, & Clarke, 2023, p. 56).
- El **despliegue** de este tipo de arquitecturas demanda un esfuerzo considerable para gestionar la trazabilidad y mantener la coherencia de los cambios, debido a la gran cantidad de archivos y componentes que deben actualizarse (Monti, et al., 2008, p. 1066). Además, cualquier modificación implica la recompilación del sistema completo, lo que incrementa los tiempos de implementación (Prasandy, Titan, Murad, & Darwis, 2020, p. 726). A esto se suma que los errores durante el despliegue pueden ser más difíciles de aislar y resolver, afectando negativamente los tiempos de entrega (Mulahuwaish, Korbel, & Qolomany, 2022, p. 4).
- Entre otras dificultades que afectan los tiempos de entrega se encuentra la duplicación de esfuerzos debido a la **escasa reutilización de código** (Tekinerdoğan, Yakın, Yağız, & Özcan, 2020, p. 1). Asimismo, los **despliegues** de esta arquitectura son de todos sus componentes, lo que genera recompilaciones extensas en tiempo (Breivold, Crnkovic, Land, & Larsson, 2008). A esto se suma la **falta de innovación** de estos diseños, lo que contribuye a la falta de evolución de este tipo de arquitecturas (Fernández, Le Mouël, Lin, & Escudié, 2023, p. 1).

6.2.2. ¿Cómo las prácticas, métodos o herramientas identificados en el ciclo de vida del software facilitan el delivery de aplicaciones con arquitectura monolítica? (PI2)

La mejora en los tiempos de entrega se debe a diversos tipos de cambios que impactan múltiples etapas del ciclo de vida del software, permitiendo resolver o mitigar varios de los problemas previamente descritos. A continuación, se describen las prácticas identificadas en los artículos académicos seleccionados y cómo estas contribuyen a optimizar los tiempos de entrega en soluciones basadas en arquitecturas monolíticas.

- La mayoría de las dificultades identificadas están relacionadas con el **diseño arquitectónico** de este tipo de soluciones. Por ello, adoptar una arquitectura modular permite reutilizar código de manera eficiente (Sarda, 2015, p. 1), mientras que una arquitectura en capas brinda mayor independencia para realizar cambios sin afectar toda la solución (Tekinerdoğan, Yakın, Yağız, & Özcan, 2020, p. 7). Este tipo de diseño facilita la migración o integración de microservicios (Shabani, Hiseni, Hyseni, & Çiço, 2024, p. 6). Para asegurar una integración óptima entre componentes, es fundamental contar con interfaces bien definidas que favorezcan la flexibilidad y escalabilidad (Sehestedt, Giannopoulou, Monot, & Wahler, 2019, p. 221).
- Los estudios también destacan el uso de diversas **prácticas de desarrollo**, como la automatización de pruebas y despliegues. Estas prácticas permiten validar exhaustivamente el producto (Salii, Ajdari, & Zenuni, 2023, pág. 1672), reducir el trabajo manual (Bataieneh, Ziadeh, & Al-Qora'n, 2024, p. 67009) y minimizar los errores humanos (Raja, 2024, p. 130). Asimismo, la refactorización periódica mejora la calidad del código, lo que disminuye el esfuerzo requerido en futuras entregas (Assunção, Marchezan, Arkoh, Egyed, & Ramler, 2024, p. 14).
- El uso de **herramientas tecnológicas** representa una alternativa eficaz para mejorar la productividad de los equipos. Los generadores de código (Saxena & Bhowmik, 2023, p. 2) y las plataformas de integración y despliegue continuo facilitan la entrega de nuevas funcionalidades (Assunção, Marchezan, Arkoh, Egyed, & Ramler, 2024, p. 18). Además, los contenedores aseguran la consistencia entre entornos de trabajo y la virtualización de infraestructura permite una mejor asignación de recursos, incrementando la tolerancia a fallos (Gouigoux & Tamzalit, 2017, p. 63; Manjiyani, Iatrou, Graube, & Urbas, 2019, p. 1673).
- Una **documentación** adecuada facilita la comprensión del sistema, mejora la comunicación entre equipos y reduce los errores durante el desarrollo. Esto contribuye a implementar cambios de manera más eficiente (Salii, Ajdari, & Zenuni, 2023, pág. 1672; Nogueira, Felizardo, Amaral, Assunção, & Colanzi, 2024, pág. 767). Por otro lado, el **monitoreo** continuo genera información valiosa que permite identificar y resolver problemas con mayor rapidez (Sotomayor, Allala, Santiago, King, & Clarke, 2023, p. 61).
- Finalmente, la **adopción de principios ágiles**, como los propuestos por Scrum y DevOps, promueve ciclos de desarrollo más cortos, lo que mejora significativamente la entrega de software (Mallidi, Sharma, & Singh, 2021, p. 166).

6.2.3. ¿Qué resultados han dado la aplicación de las prácticas o métodos encontrados en esta investigación sobre los tiempos de entrega? (PI3)

En el protocolo de la revisión sistemática se estableció que los artículos seleccionados debían incluir el tiempo de entrega como uno de los resultados de sus

investigaciones. Si bien muchos estudios afirman que las prácticas aplicadas contribuyen a reducir los tiempos de entrega, la mayoría no proporciona datos cuantitativos específicos, limitándose a señalar beneficios generales en este aspecto. A continuación, se presentan los resultados de aquellas prácticas identificadas en este estudio que sí cuantifican, de alguna manera, la reducción del tiempo de entrega:

- La **reutilización de componentes** y el **desarrollo modular** han demostrado ser prácticas eficientes, ya que pueden reducir el costo del trabajo por la reducción del tiempo de desarrollo entre un 20% y un 40%, y el tiempo de despliegue entre un 50% y un 75%. Además, mejoran la escalabilidad y la calidad del software (Witman & Ryan, 2007, p. 6).
- La **reducción del tamaño del código** mediante el uso de una arquitectura de microcomponentes puede disminuirlo entre un 45% y un 72%, lo cual facilita una mejor reutilización del código y reduce los tiempos de entrega (Yamakami, 2005, p. 6).
- La **virtualización** permite realizar pruebas en entornos locales como en nube, lo que acelera los tiempos de implementación pasando de desarrollar entre uno a cinco días a menos de un día (Sehestedt, Giannopoulou, Monot, & Wahler, 2019, p. 222).
- La **reingeniería estructurada** mejora los trabajos de mantenimiento y la reutilización de código. Está práctica reduce los defectos potenciales a cero, dejando un sistema más eficiente y bien documentado. En la experimentación pasaron de 52 defectos potenciales a cero (Ko, Chang, & Kim, 2007, pág. 8).
- Finalmente, las **actualizaciones parciales** permiten actualizar parte del software sin necesidad de actualizar todo. Esto puede reducir los tiempos de desarrollo hasta en un 80% (Matijević, Herceg, Milošević, & Četić, 2021).

6.2.4. ¿Qué consideraciones debemos tomar para implementar las prácticas identificadas de manera exitosa? (PI4)

Las prácticas descritas en el apartado 6.2.2. pueden parecer intuitivas para optimizar los tiempos de entrega en arquitecturas monolíticas. No obstante, es fundamental comprender cómo estas prácticas pueden ser adoptadas de manera efectiva en distintos contextos organizacionales y proyectos específicos. Por ello, la siguiente pregunta tiene como propósito identificar las recomendaciones y consideraciones que diversos investigadores han planteado en sus estudios para lograr una implementación exitosa de dichas prácticas.

- Las **herramientas y tecnologías** juegan un papel esencial en la ejecución de tareas a lo largo del ciclo de vida del software. Por ello, su adecuada selección resulta crítica. Es fundamental considerar aspectos como la compatibilidad con el código existente, la disponibilidad de soporte técnico y la capacidad de integración con el entorno de desarrollo (Sotomayor, Allala, Santiago, King, & Clarke, 2023). Este análisis cobra especial relevancia en sistemas heredados con arquitectura monolítica, donde la compatibilidad de herramientas virtualizadas puede representar un desafío considerable (Fernández, Le Mouël, Lin, & Escudié, 2023, p. 73698).
- Las buenas prácticas en **desarrollo de software** subrayan la importancia de contar con entornos homologados a producción, lo que permite obtener resultados precisos durante las pruebas (Sotomayor, Allala, Santiago, King, & Clarke, 2023, p. 59). También se recomienda realizar pruebas preliminares para detectar posibles problemas que puedan resolverse mediante refactorización (Eggert, Günther, Maletschek, Maxiniuc, & Mann-Wahrenberg, 2022, p. 177). Asimismo, la definición y ejecución de pruebas unitarias a lo largo de toda la fase de implementación

contribuye significativamente a garantizar la estabilidad y compatibilidad del desarrollo (Fernández, Le Mouël, Lin, & Escudié, 2023, p. 73690).

- La implementación de un sistema de **monitoreo** que permita observar las actividades de los usuarios, el uso de recursos y el rendimiento del sistema es clave para mantener la estabilidad operativa y la eficiencia en el uso de recursos, especialmente en arquitecturas complejas (Mulahuwaish, Korbel, & Qolomany, 2022, p. 15). Este tipo de monitoreo también ayuda a reducir la complejidad operativa, facilitando la gestión y el mantenimiento del sistema (Prakash & Arora, 2024, p. 4).
- En el ámbito de la **gestión de proyectos**, es esencial establecer flujos de trabajo claramente definidos que orienten la participación de los equipos involucrados (Monti, et al., 2008, p. 1066). Igualmente, resulta clave contar con una estrategia de escalamiento adecuada para gestionar los cambios de forma eficiente (Monti, et al., 2008, p. 1068). La alineación de los procesos organizacionales y de gobernanza con las nuevas prácticas o herramientas a adoptar es necesaria para asegurar su integración efectiva (Menychtas, et al., 2013, p. 428). Finalmente, la gestión de la obsolescencia debe contemplarse en la planificación, ya que permite optimizar el uso de los recursos de hardware disponibles (Prakash & Arora, 2024, p. 8).
- La **gestión del conocimiento** es un factor determinante para reducir la complejidad inherente a las arquitecturas monolíticas. Es crucial asegurar que la documentación de los requerimientos sea clara (Ferreira, Assunção, Martinez, & Figueiredo, 2022, pág. 23) y que exista trazabilidad bidireccional entre dicha documentación y los artefactos generados (Monti, et al., 2008, p. 1069). Además, la capacitación continua de los equipos en nuevas tecnologías o prácticas es esencial para disminuir la curva de aprendizaje y facilitar la adopción efectiva de cambios (Rocha & Teixeira, 2024, p. 475).
- La priorización de **lineamientos estratégicos** constituye un mecanismo eficaz para garantizar la adopción coherente de prácticas y herramientas a lo largo de las distintas etapas del ciclo de vida del software (Gouigoux & Tamzalit, 2017, p. 64). En este contexto, las directrices arquitecturales —como **los patrones de diseño** (Chondamrongkul, Sun, & Warren, 2021, págs. 8-12) — y la definición de un flujo claro para la gestión de cambios en estos artefactos contribuyen significativamente a mejorar la calidad y flexibilidad del diseño arquitectónico (Kim, 2006, p. 304). Asimismo, establecer estándares de seguridad permite alinear las prácticas con los requisitos de protección definidos por la organización (Martorell, Fabre, Lauer, Roy, & Valentin, 2015, p. 77). La alineación entre los objetivos técnicos y los de negocio es esencial para enfocar los esfuerzos en la satisfacción de las necesidades reales de los usuarios (Prakash & Arora, 2024, p. 7).
- Finalmente, la adopción de nuevas prácticas vinculadas a la **cultura organizacional** es clave para promover enfoques modernos como DevOps o metodologías ágiles. Para ello, es necesario preparar a los equipos para afrontar estos cambios (Ayas, Leitner, & Hebig, 2023, p. 22) y mitigar la resistencia mediante programas de capacitación y una comunicación efectiva (Salii, Ajdari, & Zenuni, 2023, pág. 1674). La definición de una estructura de gobernanza clara (Rocha & Teixeira, 2024, p. 471), respaldada por un comité ejecutivo que refuerce los estándares establecidos (Iocola, 2007, pág. 3), permitirá a los equipos optimizar esfuerzos frente a los desafíos organizacionales (Salii, Ajdari, & Zenuni, 2023, pág. 1674) y concentrarse en resolver sus preocupaciones, mejorando así los procesos de implementación (Premchand & Choudhry, 2018, pág. 31).

