Universidad
Mariana

Prácticas de un arquitecto de software en un equipo Scrum

José Santiago Zambrano Torres

Universidad Mariana

Facultad de Ingeniería

Programa de Ingeniería de Sistemas

San juan de Pasto

2024

Prácticas de un arquitecto de software en un equipo Scrum

José Santiago Zambrano Torres

Informe de investigación para optar al título de: Ingeniero de Sistemas

Asesor
Mg. Giovanni Albeiro Hernández Pantoja

Universidad Mariana
Facultad de Ingeniería
Programa de Ingeniería de Sistemas
San Juan de Pasto
2024

Artículo 71: los conceptos, afirmaciones y opiniones emitidos en el Trabajo de Grado son responsabilidad única y exclusiva del (los) Educando (s)

Reglamento de Investigaciones y Publicaciones, 2007
Universidad Mariana

Dedicatoria

Con todo mi amor y gratitud, dedico este trabajo de grado a mis padres, quienes han sido el pilar fundamental en mi vida. Gracias por su amor incondicional, su apoyo constante y por enseñarme a luchar por mis sueños.

A mi hermano, por ser mi compañero de vida, mi amigo y mi ejemplo de perseverancia.

A mi familia en general, por estar siempre presentes, brindándome su cariño, confianza y motivación en cada paso de este camino.

Este logro es tan mío como de ustedes.

José Santiago Zambrano Torres

Agradecimientos

En primer lugar, agradezco profundamente a mis padres por su guía, sacrificios y por creer en mí incluso en los momentos más desafiantes. Sin su esfuerzo y dedicación, este logro no habría sido posible.

A los docentes que me acompañaron en este proceso académico, quienes con su paciencia, conocimientos y enseñanzas marcaron una huella invaluable en mi formación.

A todos aquellos que de alguna manera contribuyeron a mi desarrollo personal y profesional, les extiendo mi más sincero agradecimiento.

Contenido

Índice de Tablas

Índice de Figuras

Introducción

El presente trabajo aborda la integración de prácticas de arquitectura en equipos Scrum, un tema fundamental en el desarrollo de proyectos medianamente complejos dentro de la industria tecnología. El área temática de la investigación se centra en la ingeniería de software específicamente en la concurrencia entre las metodologías agiles de desarrollo y el diseño arquitectónico, un campo con creciente adopción en las empresas de desarrollo debido a su flexibilidad y enfoque iterativo.

La arquitectura de software desempeña un papel fundamental en la construcción de sistemas robustos, al proporcionar instrucciones para la estructura y el comportamiento del software. Sin embargo, en metodologías agiles como Scrum, su incorporación es limitada debido a la falta de roles y actividades específicas para el diseño de arquitectura. Estudios realizados como (Yang et al., 2018) y (Navarro et al., 2018) destacan que, aunque las metodologías agiles incrementan la adaptabilidad y colaboración, su implementación con prácticas de arquitectura de software sigue siendo un desafío.

La arquitectura de software se define como el conjunto de decisiones estructurales que determinan como los componentes de un sistema interactúan entre sí para cumplir con los requisitos funcionales y no funcionales, (Bass et al., 2013). Por su parte, Scrum, es un marco ágil enfocado en la organización de equipos pequeños y multifuncionales en iteraciones cortas para promover la colaboración y la adaptabilidad dentro del desarrollo de software (Schwaber & Sutherland, 2020). La interacción entre estos dos conceptos constituye el eje de esta investigación.

El rol de arquitecto de software no se encuentra definido explícitamente en Scrum, lo que genera desafíos en proyectos de mediana complejidad los cuales requieren decisiones arquitectónicas críticas. La falta de prácticas estandarizadas y la dependencia de recursos externos incrementan los costos y dificultan la multifuncionalidad del equipo.

La motivación principal de este estudio radica en ofrecer una solución practica para integrar las prácticas de arquitectura de software en equipos Scrum, permitiendo que estos sean completamente

autónomos y multifuncionales. Al abordar la problemática planteada, se busca reducir costos, la optimización de recursos y establecer un modelo el cual se pueda replicar en proyectos de mediana complejidad. Esto, a su vez, aporta valor tanto a la academia como a la industria de desarrollo, promoviendo mejores prácticas en el desarrollo ágil de software

El trabajo finalizó con el diseño del proceso de Incorporación de Prácticas de Arquitectura en un Equipo Scrum (IPARES), el cual organiza las prácticas de arquitectura de software en tres fases clave que pertenecen a las 3 primeras etapas del ciclo de vida de la arquitectura de un producto software propuesto por Babar et al. (2014): Analizar el dominio del problema, diseñar y describir la arquitectura de software y evaluar la arquitectura de software. Este proceso fue validado por tres expertos en arquitectura de software demostrando ser eficiente, efectivo suficiente, que puede contribuir al aseguramiento de la calidad y de bajo costo.

El documento cuenta con elementos del proceso investigativo, donde se proporciona una visión general de la problemática y los objetivos de investigación. También cuenta con marco teórico, donde se describen los conceptos clave, antecedentes y fundamentos que sustentan este estudio. Metodología, que expone el enfoque empírico-analítico utilizado, detallando los instrumentos, la población y el proceso investigativo. Resultados, que presentan las practicas identificadas, los procesos diseñados y su validación, junto con los datos recolectados. Conclusiones y recomendaciones, que resumen los hallazgos, destaca el impacto y propone líneas futuras de investigación.

Este estudio no solo contribuye al conocimiento teórico, sino que también ofrece una herramienta practica para la industria del desarrollo de software en proyectos de mediana complejidad, facilitando la integración de la arquitectura en equipos Scrum, lo que resulta una mejora significativa en la calidad de los productos y en la eficiencia de los equipos de trabajo.

1. Resumen del proyecto

1.1 Antecedentes y estado del conocimiento

En el estudio realizado por Lopes y Junior (2017), se plantea que la arquitectura de software es la composición de un conjunto de decisiones de diseño arquitectónico, que incluye, entre otras cosas, el conocimiento capturado por esas decisiones y su justificación. Por lo tanto, diseñar una arquitectura de software es un proceso de toma de decisiones. El proceso de diseño de la arquitectura de software comprende tres fases: comprender el problema, encontrar una solución para el problema y evaluar la solución propuesta. Especialmente en la fase dos, se toman varias decisiones de diseño arquitectónico para satisfacer las necesidades de las partes interesadas. Varios estudios ((Rekha y Muccini, 2014), (Tofan et al., 2013)) revelan que las decisiones de diseño arquitectónico son el resultado de un esfuerzo de grupo. La composición del equipo suele ser heterogénea, e incluye personas con diferentes responsabilidades, roles y habilidades. Estas características imponen muchos desafíos al proceso de toma de decisiones, donde, en este caso, ya no son más importantes las opiniones individuales sino el consenso de todo el grupo sobre una solución (Lopes y Junior, 2017).

El estudio realizado por Navarro et al. (2018) las metodologías agiles se enfocan en el trabajo en equipo, la adaptabilidad y colaboración dentro del equipo de desarrollo de software y también en los demás miembros del proyecto y los usuarios finales. El uso de estas metodologías ha marcado una tendencia para su adopción en el desarrollo de los proyectos de software donde se presenten necesidades cambiantes y en los que se espera beneficios en menor tiempo. La Arquitectura de Software, en tanto, describe la solución de un sistema y por lo tanto tradicionalmente se piensa como una parte temprana de la fase de diseño ya que reúne todos los requerimientos técnicos y operacionales y que son difíciles de cambiar durante el proceso de desarrollo. Los beneficios al lograr integrar la Arquitectura de Software en las Metodologías agiles, no es un tema que se haya explorado en ambientes académicos, como consecuencia de esto, son muy escasas las publicaciones relacionadas a estos enfoques (Navarro et al., 2018).

En Yang et al. (2018) habla de que en la actualidad se denota que el uso de las metodologías

agiles han aumentado significativamente sirviendo de mecanismo para aumentar la adaptación a los requisitos cambiantes del mercado de software, mostrando que la agilidad es la metodología de desarrollo de software que más está dominando sobre las empresas de software. Independientemente de la metodología de desarrollo de software, la información arquitectónica debe mantenerse bajo documentación para generar un entendimiento común entre las partes interesadas. Sin embargo, la filosofía de agilidad recomienda tener menos documentación, la generación de esta documentación de la arquitectura de software brinda grandes beneficios al producto de software. Esto nos indica que la documentación crea un canal de comunicación alterno y va construyendo memoria histórica. Haciendo que la documentación sirva para cuando se tenga algún cambio en el equipo de desarrollo, el proyecto tenga un gran crecimiento o cuando este tenga varios años de creación(Yang et al., 2018).

Para la investigación de Mekni et al., (2018) los proyectos de desarrollo de software buscan agilidad y sostenibilidad con combinación de prácticas agiles y de arquitectura para poder gestionar objetivos competitivos de agilidad a corto plazo con estabilidad de largo plazo. Además, se observa que cada proyecto de desarrollo tendrá un ciclo de vida con una serie de pasos para la construcción de software (especificación de requisitos, diseño de software, verificación y validación de software, implementación de software), los cuales proporcionaran el modelo del software para su desarrollo y gestión. También, se habla del diseño arquitectónico del software, que es el proceso de aplicación de diferentes técnicas con el fin de definir un sistema para su respectiva codificación física. Este enfoque convencional del diseño de software se centra en la división de un problema y su solución en partes detalladas antes de entrar a las fases de construcción. Pero se observa que los métodos agiles sin racionalización, llegan a cubrir ciertas fases del ciclo de vida del proyecto de desarrollo; estos no proporcionaran un verdadero soporte para el diseño de la arquitectura de software en la gestión del proyecto. Si bien algunas soluciones tendrán aportes desde la literatura respectiva, las evidencias empíricas sobre su adaptación y uso en los entornos de desarrollo ágil siguen siendo muy limitados en la actualidad (Mekni et al., 2018).

En la investigación realizada por Galster et al. (2017) se muestra que, cuando se trata de arquitectura y diseño de software, la industria se inclina hacia diseños flexibles y métodos de diseño de arquitectura ligera. Sin embargo, aunque el desarrollo ágil de software se opone a un gran diseño

inicial, "alguna" arquitectura inicial también es útil en proyectos ágiles. En general, hay dos perspectivas sobre la arquitectura de software. En primer lugar, existe la perspectiva de que las arquitecturas deberían surgir y evolucionar durante el desarrollo. Esta perspectiva puede ser cuestionada porque la arquitectura abarca las decisiones significativas sobre la estructura y el comportamiento del sistema software. En segundo lugar, existe la perspectiva del “gran diseño por adelantado" que conlleva el riesgo de hacer muchas suposiciones arquitectónicas que pueden ser costosas de revertir más adelante (Galster et al., 2017).

Las anteriores investigaciones indican que para todo proyecto de software se debe definir una arquitectura base, y se trata de buscar los diseños más flexibles y métodos de arquitectura adaptativos. En este camino o búsqueda, se han alcanzado varios avances en la ejecución de proyectos de creación de software, donde los métodos agiles como Scrum, no explicitan: el quehacer o las prácticas, los roles y las responsabilidades, dentro de un equipo pequeño para definir una arquitectura de software; no obstante día a día los métodos ágiles se vuelven más populares y se amplía su campo de adopción por parte de las empresas.

Las diferencias encontradas entre esta investigación y los antecedentes consultados radica principalmente en que la adopción de métodos ágiles se viene incrementando dentro de las empresas que desarrollan software (Digital.ai, 2021); y la complejidad cada vez mayor de los sistemas hace que la arquitectura se convierta en un elemento clave para el éxito del producto; sin embargo existe literatura muy reducida de cómo abordar las decisiones arquitectónicas en proyectos de desarrollo de software que requieran entre dos o diez atributos(a este tipos de proyectos se los clasifica como medianamente complejos) realizados por equipos pequeños que hace uso de métodos ágiles.

En la revisión de antecedentes realizada no se logró identificar estudios o investigaciones relacionadas con prácticas de arquitectura de software en métodos ágiles como Scrum. Este hallazgo provisional permite establecer que existe información reducida sobre el tema.

1.2 Descripción del problema

El uso de métodos agiles viene creciendo ampliamente en las empresas de base tecnológica por los beneficios que este enfoque proporciona. A continuación, se presenta uno de los temas poco abordados en los métodos ágiles, pero necesario para solucionar una problemática que se viene presentando.

La aceleración en la adopción de tecnología y los procesos de transformación digital, en las organizaciones se ha convertido en un tema de interés en la investigación en los últimos años. Uno de los aspectos de mayor impacto y factor fundamental en un equipo que desarrolla software está relacionado con la productividad (Oliveira et al., 2016).

Para llegar a ser más productivas, las organizaciones han venido incorporando métodos que se basan en los valores y principios propuestos en el manifiesto por el desarrollo ágil de software (ASD por sus siglas en inglés) y han ganado popularidad en los últimos años, fundamentalmente Scrum (Digital.ai, 2020, 2021). Según Digital.ai (2021) de los equipos que hace uso de métodos ágiles, el 81% utilizan Scrum para la gestión de los proyectos de construcción de software.

Scrum es considerado como un método ágil fácil de comprender, pero difícil de adoptar; que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables a problemas complejos (Schwaber y Sutherland, 2020). Este método plantea 3 roles: product owner, Scrum master y developers, sin embargo, se presenta una dificultad en relación con el diseño de la arquitectura de un producto software, porque en ninguno de los roles se establecen explícitamente actividades relacionadas con esta actividad.

Existen varios estudios donde se analiza el rol de la arquitectura de software en los proyectos de construcción de software que hace uso de métodos ágiles (Yang et al., 2016). No obstante, es reducida la literatura que aborda el rol del arquitecto de software en un método como Scrum.

De otro lado, la arquitectura de software es esencial en el diseño de software y definirla en un equipo Scrum ha sido un tema de interés en los últimos años. En estudios como los de Angelov et

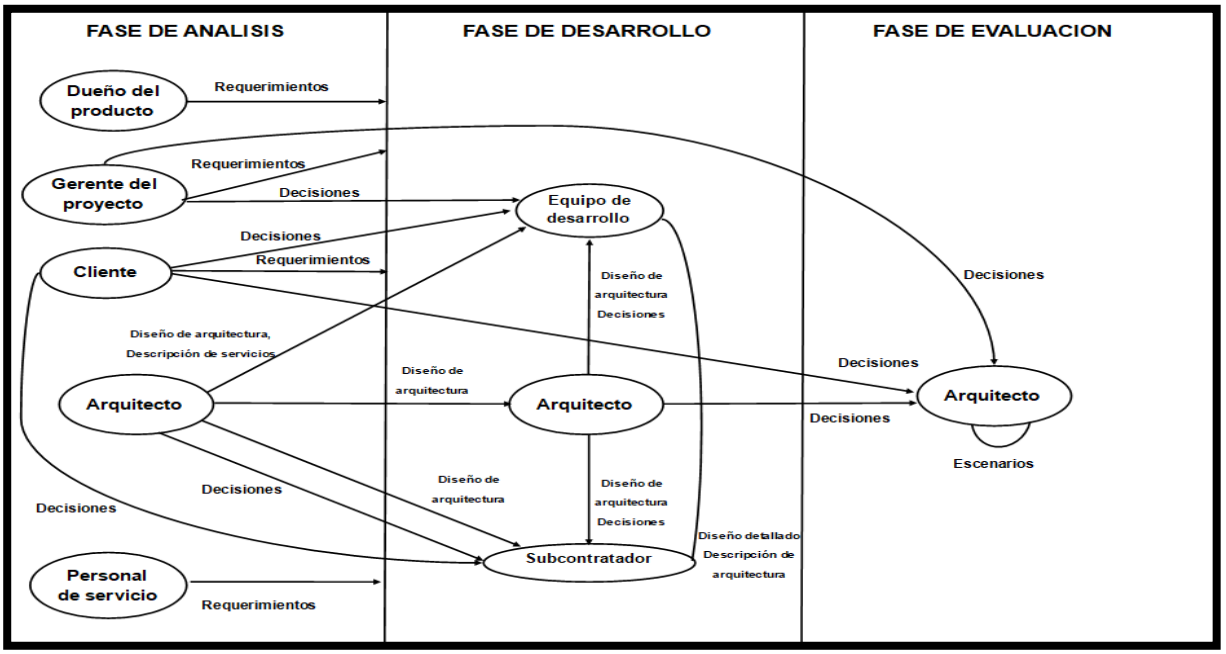
al. (2016) se muestra que no se describe explícitamente el rol del arquitecto de software en un equipo Scrum, pero dependiendo del tipo de producto, la madurez/experiencia/habilidades de los miembros del equipo, estos pueden ser más o menos influyentes. Sin embargo, actualmente faltan conocimientos detallados y sistemáticos sobre el papel del arquitecto.