CAPÍTULO 7. CONCLUSIONES Y TRABAJOS FUTUROS

A partir de la revisión sistemática de 80 artículos académicos, se identificaron diversas prácticas que contribuyen a resolver o mitigar muchos de los problemas inherentes a las arquitecturas monolíticas. Estas prácticas permiten reducir los tiempos de entrega de cambios o nuevas funcionalidades. Entre ellas destacan: el desacoplamiento de la arquitectura hacia un modelo modular que facilite la independencia de los cambios; el uso de herramientas de automatización y virtualización para minimizar errores en distintos procesos; el aprovechamiento de la información del sistema y la documentación para reducir la complejidad en su comprensión; y la adopción de principios ágiles que promuevan ciclos cortos de entrega. Todas estas estrategias contribuyen significativamente a mejorar la eficiencia en este tipo de diseños arquitectónicos.

Sin embargo, la mayoría de los estudios revisados no cuantifica el impacto en tiempo de estas prácticas en los distintos contextos en los que se aplican. Además, es importante considerar que la implementación de estas prácticas requiere inversiones en herramientas de automatización y virtualización, así como en la capacitación de los equipos para que comprendan y apliquen adecuadamente los nuevos modelos arquitectónicos y metodologías de desarrollo. También es fundamental establecer lineamientos que aseguren una adopción uniforme de estas prácticas, respaldados por una estructura organizacional sólida que facilite y sostenga el cambio.

Se recomienda que futuros trabajos que implementen alguna de estas prácticas durante el ciclo de vida del software incluyan métricas que permitan medir la reducción de tiempo lograda. Esto permitirá reforzar de manera objetiva los beneficios observados. Por otro lado, no se encontró ningún estudio similar al caso de Prime Video, en el que se haya migrado de una arquitectura de microservicios a una monolítica. Sería valioso que investigaciones futuras exploren este tipo de transición, identificando las consideraciones necesarias para llevarla a cabo, así como sus ventajas y desventajas.

REFERENCIAS BIBLIOGRÁFICAS

Ahmad, A., Jamshidi, P., & Pahl, C. (2012). *A Classification and Comparison of Software Architecture Evolution Reuse-Knowledge: Protocol for Systematic Literature Review.* Dublin, Ireland: Dublin City University.

Aishwarya, V., Pediredla, S., Radhika, B., Vasanthi, B., Padmanaban, K., & Velmurugan, A. (2023). Incorporating of Security Methods into the Software Development Lifecycle Process (SDLC). *2023 International Conference on Computer Communication and Informatics (ICCCI).* Coimbatore, India: IEEE.

Amiri, Z., Heidari, A., & Navimipour, N. J. (2022). Resilient and dependability management in distributed environments: a systematic and comprehensive literature review. *Cluster Computing*, 1565–1600.

Assunção, W. K., Marchezan, L., Arkoh, L., Egyed, A., & Ramler, R. (2024). Contemporary Software Modernization: Strategies, Driving Forces, and Research Opportunities. *ACM Transactions on Software Engineering and Methodology, Volume 34, Issue 5*, 1-35.

Ayas, H. M., Leitner, P., & Hebig, R. (2023, 05 22). An empirical study of the systemic and technical migration towards microservices. *Empirical Software Engineering*, pp. 1-50.

Bajaj, D., Goel, A., & Gupta, S. C. (2022). GreenMicro: Identifying Microservices From Use Cases in Greenfield Development. *IEEE Access (Volume: 10)*, 67008 - 67018.

Bataieneh, S., Ziadeh, A., & Al-Qora'n, L. F. (2024). Microservices Architecture for Improved Maintainability and Traceability in MVC-Based E-Learning Platforms: RoadMap for Future Developments. *2024 15th International Conference on Information and Communication Systems (ICICS)* (pp. 1-6). Irbid, Jordania: IEEE.

Breivold, H. P., Crnkovic, I., Land, R., & Larsson, M. (2008). Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study. *2008 The Third International Conference on Software Engineering Advances* (págs. 205-213). Sliema, Malta: IEEE.

Chondamrongkul, N., Sun, J., & Warren, I. (23 de Julio de 2021). Software Architectural Migration: An Automated Planning Approach. *ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 30, Issue 4*, págs. 1 - 35.

Christian, J., Steven, Kurniawan, A., & Anggreainy, M. S. (2023). Analyzing Microservices and Monolithic Systems: Key Factors in Architecture, Development, and Operations. *2023 6th International Conference of Computer and Informatics Engineering (IC2IE)* (pp. 64-69). Lombok, Indonesia: IEEE.

Dersten, S., Axelsson, J., & Froberg, J. (2011). Effect Analysis of the Introduction of AUTOSAR: A Systematic Literature Review. *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 239-246). Oulu, Finland: IEEE.

Eggert, M., Günther, K., Maletschek, J., Maxiniuc, A., & Mann-Wahrenberg, A. (2022). In three steps to software product lines: a practical example from the automotive industry. *SPLC '22: Proceedings of the 26th ACM International Systems and Software Product Line Conference* (pp. 170 - 177). Graz, Austria: Association for Computing Machinery.

Farsi, H., Allaki, D., En-nouaary, A., & Dahchour, M. (2023). Dealing with Anti-Patterns When Migrating from Monoliths to Microservices: Challenges and Research Directions. *2023 IEEE 6th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)* (p. 8). Marruecos: IEEE.

Fernández, D., Le Mouël, F., Lin, T., & Escudié, M.-P. (2023). A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems. *IEEE Access (Volume: 11)*, 73688 - 73753.

Ferreira, R. A., Assunção, W. K., Martinez, J., & Figueiredo, E. (2022). Open-source software product line extraction processes: the ArgoUML-SPL and Phaser cases. *Empirical Software Engineering* , 1-35.

Gouigoux, J.-P., & Tamzalit, D. (2017). From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture. *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)* (pp. 62-65). Gothenburg, Sweden: IEEE Xplore.

Hawilo, H., Jammal, M., & Shami, A. (2019, Abril). Exploring Microservices as the Architecture of Choice for Network Function Virtualization Platforms. *IEEE Network*, pp. 202 - 210.

Iocola, P. (2007). When Legacy Meets SOA: Achieving Business Agility by Integrating New Technology with Existing Software Asset. *2007 1st Annual IEEE Systems Conference* (págs. 1-8). Honolulu, HI, USA: IEEE.

Khan, M. H., Habaebi, M. H., & Islam, M. R. (2024). A Systematic Literature Review of Cloud Brokers for Autonomic Service Distribution. *IEEE Access (Volume: 12)*, 131164 - 131187.

Kim, J. A. (2006). Case Study of Product Line Engineering in Insurance Company. *2006 IEEE International Conference on Information Reuse & Integration* (pp. 303-306). Waikoloa, HI, USA: IEEE.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. 1-65.

Klespitz, J., Bíró, M., & Kovács, L. (2015). Aspects of improvement of software development lifecycle management. *2015 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI).* Budapest, Hungary: IEEE.

Ko, H., Chang, G., & Kim, K. (2007). A Reengineering Approach of the Legacy System in the Digital Media Domain. *International Conference on Software Engineering Advances (ICSEA 2007)* (págs. 1-9). Cap Esterel, France: IEEE.

Kolny, M. (2023, Marzo 22). *Scaling up the Prime Video audio/video monitoringservice and reducing costs by 90%.* Retrieved from web.archive.org: https://web.archive.org/web/20240325042615/https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

Mallidi, R. K., Sharma, M., & Singh, J. (2021). Legacy Digital Transformation: TCO and ROI Analysis. *International Journal of Electrical and Computer Engineering Systems Volume 12*, pp. 163 - 170.

Manjiyani, Z., Iatrou, C. P., Graube, M., & Urbas, L. (2019). Open Cognitive Control System Architecture. *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (pp. 1673-1677). Zaragoza, Spain: IEEE.

Manju Khari, V. y. (2016). Embedding security in Software Development Life Cycle (SDLC). 20.

Martorell, H., Fabre, J.-C., Lauer, M., Roy, M., & Valentin, R. (2015). Partial Updates of AUTOSAR Embedded Applications -- To What Extent? *2015 11th European Dependable Computing Conference (EDCC)* (pp. 73-84). Paris, France: IEEE.

Matijević, B., Herceg, M., Milošević, M., & Četić, N. (2021). Partial software update on the AURIX TC397 platform. *2021 Zooming Innovation in Consumer Technologies Conference (ZINC)* (pp. 64-69). Novi Sad, Serbia: IEEE.

Menychtas, A., Santzaridou, C., Kousiouris, G., Varvarigou, T., Orue-Echevarria, L., & Juncal, A. (2013). ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud. *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (pp. 424-431). Timisoara, Romania: IEEE Xplore.

Mittal, V., & Chopra, C. (2011). Potter Model - A Change Compliant Software Development Lifecycle Model. *2011 Second International Conference on Intelligent Systems, Modelling and Simulation.* Phnom Penh, Cambodia: IEEE.

Monti, S., Nesci, W., Angellotti, S., Schellino, C., Seminara, M., & Wuesthenagen, R. (2008). Configuration and Change Management of the Outcomes of an Automotive Engine Control Model Based Software Design Process. *2008 32nd Annual IEEE International Computer Software and Applications Conference* (pp. 1065-1069). Turku, Finland: IEEE.

Mparmpoutis, A., & Kakarontzas, G. (2023). Using Database Schemas of Legacy Applications for Microservices Identification: A Mapping Study. *ICACS '22: Proceedings of the 6th International Conference on Algorithms, Computing and Systems* (pp. 1 - 7). New York, United States: Association for Computing Machinery.

Mulahuwaish, A., Korbel, S., & Qolomany, B. (2022). Improving datacenter utilization through containerized service-based architecture. *Journal of Cloud Computing* , 1-29.

Munjal, P., Gangodkar, D., & Lohumi, Y. (2023). Microservices based Ticket Booking Platform deployed on Cloud. *2023 Second International Conference on Informatics (ICI)* (pp. 1-5). Noida, India: IEEE.

Nam, L. H., & Hung, P. D. (2021). Building a Big Data Oriented Architecture for Enterprise Integration. *Cooperative Design, Visualization, and Engineering (CDVE 2021)* (pp. 172–182). Virtual Event: Springer.

Nogueira, V. L., Felizardo, F. S., Amaral, A. M., Assunção, W. K., & Colanzi, T. E. (2024). Insights on Microservice Architecture Through the Eyes of Industry Practitioners. *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (págs. 765-777). Flagstaff, AZ, USA: IEEE.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Addison-Wesley.

Prakash, C., & Arora, S. (2024). Systematic Analysis of Factors Influencing Modulith Architecture Adoption over Microservices. *2024 TRON Symposium (TRONSHOW)* (pp. 1-8). Tokyo, Japan: IEEE.

Prasandy, T., Titan, Murad, D. F., & Darwis, T. (2020). Migrating Application from Monolith to Microservices. *2020 International Conference on Information Management and Technology (ICIMTech)* (pp. 726-731). Bandung, Indonesia: IEEE.

Premchand, A., & Choudhry, A. (2018). Architecture Simplification at Large Institutions using Micro Services. *2018 International Conference on Communication, Computing and Internet of Things (IC3IoT)* (págs. 30-35). Chennai, India: IEEE.