En el estudio realizado por Eloranta y Koskimies (2014) se muestra que en las empresas que hacen uso de Scrum se presentan cuatro prácticas para definir la arquitectura de un producto software, a saber: diseño inicial grande, *sprint zero*, durante los *sprints* y equipo de arquitectura separada. La primera práctica corresponde con proponer la arquitectura de software antes de que el sistema sea implementado en los *sprints*. La segunda práctica corresponde con designar el primer sprint para definir la arquitectura. La tercera práctica obedece a ir definiendo la arquitectura de manera iterativa e incremental a través del desarrollo de los *sprints*; y la última práctica, busca definir de manera separada un equipo que se dedica exclusivamente al diseño de la arquitectura de software.

En Eloranta y Koskimies (2014) se plantean unos momentos para el desarrollo de las prácticas para definir la arquitectura de un producto software en Scrum. Siendo el *sprint* el corazón de Scrum (Schwaber & Sutherland, 2020), los momentos corresponden con la planeación, el desarrollo y la evaluación. En las prácticas de diseño inicial grande, *sprint zero* y durante los *sprints*; surge el rol del arquitecto de software en los tres momentos (Ver Figura 1). No obstante, cuando la arquitectura de software se la define de manera externa, también surge el rol de arquitecto de software, pero en la planeación y evaluación del sprint (Ver Figura 2).

Figura 1

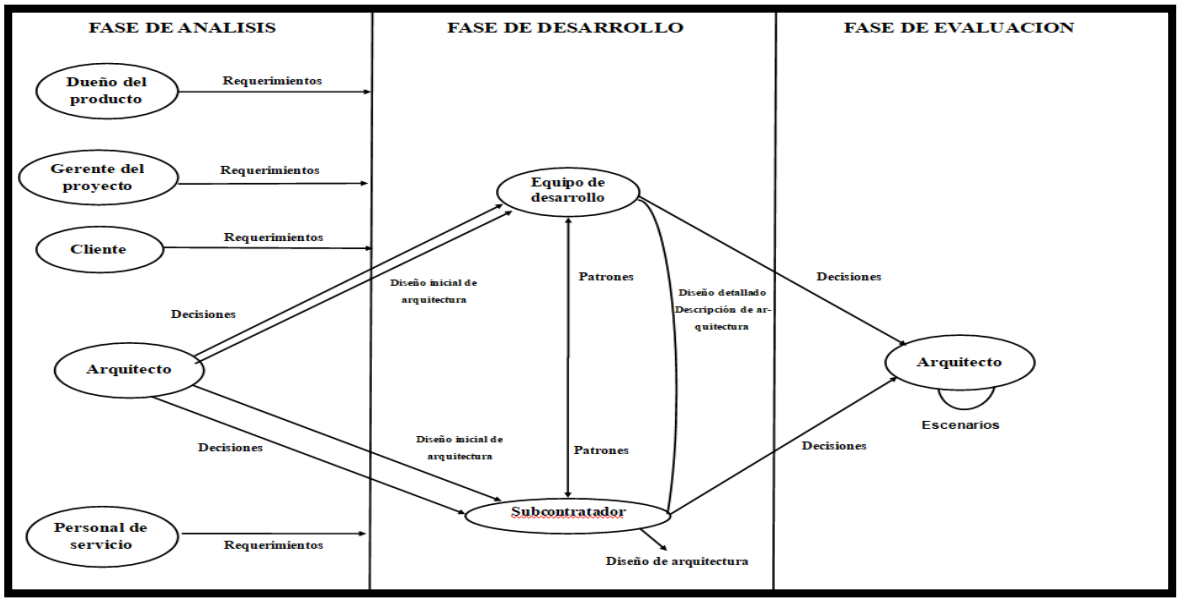
Flujo de la información para la arquitectura de software con el rol de arquitecto durante el sprint



Fuente: Una adaptación del estudio realizado por Eloranta y Koskimies (2014).

Figura 2

Flujo de la información para la arquitectura de software con el rol de arquitecto externo



Fuente: Una adaptación del estudio realizado por Eloranta y Koskimies (2014).

Lo anteriormente descrito, permite evidenciar que, si bien los equipos Scrum de las empresas de la industria de software adoptan prácticas que les permiten definir la arquitectura de un producto software, la forma como lo han venido asumiendo es a través de la incorporación (Puede ser interna o externa) del rol de arquitecto de software. Esta decisión implica dos situaciones problemáticas. La primera obedece a que Scrum está diseñado para trabajar con equipos pequeños (Schwaber y Sutherland, 2020) y el reto de incorporar un arquitecto de software se verá reflejado en los costos de los proyectos. Según el estudio realizado por (CENISOFT & Fedesoft, 2020) para el año 2019 unos de los perfiles más demandados en la Industria TI en Colombia es el arquitecto de software con un salario promedio de $ 1.794 USD (Aproximadamente 7.371.850.oo pesos colombianos a la tasa representativa de la elaboración de este documento). La segunda situación, se presenta cuando las decisiones de arquitectura son externas al equipo Scrum, acción que va en contra de una de las características de este método ágil, la cual es la multifuncionalidad, es decir, el equipo no debe depender de personal ajeno o de terceros. Además, esta decisión implicaría tiempo adicional, introducción de nuevos artefactos y capacitaciones en el uso de los mismos.

1.2.1 Formulación del problema

¿Cómo incorporar las prácticas de un arquitecto de software en un equipo Scrum que desarrolla proyectos medianamente complejos?

1.3 Objetivos

1.3.1 Objetivo general

Incorporar las prácticas de un arquitecto de software en un equipo Scrum que desarrolla proyectos medianamente complejos

1.3.2 Objetivos específicos

- Identificar prácticas que realiza un arquitecto de software en un equipo Scrum mediante un mapeo sistemático de literatura.

- Diseñar un proceso que incorpore las prácticas de un arquitecto de software a los roles de un equipo Scrum.

- Validar el proceso propuesto en un ambiente experimental donde se desarrollen proyectos medianamente complejos.

1.4 Justificación

En esta investigación al identificar prácticas que realiza un arquitecto de software en un equipo Scrum se puede sistematizar teoría que servirá de guía para que otros equipos Scrum la puedan utilizar cuando se encuentren desarrollando software y se basen para poder recrear las prácticas de los arquitectos de software y se deleguen a los miembros del equipo. Además, al diseñar un proceso que incorpore las prácticas de un arquitecto de software a los roles de un equipo Scrum propiciará la característica de multifuncionalidad de equipo, es decir, que no dependerá de externos o terceros para cumplir con los objetivos propuestos durante el trayecto del proyecto de desarrollo de software, además de que todos los roles del equipo Scrum tendrán conocimiento sobre todas las funcionalidades que debe tener el software. Así también, al validar el proceso propuesto en un ambiente experimental permitirá identificar aspectos por mejorar de la propuesta realizada la cual busca aportar a la comunidad de desarrolladores y en especial a las empresas de la industria de desarrollo de software en Nariño y Colombia para mejorar el proceso de construcción de software en especial en empresas con proyectos de mediana complejidad y donde no se pueda lograr el costo adecuado para la contratación de un arquitecto de software.

Además, al lograr identificar las prácticas que realiza un arquitecto de software en un equipo scrum en una microempresa se sistematiza una guía de control sobre las prácticas que se identifiquen de un arquitecto de software dentro de un equipo de desarrollo. Además, en el diseño de un proceso que incorpore las prácticas de un arquitecto de software a los roles de un equipo scrum la cual beneficiaria a los equipos scrum para aprender de las prácticas y poder trabajar de forma simultánea sobre estas actividades que son otorgadas al arquitecto de software también generando conocimiento sobre la estructura de los proyectos y facilitando la realización de estos dentro de las microempresas a nivel regional, nacional e internacional. De esta forma también se

tiene que el validar el proceso propuesto en un ambiente experimental, fomentará en las microempresas de desarrollo de software como pueden adoptar de mejor forma estos objetivos y de cómo se podrán llevar a cabo los cambios necesarios para que los posibles proyectos de estas microempresas no tiendan a quedar estancados o terminen en fracaso.

Por último, la identificación de prácticas que realiza un arquitecto de software en un equipo Scrum se desarrollará mediante el protocolo propuesto en el mediante un mapeo sistemático de literatura en Ingeniería de Software, ampliamente utilizado por la comunidad académica y científica a nivel nacional e internacional. No obstante, como se ha demostrado en la descripción del problema, es reducida la información que existe sobre prácticas de arquitectura en equipos Scrum. Además de cómo se puede validar este proceso en un ambiente experimental conformado por los proyectos que desarrollen los estudiantes de Ingeniería de Sistemas de las Universidades Mariana y Nariño en la materia de Ingeniería de Software II, común en los dos currículos; donde se pondrá en marcha el cómo las actividades que realiza un arquitecto de software a la hora de tomar decisiones dentro de un equipo de desarrollo que implementa la metodología scrum en proyectos de mediana complejidad, se identificara el cómo se van a distribuir estas actividades a todos los miembros del equipo scrum y como esto también ayudara a que los miembros de este equipo adquieran multifuncionalidad en el proyecto y que se tenga conocimiento global de cómo es el funcionamiento del software desarrollado.

1.5 Marcos de referencia

1.5.1 Marco teórico

A continuación, se presentan los principales conceptos teóricas que sirven de constructo y guía para el desarrollo de esta investigación.

Arquitectura de Software

Hay muchas definiciones de arquitectura de software, parafraseando a Bass et al. (2013) manifiesta que la arquitectura de software de un sistema es el conjunto de estructuras que se

necesitan para reflexionar sobre el mismo sistema, que comprenden diferentes elementos de software, que se relacionan entre ellos y comparten propiedades en el mismo. Esta definición de arquitectura de software se asemeja a muchas otras que hablan sobre la toma de decisiones en el diseño principal del sistema. Si bien se dice que muchas de estas decisiones con respecto a la arquitectura son tomadas al inicio de cada proyecto, no todas se enfocan en el diseño principal, esto se evidencia especialmente en proyectos de desarrollo con metodologías agiles y en desarrollo en espiral. Además, también se puede observar que muchas de las decisiones tomadas al inicio de cada proyecto no se relacionan con la arquitectura. También, se muestra que se dificulta separar las decisiones con respecto a la importancia que genera en el proyecto, la importancia de las decisiones en cada proyecto se definirá con el paso del tiempo. Y dado que el escribir una arquitectura se define como una de las obligaciones más importantes del rol de arquitecto, se debe saber cuáles son las decisiones que va a comprender la arquitectura. Por otro lado, las estructuras son fáciles de identificar dentro de un software y conforman una gran herramienta para el diseño de un sistema.

El proceso de arquitectura definido (Yang et al., 2016) lo componen una lista de actividades de arquitectura especificas (cubren todo el ciclo de vida de la arquitectura) y además una serie de actividades de arquitectura generales las cuales respaldan a las actividades específicas. Las actividades específicas están definidas de la siguiente forma:

- El Análisis Arquitectónico (AA): El cual tiene como objetivo definir las problemáticas que debe resolver una arquitectura. El resultado de esta actividad son los conjuntos de requisitos arquitectónicamente significativos (ASR) (Hofmeister et al., 2007).
- La Síntesis Arquitectónica (AS): es la que propone las soluciones de arquitectura candidatas para solucionar los ASR recogidos en el AA por lo tanto esta actividad se mueve del espacio del problema al de la solución (Hofmeister et al., 2007).
- La Evaluación Arquitectónica (AE): asegura que las decisiones tomadas de diseño arquitectónico sean las correctas, y las soluciones arquitectónicas propuestas en la AS se midan contra los ASR obtenidos en el AA (Hofmeister et al., 2007).
- La implementación arquitectónica (IA): implementa la arquitectura por medio de la creación de un diseño detallado (Tang et al., 2010).
- Mantenimiento y Evolución Arquitectónica (AME): El mantenimiento arquitectónico es

> realizar ajustes a una arquitectura con el propósito de corregir fallas o errores encontrados (ISO, 2006, ISO, 2011) y la evolución arquitectónica se usa para responder a nuevos requisitos a nivel arquitectónico (Postma et al., 2004). En este medio, se considera el mantenimiento de la arquitectura y la evolución de la arquitectura como una actividad, en la que se cambia una arquitectura para arreglar las fallas presentadas o implementar los nuevos requisitos solicitados.

Según el estudio de Matthias Galster (Galster et al., 2016), los arquitectos tienen tres tareas fundamentales a la hora de generar un sistema: Se debe obtener la información del mundo exterior (escuchar a los clientes, partes interesadas, aprender sobre tecnologías, etc.), realizar el diseño (tomar las decisiones arquitectónicas para descomponer sistemas, seleccionar las tecnologías a usar, decidir sobre patrones y estilos arquitectónicos, etc.) y proporcionar información sobre los anteriores pasos (comunicar sobre la arquitectura, ayudar a las partes interesadas, etc.). Las actividades de los arquitectos y las habilidades requeridas se centran en procesos, prácticas y tecnologías. La línea entre "desarrollo" y "arquitectura" es muy delgada. Sin embargo, a diferencia del rol de arquitectos, los desarrolladores son los que implementan, prueban y mantienen la máquina de software relacionada con el código y son los que pasan la mayor parte de su tiempo creando o modificando el código (en relación con otras actividades asignadas). El rol de desarrollador y arquitecto no están esencialmente separado puesto que son roles dentro del proyecto, y cada participante puede asumir más de un rol en específico.

Practica de arquitectura de software

Para definir que es una práctica de arquitectura de software, en primer lugar se hace necesario definir que son las prácticas de software, las cuales no solo son descripciones para que se lean en los procesos de desarrollo, estos métodos deben ser dinámicos y respaldan las actividades diarias (Object Managment Group, 2015). También se define las prácticas como un enfoque repetible para alcanzar un objetivo específico (Jones, 2014). Estas proporcionan una vía sistemática y verificable para abordar un aspecto particular del trabajo en cuestión (Barón, 2019).

Con la definición de práctica ya establecida, se puede definir las prácticas de arquitectura de

software, la cuales son acciones basadas en las decisiones que se deben tomar a la hora de empezar a desarrollar un proyecto, enfocándose en recolectar la información de cada ambiente que pueda interactuar con el proyecto y además desglosan todas los atributos de calidad para generar conocimiento sobre el proyecto que se quiere crear (Bass et al., 2013). Generalmente, como resultado del desarrollo de las prácticas de arquitectura de software se estructurar un informe con la información con los atributos y escenarios de calidad que van surgiendo de acuerdo con el avance en la recopilación de información (Bass et al., 2013). Según Barón (2019) algo importante a resaltar es que una práctica de arquitectura debe mostrar que es transferible a varios proyectos de desarrollo y su adopción debe ser en cualquier método ágil para generar conocimiento y además de esto, determina con mayor facilidad las funciones del rol de arquitecto dentro de un equipo de desarrollo.

Scrum

Para definir qué es Scrum, se hace necesario hablar de metodologías, las cuales según (Cadavid et al., 2013) se hace referencia a las metodologías de desarrollo de software tradicionales. Estas metodologías tradicionales de desarrollo de software están orientadas por la planeación. Estas inician el desarrollo de un proyecto con un riguroso proceso de solicitud de requerimientos, esto previo a las etapas de análisis y de diseño. Con esto se busca asegurar resultados de alta calidad ligados a un calendario. En las metodologías tradicionales se determina un solo proyecto, de gran dimensión y estructura definida; donde se sigue un proceso en secuencia en una sola dirección y sin marcha atrás; este proceso se mantiene estático y no puede alterarse; los requerimientos son acordados y se mantienen para todo el desarrollo del proyecto, demandando grandes avances en la planeación previa y muy poca interacción con el cliente una vez acabe esta fase.

Las metodologías ágiles obtienen el factor de flexibilidad, estas pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. Los proyectos agiles se pueden dividir en proyectos más pequeños mediante una lista con características ordenadas. Cada uno de estos es desarrollado de manera independiente y maneja un subconjunto de características durante un periodo de tiempo corto definido de dos a seis semanas. En esta parte la comunicación con el cliente se debe mantener de forma constante a tal punto de requerir un representante para que este durante el desarrollo. Los proyectos son altamente colaborativos y se pueden adaptar mejor a los cambios

que se puedan presentar; estos cambios se evidencian en los requerimientos y se determinan como una característica que es esperada y deseada dentro del desarrollo, al igual que las entregas constantes del producto al cliente y la respectiva retroalimentación de cada una de estas, Tanto el producto como este proceso es mejorado frecuentemente (Cadavid et al., 2013).

La metodología Scrum enfocada en el desarrollo ágil de software es un marco de trabajo que está diseñado para que los equipos de desarrollo de cada proyecto que se encuentren usando esta metodología puedan lograr una colaboración más eficaz en el desarrollo de cada uno, en el cual se usa un conjunto de normas y dispositivos, además de definir roles que general la estructura que se necesita para su correcto funcionamiento (Cadavid et al., 2013). Scrum utiliza un enfoque incremental que tiene como fundamento principal la teoría de control empírico de procesos. Esta teoría está fundamentada en la transparencia, inspección y adaptación; donde la transparencia garantiza la visibilidad en los procesos de las cosas que pueden alterar el resultado, la inspección que ayuda a detectar los cambios indeseados en el proceso y la adaptación que realiza los ajustes necesarios para disminuir el impacto de los mismos (Cadavid et al., 2013).

Los equipos Scrum son autogestionados, multifuncionales y realizan su trabajo en iteraciones. La autogestión les permite elegir el cómo pueden efectuar su trabajo de una mejor forma, en vez de estar ligado a lineamientos de personas que no pertenecen a los equipos y carecen del contexto del manejo de equipos scrum (Cadavid et al., 2013). Los integrantes de estos equipos tienen los conocimientos necesarios al ser multifuncionales, para poder llevar a cabo los trabajos solicitados. La entrega de los productos se realiza por medio de iteraciones, donde cada iteración va a requerir de nuevas funcionalidades o se modifica las que el dueño del producto pueda requerir (Cadavid et al., 2013).

Según Schwaber y Sutherland (2020) en la metodología Scrum se definen tres roles principales: el Scrum master, el product owner y los developers. El rol de Scrum master tiene como función asegurarse de que el equipo este adoptando de buena forma la metodología con sus prácticas, valores y normas, se entiende como Scrum master que es el líder del equipo, pero no gestiona el desarrollo (Schwaber y Sutherland, 2020). El product owner es una sola persona y representa a los clientes y a las partes interesadas, este rol es el responsable de maximizar el valor del producto y

el trabajo que realiza el equipo de desarrollo (Schwaber y Sutherland, 2020). Entre sus funciones se encuentra el gestionar la lista de funcionalidades requeridas o producto backlog. En los desarrolladores o developers, por su parte tienen la responsabilidad de transformar lo que el cliente quiere, el producto backlog, en las iteraciones funcionales del producto, los desarrolladores no tienen una jerarquía en específico, todos los miembros se mantienen en el mismo nivel y cargo de desarrollador y el tamaño óptimo de un equipo Scrum puede estar entre tres y hasta nueve personas (Schwaber y Sutherland, 2020).

1.6 Metodología

1.6.1 Paradigma, enfoque y tipo de investigación

Esta investigación es de corte cuantitativo, porque según Hernández et al. (2010) representa un conjunto de procesos secuenciales y probatorios, por lo tanto no se pueden omitir pasos debido a que el orden es riguroso que parte de una idea que una vez delimitada se derivan en objetivos y preguntas de investigación, de las preguntas se establece una hipótesis, determinan variables y la validez de los resultados se soportarán a través de un proceso estadístico descriptivo debido a que de manera sistemática se desea incorporar prácticas de un arquitecto de software en un equipo Scrum que desarrolla proyectos medianamente complejos. Para lograr este fin, se utilizarán técnicas de recolección y se analizarán los datos procediendo de manera inductiva donde se obtendrán conclusiones empíricas.

El enfoque para esta investigación es Empírico-analítico, porque involucra la experiencia propia con el positivismo lógico (Sampieri et. Al., 2011). Además, con esta investigación se pretende examinar el área de estudio a través de una comprensión instrumental y técnica para analizar las prácticas de los arquitectos de software como un fenómeno que debe ser estudiado de forma objetiva (Tamayo, 2004). Entre los presupuestos que caracterizan este enfoque y se aplican a esta investigación se destaca que, las prácticas de un arquitecto de software en un equipo Scrum son identificables y medibles (Paitán, 2014).

El tipo de investigación es descriptiva porque como plantea Sampieri et al. (2011) se pretende

a través del mapeo sistemático de literatura, identificar prácticas que realiza un arquitecto de software en un equipo Scrum; para posteriormente diseñar y validar un proceso que incorpore estas prácticas en un equipo Scrum. Según Fernández (2002) el comportamiento de las prácticas que desarrollan los arquitectos de software es susceptible de describirse a través de las propiedades que implícita o explícitamente exhiben para generalizar dentro del contexto de la investigación.

1.6.2 Línea y áreas temáticas de investigación

A continuación, se presenta la línea y área temática que se encuentra relacionada con este estudio:

- Línea de investigación: Ingeniería, Informática y computación.
- Áreas Temáticas de investigación: Innovación, modelamiento y desarrollo de software.

1.6.3 Población y muestra

Para identificar prácticas que realiza un arquitecto de software en un equipo Scrum, la población será los artículos de investigación sobre prácticas que realiza un arquitecto de software en un equipo Scrum que se encuentren en los repositorios de ACM Digital Library, IEEExplorer Digital Library y Springer. El muestreo es no probabilístico de tipo intencional, donde los criterios de inclusión para los artículos corresponden con las fases de: búsqueda, selección y evaluación de la calidad, propuestas en el protocolo de Petersen et al. (2008) para el mapeo sistemático de literatura en Ingeniería de Software.

En la validación del proceso propuesto que incorpore las prácticas de un arquitecto de software a los roles de un equipo Scrum en un ambiente experimental, como población se ha seleccionado a los estudiantes el último semestre en las Universidad Mariana y Nariño. El muestreo es no probabilístico de tipo intencional, donde los criterios de inclusión se muestran en la Tabla 1.

Tabla 1

Criterios de inclusión y exclusión

Criterios de inclusión	Criterios de exclusión
• Estudiantes de las Universidades Marian y Nariño. • Estudiantes que este cursando la carrera profesional de Ingeniería de Sistemas. • Estudiantes en último semestre. • Estudiantes que estén cursando una materia electiva del área de Ingeniería de Software.	• Estudiantes de carreras profesionales diferentes a la Ingeniería de Sistemas • Estudiantes que no han aprobado el 82% de los créditos académicos de la carrera profesional.

1.6.4 Proceso de investigación

En la Tabla 2, se presenta la descripción del proceso que se pretende realizar y los medios para lograrlo.

Tabla 2

Proceso de investigación

Objetivos específicos	Fuente	Técnica de recolección	Instrumento	Técnica de Procesamiento	Resultado
Identificar prácticas que realiza un arquitecto de software en un equipo Scrum mediante un mapeo sistemático de literatura.	ACM Digital Library, IEEExplorer Digital Library y Springer.	Fases de búsqueda, selección y evaluación de calidad de los artículos propuestas por Petersen et al. (2008) en el mapeo sistemático de literatura en Ingeniería de Software.	Matrices de: selección de estudios y evaluación de la calidad.	Fases de extracción y síntesis de información propuestas por Petersen et al. (2008) en el mapeo sistemático de literatura en Ingeniería de Software.	Matriz con información sobre las prácticas de multifuncionalidad de equipo.
Validar el proceso propuesto en un ambiente experimental donde se desarrollen proyectos medianamente complejos.	Estudiantes de Ingeniería de Sistemas de la Universidad Mariana y la Universidad de Nariño.	Encuesta	Cuestionario	Análisis de frecuencias y medidas de tendencias central de la estadística descriptiva	Informe con el análisis de los resultados

1.6.5 Variables e hipótesis

1.6.5.1 Hipótesis. Es posible incorporar las prácticas de un arquitecto de software en un equipo Scrum que desarrolla proyectos medianamente complejos.

1.6.5.2 Variables. En la Tabla 3, se pueden observar las variables de la presente investigación.

Tabla 3

Variables de la investigación

Variable	Descripción	Tipo de Variable	Objetivo específico	Indicador	Naturaleza	Fuente	Tr*	Ta**
Práctica arquitectónica de software en Scrum	Aproximación repetible que aprovecha las habilidades de los integrantes de un equipo, con un alto grado de interdependencia para hacer entregas efectivas de software	Independiente	Identificar prácticas que realiza un arquitecto de software en un equipo Scrum mediante un mapeo sistemático de literatura	Práctica	Cuantitativa	Artículos de los repositorios de ACM Digital Library, IEEExplorer Digital Library y Springer	Fases de búsqueda, extracción y síntesis de información propuestas por Petersen et al. (2008) en el mapeo sistemático de literatura en Ingeniería de Software	Fases de búsqueda, extracción y síntesis de información propuestas por Petersen et al. (2008) en el mapeo sistemático de literatura en Ingeniería de Software
				Representación	Cuantitativa			
				Competencia	Cuantitativa			
				Nivel de efectividad	Cuantitativa			

Variable	Descripción	Tipo de Variable	Objetivo específico	Indicador	Naturaleza	Fuente	Tr*	Ta**
Nivel de aceptación	Validar el proceso propuesto en un ambiente experimental donde se desarrollen proyectos medianamente complejos	Independiente	Grado en el cual la propuesta cumple con los criterios definidos.	Aceptación	Cuantitativa	Estudiantes de Ingeniería de Sistemas de la Universidad Mariana y la Universidad de Nariño	Encuesta	Análisis de frecuencias y medidas de tendencias central de la estadística descriptiva
				Mejora	Cuantitativa			
				Incorporación	Cuantitativa			

2. Resultados

En esta sección se presentan los resultados obtenidos en el desarrollo de los objetivos específicos del proyecto. A continuación, se muestra las prácticas de arquitectura que se desarrollan en un equipo Scrum las cuales fueron identificadas a través de un mapeo sistemático de literatura. Posteriormente, se definen los procesos para la incorporación de las prácticas de arquitectura de software a los roles de un equipo Scrum. Finalmente, se realiza la validación del proceso de incorporación con expertos en arquitectura de software.

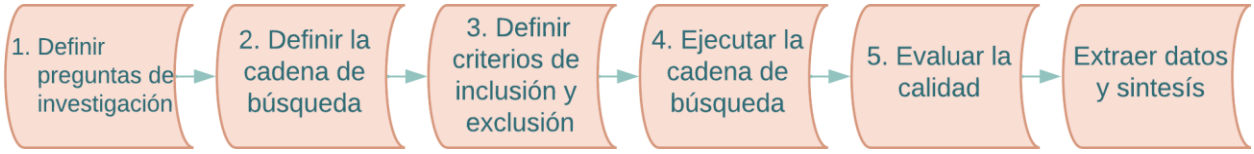
2.1 Prácticas de un arquitecto de software en Scrum

Para identificar prácticas que realiza un arquitecto de software en un equipo Scrum, se utilizó como técnica el protocolo para el mapeo sistemático de literatura descrito en (Petersen et al., 2008).

El protocolo planteado por Petersen et al. (2008) se complementa con el trabajo realizado por (Hernández et al., 2019) (Guerrero-Calvache, & Hernández, 2022) y consta de las fases que se muestran en la figura 3.

Figura 3

Proceso de revisión sistemática de literatura



2.1.1 Definición de las preguntas de investigación

Las preguntas de investigación formuladas se basan en la pregunta principal: ¿Cuáles son las prácticas que realiza un arquitecto de software en un equipo Scrum?

A partir de la pregunta principal se derivaron las preguntas secundarias que permitieron analizar

y categorizar los estudios primarios como se observan en la Tabla 4.

Tabla 4

Preguntas de investigación del SMS

Id	Pregunta de investigación	Justificación
RQ1	¿Cuáles son las prácticas que realiza un arquitecto de software en un equipo Scrum?	En un equipo Scrum, el rol del arquitecto de software no está definido explícitamente. Sin embargo, el arquitecto de software puede ser parte del equipo Scrum y desempeñar un papel importante en el desarrollo de software
RQ2	. ¿Cómo se clasifican las prácticas que realiza un arquitecto de software en un equipo Scrum?	Las prácticas que lleva a cabo un arquitecto de software se dividen en diversos tipos, cada una de ellas centrada en una actividad específica.
RQ3	¿Qué habilidades (Competencias) se requieren para desarrollar las prácticas que realiza un arquitecto de software en un equipo Scrum?	Las habilidades son esenciales para poder comprender los requisitos del rol de arquitecto de software donde se puede asegurar colaboración y éxito al momento de optimizar las actividades del arquitecto de software

1.1.2 Definición de la cadena de búsqueda

Como parte del protocolo para buscar estudios primarios, se identificaron las fuentes de información y se definió la cadena de búsqueda, según la pregunta de investigación principal. Las siguientes bases de datos se utilizaron como fuentes de información: *ACM Digital Library, IEEExplorer Digital Library, Scopus* y *Springer*. En el momento en que se desarrolló el mapeo sistemático, se contaba con acceso libre a las fuentes de información descritas anteriormente. Por esta razón, se tomó la decisión de trabajar con ellas.

La cadena general creada para establecer los criterios de búsqueda se configura a partir de los datos mostrados en la Tabla 5.

Tabla 5

Cadena de Búsqueda

Operador	Parámetros
	"Practice" OR "Practicing" OR "Praxis"
AND	"Software Architect" OR "Software Architecture"
AND	"Scrum" OR "Agile"

La cadena de búsqueda fue elaborada construyendo expresiones que utilizan los operadores booleanos OR y AND. El operador OR se utilizó para incorporar sinónimos del concepto de búsqueda, mientras que el operador AND permite agregar las palabras relacionadas en la cadena de búsqueda.

Para la ejecución de las búsquedas se examinó: título, resumen y palabras clave, dentro de los resultados obtenidos a través del uso de los motores de búsqueda de cada fuente de datos seleccionada.

2.1.3 Definición de criterios de inclusión y exclusión

Después de realizar el proceso de búsqueda, se procedió a seleccionar los estudios relevantes, es decir, aquellos artículos que permiten dar respuesta a las preguntas de investigación planteadas. Para determinar la relevancia de los estudios, se establecieron unos criterios de inclusión y exclusión, como se puede observar en la Tabla 6.

Para la aplicación de los criterios de inclusión y exclusión, como parte de la selección, se definieron los filtros (F) que se muestran en la Tabla 7.

Tabla 6

Criterios de selección de estudios

Criterios de inclusión	Criterios de exclusión
1. Artículos relacionados con prácticas que realiza un arquitecto de software en un equipo Scrum. 2. Artículos publicados en una ventana de observación entre 2001 y 2022 3. Artículos escritos en inglés. 4. Artículos, resultado de estudios primarios. 5. Estudios completos publicados en revistas, conferencias o congresos con revisión por pares	1. Artículos que no se relacionen con prácticas que realiza un arquitecto de software en un equipo Scrum. 2. Reportes técnicos. 3. Estudios duplicados.

Tabla 7

Estrategia de selección

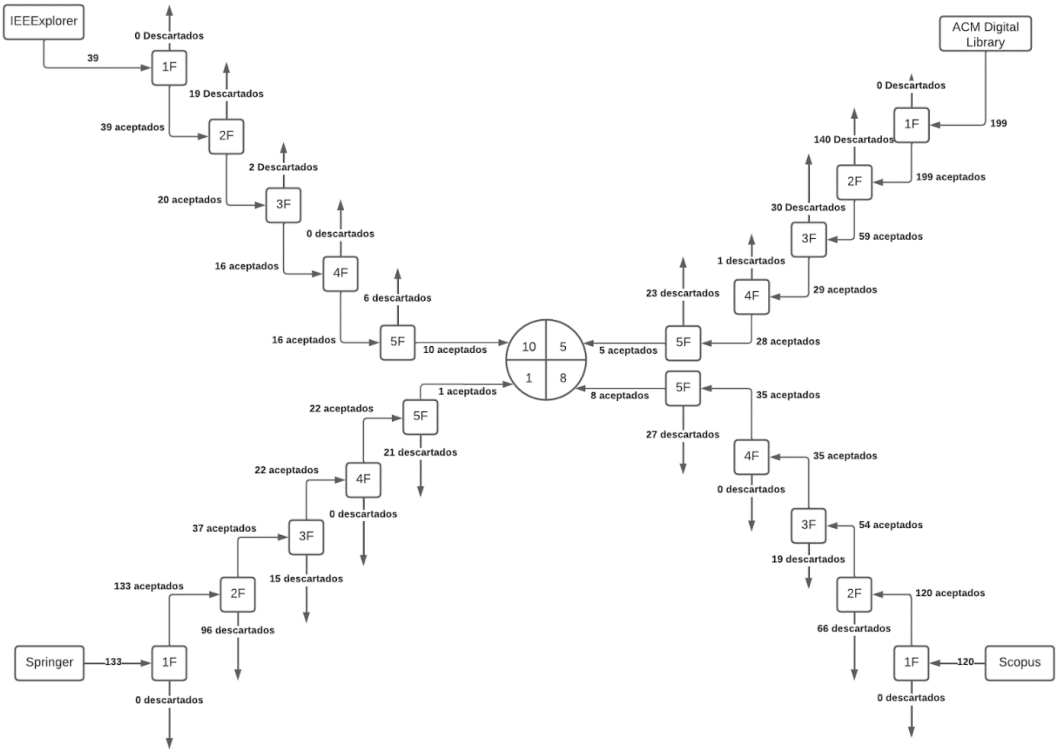
Código de Filtro (F)	Descripción	Criterio aplicado	
		Inclusión	Exclusión
1F	Buscar los artículos aplicando la cadena de búsqueda en los motores de las fuentes seleccionadas.	2, 3	
2F	Leer el título, palabras clave y resumen del artículo aplicando los criterios de inclusión y exclusión.	1 y 2	1 y 2
3F	Leer los resultados y conclusiones del artículo aplicando los criterios de inclusión y exclusión.	1, 4 y 5	1 y 2
4F	Eliminar los estudios duplicados.		3
5F	Leer el artículo completo y aplicar los criterios de inclusión y exclusión.	1, 4 y 5	1 y 2

2.1.4 Ejecución de la cadena de búsqueda

En esta etapa del proceso se ejecutan las etapas para la búsqueda y selección de los artículos de acuerdo con las fases descritas en la estrategia de selección. Los resultados de la búsqueda se pueden observar en la Figura 4.