Raghi, K., Sudha, K., Sreeram, A., & Steve Joshua, S. (2024). Software Development Automation Using Generative AI. *2024 International Conference on Emerging Research in Computational Science (ICERCS).* Coimbatore, India: IEEE.

Raja, R. (2024). Software Architecture for Agricultural Robots: Systems, Requirements, Challenges, Case Studies, and Future Perspectives. *IEEE Transactions on AgriFood Electronics (Volume: 2, Issue: 1)*, 125 - 137.

Rocha, R., & Teixeira, A. (2024). Accelerating the software development process through the application of MVP and monolithic architecture. *SBQS '24: Proceedings of the XXIII Brazilian Symposium on Software Quality* (pp. 469 - 477). Salvador Bahia, Brazil: Association for Computing Machinery.

Salii, S., Ajdari, J., & Zenuni, X. (2023). Migrating to a microservice architecture: benefits and challenges. *2023 46th MIPRO ICT and Electronics Convention (MIPRO)* (págs. 1670-1677). Opatija, Croatia: IEEE.

Sarda, T. (2015). Frugal embedded software development methodology for instrument cluster and load center. *2015 IEEE International Transportation Electrification Conference (ITEC)* (pp. 1-5). Chennai, India: IEEE.

Saxena, D., & Bhowmik, B. (2023). Paradigm Shift From Monolithic To Microservices. *2023 IEEE International Conference on Recent Advances in Systems Science and Engineering* (pp. 1-7). Kerala, India: IEEE Xplorer.

Sehestedt, S., Giannopoulou, G., Monot, A., & Wahler, M. (2019). Virtualizing Embedded Firmware to Boost Innovation Cycles. *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)* (pp. 218-225). Hamburg, Germany: IEEE.

Shabani, I., Hiseni, N., Hyseni, D., & Çiço, B. (2024). Optimizing HR Monolithic Systems to Modern HR Systems using Microservices Architecture. *2024 13th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1-7). Budva, Montenegro: IEEE.

Sotomayor, J. P., Allala, S. C., Santiago, D., King, T. M., & Clarke, P. J. (2023). Comparison of open-source runtime testing tools for microservices. *Software Quality Journal, Volume 31*, 55–87.

statista.com. (2025, 6 4). *Software - Worldwide*. Retrieved from Statista: https://www.statista.com/outlook/tmo/software/worldwide

Storm, M., Venegas, M., Gocinski, A., Myers, A., Brooks, J., & Fortuna, K. (2021). Stakeholders' Perspectives on Partnering to Inform the Software Development Lifecycle of Smartphone Applications for People with Serious Mental Illness: Enhancing the Software Development Lifecycle Through Stakeholder Engagement. *2021 IEEE Global Humanitarian Technology Conference (GHTC).* Seattle, WA, USA: IEEE.

Tekinerdoğan, B., Yakın, İ., Yağız, S., & Özcan, K. (2020). Product Line Architecture Design of Software-Intensive Physical Protection Systems. *2020 IEEE International Symposium on Systems Engineering (ISSE)* (pp. 1-8). Vienna, Austria: IEEE.

Witman, P. D., & Ryan, T. (2007). Innovation in Large-Grained Software Reuse: A Case from Banking. *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)* (pp. 1-10). Waikoloa, HI, USA: IEEE.

Yamakami, T. (2005). A micro-component architecture approach for next generation embedded browsers. *Second International Conference on Embedded Software and Systems (ICESS'05)* (pp. 1-7). Xi'an, China: IEEE.

Yu, W., & Le, K. (2012). Towards a Secure Software Development Lifecycle with SQUARE+R. *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops.* Izmir, Turkey: IEEE.

ANEXOS

ANEXO A. CADENAS DE BÚSQUEDA POR BASE DE DATOS

Base de datos	Cadena de búsqueda
IEEE Explore	(("Full Text & Metadata":"Software Architecture" OR "Full Text & Metadata":"System Architecture" OR "Full Text & Metadata":"Application Architecture") AND ("Full Text & Metadata":"Monolithic Architecture" OR "Full Text & Metadata":"Traditional Monolith" OR "Full Text & Metadata":"Layer* Monolith" OR "Full Text & Metadata":"Modular Monolith" OR "Full Text & Metadata":"Microkernel" OR "Full Text & Metadata":"Plugin Architecture" OR "Full Text & Metadata":"Pipes and Filters Architecture" OR "Full Text & Metadata":"Layered Architecture") AND ("Full Text & Metadata":"Software Development Lifecycle" OR "Full Text & Metadata":SDLC OR ("Full Text & Metadata":Software AND ("Full Text & Metadata":"Requirement*" OR "Full Text & Metadata":"Design*" OR "Full Text & Metadata":"Develop*" OR "Full Text & Metadata":"Test*" OR "Full Text & Metadata":"Deploy*"))) AND ("Full Text & Metadata":"Lead Time" OR "Full Text & Metadata":"Delivery Time" OR "Full Text & Metadata":"Leadtime" OR "Full Text & Metadata":"Time-to-Market" OR "Full Text & Metadata":"Time to Market" OR "Full Text & Metadata":TTM OR "Full Text & Metadata":T2M))
ACM Digital Library	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND ("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
Scopus	ALL("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ALL("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND ALL("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ALL("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
SpringerLink	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND ("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")

Tabla A.1. Cadena de búsqueda #1.

Para las preguntas de investigación que utilizan la cadena 2 se muestra en la Tabla A.2.

Base de datos	Cadena de búsqueda
IEEE Explore	(("Full Text & Metadata":"Software Architecture" OR "Full Text & Metadata":"System Architecture" OR "Full Text & Metadata":"Application Architecture") AND ("Full Text & Metadata":"Monolithic Architecture" OR "Full Text & Metadata":"Traditional Monolith" OR "Full Text & Metadata":"Layer* Monolith" OR "Full Text & Metadata":"Modular Monolith" OR "Full Text & Metadata":"Microkernel" OR "Full Text & Metadata":"Plugin Architecture" OR "Full Text & Metadata":"Pipes and Filters Architecture" OR "Full Text & Metadata":"Layered Architecture") AND **("Full Text & Metadata":"Difficult" OR "Full Text & Metadata":"Problem" OR "Full Text & Metadata":"Incident")** AND ("Full Text & Metadata":"Software Development Lifecycle" OR "Full Text & Metadata":SDLC OR ("Full Text & Metadata":Software AND ("Full Text & Metadata":"Requirement*" OR "Full Text & Metadata":"Design*" OR "Full Text & Metadata":"Develop*" OR "Full Text & Metadata":"Test*" OR "Full Text & Metadata":"Deploy*"))) AND ("Full Text & Metadata":"Lead Time" OR "Full Text & Metadata":"Delivery Time" OR "Full Text & Metadata":"Leadtime" OR "Full Text & Metadata":"Time-to-Market" OR "Full Text & Metadata":"Time to Market" OR "Full Text & Metadata":TTM OR "Full Text & Metadata":T2M))
ACM Digital Library	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **(Difficult* OR Problem* OR Incident*)** AND ("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
Scopus	ALL("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ALL("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **ALL(Difficult* OR Problem* OR Incident*)** AND ALL("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ALL("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
SpringerLink	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **(Difficult* OR Problem* OR Incident*)** AND ("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")

Tabla A.2. Cadena de búsqueda #2.

Base de datos	Cadena de búsqueda
IEEE Explore	(("Full Text & Metadata":"Software Architecture" OR "Full Text & Metadata":"System Architecture" OR "Full Text & Metadata":"Application Architecture") AND ("Full Text & Metadata":"Monolithic Architecture" OR "Full Text & Metadata":"Traditional Monolith" OR "Full Text & Metadata":"Layer* Monolith" OR "Full Text & Metadata":"Modular Monolith" OR "Full Text & Metadata":"Microkernel" OR "Full Text & Metadata":"Plugin Architecture" OR "Full Text & Metadata":"Pipes and Filters Architecture" OR "Full Text & Metadata":"Layered Architecture") AND **("Full Text & Metadata":Practice OR "Full Text & Metadata":Standard OR "Full Text & Metadata":Guideline OR "Full Text & Metadata":Method OR "Full Text & Metadata":Technique OR "Full Text & Metadata":Procedure)** AND ("Full Text & Metadata":"Software Development Lifecycle" OR "Full Text & Metadata":SDLC OR ("Full Text & Metadata":Software AND ("Full Text & Metadata":"Requirement*" OR "Full Text & Metadata":"Design*" OR "Full Text & Metadata":"Develop*" OR "Full Text & Metadata":"Test*" OR "Full Text & Metadata":"Deploy*"))) AND ("Full Text & Metadata":"Lead Time" OR "Full Text & Metadata":"Delivery Time" OR "Full Text & Metadata":"Leadtime" OR "Full Text & Metadata":"Time-to-Market" OR "Full Text & Metadata":"Time to Market" OR "Full Text & Metadata":TTM OR "Full Text & Metadata":T2M))
ACM Digital Library	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **(Practice* OR Standard* OR Guideline* OR Method* OR Technique* OR Procedure)** AND("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
Scopus	ALL("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ALL("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **ALL(Practice* OR Standard* OR Guideline* OR Method* OR Technique* OR Procedure)** AND ALL("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ALL("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
SpringerLink	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **(Practice* OR Standard* OR Guideline* OR Method* OR Technique* OR Procedure)** AND("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")

Tabla A.3. Cadena de búsqueda #3.

Base de datos	Cadena de búsqueda
IEEE Explore	(("Full Text & Metadata":"Software Architecture" OR "Full Text & Metadata":"System Architecture" OR "Full Text & Metadata":"Application Architecture") AND ("Full Text & Metadata":"Monolithic Architecture" OR "Full Text & Metadata":"Traditional Monolith" OR "Full Text & Metadata":"Layer* Monolith" OR "Full Text & Metadata":"Modular Monolith" OR "Full Text & Metadata":"Microkernel" OR "Full Text & Metadata":"Plugin Architecture" OR "Full Text & Metadata":"Pipes and Filters Architecture" OR "Full Text & Metadata":"Layered Architecture") AND **("Full Text & Metadata":Recommend OR "Full Text & Metadata":Suggest OR "Full Text & Metadata":Indicate OR "Full Text & Metadata":Propose)** AND ("Full Text & Metadata":"Software Development Lifecycle" OR "Full Text & Metadata":SDLC OR ("Full Text & Metadata":Software AND ("Full Text & Metadata":"Requirement*" OR "Full Text & Metadata":"Design*" OR "Full Text & Metadata":"Develop*" OR "Full Text & Metadata":"Test*" OR "Full Text & Metadata":"Deploy*"))) AND ("Full Text & Metadata":"Lead Time" OR "Full Text & Metadata":"Delivery Time" OR "Full Text & Metadata":"Leadtime" OR "Full Text & Metadata":"Time-to-Market" OR "Full Text & Metadata":"Time to Market" OR "Full Text & Metadata":TTM OR "Full Text & Metadata":T2M))
ACM Digital Library	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **(Practice* OR Suggest* OR Indicate* OR Propose*)** AND("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
Scopus	ALL("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ALL("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **ALL(Practice* OR Suggest* OR Indicate* OR Propose*)** AND ALL("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ALL("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")
SpringerLink	("Software Architecture" OR "System Architecture" OR "Application Architecture") AND ("Monolithic Architecture" OR ("Traditional Monolith" OR "Layer Monolith" OR "Modular Monolith" OR "Microkernel" OR "Plugin Architecture" OR "Pipes and Filters Architecture" OR "Layered Architecture")) AND **(Practice* OR Suggest* OR Indicate* OR Propose*)** AND("Software Development Lifecycle" OR "SDLC" OR (Software AND (Requirement* OR Design* OR Develop* OR Test* OR Deploy*))) AND ("Delivery Time" OR "Lead Time" OR "Leadtime" OR "Time-To-Market" OR "Time to Market" OR "TTM" OR "T2M")

Tabla A.4. Cadena de búsqueda #4.

ANEXO B. RESULTADOS DEL FILTRADO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE001	2022	Product Agnostic Generic Software Design and Development	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE002	2008	A Case Study on SW Product Line Architecture Evaluation: Experience in the Consumer Electronics Domain	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE003	2003	Embedded software engineering: the state of the practice	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE004	2005	Comparing design alternatives from field-tested systems to support product line architecture design	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE005	2012	Architecture for Large-Scale Innovation Experiment Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE006	2006	Research of Component-Based Hybrid Design Pattern for Real-Time Microkernel	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE007	2023	Paradigm Shift From Monolithic to Microservices	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE008	2024	Model Based Design Approach for ePWT Software & System ,Battery Management System for xEVs	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE009	2009	Trends in Embedded Software Engineering	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE010	2008	Configuration and Change Management of the Outcomes of an Automotive Engine Control Model Based Software Design Process	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE011	2011	Using Guidelines to Improve Quality in Software Nonfunctional Attributes	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE012	2009	Applying software product line technology to prototyping of real-time object tracking	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE013	2021	Stability in Software Engineering: Survey of the State-of-the-Art and Research Directions	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE014	2012	Sustainability guidelines for long-living software systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE015	2004	Establishing a software architecting environment	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE016	2005	Reflection on Software Architecture Practices - What Works, What Remains to Be Seen, and What Are the Gaps	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE017	2016	Software architecture techniques and emergence of problem domain in E-Governance	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE018	2005	Software reuse in product populations	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE019	2014	Test logic reuse through unit test patterns a test automation framework for software product lines	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE020	2009	Efficient and Adapted Component-Based Strategies for Embedded Software Device Drivers Development	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE021	2005	Development of architecture and software technologies in high-performance low-power SoC design	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE022	2022	Model-Based Software Development - Autocode to Autosar	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE023	2007	Flexible Application Software Generation for Heterogeneous Multi-Processor System-on-Chip	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE024	2015	Preventing software architecture erosion through static architecture conformance checking	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE025	2017	Towards a Process Model for Service-Oriented Development of Embedded Software Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE026	2004	Real world influences on software architecture - interviews with industrial system experts	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE027	2022	Software Development Models for IoT	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE028	2005	Toward an architectural knowledge base for wireless service engineering	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE029	2007	Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE031	2004	An extensible monolithic software design approach for Internet-convergent embedded applications	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE032	2005	Component-based development process and component lifecycle	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE034	2019	Virtualizing Embedded Firmware to Boost Innovation Cycles	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE035	2004	UML-based reverse engineering and model analysis approaches for software architecture maintenance	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE037	2005	Using MAP for Recovering the Architecture of Web Systems of a Spanish Insurance Company	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE038	2007	Reconfigurable Software Architecture for Voice Access to Data Services	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE040	2024	Software Architecture for Agricultural Robots: Systems, Requirements, Challenges, Case Studies, and Future Perspectives	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE041	2004	Marketplace issues in software planning and design	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE043	2015	Frugal embedded software development methodology for instrument cluster and load center	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE044	2006	Component-Based Development Process and Component Lifecycle	SI	SI	-	-	NO	NO	NO	SI	NO	NO
IEEE045	2006	The feature-architecture mapping (FArM) method for feature-oriented development of software product lines	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE046	2004	Building a service assurance system in KT	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE047	2023	A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE048	2003	Exploiting programmable bist for the diagnosis of embedded memory cores	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE049	2021	Emphasis on Evaluative Prerequisites for Decisive Software-in-the-Loop (SiL) Environments	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE050	2018	Feature Oriented Software Development Framework for Stock Exchange Systems	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE051	2007	Extending transaction level modeling for embedded software design and validation	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE052	2024	Domain-Driven Design for Microservices: An Evidence-Based Investigation	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE053	2024	Fault Tolerance Enhancement in Microservices using Orchestration Process with Agile	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE054	2024	Analysis of Methodologies and Tools for Software Development in Different Architectures	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE055	2008	Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE056	2016	Impact of Introducing Domain-Specific Modelling in Software Maintenance: An Industrial Case Study	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE057	2009	Linking GENESYS application architecture modelling with platform performance simulation	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE059	2013	Multi-level MPSoC modeling for reducing software development cycle	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE060	2020	Microservice Decomposition via Static and Dynamic Analysis of the Monolith	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE061	2009	ICT-Emuco. An innovative solution for future smart phones	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE062	2022	GreenMicro: Identifying Microservices From Use Cases in Greenfield Development	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE063	2021	Aligning Architecture with Business Goals in the Automotive Domain	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE064	2009	A component model and layered system architecture for reconfigurable CNC systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE065	2023	Analyzing Microservices and Monolithic Systems: Key Factors in Architecture, Development, and Operations	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE066	2024	Microservices Architecture for Improved Maintainability and Traceability in MVC-Based E-Learning Platforms: RoadMap for Future Developments	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE067	2024	Navigating the Landscape: An In-Depth Exploration of Modern Application Development Methodologies and Practices	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE068	2017	From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE071	2011	Lessons from Space	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE072	2007	Innovation in Large-Grained Software Reuse: A Case from Banking	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE074	2005	A micro-component architecture approach for next generation embedded browsers	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE075	2006	Eclipse-based management system for process innovation & methodology enhancement	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE076	2022	On Trustworthy Scalable Hardware/Software Platform Design	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE077	2004	On the expected synergies between component-based software engineering and best practices in product integration	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE078	2016	BaaS-4US: A Framework to Develop Standard Backends as a Service for Ubiquitous Applications	SI	SI			NO	NO	NO	NO	NO	NO
IEEE081	2010	Net centric radar technology and development using an open system architecture approach	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE082	2001	Implementing an agent system using N-tier pattern-based framework	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE083	2024	Adaptive CI/CD: A Flexible Architecture for Software Development	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE084	2008	A Highly Automated Documentation System: Component Design	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE085	2013	A Model-Driven Infrastructure for Developing Product Line Architectures Using CVL	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE086	2007	Introducing Impact Analysis for Architectural Decisions	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE087	2011	An Architectural Approach to Support Online Updates of Software Product Lines	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE088	2018	Architecture Patterns, Quality Attributes, and Design Contexts: How Developers Design with Them	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE089	2008	Exploring the Service-Oriented Enterprise: Drawing Lessons from a Case Study	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE090	2001	A survey of configurable, component-based operating systems for embedded applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE091	2013	Towards a Scalable PaaS for Service Oriented Software	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE092	2008	Enabling Verifiable Conformance for Product Lines	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE093	2016	Extractive SPL adoption applied into a small software company	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE094	2006	Asynchronous mediation for integrating business and operational processes	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE095	2020	Product Line Architecture Design of Software-Intensive Physical Protection Systems	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE096	2013	Robust and integrated diagnostics for safety systems in the industrial domain	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE098	2024	Smart Agriculture: Software Platform for Telematics Monitoring in Farm Machinery	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE099	2023	Microservices based Ticket Booking Platform deployed on Cloud	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE100	2023	The Case for the Simpler Cyber-Physical System	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE101	2012	Qualitative Analysis of the Impact of SOA Patterns on Quality Attributes	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE103	2023	From Monolith to Service-Oriented Architecture: A Model-Based Design Approach towards Software-Defined Vehicles	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE104	2018	Platform-Centric Self-Awareness as a Key Enabler for Controlling Changes in CPS	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE105	2008	An Embedded Computing Platform for Robot	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE106	2019	Flow Constraint Language for Coordination by Exogenous Connectors	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE107	2023	Migrating to a microservice architecture: benefits and challenges	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE108	2021	Quality Assurance for Microservice Architectures	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE110	2017	On Service-Orientation for Automotive Software	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE111	2010	Applied MDA for Embedded Devices: Software Design and Code Generation for a Low-Cost Mobile Phone	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE112	2007	A Reengineering Approach of the Legacy System in the Digital Media Domain	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE113	2002	T-Engine: the open, real-time embedded-systems platform	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE114	2021	Partial software update on the AURIX TC397 platform	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE115	2004	Electronic simulation for virtual reality: virtual prototyping	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE116	2007	When Legacy Meets SOA: Achieving Business Agility by Integrating New Technology with Existing Software Asset	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE119	2024	Systematic Analysis of Factors Influencing Modulith Architecture Adoption over Microservices	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE120	2022	Architecture as a Backbone for Safe DevOps in Automotive Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE121	2023	Implementing Cross-Organizational FDA Medical Device Design Controls Using Blockchain	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE123	2009	Component Based Middleware-Synthesis for AUTOSAR Basic Software	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE124	2007	TVSkin: A Skin-able User Interface for Digital TV Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE125	2020	Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE126	2024	DevSec-GPT ,Äî Generative-AI (with Custom-Trained Meta's Llama2 LLM), Blockchain, NFT and PBOM Enabled Cloud Native Container Vulnerability Management and Pipeline Verification Platform	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE129	2002	Does Q = MC/sup 2/? (On the relationship between Quality in electronic design and the Model of Colloidal Computing)	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE130	2024	Smart City Middleware: A Survey and a Conceptual Framework	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE134	2002	"Linux on ITRON": a hybrid operating system architecture for embedded systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE135	2015	Designing applications for heterogeneous many-core architectures with the FlexTiles Platform	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE137	2011	Decentralized Autonomous-Agent-Based Infrastructure for Agile Multiparallel Manufacturing	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE140	2022	The software solution for precise agriculture using the NDVI index	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE141	2009	Application driven Policy Based Resource Management for IP multimedia subsystems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE142	2024	EMDRIVE Architecture: Embedded Distributed Computing and Diagnostics from Sensor to Edge	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE144	2021	RT-RMS: A Real-Time Resiliency Management System for Operational Decision Support	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE145	2011	Virtual Hellfire Hypervisor: Extending Hellfire Framework for embedded virtualization support	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE146	2013	ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE149	2012	A Hardware/Software CBSE Framework for RTOS Services: The Timing Service Case Study	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE150	2018	A Method to Assess Assembly Complexity of Industrial Products in Early Design Phase	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE151	2009	Towards Lightweight Application Integration Based on Mashup	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE152	2024	Optimizing HR Monolithic Systems to Modern HR Systems using Microservices Architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE153	2011	A Supportability Framework	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE154	2017	Agile design of low-cost autonomous underwater vehicles	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE155	2010	Radar Open System Architecture provides net centricity	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE157	2001	The application of component-based methodology in developing visual power system analysis tool	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE159	2024	On the Usefulness of Automatically Generated Microservice Architectures	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE161	2020	Better Data Structures for Co-simulation of Distribution System with GridLAB-D and Python	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE162	2021	Layered Architecture and Virtualization for 5G Slicing	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE163	2010	An intelligent agent-based distributed architecture for Smart-Grid integrated network management	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE164	2013	Session Border Controller Virtualization Towards "Service-Defined" Networks Based on NFV and SDN	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE166	2015	ALFRED: A Methodology to Enable Component Fault Trees for Layered Architectures	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE168	2011	Decoupling variability management in multi-tenant SaaS applications	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE169	2021	Emergent Design Challenges for Embedded Systems and Paths Forward: Mixed-criticality, Energy, Reliability and Security Perspectives: Special Session Paper	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE171	2024	Insights on Microservice Architecture Through the Eyes of Industry Practitioners	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE173	2011	Net centric radar technology & development using an open system architecture approach	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE174	2021	Migrating Monoliths to Microservices-based Customizable Multi-tenant Cloud-native Apps	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE175	2021	Micraspis: A Computer-Aided Proposal Toward Programming and Architecting Smart IoT Wearables	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE177	2021	Making the Case for Centralized Automotive E/E Architectures	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE178	2009	Knowledge-Based Integration Between Virtual and Physical Prototyping for Identifying Behavioral Constraints of Embedded Real-Time Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE179	2008	A Novel Practical Convergence Service Model for Next Generation Networks	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE180	2021	Convergence of SoC architecture and semiconductor manufacturing through AI/ML systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE181	2023	Moore,Äôs Law: Accelerating the Pace of Innovation A Future Look into Ocean Observing Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE182	2013	Exploiting Template Metaprogramming to customize an object-oriented operating system	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE184	2011	Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE185	2023	Design Pattern for Industrial Control Applications Based on One-Line IEC 61499 Adapter Connections	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE187	2022	A High-level Model to Leverage NoC-based Many-core Research	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE188	2018	Software Engineering Practices for the Smart Mobility Market	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE189	2015	A hypervisor approach with real-time support to the MIPS M5150 processor	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE191	2001	Toward an all-IP-based UMTS system architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE192	2011	An optimization process for adaptation space exploration of service-oriented applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE193	2024	Furthering Moore,Äôs Law Integration Benefits in the Chiplet Era	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE195	2014	Virtual CAN Lines in an Integrated MPSoC Architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE196	2015	Declaration language-based graphical user interface	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE197	2017	Development and Application of a Real-Time Test Bed for Cyber,ÄìPhysical System	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE198	2019	Open Cognitive Control System Architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE200	2009	A Portable Feather-Weight Java-Based Graphic Library for Embedded Systems on Java Processor	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE201	2006	Case Study of Product Line Engineering in Insurance Company	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE203	2005	Understanding cloned patterns in Web applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE204	2015	A Platform for Context-Aware Application Development: PCAD	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE206	2009	A Hardware Abstraction Layer for Integrating Real-Time and General-Purpose with Minimal Kernel Modification	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE210	2023	Dealing with Anti-Patterns When Migrating from Monoliths to Microservices: Challenges and Research Directions	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE211	2012	Lessons from VAX/SVS for High-Assurance VM Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE213	2021	Cloud-Native Network Slicing Using Software Defined Networking Based Multi-Access Edge Computing: A Survey	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE214	2019	Firmware Synthesis for Ultra-Thin IoT Devices Based on Model Integration	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE216	2012	MPSoCDK: A framework for prototyping and validating MPSoC projects on FPGAs	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE217	2021	Design and Implementation of an Ethernet-Based Linear Motor Drive for Industrial Transport Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE219	2014	A Transformation Approach to Enact the Design-Time Simulation of BPMN Models	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE225	2003	Design of an efficient distributed GIS application	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE228	2023	Statistical Analysis of Remote Health Monitoring Based IoT Security Models & Deployments From a Pragmatic Perspective	SI	SI	NO		NO	NO	NO	NO	NO	NO
IEEE231	2002	Principles of multi-level reflection for fault tolerant architectures	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE233	2011	A Reconfigurable Computing System Based on a Cache-Coherent Fabric	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE234	2023	Architecture of the Graphics System for Embedded Real-Time Operating Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE237	2021	Digital Engineering Hub Pathfinder	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE238	2010	R3TOS: A reliable reconfigurable real-time operating system	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE239	2001	Mobile management of network files	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE240	2009	Auction Service Discovery Model - An Agent Based Approach	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE242	2011	A Review of Aeronautical Electronics and Its Parallelism With Automotive Electronics	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE247	2024	A Survey on Beyond 5G Network Slicing for Smart Cities Applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE249	2005	Rapid embedded control prototyping by Petri nets	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE251	2020	Migrating Application from Monolith to Microservices	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE252	2015	Partial Updates of AUTOSAR Embedded Applications -- To What Extent?	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE256	2012	A Survey of SOA Technologies in NGN Network Architectures	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE257	2012	A customizable and ARINC 653 quasi-compliant hypervisor	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE258	2009	Beyond multiplayer games: Engaging cognitive virtual partners in collaborative decision-making and problem-solving situations	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE259	2023	Decision as a Service for Transaction Banking Using Service-Oriented Modeling Architecture Methodology	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE260	2019	Exploring Microservices as the Architecture of Choice for Network Function Virtualization Platforms	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE261	2012	Sky Computing Platform for Legacy Distributed Application	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE262	2021	Journey toward software-defined passive optical networks with multi-PON technology: an industry view [Invited]	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE264	2019	Virtualization of Residential Gateways: A Comprehensive Survey	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE265	2016	A Kernel for embedded systems development and simulation using the Boost library	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE268	2018	A Novel SDN-based Architecture and Traffic Steering Method for Service Function Chaining	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE269	2022	Can We Put the ‚ÄúS‚Äù Into IoT?	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE272	2020	Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE275	2018	Routing in Fog-Enabled IoT Platforms: A Survey and an SDN-Based Solution	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE276	2011	TSaaS ‚Äì Customized telecom app hosting on cloud	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE278	2022	Container Based Scalability and Performance Analysis of Multitenant SaaS Applications	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE280	2007	Auto-adaptive reconfigurable architecture for scalable multimedia applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE281	2007	Platform concept: A breakthrough in surface radar architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE282	2015	Impact of Interdisciplinary Research on Planning, Running, and Managing Electromobility as a Smart Grid Extension	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE283	2012	Remote health monitoring by OSGi technology and digital TV integration	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE284	2006	Web Services for Production Planning and Optimization	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE287	2001	Applying a mediator architecture employing XML to retail inventory control	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE290	2008	An implementation of context-aware partner selection to support supply chain collaborations	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE291	2010	Embedded systems' virtualization: The next challenge?	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE293	2014	Network Security Function Virtualization(NSFV) towards Cloud computing with NFV Over Openflow infrastructure: Challenges and novel approaches	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE294	2024	On the Interplay of Artificial Intelligence and Space-Air-Ground Integrated Networks: A Survey	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE297	2022	Measurement Campaign on 5G Indoor Millimeter Wave and Visible Light Communications Multi Component Carrier System	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE298	2021	Cyber Physical Systems Architecture for Collaborative Services	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE300	2021	A Survey of Wireless Networks for Future Aerial Communications (FACOM)	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE302	2020	Laying the Foundations for an IoT Reference Architecture for Agricultural Application Domain	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE304	2013	What?s New in Intelligent Transportation Systems?: An Overview of European Projects and Initiatives	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE305	2022	Profit Maximizing Smart Manufacturing Over AI-Enabled Configurable Blockchains	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE308	2017	Concept for AutomationML-based interoperability between multiple independent engineering tools without semantic harmonization: Experiences with AutomationML	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE309	2003	Processing and scheduling components in an innovative network processor architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE310	2018	Architecture Simplification at Large Institutions using Micro Services	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
IEEE311	2011	RACME: A Framework to Support V&V and Certification	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE313	2018	360¬∞ Integration Gateway Approach Applied in a Multi-Layer Cloud Management Platform for Internet TV Applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE317	2010	Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE318	2024	HELICS: A Co-Simulation Framework for Scalable Multi-Domain Modeling and Analysis	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE325	2008	Abstracts by ID	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE326	2013	Flexible collaboration and control of heterogeneous mechatronic devices and systems by means of an event-driven, SOA-based automation concept	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE327	2021	Sensor Cloud Frameworks: State-of-the-Art, Taxonomy, and Research Issues	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE328	2007	A Verifiable Language for Programming Real-Time Communication Schedules	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE329	2019	Adopting SOA and Microservices for Inter-enterprise Architecture in SME Communities	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE330	2006	Domain management of IMS	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE338	2020	ML-based Data Anomaly Mitigation and Cyber-Power Transmission Resiliency Analysis	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE339	2024	Securing the Shared Kernel: Exploring Kernel Isolation and Emerging Challenges in Modern Cloud Computing	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE340	2016	Docker container security via heuristics-based multilateral security-conceptual and pragmatic study	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE341	2011	Nano-CF: A coordination framework for macro-programming in Wireless Sensor Networks	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE343	2012	Digital Control of Power Converters,ÄîA Survey	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE344	2024	Drawing the Boundaries Between Blockchain and Blockchain-Like Systems: A Comprehensive Survey on Distributed Ledger Technologies	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE348	2005	Large scale multiagent simulation on the grid	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE355	2013	IMS: The New Generation of Internet-Protocol-Based Multimedia Services	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE359	2003	The performance and energy consumption of embedded real-time operating systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE360	2014	A communicable disease prediction benchmarking platform	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE361	2024	Contextualizing Formal Verification for Systems Security Engineering	SI	SI	NO	-	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
IEEE362	2020	Business Process Management and Service Oriented Architecture Integration for Transactional Banking Application	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE364	2025	Distributed Energy Resource Management System (DERMS) Cybersecurity Scenarios, Trends, and Potential Technologies: A Review	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE365	2007	The Operator's Response to P2P Service Demand	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE366	2024	PenChain: A Blockchain-Based Platform for Penalty-Aware Service Provisioning	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE367	2019	An Architecture for Delivering Graphical Web Applications in Constrained IoT Devices	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE371	2019	Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE372	2012	From Delay-Tolerant Networks to Vehicular Delay-Tolerant Networks	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE373	2019	Edge Computing and Networking: A Survey on Infrastructures and Applications	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE374	2022	Data Collection in Studies on Internet of Things (IoT), Wireless Sensor Networks (WSNs), and Sensor Cloud (SC): Similarities and Differences	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE375	2022	An Extensive Blockchain Based Applications Survey: Tools, Frameworks, Opportunities, Challenges and Solutions	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE376	2012	Transparent Synchronization Protocols for Compositional Real-Time Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE383	2013	Book of abstracts	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
IEEE393	2025	Compounded real-time operating systems for rich real-time applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
IEEE398	2009	The best of both worlds: Merging the benefits of Rack&Stack and universal ATE	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM001	2011	A critical review of applied MDA for embedded devices: identification of problem classes and discussing porting efforts in practice	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM002	2019	A generic traceability metamodel for enabling unified end-to-end traceability in software product lines	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM003	2016	A genetic approach to architectural pattern discovery	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM004	2015	A holistic optimization approach for the synthesis of AUTOSAR E/E architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM005	2004	A multimedia workflow-based collaborative engineering environment for oil & gas industry	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM006	2024	A proposed catalog of development patterns for fault-tolerant microservices	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM007	2020	A secure hardware-software solution based on RISC-V, logic locking and microkernel	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM008	2024	Accelerating the software development process through the application of MVP and monolithic architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM010	2019	ACRN: a big little hypervisor for IoT development	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM013	2005	AgentSteel: an agent-based online system for the planning and observation of steel production	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM016	2018	An automatic extraction approach: transition to microservices architecture from monolithic application	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
ACM017	2008	An empirical investigation of software reuse benefits in a large telecom product	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM018	2018	An experience report on the adoption of microservices in three Brazilian government institutions	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM019	2007	An online help framework for web applications	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM020	2010	Application frameworks: how they become your enemy	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM032	2016	Assessing Dependability with Software Fault Injection: A Survey	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM033	2007	Assessing the influence on processes when evolving the software architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM036	2023	Beyond Smart Homes: An In-Depth Analysis of Smart Aging Care System Security	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM037	2024	Bridge: A Leak-Free Hardware-Software Architecture for Parallel Embedded Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM038	2009	Building the Senceive System	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM044	2024	Contemporary Software Modernization: Strategies, Driving Forces, and Research Opportunities	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM045	2020	Continuous Data-driven Software Engineering - Towards a Research Agenda: Report on the Joint 5th International Workshop on Rapid Continuous Software Engineering (RCoSE 2019) and 1st International Works	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM049	2019	Demystifying Arm TrustZone: A Comprehensive Survey	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM052	2021	Emergent design challenges for embedded systems and paths forward: mixed-criticality, energy, reliability and security perspectives	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM053	2023	Empirical evidence on technical challenges when adopting continuous practices	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM058	2005	Frontmatter (TOC, Letters, Election results, Software Reliability Resources!, Computing Curricula 2004 and the Software Engineering Volume SE2004, Software Reuse Research, ICSE 2005 Forward)	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM059	2005	Frontmatter (TOC, Letters, Philosophy of computer science, Interviewers needed, Taking software requirements creation from folklore to analysis, SW components and product lines: from business to systems and technology, Software engineering survey)	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM064	2007	Grand challenges in embedded software	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM065	2022	Hacking or Engineering? Towards an Extended Entrepreneurial Software Engineering Model	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM066	2014	Hardware-software collaboration for secure coexistence with kernel extensions	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM068	2014	How do professionals perceive legacy systems and software modernization?	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM093	2002	Impact of the research community on the field of software configuration management: summary of an impact project report	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM094	2014	Improving adaptiveness of AUTOSAR embedded applications	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM095	2022	In three steps to software product lines: a practical example from the automotive industry	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM103	2023	Leading a Software Architecture Revolution - "Part 1: Creating Awareness, Preparing and Measuring"	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM105	2016	Metamodeling of reference software architecture and automatic code generation	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
ACM106	2018	Microservice architecture in industrial software delivery on edge devices	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM107	2023	Microservice Architecture Patterns for Enterprise Applications Supporting Business Agility	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM110	2019	NetDIMM: Low-Latency Near-Memory Network Interface Architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM111	2010	Performance study of active tracking in a cellular network using a modular signaling platform	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM113	2023	Placement of Microservices-based IoT Applications in Fog Computing: A Taxonomy and Future Directions	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM114	2015	Pluggable Systems as Architectural Pattern: An Ecosystemability Perspective	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM115	2015	Process model for rapid implementation of features using flexible architecture	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM117	2018	Protocol emulation platform based on microservice architecture for underwater acoustic networks	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM118	2021	Real-time Data Infrastructure at Uber	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM130	2010	SE-155 DBSA: a device-based software architecture for data mining	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM135	2021	Software Architectural Migration: An Automated Planning Approach	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM137	2009	Taming subsystems: capabilities as universal resource access control in L4	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM138	2023	The Core Industry Manufacturing Process of Electronics Assembly Based on Smart Manufacturing	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM140	2010	Think big for reuse	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM141	2011	Towards a lean-government using new IT-architectures for compliance monitoring	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM142	2021	Towards an Anatomy of Software Craftsmanship	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
ACM143	2017	Towards Dynamic Cross-platform Component-driven Applications with Girders Elements Framework	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
ACM146	2023	Using Database Schemas of Legacy Applications for Microservices Identification: A Mapping Study	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
ACM147	2024	Vertically Autoscaling Monolithic Applications with CaaSPER: Scalable Container-as-a-Service Performance Enhanced Resizing Algorithm for the Cloud	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SCO001	2012	A system-level modeling methodology for performance-driven component selection in multicore architectures	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SCO002	2008	Dependability assessment for the selection of embedded cores	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SCO003	2024	Fault Tolerance Enhancement in Microservices using Orchestration Process with Agile	SI	SI	-	-	NO	NO	NO	SI	NO	NO
SCO005	2023	Paradigm Shift from Monolithic to Microservices	SI	SI	-	-	NO	NO	NO	SI	NO	NO
SCO006	2021	Legacy Digital Transformation: TCO and ROI Analysis	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SCO008	2018	An architectural view to computer integrated manufacturing systems based on Axiomatic Design Theory	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SCO010	2015	Designing applications for heterogeneous many-core architectures with the FlexTiles Platform	SI	SI	-	-	NO	NO	NO	SI	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
SPI001	2024	Modelling the quantification of requirements technical debt	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI002	2022	Open-source software product line extraction processes: the ArgoUML-SPL and Phaser cases	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI003	2023	Product representation via networks methodology for exposing project risks	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI005	2022	Improving datacenter utilization through containerized service-based architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI006	2024	Towards a seamless data cycle for space components: considerations from the growing European future digital ecosystem Gaia-X	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI011	2023	An empirical study of the systemic and technical migration towards microservices	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI012	2021	A microservices persistence technique for cloud-based online social data analysis	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI015	2021	Serverless and Containerization Models and Methods in Challenger Banks Software	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI016	2022	Comparison of open-source runtime testing tools for microservices	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI017	2024	ChamelIoT: a tightly- and loosely-coupled hardware-assisted OS framework for low-end IoT devices	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI018	2024	Black widow optimization algorithm for efficient task assignment in cloud computing	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI020	2021	The Migration Journey Towards Microservices	SI	SI			NO	NO	NO	NO	NO	NO
SPI021	2023	ReNo: novel switch architecture for reliability improvement of NoCs	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI022	2023	A location-based fog computing optimization of energy management in smart buildings: DEVS modeling and design of connected objects	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI023	2022	Designing and developing smart production planning and control systems in the industry 4.0 era: a methodology and case study	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI024	2023	Cloud manufacturing adoption: a comprehensive review	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI025	2021	Building a Big Data Oriented Architecture for Enterprise Integration	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI026	2023	A low-cost intelligent tracking system for clothing manufacturers	SI	SI			NO	NO	NO	NO	NO	NO
SPI028	2019	A versatile hardware/software platform for personalized driver assistance based on online sequential extreme learning machines	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI029	2020	The Quest for Introducing Technical Debt Management in a Large-Scale Industrial Company	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI030	2019	Designing smart city mobile applications	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI031	2023	Abstracts of the 12th DACH+ Conference on Energy Informatics 2023	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI032	2020	Architectural Concerns for Digital Twin of the Organization	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI033	2019	Strategies Reported in the Literature to Migrate to Microservices Based Architecture	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI034	2019	A near-infrared camera for iRobo-AO on the IUCAA 2-m telescope	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI035	2019	Trustworthy Isolation of DMA Enabled Devices	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI036	2018	Challenges When Moving from Monolith to Microservice Architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
SPI037	2018	RETRACTED ARTICLE: Research on Monitoring Platform of Agricultural Product Circulation Efficiency Supported by Cloud Computing	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI038	2019	RETRACTED ARTICLE: Agricultural product monitoring system supported by cloud computing	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI039	2020	Digital twin-based cyber physical production system architectural framework for personalized production	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI040	2017	Development of an industrial Internet of things suite for smart factory towards re-industrialization	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI041	2017	Enabling Legacy Applications for Multi-tenancy Without Reengineering	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI042	2017	Using clustering approaches for response time aware job scheduling model for internet of things (IoT)	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI043	2016	Migrating to Cloud-Native Architectures Using Microservices: An Experience Report	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI044	2016	On the use of models for high-performance scientific computing applications: an experience report	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI047	2016	Implementing an Event-Driven Enterprise Information Systems Architecture: Adoption Factors in the Example of a Micro Lending Case Study	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI048	2015	Biofuels for Combustion Engines	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI049	2015	A Service Oriented Architecture for Total Manufacturing Enterprise Integration	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI050	2015	Interoperability Between Machine-to-Machine Communication System and IP Multimedia Subsystem	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI051	2015	Implementing Architectural Thinking	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI052	2013	Connecting and Managing M2M Devices in the Future Internet	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI053	2013	Designing a flood forecasting and inundation-mapping system integrated with spatial data infrastructures for Turkey	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI054	2014	Similar Concepts, Distinct Solutions, Common Problems: Learning from PLM and BIM Deployment	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI055	2013	Managing Run-Time Variability in Robotics Software by Modeling Functional and Non-functional Behavior	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI056	2013	Towards Virtualization Concepts for Novel Automotive HMI Systems	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI057	2013	An extensible six-step methodology to automatically generate fuzzy DSSs for diagnostic applications	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI058	2012	Mining the Traffic Cloud: Data Analysis and Optimization Strategies for Cloud-Based Cooperative Mobility Management	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI059	2012	Towards Model-Based System Engineering for Simulation-Based Design in Product Data Management Systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI060	2012	A new classification approach for neural networks hardware: from standards chips to embedded systems on chip	SI	SI	NO	-	NO	NO	NO	NO	NO	NO