Figura 4

Resultados de la revisión sistemática



2.1.5 Evaluación de la calidad

Posterior al proceso de selección, se realiza una nueva tarea para asegurar la calidad de los artículos encontrados, que corresponde con la inspección de un conjunto de criterios que se muestran en la Tabla 8.

Tabla 8

Criterios de evaluación de la calidad

Criterio	Criterio	Categoría
C1	Los objetivos y preguntas de investigación se describen de forma explícita, son claros y relevantes.	Calidad del reporte
C2	La investigación presenta un diseño metodológico que le permite alcanzar los objetivos.	Rigor
C3	El procedimiento de recopilación de datos es coherente con el diseño metodológico.	Rigor
C4	Los resultados presentados son claros y coherentes con el diseño metodológico propuesto.	Credibilidad
C5	El estudio es valorado por otros investigadores.	Relevancia

Para evaluar la calidad de los artículos se estableció una escala para inspeccionar el nivel de cumplimiento de los criterios, de la siguiente manera: Alto (2 puntos), Medio (1 punto) y Bajo (0 puntos). Los artículos que cumplieron con una valoración igual o superior al 60% (Ver Tabla 13), del total de los puntos posibles, son lo que finalmente se eligieron.

Al finalizar el proceso de evaluación de la calidad, se incluyeron 10 artículos. En las Tabla 9 Tabla 10, Tabla 11, Tabla 12, se presenta la evaluación de la calidad de los estudios identificados.

Tabla 9

Criterios de evaluación de calidad artículos IEEExplorer

ID	Artículo	C1	C2	C3	C4	C5	F.O.	% F.O.
S1	Software architecture and agile software development: a clash of two cultures?	1	1	1	0	2	5	50%
S2	Peaceful Coexistence: Agile Developer Perspectives on Software Architecture	0	0	1	2	2	5	50%

ID	Artículo	C1	C2	C3	C4	C5	F.O.	% F.O.
S3	Agile vs. plan-driven perceptions of software architecture	1	1	1	2	2	7	70%
S4	A Multiple Case Study of Continuous Architecting in Large Agile Companies: Current Gaps and the CAFFEA Framework	2	2	2	1	2	9	90%
S5	A Contemporary View on Software Quality Requirements in Agile and Software Architecture Practices	2	1	2	1	1	7	70%
S6	Change-Impact Driven Agile Architecting	2	2	2	2	2	10	100%
S7	Code Matters!	0	0	0	0	1	1	10%
S8	Architecting Session Report	1	1	0	0	1	3	30%
S9	Architectural Knowledge Management Practices in Agile Global Software Development	1	2	1	1	2	7	70%
S10	Towards Automatic Classification of Design Decisions from Developer Conversations	1	1	1	1	0	4	40%

Tabla 10

Criterios de evaluación de calidad artículos Springer

ID	Artículo	C1	C2	C3	C4	C5	F.O.	%F.O.
S11	An architecture governance approach for Agile development by tailoring the Spotify model	1	2	2	1	1	7	70%

Tabla 11

Criterios de evaluación de calidad artículos Scopus

ID	Articulo	C1	C2	C3	C4	C5	F.O.	% F.O.
S12	Love Unrequited: The Story of Architecture, Agile, and How Architecture Decision Records Brought Them Together	0	0	0	0	1	1	10,0%
S13	The Pragmatic Architect Evolves	0	0	0	0	1	1	10,0%
S14	Architecture enforcement concerns and activities - An expert study	2	2	1	1	2	8	80,0%
S15	Software Architecture in a Changing World	1	0	1	0	2	4	40,0%
S16	Software Architects in Large-Scale Distributed Projects: An Ericsson Case Study	0	0	0	0	2	2	20,0%
S17	The Architect's Role in Community Shepherding	0	0	2	1	2	5	50,0%
S18	What's the Architect's Role in an Agile, Cloud-Centric World?	0	0	0	0	2	2	20,0%
S19	Aligning Architecture Work with Agile Teams	1	0	0	0	2	3	30,0%

Tabla 12

Criterios de evaluación de calidad artículos ACM Digital Library

ID	Articulo	C1	C2	C3	C4	C5	F.O.	%F.O.
S20	An Industrial Case of Exploiting Product Line Architectures in Agile Software Development	1	2	2	1	2	8	80%
S21	Industrial-Scale Agile: From Craft to Engineering	0	1	0	0	2	3	30%

ID	Articulo	C1	C2	C3	C4	C5	F.O.	%F.O.
S22	Integrating Agile Practices into Architectural Assumption Management: An Industrial Survey	2	1	2	2	1	8	80%
S23	Variability in Software Architecture: The Road Ahead	1	1	1	1	1	5	50%
S24	On the Responsibilities of Software Architects and Software Engineers in an Agile Environment: Who Should Do What?	2	2	1	2	1	8	80%

Una vez, se han aplicado los criterios para evaluar la calidad de los artículos, en la Tabla 13, se pueden apreciar los estudios finales que superan este proceso.

Tabla 13

Artículos finales

ID	Artículo	C1	C2	C3	C4	C5	FO	%F.
S1	An Industrial Case of Exploiting Product Line Architectures in Agile Software Development	1	2	2	1	2	8	80%
S2	Integrating Agile Practices into Architectural Assumption Management: An Industrial Survey	2	1	2	2	1	8	80%
S3	On the Responsibilities of Software Architects and Software Engineers in an Agile Environment: Who Should Do What?	2	2	1	2	1	8	80%
S4	An architecture governance approach for Agile development by tailoring the Spotify model	1	2	2	1	1	7	70%
S5	Architecture enforcement concerns and activities - An expert study	2	2	1	1	2	8	80%
S6	Agile vs. plan-driven perceptions of software architecture	1	1	1	2	2	7	70%
S7	A Multiple Case Study of Continuous Architecting in Large Agile Companies: Current Gaps and the CAFFEA Framework	2	2	2	1	2	9	90%

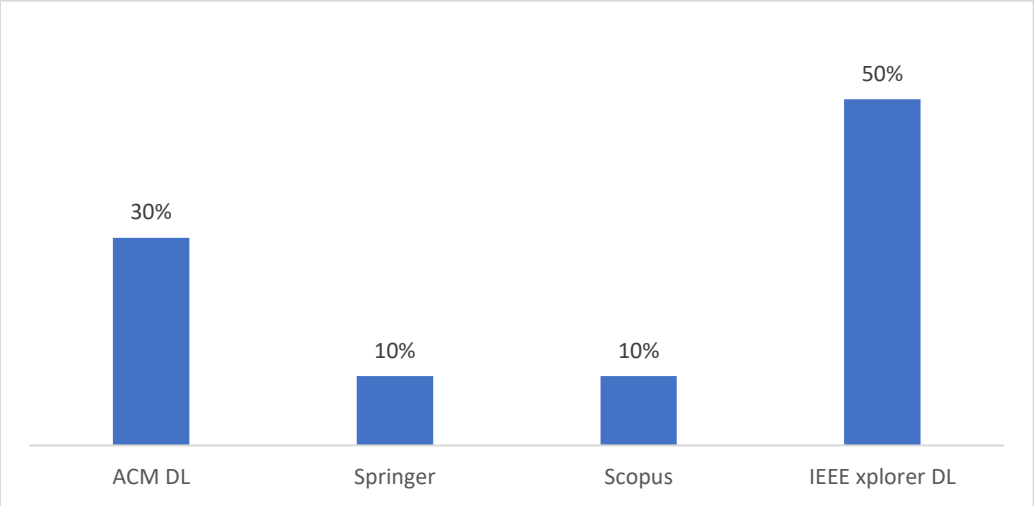
S8	A Contemporary View on Software Quality Requirements in Agile and Software Architecture Practices	2	1	2	1	1	7	70%
S9	Change-Impact Driven Agile Architecting	2	2	2	2	2	10	100%
S10	Architectural Knowledge Management Practices in Agile Global Software Development	1	2	1	1	2	7	70%

2.1.6 Extracción de datos

En esta etapa se procede a la extracción de datos considerando las preguntas de investigación definidas en la primera fase del mapeo. De acuerdo con la fuente de los datos, la mayoría de los artículos relacionadas con prácticas que realiza un arquitecto de software, pertenecen a la fuente IEEExplorer Digital Library, como se puede observar en la Figura 5.

Figura 5

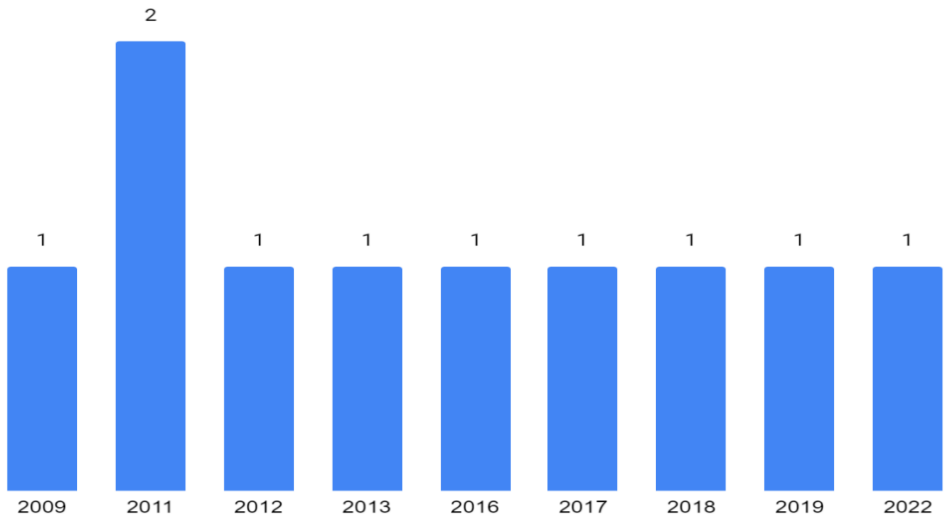
Artículos identificados por repositorio



De acuerdo con la fuente de datos, los artículos seleccionados fueron publicados entre los años de 2000 al año 2022, teniendo la mayoría de los artículos en el 2011 su publicación como se observa en la Figura 6.

Figura 6

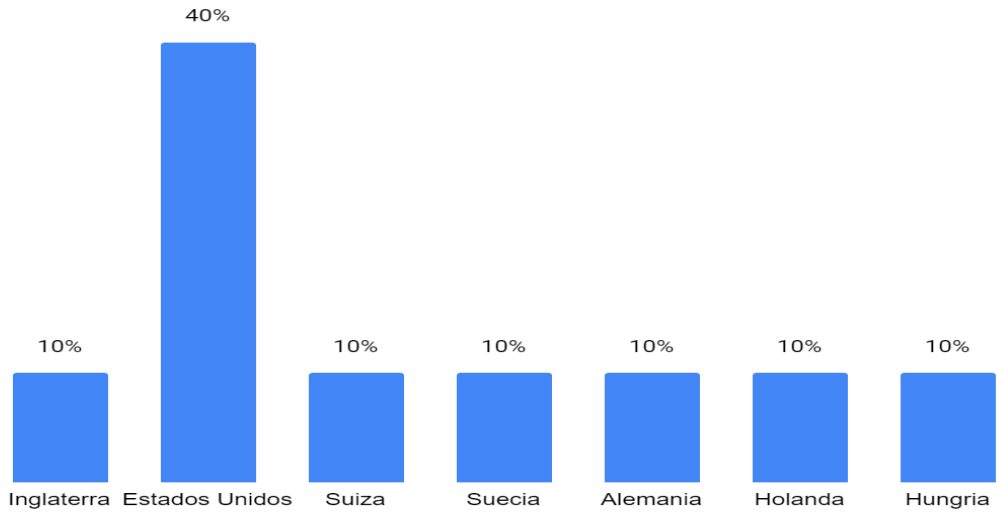
Artículos identificados por año



Por otro lado, de acuerdo con la filiación de los investigadores, es decir, el país de origen se encuentra que, la mayoría de los artículos publicados son de Estados Unidos. Esto se puede evidenciar en la Figura 7.

Figura 7

Artículos Identificados por afiliación



- **Prácticas de un arquitecto de software en un equipo Scrum (RQ1)**

A partir de los artículos seleccionas en la Tabla 14, se extrajeron 40 prácticas que realiza un arquitecto de software en un equipo Scrum indicadas en la Tabla 15. Para especificar las prácticas, se empleó el enfoque general propuesto por (Barón, 2019) para identificar las prácticas, el cual consiste en un modelo que comprende un verbo nominalizado, un adjetivo y un sustantivo (consultar Figura 7). Según Barón (2019), el verbo nominalizado representa el conjunto de acciones que guían el progreso hacia el estado señalado en el criterio de conclusión de la práctica. El adjetivo describe la cualidad que abarca los enfoques empleados para llevar a cabo cada una de estas acciones, mientras que el sustantivo indica el criterio de conclusión de la práctica. Además, se incluye el producto de trabajo de entrada que se requiere para desarrollar la práctica y el producto de trabajo de salida, el cual sería el resultado de la ejecución de la práctica.

Figura 8

Nombramiento de las practicas

Tabla 14

Prácticas que realiza un arquitecto de software en un equipo Scrum

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR1	Diseñar	de manera detallada	la arquitectura de software	Arquitecto de software	Diseño de arquitectura usado en el software	Documento con el diseño arquitectónico
PR2	Crear	de forma explicita	el diseño para la funcionalidad no documentada	Arquitecto de software	Software existente	Diseño para la funcionalidad no documentada
PR3	Describir	detalladamente	las desviaciones de las reglas arquitectónicas para el equipo de desarrollo	Arquitecto de software	Reglas arquitectónicas del producto software	Documento con las desviaciones de las reglas arquitectónicas
PR4	Documentar	de manera clara	el diseño arquitectónico (Diseño de alto nivel)	Developers	Requerimientos no funcionales significativos de los Stakeholders	Documento con el diseño arquitectónico
PR5	Documentar	de manera precisa	la arquitectura actual	Arquitecto de software	Documentación y código existente	Documento con la arquitectura actual

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
					del software	
PR6	Escribir	detalladamente	el código critico	Developers	Requerimientos no funcionales significativos de los stakeholders	Código Critico del Software
PR7	Acordar	de manera detallada	la gestión de supuestos arquitectónicos	Arquitecto de Software	Supuestos arquitectónicos	Documento con la gestión de los supuestos arquitectónicos
PR8	Analizar	detalladamente	los requerimientos del software	Product owner	Requerimientos no funcionales significativos de los stakeholders	Documento con los requerimientos a realizar en el software
PR9	Aplicar	de manera precisa	la técnica CIA (Change-Impact Analysis) a la arquitectura de trabajo del sprint anterior, para obtener conocimiento sobre el impacto de los cambios	Arquitecto de software	Técnica CIA	Aplicación de Técnica Cía.