ID	Año	Título	CI1	CI2	CI3	CI4	CE1	CE2	CE3	CE4	CE5	CE6
SPI065	2010	Data exchange and multi-layered architecture for a collaborative design process in virtual environments	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI067	2009	Enhancing a dependable multiserver operating system with temporal protection via resource reservations	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI069	2009	Formal specification of non-functional properties of component-based software systems	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI072	2009	Constructing a Multi-OS Platform with Minimal Engineering Cost	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI073	2007	Automatic generation of PLC automation projects from component-based models	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI074	2007	Quality, productivity and economic benefits of software reuse: a review of industrial studies	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI077	2007	Adaptive Scheduling for Real-Time Network Traffic Using Agent-Based Simulation	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI078	2006	Global Grids – Making a Case for Self-organization in Large-Scale Overlay Networks	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI083	2006	A Meshing Tool Product Line Architecture	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
SPI084	2005	An Investigation of a Method for Identifying a Software Architecture Candidate with Respect to Quality Attributes	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI086	2005	Component-based development of Web-enabled eHome services	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI087	2005	Using Software Agents to Personalize Access to E-Offices	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI088	2005	An Engineering Approach to Cooperating Agents for Distributed Information Systems	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI091	2004	Assessing agile methods: An empirical study	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
SPI096	2002	Software radio: The challenges for reconfigurable terminals	SI	SI	NO	-	NO	NO	NO	NO	NO	NO
SPI104	2002	Distributed Video Production: Tasks, Architecture and QoS Provisioning	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO

ANEXO C. LISTADO DETALLADO DE ESTUDIOS PRIMARIOS

ID	Fuente	Año	País	Titulo	Autores	Tipo
ACM003	ACM Digital Library	2016	Países Bajos	A genetic approach to architectural pattern discovery	Joeri Peters, Jan Martijn E. M. van der Werf	Conference paper
ACM008	ACM Digital Library	2024	Brasil	Accelerating the software development process through the application of MVP and monolithic architecture	Rayfran Rocha Lima; Alexandre Teixeira da Silva	Conference paper
ACM017	ACM Digital Library	2008	Noruega	An empirical investigation of software reuse benefits in a large telecom product	Parastoo Mohagheghi, Reidar Conradi	Article-Journal
ACM018	ACM Digital Library	2018	Brasil	An experience report on the adoption of microservices in three Brazilian government institutions	Welder Luz, Everton Agilar, Marcos César de Oliveira, Carlos Eduardo R. de Melo, Gustavo Pinto, Rodrigo Bonifácio	Conference paper
ACM019	ACM Digital Library	2007	Corea del Sur	An online help framework for web applications	KwangChun Lee, Dan Hyung Lee	Conference paper
ACM033	ACM Digital Library	2007	Suecia	Assessing the influence on processes when evolving the software architecture	Stig Larsson, Anders Wall, Peter Wallin	Conference paper
ACM044	ACM Digital Library	2024	Estados Unidos	Contemporary Software Modernization: Strategies, Driving Forces, and Research Opportunities	Wesley K. G. Assunção, Luciano Marchezan, Lawrence Arkoh, Alexander Egyed, Rudolf Ramler	Article-Journal
ACM095	ACM Digital Library	2022	Alemania	In three steps to software product lines: a practical example from the automotive industry	Matthias Eggert, Karsten Günther, Jochen Maletschek, Alexandru Maxiniuc, Alexander Mann-Wahrenberg	Conference paper
ACM135	ACM Digital Library	2021	Tailandia	Software Architectural Migration: An Automated Planning Approach	Nacha Chondamrongkul, Jing Sun, Ian Warren	Article-Journal
ACM146	ACM Digital Library	2023	Grecia	Using Database Schemas of Legacy Applications for Microservices Identification: A Mapping Study	Antonios Mparmpoutis, George Kakarontzas	Conference paper
ACM147	ACM Digital Library	2024	Estados Unidos	Vertically Autoscaling Monolithic Applications with CaaSPER: Scalable Container-as-a-Service Performance Enhanced Resizing Algorithm for the Cloud	Anna Pavlenko, Joyce Cahoon, Yiwen Zhu, Brian Kroth, Michael Nelson, Andrew Carter, David Liao, Travis Wright, Jesús Camacho-Rodríguez, Karla Saur	Conference paper
IEEE007	IEEE Xplore	2023	India	Paradigm Shift From Monolithic to Microservices	Divarshana Saxena; Biswajit Bhowmik	Conference paper
IEEE009	IEEE Xplore	2009	Alemania	Trends in Embedded Software Engineering	Peter Liggesmeyer; Mario Trapp	Article-Journal
IEEE010	IEEE Xplore	2008	Italia	Configuration and Change Management of the Outcomes of an Automotive Engine Control Model Based Software Design Process	Stefano Monti; Walter Nesci; Serino Angellotti; Claudio Schellino; Massimo Seminara; Rainer Wuesthenagen	Conference paper
IEEE015	IEEE Xplore	2004	Finlandia	Establishing a software architecting environment	C. Riva; P. Selonen; T. Systa; A.-P. Tuovinen; Jianli Xu; Yaojin Yang	Conference paper