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
			planificados para el sprint actual)			
PR10	Comparar	de manera detallada	el diseño arquitectónico y el código del software	Scrum master	Software existente	Documento con la comparación de modelo y código del software desarrollado
PR11	Comprender	detalladamente	el producto final y el estado actual de la implementación de las características	Arquitecto de software	Proyecto final	implementación de nuevas características
PR12	Comunicar	de forma coherente	la arquitectura con la que se está trabajando	Arquitecto de software	Modelo de la arquitectura de software actual	Documento con la arquitectura de software a comunicar
PR13	Concientizar	de manera detallada	sobre los efectos de los cambios en las decisiones de diseño anteriores, restricciones, compensaciones y riesgos	Arquitecto de software	Cambios en las decisiones tomadas en el proyecto	Documento con los riesgos sobre los cambios que se quieren realizar al proyecto

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
			para preservar la integridad de la arquitectura			
PR14	Construir	de forma detallada	las iteraciones e incrementos de arquitecturas de trabajo usando el metamodelo Flexible -PLA	Arquitecto de software	Arquitecturas nuevas	Documento con las iteraciones de las arquitecturas
PR15	Convertir	de manera detallada	las decisiones arquitectónicas en decisiones descentralizadas	Arquitecto de software	Toma de decisiones arquitectónicas	Toma de decisiones descentralizada
PR16	Crear	completamente	la alineación técnica y empresarial para el producto software	Arquitecto de software	Objetivos estratégicos de la empresa Elementos tecnológicos	Documento con la alineación técnica y empresarial del software
PR17	Definir	de manera clara	las tecnologías a usar en el proyecto software	Stakeholders	Tecnologías existentes	Documento de tecnologías que se requieren para realizar el software
PR18	Definir	de manera clara	los objetivos de la	Arquitecto de	Proyecto de	Documento con los

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
			arquitectura del software	software	desarrollo	objetivos de la arquitectura a usar en el proyecto
PR19	Dividir	de forma detallada	el sistema en capas o niveles más pequeños	Arquitecto de software	Sistema	división del sistema
PR20	Documentar	de manera clara	la justificación del diseño arquitectónico	Arquitecto de software	Requerimientos no funcionales significativos de los Stakeholders	Documento con justificación del diseño
PR21	Documentar	de manera clara	el diseño detallado	Developers	Documento con el diseño arquitectónico	Documento del diseño detallado
PR22	Elaborar	rápidamente	el estudio de viabilidad del producto	Product Owner	Requerimientos no funcionales significativos de los stakeholders	Documento con el estudio de viabilidad de proyecto
PR23	Entregar	de manera puntual	documentos de diseño de alto nivel	Product Owner	Documentos de diseño de alto nivel	Entrega de documentos de diseño de alto nivel

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR24	Fomentar	de manera eficiente	debates sobre los supuestos arquitectónicos	Arquitecto de software	Actividades realizadas	Informe de debate realizado
PR25	Gestionar	permanentemente	decisiones y cambios de los stakeholders	Product owner	Requerimientos no funcionales significativos de los stakeholders	Software adaptable y funcional
PR26	Gestionar	permanentemente	configuración de producto y código fuente	Developers	Software existente	Software adaptable y funcional
PR27	Integrar	de manera eficiente	las funcionalidades en un solo sistema	Arquitecto de Software	Software existente	Integración de las funcionalidades al software existente
PR28	Interconectar	de forma precisa	los componentes del sistema	Arquitecto de software	Componentes del sistema	Componentes comunicados entre si
PR29	Investigar	de manera exhaustiva	las tecnologías disponibles para la implementación del diseño	product owner	Software a desarrollar	Selección de tecnologías a usar en el proyecto de desarrollo
PR30	Mejorar	progresivamente	la calidad del software	Developers	Software existente	Mejorías en la calidad del software

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR31	Priorizar	de manera oportuna	requerimientos no funcionales de la aplicación	Scrum master	Requisitos de la aplicación	Documento de los requerimientos priorizados
PR32	Priorizar	de manera clara	las reglas arquitectónicas	Arquitecto de software	Reglas arquitectónicas	Documento con reglas arquitectónicas priorizadas
PR33	Probar	de manera optima	el rendimiento del sistema	Developers	Sistema a verificar	Rendimiento del sistema
PR34	Resolver	de manera rápida	problemas arquitectónicos críticos de los clientes	Scrum master	Problemas presentados en el software	Documento con los problemas resueltos en el software
PR35	Revisar	Detalladamente	La arquitectura del software desarrollado	Arquitecto de software	Arquitectura del software	Documento con la arquitectura actual
PR36	Revisar	de forma completa	El código del software	Scrum master	Código de la aplicación	Documento con la revisión de código y aprobación
PR37	Revisar	de manera precisa	el diseño detallado del software	Product owner	Requerimientos no funcionales	Documento con la revisión del diseño

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
					significativos de los stakeholders	de la aplicación
PR38	Tomar	de manera oportuna	decisiones finales en situaciones arquitectónicas conflictivas	Arquitecto de software	Decisiones finales	Documento con la solución de las situaciones arquitectónicas conflictivas
PR39	Validar	detalladamente	las reglas arquitectónicas	Arquitecto de software	Reglas de arquitectura	Documento con las reglas arquitectónicas validas
PR40	Verificar	detalladamente	el cumplimiento de las reglas arquitectónicas	Arquitecto de software	Reglas arquitectónicas	Documento con la verificación de las reglas arquitectónicas

Cada practica que realiza un arquitecto de software está relacionada con una frecuencia observada por los artículos presentados en la Tabla 15., con el fin de evidenciar las prácticas que se repiten por estudio (Ver Tabla 17) En ella, se evidencia que la PR1, definida como el diseño detallado de la arquitectura del software, se repite en cuatro artículos; la PR2, que consiste en crear de manera explícita el diseño para la funcionalidad no documentada; la PR3, que implica describir detalladamente las desviaciones de las reglas arquitectónicas para el equipo de desarrollo; la PR4, que se refiere a documentar de manera clara el diseño arquitectónico (diseño de alto nivel); la PR5, relacionada con documentar de manera precisa la arquitectura final; y la PR6, que implica escribir detalladamente el código crítico, se repiten en dos artículos cada una.

Tabla 15

Frecuencia Observada de las practicas que realiza un arquitecto de software

No	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Frecuencia Observada
PR1			1		1	1		1			4
PR2			1			1					2
PR3		1								1	2
PR4			1							1	2
PR5	1						1				2
PR6					1	1					2

- **Clasificación de las prácticas de un arquitecto de software en un equipo Scrum (RQ2)**

Después de encontrar las practicas que realiza un arquitecto de software se procede a darles una clasificación según sus funcionalidades, por ende, las practicas establecidas en la tabla 16 se clasifican en las siguientes categorías: análisis y documentación, calidad y pruebas, comunicación y presentación, diseño arquitectónico, desarrollo y codificación, entrega, gestión y planificación, investigación y evaluación tecnológica.

La Tabla 16 contiene las practicas relacionadas con el análisis y la documentación de una arquitectura de software, las cuales se orientan con encontrar los problemas a resolver, establecer requerimientos arquitectónicos y especificar todos los resultados del análisis realizado.

Tabla 16

Categoría Análisis y Documentación

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR7	Acordar	de manera detallada	la gestión de supuestos arquitectónicos	Arquitecto de Software	Supuestos arquitectónicos	Documento con la gestión de los supuestos arquitectónicos
PR8	Analizar	detalladamente	los requerimientos del software	Product owner	Requerimientos no funcionales significativos de los stakeholders	documento con los requerimientos a realizar en el software
PR10	Comparar	de manera detallada	el diseño arquitectónico y el código del software	Scrum master	Software existente	Documento con la comparación de modelo y código del software desarrollado
PR11	Comprender	detalladamente	el producto final y el estado actual de la implementación de las características	Arquitecto de software	Proyecto final	Implementación de nuevas características
PR13	Concientizar	de manera	sobre los efectos de los	Arquitecto	Cambios en las	Documento con los

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
		detallada	cambios en las decisiones de diseño anteriores, restricciones, compensaciones y riesgos para preservar la integridad de la arquitectura	de software	decisiones tomadas en el proyecto	riesgos sobre los cambios que se quieren realizar al proyecto
PR4	Documentar	de manera clara	el diseño arquitectónico (Diseño de alto nivel)	Developers	Requerimientos no funcionales significativos de los Stakeholders	Documento con el diseño arquitectónico
PR5	Documentar	de manera precisa	la arquitectura actual	Arquitecto de software	Documentación y código existente del software	Documento con la arquitectura actual
PR20	Documentar	de manera clara	la justificación del diseño arquitectónico	Arquitecto de software	Requerimientos no funcionales significativos de los Stakeholders	Documento con justificación del diseño
PR21	Documentar	de manera clara	el diseño detallado	Developers	Documento con el diseño	Documento del diseño detallado

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
					arquitectónico	
PR35	Revisar	Detalladamente	La arquitectura del software desarrollado	Arquitecto de software	Arquitectura del software	Documento con la arquitectura actual
PR37	Revisar	de manera precisa	el diseño detallado del software	Product owner	Requerimientos no funcionales significativos de los stakeholders	Documento con la revisión del diseño de la aplicación
PR39	Validar	detalladamente	las reglas arquitectónicas	Arquitecto de software	Reglas de arquitectura	Documento con las reglas arquitectónicas validas
PR40	Verificar	detalladamente	el cumplimiento de las reglas arquitectónicas	Arquitecto de software	Reglas arquitectónicas	Documento con la verificación de las reglas arquitectónicas

En la Tabla 17 se presentan las prácticas orientadas al aseguramiento de la calidad y al desarrollo de pruebas de una arquitectura de software con el fin de probar el estado de la solución arquitectónica y si llega a presentar problemas solucionarlos de forma eficaz, además de hacer seguimiento a la solución para mejorar constantemente su calidad

Tabla 17

Categoría Calidad y Pruebas

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR30	Mejorar	progresivamente	la calidad del software	Developers	Software existente	Documento con las mejoras de la calidad del software
PR33	Probar	de manera optima	el rendimiento del sistema	Developers	Sistema a verificar	Documento con los datos del desempeño del sistema
PR34	Resolver	de manera rápida	problemas arquitectónicos críticos de los clientes	Scrum master	Problemas presentados en el software	Documento con los problemas resueltos en el software

En la Tabla 18 contiene las practicas relacionadas con la categoría de comunicación y presentación de una arquitectura de software. Estas acciones se orientan a la comunicación de una arquitectura de software al cliente para identificar cambios de la misma.

Tabla 18

Categoría Comunicación y Presentación

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR12	Comunicar	de forma coherente	la arquitectura con la que se está trabajando	Arquitecto de software	Modelo de la arquitectura de software actual	Documento con la arquitectura de software a comunicar
PR24	Fomentar	de manera eficiente	debates sobre los supuestos arquitectónicos	Arquitecto de software	Actividades realizadas	Informe de debate realizado

En la Tabla 19 contiene las prácticas sobre la elaboración del diseño arquitectónico con el fin de estructurar, definir, crear y diseñar la arquitectura de un producto software.

Tabla 19

Categoría Diseño Arquitectónico

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR15	Convertir	de manera detallada	las decisiones arquitectónicas en decisiones descentralizadas	Arquitecto de software	Toma de decisiones arquitectónicas	Toma de decisiones descentralizada
PR2	Crear	de forma explicita	el diseño para la funcionalidad no documentada	Arquitecto de software	Software existente	Diseño para la funcionalidad no documentada
PR3	Describir	detalladamente	las desviaciones de las reglas arquitectónicas para el equipo de desarrollo	Arquitecto de software	Reglas arquitectónicas del producto software	Documento con las desviaciones de las reglas arquitectónicas
PR1	Diseñar	de manera detallada	la arquitectura de software	Arquitecto de software	Diseño de arquitectura usado en el software	Documento con el diseño arquitectónico
PR19	Dividir	de forma detallada	el sistema en capas o niveles más pequeños	Arquitecto de software	Sistema	División del sistema
PR27	Integrar	de manera eficiente	las funcionalidades en un solo sistema	Arquitecto de Software	Software existente	Integración de las funcionalidades al software existente
PR28	Interconectar	de forma precisa	los componentes del sistema	Arquitecto de software	Componentes del sistema	Componentes comunicados entre si

La Tabla 20 contiene las practicas relacionadas con el desarrollo y codificación en donde las practicas se enfocan en la construcción y revisión de código del software siguiendo un conjunto de lineamientos.

Tabla 20

Categoría Desarrollo y Codificación

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR14	Construir	de forma detallada	las iteraciones e incrementos de arquitecturas de trabajo usando el metamodelo Flexible -PLA	Arquitecto de software	Arquitecturas nuevas	Documento con las iteraciones de las arquitecturas
PR16	Crear	completamente	la alineación técnica y empresarial para el producto software	Arquitecto de software	Objetivos estratégicos de la empresa Elementos tecnológicos	Documento con la alineación técnica y empresarial del software
PR6	Escribir	detalladamente	el código critico	Developers	Requerimientos no funcionales significativos de los stakeholders	Código Critico del Software
PR36	Revisar	de forma completa	el código del software	Scrum master	Código de la aplicación	Documento con la revisión de código y aprobación

En la Tabla 21 se evidencian las prácticas orientadas con las entregas donde se evidencia la práctica definida como la entrega de manera puntual de los documentos del diseño de alto nivel del software.

Tabla 21

Categoría Entrega

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR23	Entregar	de manera puntual	documentos de diseño de alto nivel	Product Owner	Documentos de diseño de alto nivel	Entrega de documentos de diseño de alto nivel

En la Tabla 22, se evidencian las practicas relacionadas con la gestión y planificación de una arquitectura de software, las cuales establecen los parámetros para poder iniciar con la definición de la arquitectura y organizar el trabajo con los clientes para este fin.

Tabla 22

Categoría Gestión y Planificación

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR9	Aplicar	de manera precisa	la técnica CIA (Change-Impact Analysis) a la arquitectura de	Arquitecto de software	Técnica CIA	Aplicación de Técnica Cía.

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
			trabajo del sprint anterior, para obtener conocimiento sobre el impacto de los cambios planificados para el sprint actual)			
PR22	Elaborar	rápidamente	el estudio de viabilidad del producto	Product Owner	Requerimientos no funcionales significativos de los stakeholders	Documento con el estudio de viabilidad de proyecto
PR25	Gestionar	permanentemente	decisiones y cambios de los stakeholders	Product owner	Requerimientos no funcionales significativos de los stakeholders	Software adaptable y funcional
PR26	Gestionar	permanentemente	configuración de producto y código fuente	Developers	Software existente	Software adaptable y funcional
PR31	Priorizar	de manera oportuna	requerimientos no funcionales de la aplicación	Scrum master	Requisitos de la aplicación	Documento de los requerimientos priorizados
PR32	Priorizar	de manera clara	las reglas arquitectónicas	Arquitecto	Reglas	Documento con

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
				de software	arquitectónicas	reglas arquitectónicas priorizadas
PR38	Tomar	de manera oportuna	decisiones finales en situaciones arquitectónicas conflictivas	Arquitecto de software	Decisiones finales	Documento con la solución de las situaciones arquitectónicas conflictivas
PR18	Definir	de manera clara	los objetivos de la arquitectura del software	Arquitecto de software	Proyecto de desarrollo	Documento con los objetivos de la arquitectura a usar en el proyecto

Finalmente, en la Tabla 23 se destacan las prácticas relacionadas con la investigación y evaluación tecnológica, donde, se determinan las tecnologías necesarias para definir la arquitectura de software y desarrollar el producto software.

Tabla 23

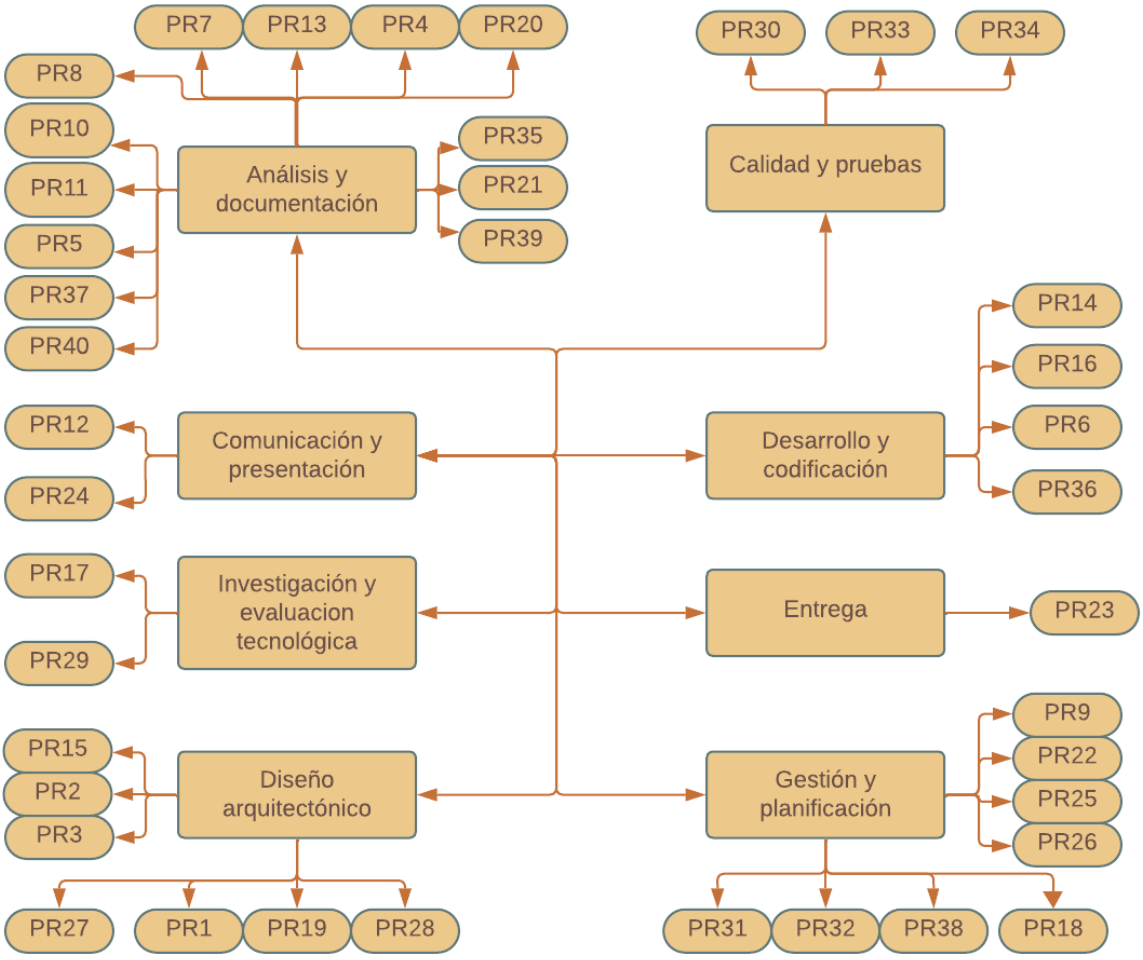
Categoría Investigación y Evaluación Tecnológica

No	Verbo	Adjetivo	Sustantivo	Quien realiza la practica	Producto de trabajo de entrada	Producto de trabajo de salida
PR17	Definir	de manera clara	las tecnologías a usar en el proyecto software	Stakeholders	Tecnologías existentes	Documento de tecnologías que se requieren para realizar el software
PR29	Investigar	de manera exhaustiva	las tecnologías disponibles para la implementación del diseño	Product owner	Software a desarrollar	Selección de tecnologías a usar en el proyecto de desarrollo

En la Figura 8, se sintetiza y hace una representación de las categorías con sus prácticas.