ID	Fuente	Año	País	Titulo	Autores	Tipo
IEEE018	IEEE Xplore	2005	Países Bajos	Software reuse in product populations	R. van Ommering	Article-Journal
IEEE019	IEEE Xplore	2014	Brasil	Test logic reuse through unit test patterns a test automation framework for software product lines	Glauco Silva Neves; Patrícia Vilain	Conference paper
IEEE031	IEEE Xplore	2004	Japón	An extensible monolithic software design approach for Internet-convergent embedded applications	T. Yamakam	Conference paper
IEEE034	IEEE Xplore	2019	Suiza	Virtualizing Embedded Firmware to Boost Innovation Cycles	Stephan Sehestedt; Georgia Giannopoulou; Aurélien Monot; Michael Wahler	Conference paper
IEEE035	IEEE Xplore	2004	Finlandia	UML-based reverse engineering and model analysis approaches for software architecture maintenance	C. Riva; P. Selonen; T. Systa; Jianli Xu	Conference paper
IEEE037	IEEE Xplore	2005	España	Using MAP for Recovering the Architecture of Web Systems of a Spanish Insurance Company	R. Capilla	Conference paper
IEEE038	IEEE Xplore	2007	Australia	Reconfigurable Software Architecture for Voice Access to Data Services	Alex Talevski; Elizabeth Chang	Conference paper
IEEE040	IEEE Xplore	2024	Países Bajos	Software Architecture for Agricultural Robots: Systems, Requirements, Challenges, Case Studies, and Future Perspectives	Rekha Raja	Article-Journal
IEEE045	IEEE Xplore	2006	Alemania	The feature-architecture mapping (FArM) method for feature-oriented development of software product lines	P. Sochos; M. Riebisch; I. Philippow	Conference paper
IEEE047	IEEE Xplore	2023	Francia	A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems	David Fernández Blanco; Frédéric Le Mouël; Trista Lin; Marie-Pierre Escudié	Article-Journal
IEEE050	IEEE Xplore	2018	Sri Lanka	Feature Oriented Software Development Framework for Stock Exchange Systems	Lasitha Konara; Indika Perera; Sujith Gunewardhane	Conference paper
IEEE054	IEEE Xplore	2024	Eslovenia	Analysis of Methodologies and Tools for Software Development in Different Architectures	Maksim Nikitashin; Marin Kaluža; Borut Werber	Conference paper
IEEE055	IEEE Xplore	2008	Suecia	Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study	Hongyu Pei Breivold; Ivica Crnkovic; Rikard Land; Magnus Larsson	Conference paper
IEEE057	IEEE Xplore	2009	Finlandia	Linking GENESYS application architecture modelling with platform performance simulation	Subayal Khan; Susanna Pantsar-Syväniemi; Jari Kreku; Kari Tiensyrjä; Juha-Pekka Soininen	Article-Journal
IEEE060	IEEE Xplore	2020	Alemania	Microservice Decomposition via Static and Dynamic Analysis of the Monolith	Alexander Krause; Christian Zirkelbach; Wilhelm Hasselbring; Stephan Lenga; Dan Kröger	Conference paper
IEEE065	IEEE Xplore	2023	Indonesia	Analyzing Microservices and Monolithic Systems: Key Factors in Architecture, Development, and Operations	Juan Christian; Steven; Afdhal Kurniawan; Maria Susan Anggreainy	Conference paper
IEEE066	IEEE Xplore	2024	Jordania	Microservices Architecture for Improved Maintainability and Traceability in MVC-Based E-	Safa'a Bataieneh; Ali Ziadeh; Lamis F. Al-Qora'n	Conference paper

ID	Fuente	Año	País	Titulo	Autores	Tipo
				Learning Platforms: RoadMap for Future Developments		
IEEE068	IEEE Xplore	2017	Francia	From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture	Jean-Philippe Gouigoux; Dalila Tamzalit	Conference paper
IEEE071	IEEE Xplore	2011	Grecia	Lessons from Space	Diomidis Spinellis; Henry Spencer	Conference paper
IEEE072	IEEE Xplore	2007	Estados Unidos	Innovation in Large-Grained Software Reuse: A Case from Banking	Paul D. Witman; Terry Ryan	Conference paper
IEEE074	IEEE Xplore	2005	Japón	A micro-component architecture approach for next generation embedded browsers	T. Yamakami	Conference paper
IEEE077	IEEE Xplore	2004	Suecia	On the expected synergies between component-based software engineering and best practices in product integration	S. Larsson; I. Crnkovic; F. Ekdahl	Conference paper
IEEE086	IEEE Xplore	2007	Alemania	Introducing Impact Analysis for Architectural Decisions	Matthias Riebisch; Sven Wohlfarth	Conference paper
IEEE088	IEEE Xplore	2018	China	Architecture Patterns, Quality Attributes, and Design Contexts: How Developers Design with Them	Tingting Bi; Peng Liang; Antony Tang	Conference paper
IEEE095	IEEE Xplore	2020	Países Bajos	Product Line Architecture Design of Software-Intensive Physical Protection Systems	Bedir Tekinerdoğan; İskender Yakın; Sevil Yağız; Kaan Özcan	Conference paper
IEEE099	IEEE Xplore	2023	India	Microservices based Ticket Booking Platform deployed on Cloud	Prerit Munjal; Durgaprasad Gangodkar; Yogesh Lohumi	Conference paper
IEEE103	IEEE Xplore	2023	India	From Monolith to Service-Oriented Architecture: A Model-Based Design Approach towards Software-Defined Vehicles	Sreeraj Arole	Conference paper
IEEE107	IEEE Xplore	2023	Macedonia	Migrating to a microservice architecture: benefits and challenges	Sh. Salii; J. Ajdari; Xh. Zenuni	Conference paper
IEEE108	IEEE Xplore	2021	Austria	Quality Assurance for Microservice Architectures	Thomas Schirgi; Eugen Brenner	Conference paper
IEEE112	IEEE Xplore	2007	Corea del Sur	A Reengineering Approach of the Legacy System in the Digital Media Domain	Hyunmin Ko; Gihun Chang; Kangtae Kim	Conference paper
IEEE114	IEEE Xplore	2021	Serbia	Partial software update on the AURIX TC397 platform	Barbara Matijević; Marijan Herceg; Milena Milošević; Nenad Četić	Conference paper
IEEE116	IEEE Xplore	2007	Estados Unidos	When Legacy Meets SOA: Achieving Business Agility by Integrating New Technology with Existing Software Asset	Pasquale Iocola	Conference paper
IEEE119	IEEE Xplore	2024	Estados Unidos	Systematic Analysis of Factors Influencing Modulith Architecture Adoption over Microservices	Chandra Prakash; Sunil Arora	Conference paper

ID	Fuente	Año	País	Titulo	Autores	Tipo
IEEE146	IEEE Xplore	2013	Grecia	ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud	Andreas Menychtas; Christina Santzaridou; George Kousiouris; Theodora Varvarigou; Leire Orue-Echevarria; Juncal Alonso	Conference paper
IEEE152	IEEE Xplore	2024	Kosovo	Optimizing HR Monolithic Systems to Modern HR Systems using Microservices Architecture	Isak Shabani; Nderon Hiseni; Dhuratë Hyseni; Betim Çiço	Conference paper
IEEE155	IEEE Xplore	2010	Estados Unidos	Radar Open System Architecture provides net centricity	John A. Nelson	Magazines
IEEE157	IEEE Xplore	2001	Malasia	The application of component-based methodology in developing visual power system analysis tool	K.M. Nor; T.A. Gani; H. Mokhlis	Conference paper
IEEE159	IEEE Xplore	2024	Brasil	On the Usefulness of Automatically Generated Microservice Architectures	Luiz Carvalho; Thelma Elita Colanzi; Wesley K. G. Assunção; Alessandro Garcia; Juliana Alves Pereira; Marcos Kalinowski	Article-Journal
IEEE171	IEEE Xplore	2024	Brasil	Insights on Microservice Architecture Through the Eyes of Industry Practitioners	Vinicius L. Nogueira; Fernando S. Felizardo; Aline M. M. M. Amaral; Wesley K. G. Assunção; Thelma E. Colanzi	Conference paper
IEEE174	IEEE Xplore	2021	Noruega	Migrating Monoliths to Microservices-based Customizable Multi-tenant Cloud-native Apps	Sindre Grønstøl Haugeland; Phu H. Nguyen; Hui Song; Franck Chauvel	Conference paper
IEEE177	IEEE Xplore	2021	Canadá	Making the Case for Centralized Automotive E/E Architectures	Victor Bandur; Gehan Selim; Vera Pantelic; Mark Lawford	Article-Journal
IEEE198	IEEE Xplore	2019	Alemania	Open Cognitive Control System Architecture	Zohra Manjiyani; Chris Paul Iatrou; Markus Graube; Leon Urbas	Conference paper
IEEE201	IEEE Xplore	2006	Corea del Sur	Case Study of Product Line Engineering in Insurance Company	Jeong Ah Kim	Conference paper
IEEE210	IEEE Xplore	2023	Marruecos	Dealing with Anti-Patterns When Migrating from Monoliths to Microservices: Challenges and Research Directions	Hassan Farsi; Driss Allaki; Abdeslam En-nouaary; Mohamed Dahchour	Conference paper
IEEE231	IEEE Xplore	2002	Francia	Principles of multi-level reflection for fault tolerant architectures	F. Taiani; J.-C. Fabre; M.-O. Killijian	Conference paper
IEEE251	IEEE Xplore	2020	Indonesia	Migrating Application from Monolith to Microservices	Teguh Prasandy; Titan; Dina Fitria Murad; Taufik Darwis	Conference paper
IEEE252	IEEE Xplore	2015	Francia	Partial Updates of AUTOSAR Embedded Applications -- To What Extent?	Hélène Martorell; Jean-Charles Fabre; Michael Lauer; Matthieu Roy; Régis Valentin	Conference paper
IEEE259	IEEE Xplore	2023	Indonesia	Decision as a Service for Transaction Banking Using Service-Oriented Modeling Architecture Methodology	Ford Lumban Gaol; Grahita Djati Nugraha; Tokuro Matsuo	Article-Journal
IEEE260	IEEE Xplore	2019	Canadá	Exploring Microservices as the Architecture of Choice for Network Function Virtualization Platforms	Hassan Hawilo; Manar Jammal; Abdallah Shami	Magazines
IEEE261	IEEE Xplore	2012	Rumania	Sky Computing Platform for Legacy Distributed Application	Silviu Panica; Dana Petcu; Iñigo Lazkanotegi Larrate; Tam's M'hr	Conference paper