Figura 9

Figura de las categorías de las prácticas que realiza un arquitecto de software



- **Competencias para desarrollar las prácticas de un arquitecto de software en un equipo Scrum (RQ3)**

Para establecer las competencias que un integrante de un equipo Scrum debe desarrollar o perfeccionar en el despliegue de las prácticas presentadas en la Tabla 24, se procede a establecer a identificar de manera global la acción y el conocimiento que se requiere para movilizar en la acción. De acuerdo con el trabajo realizado por Velasco-Elizondo (2021) las competencias que debe poseer un arquitecto de software pueden ser duras (*hard*) o blandas (*soft*). Las habilidades duras se guían

por las acciones que debe desarrollar en relación con el ciclo de vida de una arquitectura de software. Para Velasco-Elizondo (2021) existen cuatro fases. En una primera fase se encuentra la Identificación de Requerimientos, que se centra en establecer y priorizar los requerimientos arquitectónicos. Una segunda fase es el Diseño, que se centra en identificar y seleccionar los diferentes constructos que componen la arquitectura y satisfacen los requisitos arquitectónicos. En una tercera fase se muestra la Documentación, que se centra en la creación de documentos que describen las diferentes construcciones que componen la arquitectura, con el fin de comunicarla a los diferentes actores del sistema. Finalmente, una cuarta fase corresponde con la Evaluación, que se centra en evaluar el diseño de la arquitectura del software para determinar si satisface los requisitos arquitectónicos. Esta revisión se centra en las habilidades duras (*Hard Skills*) y propone como trabajo futuro realizar un análisis de las habilidades blandas (*Soft Skills*).

Para poder desarrollar las prácticas de arquitectura de software descritas en el RQ1 y RQ2 se han identificado 14 competencias donde su sintaxis propuesta por Hernández, et. al (2021) estructurada en: verbo, objeto sobre el cual recae la acción, la condición que se requiere para desarrollar la acción y el propósito o finalidad para la cual está hecha. Esta información está reflejada en la Tabla 28.

Tabla 24

Competencias requeridas para las prácticas de arquitectura de software

ID	Verbo	Objeto	Condición	Finalidad	Prácticas
C1	Alinear	la arquitectura de software con los objetivos empresariales	a partir de una comprensión empresarial, el análisis de los requerimientos y una visión arquitectónica	para contribuir de manera efectiva al éxito y la competitividad de la organización.	PR16
C2	Comunicar	una propuesta arquitectónica	de manera efectiva	para garantizar la comprensión, la	PR7 PR12

ID	Verbo	Objeto	Condición	Finalidad	Prácticas
		al equipo		validación, la gestión de expectativas y la toma de decisiones informadas en el desarrollo de un producto software	PR13 PR24
C3	Contrastar	el diseño arquitectónico y el producto desarrollado	mediante un análisis comparativo	para garantizar que el producto final cumpla con los estándares de calidad y las expectativas del cliente	PR10 PR11
C4	Crear	software de calidad	haciendo uso de prácticas ágiles	para entregar de manera incremental, continua, flexible y colaborativa, software con valor al cliente	PR6 PR36
C5	Diseñar	una propuesta arquitectónica	utilizando métodos, técnicas y patrones de diseño	para establecer la dirección técnica del proyecto, facilitar la comunicación y colaboración entre los miembros del equipo, mitigar riesgos, garantizar la calidad del software, y alinear el desarrollo con los objetivos del negocio	PR1 PR2 PR9 PR14 PR18 PR28 PR32
C6	Documentar	un diseño arquitectónico	utilizando artefactos	para garantizar la comunicación	PR3 PR4

ID	Verbo	Objeto	Condición	Finalidad	Prácticas
			(formatos, diagramas y visualizaciones) basados en estándares y herramientas	efectiva, el registro histórico, el mantenimiento y la evolución del software a lo largo del tiempo, así como para cumplir con los requisitos normativos y facilitar la colaboración y la integración con otros sistemas	PR5 PR20 PR21 PR23
C7	Elaborar	un estudio de viabilidad	utilizando un marco estructurado (Análisis Costo Beneficio, Retorno a la Inversión - ROI, Análisis Costo Eficiencia,)	para evaluar la viabilidad de un proyecto desde diferentes perspectivas antes de invertir recursos significativos	PR22
C8	Especificar	requerimientos del software	usando técnicas	para dar respuestas a los atributos de calidad en una arquitectura asegurando que el sistema satisfaga las necesidades y expectativas de los stakeholders	PR8 PR31
C9	Evaluar	una arquitectura de software	utilizando un marco	para validar el diseño técnico, identificar	PR33 PR35

ID	Verbo	Objeto	Condición	Finalidad	Prácticas
			estructurado, principios de diseño arquitectónico, patrones de diseño y herramientas	riesgos y oportunidades de mejora, asegurar la alineación con los objetivos del negocio, prepararse para la implementación y garantizar la calidad del software.	PR37 PR39 PR40
C10	Gestionar	cambios en una arquitectura de software	usando un proceso de gestión de cambios	para mantener la flexibilidad y la capacidad de respuesta del producto software	PR25 PR30
C11	Gestionar	la configuración de un producto software	mediante el uso de: una política de versionamiento, un sistema de control de versiones y un repositorio centralizado	para garantizar el cumplimiento de los atributos de calidad durante el desarrollo, operación y mantenimiento	PR26
C12	Proponer	soluciones arquitectónicas	utilizando un enfoque sistemático, estructurado y metódico	para abordar los desafíos y requerimientos del producto software	PR27
C13	Seleccionar	tecnologías	a partir de los atributos de calidad identificados	para diseñar un sistema que cumpla con los requisitos del proyecto de manera eficiente y efectiva	PR17 PR29

ID	Verbo	Objeto	Condición	Finalidad	Prácticas
C14	Tomar	decisiones arquitectónicas	con base en los atributos de calidad, patrones de diseño, tecnologías disponibles y restricciones del proyecto	para garantizar la implementación del producto software.	PR15 PR19 PR34 PR38

Después de la extracción de las competencias que requieren las practicas se elaboró una representación gráfica para determinar cuáles son las competencias más importantes y cuáles son las que en más practicas se aplican. En la Figura 9 se puede visualizar que, la competencia C5 es la competencia con mayor número de prácticas, 7 en total, con definición: Diseñar una propuesta arquitectónica usando métodos, técnicas y patrones de diseño. Esta competencia tiene mayor relevancia al usarse para el mayor número de prácticas de arquitectura de software en un equipo que trabaja con el método Scrum.

Figura 10

Grafica de competencias para prácticas de arquitectura de software de mayor a menor

2.1.7 Síntesis de los resultados

El mapeo sistemático de literatura efectuado en cuatro fuentes de información (*ACM Digital Library, IEEExplorer Digital Library, Scopus* y *Springer*) permitió identificar 24 estudios primarios, de los cuales superaron el proceso de evaluación la calidad 10 artículos.

A partir de estos 10 artículos se realizó la búsqueda de las practicas que realiza un arquitecto de software en un equipo Scrum apoyado del estudio realizado por (Barón, 2019), este estudio nos permitió identificar las practicas mediante la especificación de verbo, adjetivo y sustantivo, esto permitió identificar 40 prácticas de los 10 artículos, se identificó también que algunas de estas prácticas se encontraban repetidas en algunos de estos artículos, visualizando la practica 1 como la más frecuente siendo encontrada en 4 artículos.

Las 40 practicas identificadas fueron seleccionadas y divididas en categorías pudiendo identificar 8 categorías diferentes de las cuales la categoría de análisis y documentación y gestión y planificación son las que mayor número de prácticas contienen siendo 13 y 8 respectivamente.

Para poder desarrollar estas prácticas dentro de un equipo scrum se realizó la búsqueda de las competencias necesarias por los arquitectos para poder realizar las practicas, la sintaxis propuesta por Hernández, et. al (2021) ayudo a poder definir las competencias de las 40 practicas encontrando así 14 competencias donde cada practica estará relacionada con la competencia necesaria, además se evaluó la importancia de cada competencia encontrado las competencias: C5, C6 y C9 con mayor número de prácticas siendo 7, 6 y 5 respectivamente.

2.2 Proceso para incorporar prácticas de un arquitecto de software a los roles de un equipo Scrum

En esta sección, se describe cómo se diseña un proceso que integra las prácticas de un arquitecto de software en los roles de un equipo Scrum identificadas en el SMS anterior. Este proceso se realizó a través de la ejecución de dos fases. La primera corresponde con la identificación y definición de los referentes teóricos para realizar la representación del proceso. La segunda se

refiere al diseño y elaboración de la propuesta de incorporación de las prácticas elaborando la representación del proceso utilizando los referentes teóricos identificados.

2.2.1 Definición de referentes teóricos

En la elaboración de una representación de proceso en Ingeniería de Software, es esencial considerar varios referentes teóricos, entre ellos: proceso, proceso de negocio, modelado de procesos de negocio y notación para dicho modelado. A continuación, se detalla cada concepto para una comprensión más amplia.

- Proceso en la Ingeniería de Software

En el ámbito de la ingeniería de software, se define un proceso como una serie de acciones planificadas que se llevan a cabo durante un intervalo de tiempo determinado. Estas acciones requieren el uso de recursos y culminan en la producción de resultados concretos. Dentro de un proceso, las tareas específicas son asignadas a roles particulares y pueden subdividirse en actividades más pequeñas, cada una con su nivel de responsabilidad, con el propósito de alcanzar los objetivos previstos (IEEE Computer Society, 2014).

Un proceso define el rol que está haciendo la acción, cuando y como lo hace (Hernández González, 2005).

- Proceso de negocio

Un proceso de negocios proporciona una representación de las operaciones fundamentales de una organización, las cuales tienden a evolucionar con el tiempo, adquiriendo mayor madurez y capacidad de expansión (White y Miers, 2009).

Un proceso de negocio implica una secuencia estructurada y sistemática de actividades que tienen como objetivo cumplir una función específica dentro de una organización. Estas actividades están organizadas de manera secuencial y requieren ciertos recursos para producir resultados o

salidas deseadas.(White y Miers, 2009) (IBM, n.d.).

En dicho proceso, las actividades llevadas a cabo deben generar valor para el cliente. Para lograr este objetivo, se asignan roles específicos que colaboran en la consecución de la meta establecida (White y Miers, 2009).

La definición clara de los procesos en una organización es crucial para promover la agilidad y facilitar la toma de decisiones cuando sea necesario. (IBM, n.d.).

- Modelado de procesos de negocio

El modelado de procesos de negocio desempeña un papel fundamental al representar las operaciones en curso de una organización (White y Miers, 2009). Esto implica comprender la estructura y dinámica de dichos procesos, así como identificar posibles dificultades y sugerir mejoras. Por lo tanto, los modelos resultantes de esta identificación deben ser coherentes y alineados con los objetivos de la organización (IBM, n.d.).

Los modelos de proceso establecen "el contexto en el cual las métricas adquieren sentido" (White y Miers, 2009). Además, incluyen información sobre cómo se lleva a cabo el trabajo en una organización, quién lo realiza y las restricciones o consideraciones pertinentes (White y Miers, 2009).

- Notación para el modelado de procesos de negocio

Existe un estándar para el modelado de procesos de negocio conocido como BPMN (*Business Process Model and Notation*), que permite representar tanto procesos de negocio de alto nivel como procesos detallados. Este estándar surgió en 2001 como una respuesta a la necesidad de una forma sencilla de plasmar las necesidades de los usuarios (White y Miers, 2009). Sin embargo, no fue hasta 2008 que el Object Management Group (OMG) adoptó formalmente esta notación como estándar para modelar procesos de negocio. Actualmente, BPMN se encuentra en la versión 2.0 (OMG, 2010).

BPMN posibilita la representación de los procesos organizativos según los objetivos centrales de la empresa (White y Miers, 2009).

En la Tabla 25 Niveles de modelado de BPMN se describen los diferentes niveles de modelado que abarca BPMN (White y Miers, 2009).

Tabla 25

Niveles de modelado de BPMN

Niveles de modelado	Descripción
Mapas de Procesos	Un esquema de flujo que muestra las actividades desde una perspectiva global
Descripción de Procesos	Proporcionan detalles más específicos sobre el proceso, incluyendo roles, datos, información, entre otros aspectos.
Modelos de Proceso	Diagramas de flujo más detallados que permiten representar un proceso y luego simularlo.

BPMN posibilita a las empresas representar y comunicar sus procedimientos de negocio de forma visual y estandarizada (White & Miers, 2009).

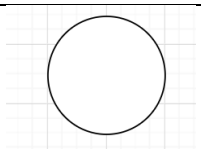
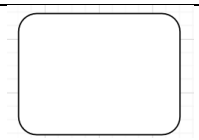
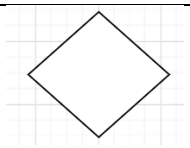
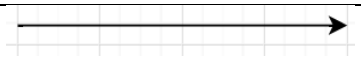
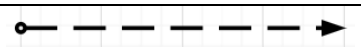
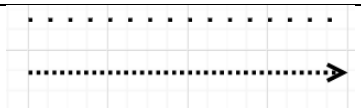
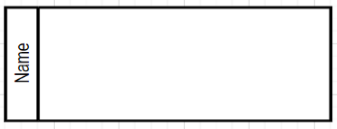
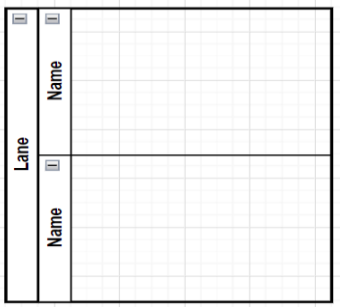
Dentro de BPMN, se pueden identificar tres categorías importantes para la definición de procesos.

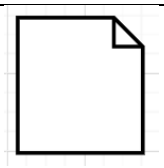
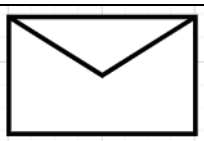
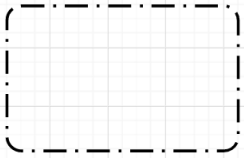
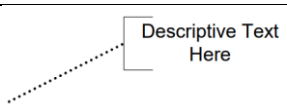
La primera es la Orquestación, que representa una vista específica del negocio u organización del proceso(White y Miers, 2009)(OMG, 2010). La segunda categoría es la Coreografía, la cual describe el comportamiento esperado de un proceso (White y Miers, 2009)(OMG, 2010). Finalmente, está la Colaboración, que incluye procesos de orquestación o coreografía y representa las interacciones entre dos o más entidades del negocio (White y Miers, 2009)(OMG, 2010).

BPMN emplea un conjunto de elementos visuales para modelar procesos, que se organizan en cinco categorías: Objetos de flujo, Datos, Conexiones de objetos, Lanes y Artefactos, como se detalla en la Tabla 26. Elementos gráficos de BPMN.

Tabla 26

Elementos gráficos de BPMN

Elemento	Representación grafica	Descripción
Evento		Muestra lo que sucede durante un proceso o una coreografía. Hay tres tipos de eventos según el flujo: Inicio, Intermedio y Fin.
Actividad		Refleja el trabajo que realiza una organización en un proceso. Puede dividirse en subprocesos o tareas.
Puerta		Cambia el flujo secuencial de las operaciones, lo que puede resultar en una ramificación, bifurcación, fusión o unión de caminos.
Flujo de secuencia		Se emplea para indicar la secuencia de las actividades dentro de un proceso.
Flujo de mensajes		Indica el intercambio de mensajes entre dos participantes del proceso.
Asociación.		Facilita la conexión de información y artefactos con otros elementos. La asociación en forma de flecha muestra la dirección del flujo.
Pool		Es un contenedor en el que un participante lleva a cabo un conjunto de actividades.
Carril		Es una subpartición dentro de un proceso donde las actividades se organizan y clasifican.

Objetos de datos		Representan los insumos o productos del trabajo necesarios para las actividades.
Mensaje		Muestra la información de comunicación entre dos participantes.
Grupo		Representa un conjunto de elementos gráficos que pertenecen a la misma categoría.
Anotaciones		Ofrece información adicional para el lector del diagrama.

Fuente: Adaptación del estándar BPMN expuesta en (OMG, 2010)

2.3 Diseño del proceso

De acuerdo con (Guerrero, 2023) la gestión de proyectos agiles se ha convertido en un desafío en la industria de software [108]. Establecer formas o maneras de incorporar prácticas de un arquitecto de software en un equipo Scrum a través de procesos posibilitan poder medir su efectividad

Por lo anterior, a través de modelos de proceso se busca propiciar la multifuncionalidad (Schwaber y Sutherland, 2020) estableciendo el uso de las prácticas identificadas en este estudio. Para el diseño del proceso es importante considerar cada uno de los elementos que lo conforman. De acuerdo con (Guerrero, 2023) un proceso tiene: Actividad, la cual es una acción que involucra el uso y/o desarrollo de elementos de trabajo. Una actividad consume ciertos recursos, genera algún producto de trabajo y es ejecutada por un rol. Los Elementos de entrada son los insumos requeridos para iniciar una actividad. Los Productos de trabajo, corresponden con las salidas generadas tras la ejecución de una actividad. Los Roles son un individuo o grupo de individuos que ejecutan una actividad o una responsabilidad en un determinado proceso. Los roles definidos para este proceso se orientan bajo marco de trabajo Scrum y son: Product Owner, Scrum Master y Developers (Schwaber y Sutherland, 2020). Además, se tiene en cuenta a los Stakeholders como un rol adicional.

Para la elaboración de los modelos de proceso, se ha seleccionado las 3 primeras etapas del ciclo de vida de la arquitectura de un producto software propuesto por Babar et al. (2014).

Figura 11

Etapas del ciclo de vida de una arquitectura de software

Fuente: una adaptación hecha a la propuesta de Babar et al. (2014)

- **Analizar el dominio del problema**

El análisis del dominio del problema es una etapa crucial en el ciclo de vida de una arquitectura de software, ya que permite comprender el contexto y las necesidades del sistema que se va a desarrollar. En esta fase, se identifican los Stakeholders clave, se definen los requerimientos iniciales, y se analizan los factores que impactan la arquitectura (Babar et al., 2014). Este análisis sirve como base para tomar decisiones informadas y alinear la arquitectura del software con los objetivos del negocio y las restricciones técnicas

En la Tabla 27, se lograron reunir las siguientes prácticas clave:

- PR8: Analizar detalladamente los requerimientos del software.
- PR31: Priorizar de manera oportuna los requerimientos no funcionales de la aplicación.
- PR18: Definir de manera clara los objetivos de la arquitectura del software.

Estas prácticas aseguran que la arquitectura esté alineada con las necesidades del dominio del problema, proporcionando un marco sólido para el diseño y desarrollo de soluciones que cumplan con las expectativas de los Stakeholders.

Tabla 27

Elementos del modelo de proceso para analizar el dominio del problema

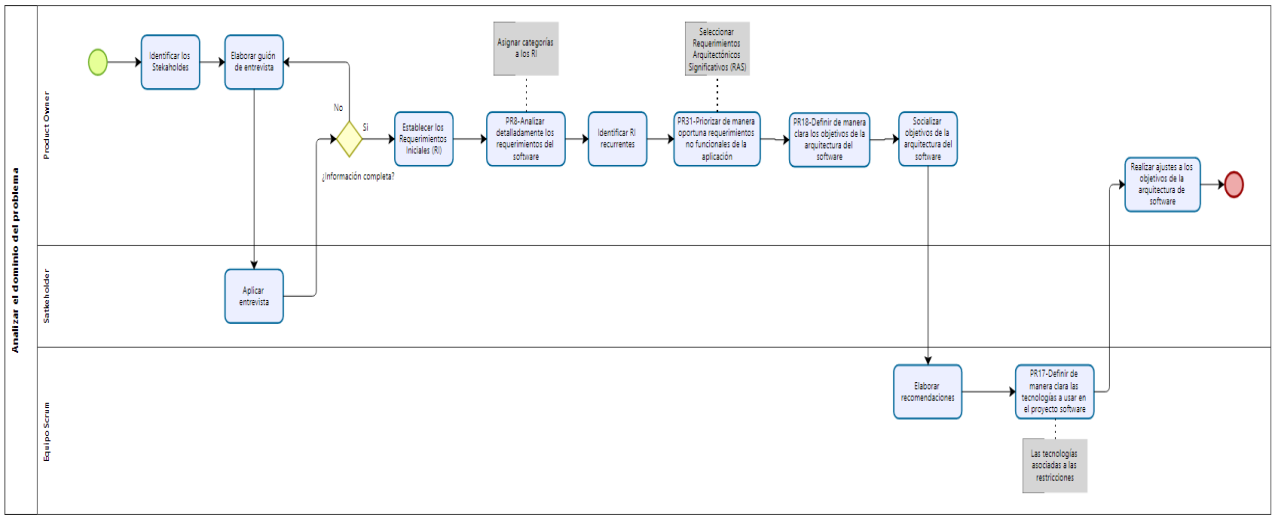
Actividad	Elemento de entrada	Producto de trabajo	Rol
Identificar los Stakeholders	Información del dominio del problema	Lista de Stakeholders clave que influirán en las decisiones arquitectónicas	Product Owner
Elaborar guion de entrevista	Información del dominio del problema, lista de Stakeholders	Guion de entrevista	Product Owner
Aplicar entrevista	Guion de entrevista	Datos sistematizados de la entrevista	Stakeholders
Establecer los requerimientos iniciales	Datos sistematizados de la entrevista	Lista de requerimientos iniciales	Product Owner
PR8-Analizar detalladamente los requerimientos del software	Lista de requerimientos iniciales	Requerimientos iniciales categorizados	Product Owner
Identificar RI recurrentes	Requerimientos iniciales categorizados	Lista de requerimientos arquitectónicos recurrentes	Product Owner
PR31-Priorizar de manera oportuna requerimientos no funcionales de la aplicación	Lista de requerimientos arquitectónicos recurrentes	Requerimientos arquitectónicos priorizados	Product Owner
PR18-Definir de manera clara los objetivos de la arquitectura del software	Requerimientos arquitectónicos priorizados	Objetivos de la arquitectura	Product Owner
Socializar Objetivos de la	Objetivos de la	Lista de participantes en	Product

Actividad	Elemento de entrada	Producto de trabajo	Rol
arquitectura de software	arquitectura	la socialización	Owner
Elaborar Recomendaciones	Objetivos de la arquitectura	Recomendaciones sobre ajustes a los objetivos de la arquitectura	Equipo Scrum
Definir de manera clara las tecnologías a usar en el proyecto software	Requerimientos no funcionales priorizados y objetivos de la arquitectura	Lista de tecnologías seleccionadas	Equipo Scrum
Realizar ajustes a los objetivos de la arquitectura de software	Objetivos de la arquitectura	Objetivos de la arquitectura de software actualizados.	Product Owner

En la Figura 12, se puede observar el modelo de proceso elaborado, a partir de los elementos del modelo descritos en la Tabla 28.

Figura 12

Modelo de proceso Análisis del dominio del problema



- **Diseñar y describir la AS**

El diseño y la descripción de las arquitecturas de software (AS) es una etapa en la cual se transforma la comprensión del dominio del problema en soluciones técnicas concretas. En este proceso, se seleccionan y traducen los requerimientos de calidad en objetivos arquitectónicos, se elaboran escenarios de calidad, y se documentan las decisiones de diseño de alto nivel (Babar, et al., 2014). El objetivo es establecer una estructura clara y coherente que guíe el desarrollo del sistema.

En la Tabla 28, se lograron reunir las siguientes prácticas clave:

- PR8: Analizar detalladamente los requerimientos del software.
- PR7: Acordar de manera detallada la gestión de supuestos arquitectónicos.
- PR4: Documentar de manera clara el diseño arquitectónico (Diseño de alto nivel).

Estas prácticas aseguran que el diseño arquitectónico sea claro, fundamentado y alineado con los requerimientos de calidad, permitiendo una transición efectiva desde el análisis del dominio del problema hacia la implementación técnica.

Tabla 28

Actividades, productos y roles del proceso de diseño y descripción de las arquitecturas de software (AS)

Actividad	Elemento de entrada	Producto de trabajo	Rol
PR8-Analizar detalladamente los requerimientos del software	Requerimientos no funcionales significativos de los Stakeholders	Análisis detallado de los requerimientos	Product Owner
Seleccionar un requerimiento de calidad	Lista de requerimientos	Requerimiento de calidad seleccionado	Product Owner
Traducir requerimiento de calidad a objetivos suaves	Requerimiento de calidad seleccionado	Objetivos suaves para los requerimientos de	Product Owner

Actividad	Elemento de entrada	Producto de trabajo	Rol
		calidad	
Elaborar escenarios de calidad para el requerimiento seleccionado	Objetivos suaves para el requerimiento	Escenarios de calidad	Developers
PR7-Acordar de manera detallada la gestión de supuestos arquitectónicos	Supuestos arquitectónicos	Acuerdos de los supuestos arquitectónicos	Developers
Especificar las decisiones de arquitectura	Requerimientos arquitectónicos significativos	Decisiones arquitectónicas especificadas	Developers
PR4 - Documentar de manera clara el diseño arquitectónico (Diseño de alto nivel)	Requerimientos arquitectónicos significativos	Diseño arquitectónico (alto nivel) documentado	Developers
Incluir las decisiones de diseño en el Product Backlog	Decisiones de arquitectura	Product Backlog de las decisiones de diseño	Product Owner

En la Figura 23, se puede observar el modelo de proceso elaborado, a partir de los elementos del modelo descritos en la Tabla 29.

Figura 13

Modelo de proceso Definir y describir la AS

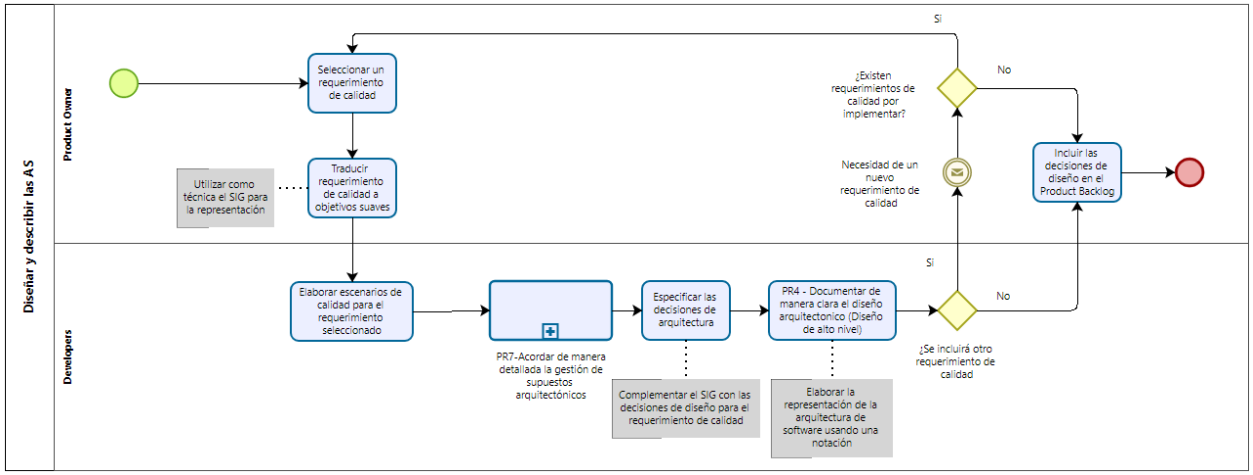
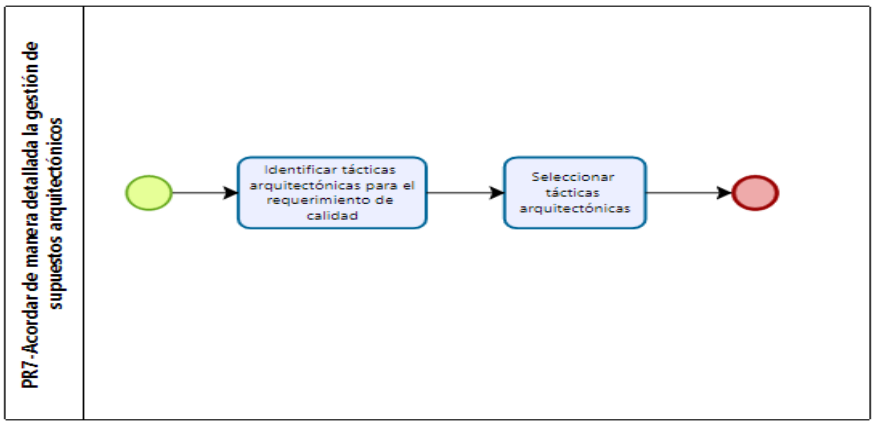


Figura 14

Subproceso definido en el modelo de diseñar y describir la AS



- **Evaluar la AS**

La evaluación de la arquitectura de un producto software (AS) es un proceso iterativo que se lleva a cabo durante todo el ciclo de vida del sistema. Este proceso incluye la selección y análisis de los atributos de calidad, la revisión de las decisiones arquitectónicas y la identificación de riesgos relacionados con los cambios en el diseño. La evaluación constante garantiza que la arquitectura continúe cumpliendo con los objetivos del proyecto y sea capaz de adaptarse a cambios o nuevos requerimientos.

En la Tabla 29, se lograron reunir las siguientes prácticas clave:

- **PR24**: Fomentar de manera eficiente debates sobre los supuestos arquitectónicos.
- **PR13**: Concientizar de manera clara sobre los riesgos de los cambios en decisiones de diseño previas.
- **PR38**: Tomar de manera oportuna decisiones finales en situaciones arquitectónicas conflictivas.
- **PR9**: Aplicar de manera precisa la técnica Change-Impact Analysis a la arquitectura del sprint previo.
- **PR34**: Resolver de manera rápida problemas arquitectónicos críticos de los clientes.

Estas prácticas aseguran que la arquitectura de un producto software sean evaluadas rigurosamente, permitiendo ajustes y mejorando la calidad global del sistema a lo largo del ciclo de vida.