ID	Fuente	Año	País	Titulo	Autores	Tipo
IEEE272	IEEE Xplore	2020	España	Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League	José Carlos González; Ángel García-Olaya; Fernando Fernández	Conference paper
IEEE276	IEEE Xplore	2011	India	TSaaS - Customized telecom app hosting on cloud	Subhankar Pal; Tirthankar Pal	Conference paper
IEEE278	IEEE Xplore	2022	India	Container Based Scalability and Performance Analysis of Multitenant SaaS Applications	Poonam Mangwani; Vrinda Tokekar	Conference paper
IEEE310	IEEE Xplore	2018	India	Architecture Simplification at Large Institutions using Micro Services	A. Premchand; A. Choudhry	Conference paper
SCO006	Scopus	2021	India	Legacy Digital Transformation: TCO and ROI Analysis	Mallidi, Ravi Kiran; Sharma, Manmohan; Singh, Jagjit	Article-Journal
SPI002	Springer Link	2022	Brasil	Open-source software product line extraction processes: the ArgoUML-SPL and Phaser cases	Rodrigo André Ferreira Moreira, Wesley K. G. Assunção, Jabier Martinez, Eduardo Figueiredo	Article-Journal
SPI005	Springer Link	2022	Estados Unidos	Improving datacenter utilization through containerized service-based architecture	Aos Mulahuwaish, Shane Korbel, Basheer Qolomany	Article-Journal
SPI011	Springer Link	2023	Suecia	An empirical study of the systemic and technical migration towards microservices	Hamdy Michael Ayas, Philipp Leitner, Regina Hebig	Conference paper
SPI015	Springer Link	2021	Ucrania	Serverless and Containerization Models and Methods in Challenger Banks Software	Yuliia Kuznetsova, Artem Kolomytsev, Maksym Somochkin, Oleksandr Vdovitchenko	Conference paper
SPI016	Springer Link	2022	Estados Unidos	Comparison of open-source runtime testing tools for microservices	Juan P. Sotomayor, Sai Chaithra Allala, Dionny Santiago, Tariq M. King, Peter J. Clarke	Article-Journal
SPI020	Springer Link	2021	Suecia	The Migration Journey Towards Microservices	Hamdy Michael Ayas, Philipp Leitner, Regina Hebig	Conference paper
SPI025	Springer Link	2021	Vietnam	Building a Big Data Oriented Architecture for Enterprise Integration	Le Hoang Nam, Phan Duy Hung	Conference paper
SPI036	Springer Link	2018	Finlandia	Challenges When Moving from Monolith to Microservice Architecture	Miika Kalske, Niko Mäkitalo, Tommi Mikkonen	Conference paper
SPI041	Springer Link	2017	Alemania	Enabling Legacy Applications for Multi-tenancy Without Reengineering	Uwe Hohenstein, Preeti Koka	Conference paper
SPI083	Springer Link	2006	Chile	A Meshing Tool Product Line Architecture	María Cecilia Bastarrica, Nancy Hitschfeld-Kahler, Pedro O. Rossel	Conference paper

ANEXO D. RESULTADOS DE LA EVALUACIÓN DE CALIDAD

Código	Título	QA1	QA2	QA3	QA4	QA5	Puntaje	Resultado
ACM003	A genetic approach to architectural pattern discovery	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
ACM008	Accelerating the software development process through the application of MVP and monolithic architecture	SI	SI	SI	SI	SI	5	Aceptado
ACM017	An empirical investigation of software reuse benefits in a large telecom product	SI	SI	SI	SI	SI	5	Aceptado
ACM018	An experience report on the adoption of microservices in three Brazilian government institutions	SI	SI	NO	SI	SI	4	Aceptado
ACM019	An online help framework for web applications	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
ACM033	Assessing the influence on processes when evolving the software architecture	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
ACM044	Contemporary Software Modernization: Strategies, Driving Forces, and Research Opportunities	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
ACM095	In three steps to software product lines: a practical example from the automotive industry	SI	SI	NO	SI	SI	4	Aceptado
ACM135	Software Architectural Migration: An Automated Planning Approach	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
ACM146	Using Database Schemas of Legacy Applications for Microservices Identification: A Mapping Study	SI	SI	SI	SI	SI	5	Aceptado
ACM147	Vertically Autoscaling Monolithic Applications with CaaSPER: Scalable Container-as-a-Service Performance Enhanced Resizing Algorithm for the Cloud	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE007	Paradigm Shift From Monolithic to Microservices	SI	SI	NO	SI	SI	4	Aceptado
IEEE009	Trends in Embedded Software Engineering	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE010	Configuration and Change Management of the Outcomes of an Automotive Engine Control Model Based Software Design Process	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE015	Establishing a software architecting environment	SI	SI	NO	SI	SI	4	Aceptado
IEEE018	Software reuse in product populations	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE019	Test logic reuse through unit test patterns a test automation framework for software product lines	SI	SI	NO	SI	SI	4	Aceptado
IEEE031	An extensible monolithic software design approach for Internet-convergent embedded applications	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE034	Virtualizing Embedded Firmware to Boost Innovation Cycles	SI	SI	NO	SI	SI	4	Aceptado

Código	Título	QA1	QA2	QA3	QA4	QA5	Puntaje	Resultado
IEEE035	UML-based reverse engineering and model analysis approaches for software architecture maintenance	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE037	Using MAP for Recovering the Architecture of Web Systems of a Spanish Insurance Company	SI	SI	NO	SI	SI	4	Aceptado
IEEE038	Reconfigurable Software Architecture for Voice Access to Data Services	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE040	Software Architecture for Agricultural Robots: Systems, Requirements, Challenges, Case Studies, and Future Perspectives	SI	SI	NO	SI	SI	4	Aceptado
IEEE045	The feature-architecture mapping (FArM) method for feature-oriented development of software product lines	SI	SI	NO	SI	SI	4	Aceptado
IEEE047	A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE050	Feature Oriented Software Development Framework for Stock Exchange Systems	SI	SI	NO	SI	SI	4	Aceptado
IEEE054	Analysis of Methodologies and Tools for Software Development in Different Architectures	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE055	Analyzing Software Evolvability of an Industrial Automation Control System: A Case Study	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE057	Linking GENESYS application architecture modelling with platform performance simulation	SI	SI	NO	SI	SI	4	Aceptado
IEEE060	Microservice Decomposition via Static and Dynamic Analysis of the Monolith	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE065	Analyzing Microservices and Monolithic Systems: Key Factors in Architecture, Development, and Operations	SI	SI	NO	SI	SI	4	Aceptado
IEEE066	Microservices Architecture for Improved Maintainability and Traceability in MVC-Based E-Learning Platforms: RoadMap for Future Developments	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE068	From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture	SI	SI	NO	SI	SI	4	Aceptado
IEEE071	Lessons from Space	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE072	Innovation in Large-Grained Software Reuse: A Case from Banking	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE074	A micro-component architecture approach for next generation embedded browsers	SI	SI	NO	PARCIAL	SI	3,5	Aceptado
IEEE077	On the expected synergies between component-based software engineering and best practices in product integration	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE086	Introducing Impact Analysis for Architectural Decisions	SI	SI	PARCIAL	SI	SI	4,5	Aceptado

Código	Título	QA1	QA2	QA3	QA4	QA5	Puntaje	Resultado
IEEE088	Architecture Patterns, Quality Attributes, and Design Contexts: How Developers Design with Them	SI	SI	SI	SI	SI	5	Aceptado
IEEE095	Product Line Architecture Design of Software-Intensive Physical Protection Systems	SI	SI	NO	SI	SI	4	Aceptado
IEEE099	Microservices based Ticket Booking Platform deployed on Cloud	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE103	From Monolith to Service-Oriented Architecture: A Model-Based Design Approach towards Software-Defined Vehicles	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE107	Migrating to a microservice architecture: benefits and challenges	SI	SI	SI	SI	SI	5	Aceptado
IEEE108	Quality Assurance for Microservice Architectures	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE112	A Reengineering Approach of the Legacy System in the Digital Media Domain	SI	SI	NO	SI	SI	4	Aceptado
IEEE114	Partial software update on the AURIX TC397 platform	SI	SI	NO	SI	SI	4	Aceptado
IEEE116	When Legacy Meets SOA: Achieving Business Agility by Integrating New Technology with Existing Software Asset	SI	SI	NO	SI	SI	4	Aceptado
IEEE119	Systematic Analysis of Factors Influencing Modulith Architecture Adoption over Microservices	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE146	ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE152	Optimizing HR Monolithic Systems to Modern HR Systems using Microservices Architecture	SI	SI	NO	SI	SI	4	Aceptado
IEEE155	Radar Open System Architecture provides net centricity	SI	SI	NO	SI	SI	4	Aceptado
IEEE157	The application of component-based methodology in developing visual power system analysis tool	SI	SI	NO	SI	SI	4	Aceptado
IEEE159	On the Usefulness of Automatically Generated Microservice Architectures	SI	SI	SI	SI	SI	5	Aceptado
IEEE171	Insights on Microservice Architecture Through the Eyes of Industry Practitioners	SI	SI	SI	SI	SI	5	Aceptado
IEEE174	Migrating Monoliths to Microservices-based Customizable Multi-tenant Cloud-native Apps	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE177	Making the Case for Centralized Automotive E/E Architectures	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE198	Open Cognitive Control System Architecture	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE201	Case Study of Product Line Engineering in Insurance Company	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE210	Dealing with Anti-Patterns When Migrating from Monoliths to Microservices: Challenges and Research Directions	SI	SI	PARCIAL	SI	SI	4,5	Aceptado

Código	Título	QA1	QA2	QA3	QA4	QA5	Puntaje	Resultado
IEEE231	Principles of multi-level reflection for fault tolerant architectures	SI	SI	NO	PARCIAL	PARCIAL	3	Aceptado
IEEE251	Migrating Application from Monolith to Microservices	SI	SI	NO	SI	SI	4	Aceptado
IEEE252	Partial Updates of AUTOSAR Embedded Applications -- To What Extent?	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE259	Decision as a Service for Transaction Banking Using Service-Oriented Modeling Architecture Methodology	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE260	Exploring Microservices as the Architecture of Choice for Network Function Virtualization Platforms	SI	SI	NO	SI	SI	4	Aceptado
IEEE261	Sky Computing Platform for Legacy Distributed Application	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE272	Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League	SI	SI	NO	SI	PARCIAL	3,5	Aceptado
IEEE276	TSaaS - Customized telecom app hosting on cloud	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE278	Container Based Scalability and Performance Analysis of Multitenant SaaS Applications	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
IEEE310	Architecture Simplification at Large Institutions using Micro Services	SI	SI	NO	SI	SI	4	Aceptado
SCO006	Legacy Digital Transformation: TCO and ROI Analysis	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI002	Open-source software product line extraction processes: the ArgoUML-SPL and Phaser cases	SI	SI	SI	SI	SI	5	Aceptado
SPI005	Improving datacenter utilization through containerized service-based architecture	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI011	An empirical study of the systemic and technical migration towards microservices	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI015	Serverless and Containerization Models and Methods in Challenger Banks Software	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI016	Comparison of open-source runtime testing tools for microservices	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI020	The Migration Journey Towards Microservices	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI025	Building a Big Data Oriented Architecture for Enterprise Integration	SI	SI	NO	SI	SI	4	Aceptado
SPI036	Challenges When Moving from Monolith to Microservice Architecture	SI	SI	PARCIAL	SI	SI	4,5	Aceptado
SPI041	Enabling Legacy Applications for Multi-tenancy Without Reengineering	SI	SI	NO	SI	SI	4	Aceptado
SPI083	A Meshing Tool Product Line Architecture	SI	SI	NO	SI	SI	4	Aceptado