Tabla 29

Actividades, productos y roles del proceso de evaluación de la arquitectura de software (AS)

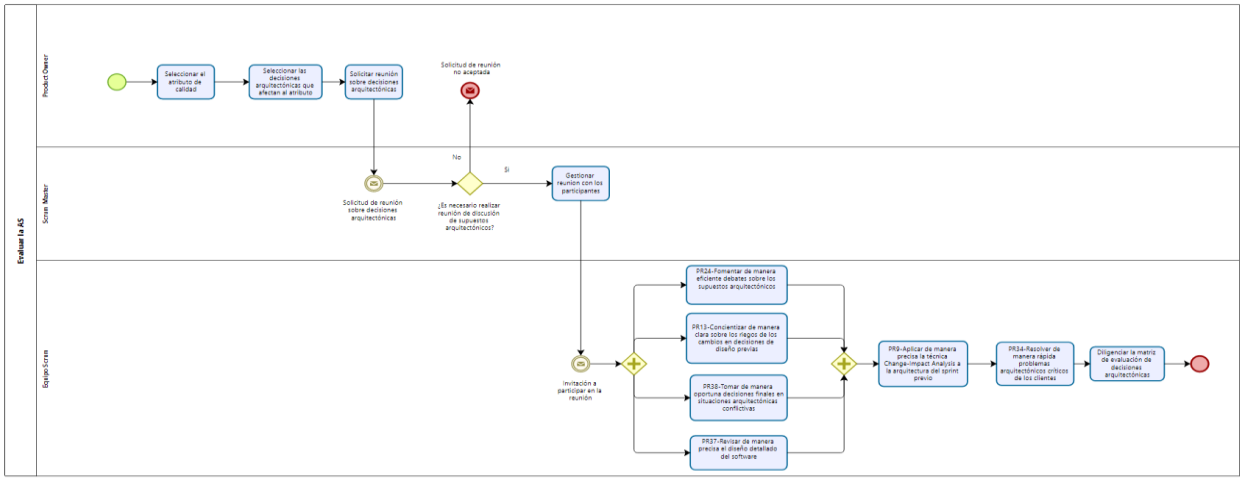
Actividad	Elemento de entrada	Producto de trabajo	Rol
Seleccionar el atributo de calidad	Atributos de calidad definidos	Atributo de calidad seleccionado	Product Owner
Seleccionar las decisiones arquitectónicas que afectan al atributo	Atributo de calidad seleccionado	Decisiones arquitectónicas relacionadas con el atributo	Product Owner
Solicitar reunión sobre decisiones arquitectónicas	Decisiones arquitectónicas identificadas	Solicitud de reunión	Product Owner
Acordar reunión sobre decisiones arquitectónicas	Solicitud de reunión	Reunión programada	Scrum Master
PR24-Fomentar de manera eficiente debates sobre los supuestos arquitectónicos	Supuestos arquitectónicos	Lista de supuestos arquitectónicos debatidos	Equipo Scrum
PR13-Concientizar de manera clara sobre los riegos de los cambios en decisiones de diseño previas	Cambios en las decisiones de diseño	Lista de riesgos de cambios en decisiones de diseño previas	Equipo Scrum
PR38-Tomar de manera oportuna decisiones finales en situaciones arquitectónicas conflictivas	Situaciones arquitectónicas conflictivas	Decisiones finales sobre situaciones arquitectónicas conflictivas	Equipo Scrum

Actividad	Elemento de entrada	Producto de trabajo	Rol
PR37-Revisar de manera precisa el diseño detallado del software	Diseño detallado del software	Listas de defectos identificados en el diseño detallado del software	Equipo Scrum
PR9-Aplicar de manera precisa la técnica Change-Impact Analysis a la arquitectura del sprint previo	Técnica CIA	Análisis de impacto de cambios (Change-Impact Analysis)	Equipo Scrum
PR34-Resolver de manera rápida problemas arquitectónicos críticos de los clientes	Problemas arquitectónicos críticos	Lista de posibles soluciones a problemas arquitectónicos críticos	Equipo Scrum
Diligenciar la matriz de evaluación de decisiones arquitectónicas	Decisiones arquitectónicas	Matriz de evaluación completada	Equipo Scrum

En la Figura 15, se puede observar el modelo de proceso elaborado, a partir de los elementos del modelo descritos en la Tabla 28.

Figura 15

Modelo de proceso Evaluar la AS



Una vez definidos los tres modelos de proceso, se evidencia que las practicas usadas contienen un orden especifico en cada una de las etapas del ciclo de vida de un software, por lo que se procede a asignarles un nuevo identificador a cada una como se evidencia en la Tabla 30.

Tabla 30

Nueva identificación practicas usadas en el diseño del proceso

N. Id	Id. Practica	Practica
PR01	PR8	Analizar detalladamente los requerimientos del software
PR02	PR31	Priorizar de manera oportuna requerimientos no funcionales de la aplicación
PR03	PR18	Definir de manera clara los objetivos de la arquitectura del software
PR04	PR17	Definir de manera clara las tecnologías a usar en el proyecto software
PR05	PR7	Acordar de manera detallada la gestión de supuestos arquitectónicos
PR06	PR4	Documentar de manera clara el diseño arquitectónico (Diseño de alto nivel)
PR07	PR24	Fomentar de manera eficiente debates sobre los supuestos arquitectónicos
PR08	PR13	Concientizar de manera clara sobre los riegos de los cambios en decisiones de diseño previas
PR09	PR38	Tomar de manera oportuna decisiones finales en situaciones arquitectónicas conflictivas
PR10	PR37	Revisar de manera precisa el diseño detallado del software
PR11	PR9	Aplicar de manera precisa la técnica Change-Impact Analysis a la arquitectura del sprint previo
PR12	PR34	Resolver de manera rápida problemas arquitectónicos críticos de los clientes

Ya realizado el nuevo listado, los modelos presentados anteriormente se actualizan para incluir el nuevo identificador por cada práctica. Esto se muestra en las siguientes figuras

Figura 16

Nuevo modelo Analizar el dominio del problema

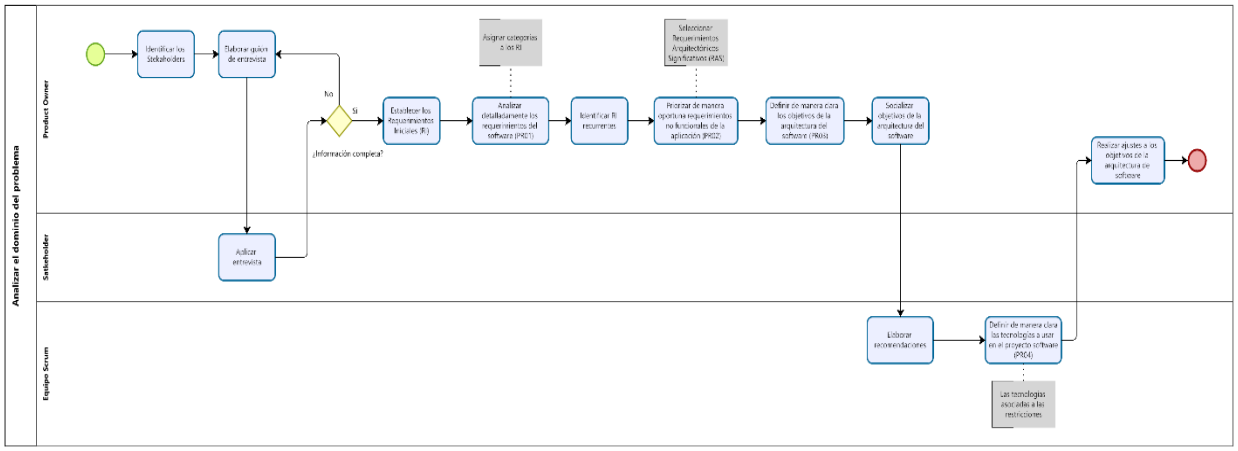


Figura 17

Nuevo modelo describir y diseñar la AS

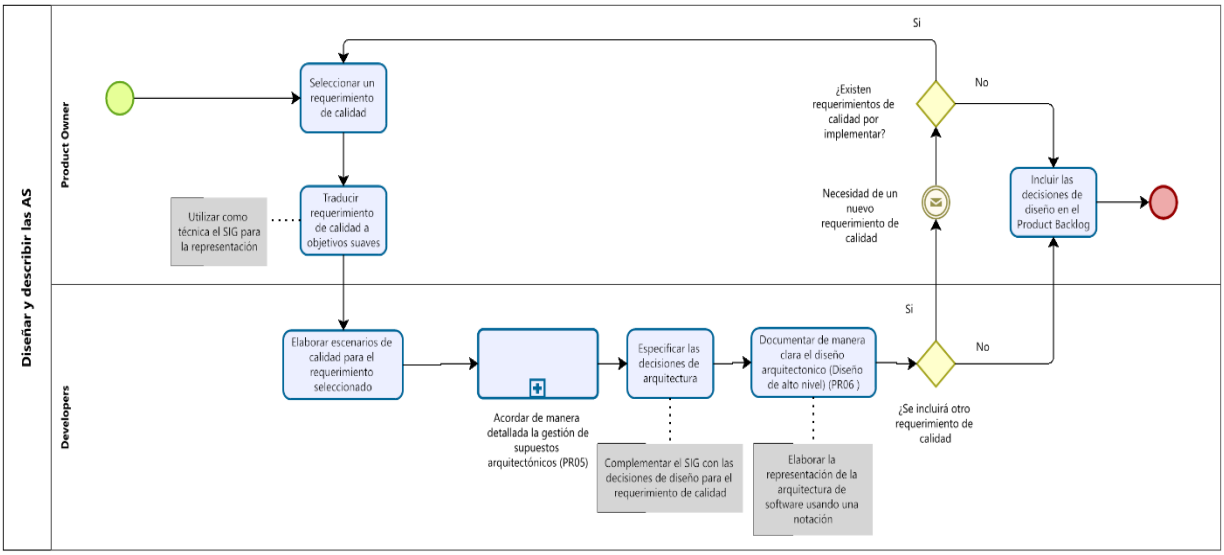


Figura 18

Subproceso definido en modelo de describir y diseñar la AS

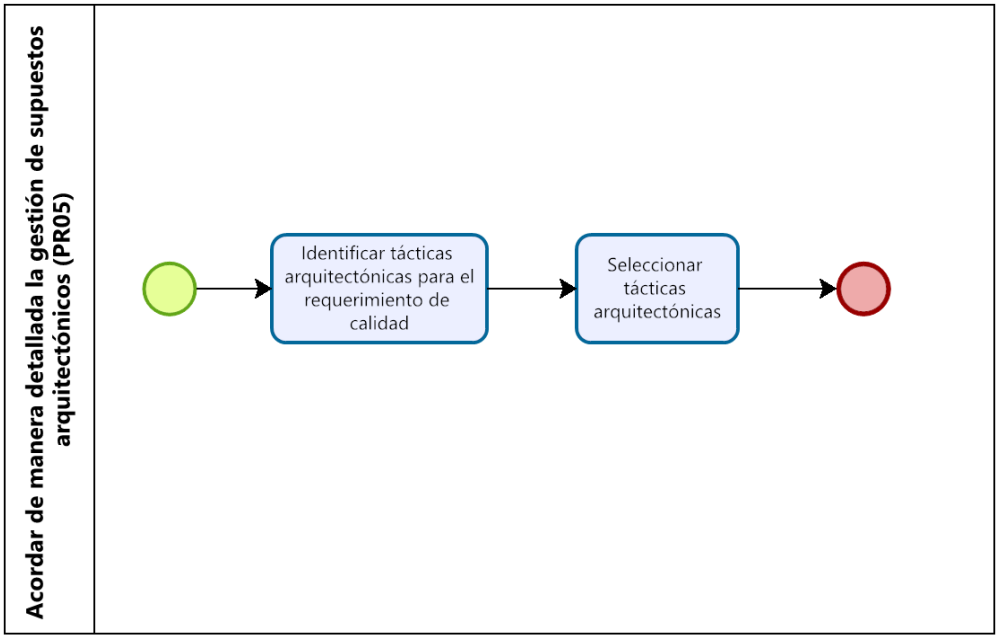
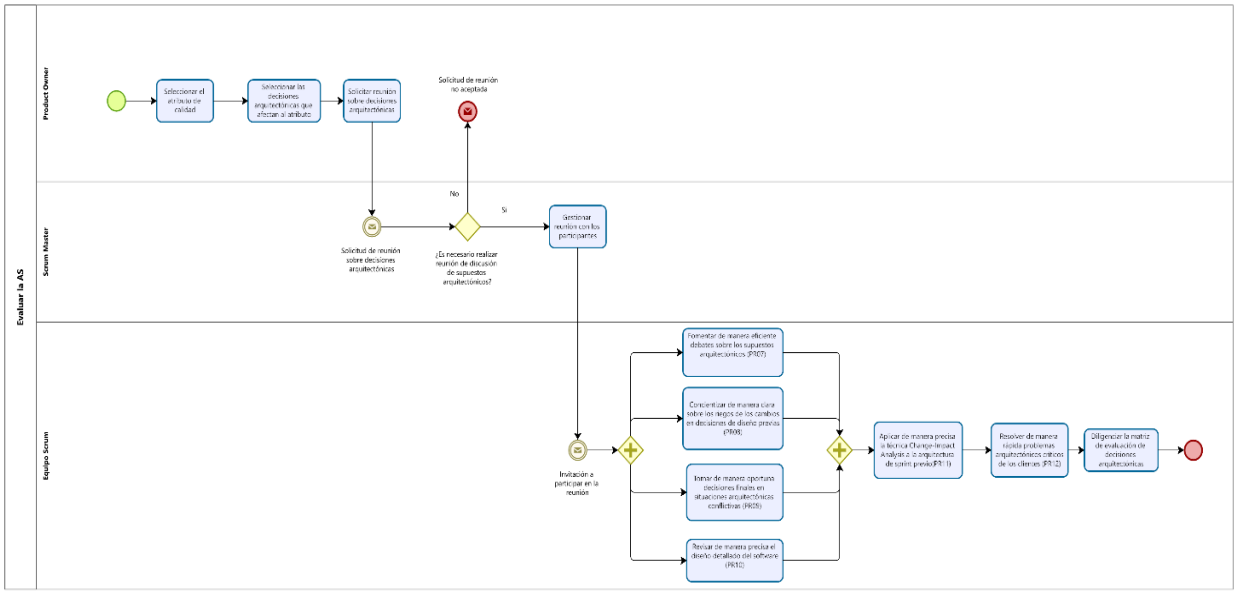


Figura 19

Nuevo modelo evaluar la AS



2.3.1 Síntesis de los resultados

En la elaboración de la representación de procesos en Ingeniería de Software, los referentes teóricos consultados han proporcionado una base sólida y clara para estructurar el enfoque. El concepto de proceso en ingeniería de software (IEEE Computer Society, 2014) establece una directriz clara sobre cómo organizar las actividades y roles, asegurando que las acciones están orientadas a alcanzar objetivos concretos. Por su parte, la definición de proceso de negocio (White & Miers, 2009; IBM, n.d.) resalta la importancia de la secuencia de actividades para generar valor, apartando un marco útil para entender cómo las operaciones deben alinearse con los resultados organizacionales deseados. El modelado de procesos de negocio facilita la visualización y análisis de las operaciones (White y Miers, 2009), lo que permite identificar cuellos de botella y sugerir mejoras. Por último, el estándar BPMN (White y Miers, 2009; OMG, 2010) proporciona una notación estandarizada que favorece la comunicación entre los diferentes actores involucrados.

Entre los aspectos positivos está la claridad y utilidad de estos conceptos para estructurar procesos eficientes. Sin embargo, uno de los retos identificados es la necesidad de profundizar en la adaptación de estas metodologías a contextos específicos, ya que el uso general de BPMN, por ejemplo, puede volverse complejo en procesos muy particulares sin ajustes adecuados.

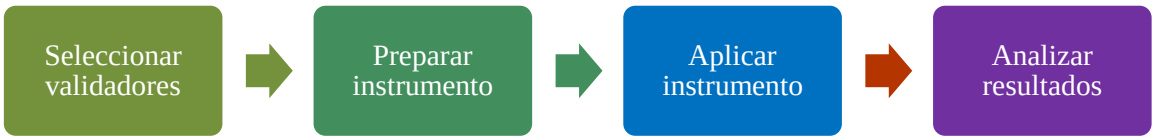
En los modelos de proceso propuestos para el análisis del dominio del problema, diseño y evaluación de arquitecturas de software (AS), se identificaron diversas prácticas clave. En el contexto de un equipo Scrum, fue posible incorporar aproximadamente el 27% de estas prácticas dentro de las actividades regulares del equipo. Las prácticas como la priorización de requerimientos no funcionales (PR31), la definición de objetivos claros de la arquitectura (PR18), y la aplicación de técnicas de análisis de impacto de cambios (PR9), se alinearon bien con las responsabilidades del Product Owner y los desarrolladores en Scrum. Sin embargo, prácticas más formales como la documentación detallada del diseño arquitectónico (PR4) y la evaluación de riesgos arquitectónicos (PR13) resultaron más difíciles de integrar plenamente debido al enfoque iterativo y ágil de Scrum, que prioriza entregas rápidas sobre una documentación exhaustiva.

2.4 Validación del proceso de incorporación de prácticas de arquitectura en un equipo Scrum

En esta sección, se presenta la forma que se validó el proceso de Incorporación de Prácticas de Arquitectura en un Equipo Scrum (IPARES). En la Figura 20, se presenta las actividades realizadas.

Figura 20

Forma de validación de la propuesta



2.4.1 Selección de validadores

Para desarrollar la selección de los validadores, se establecieron unos criterios de inclusión y exclusión que posibilitaron identificar los profesionales idóneos para realizar una retroalimentación al proceso IPARES. En la tabla 31, se presentan los criterios establecidos.

Tabla 31

Criterios de inclusión y exclusión para la selección de validadores

Criterios de inclusión	Criterios de exclusión
Desempeñarse como profesional en el uso de tecnología digitales para la solución de problemas.	Desempeñarse como profesional que no realiza proceso de abstracción sobre el uso de tecnología digitales para la solución de problemas.
Tener 3 o más años de experiencia en el desarrollo de software.	Disponer de una experiencia menor a 3 años de experiencia en el desarrollo de software.
Hacer parte de un equipo de desarrollo de software que trabaja con los lineamientos de	No desempeñar un rol dentro de un equipo de desarrollo de software.

Criterios de inclusión	Criterios de exclusión
Scrum.	
Haber finalizado al menos 3 proyectos de desarrollo, operación o mantenimiento de software.	Ser parte de un equipo que no sigue los lineamientos de Scrum.
Hacer uso de prácticas de arquitectura en el desarrollo de los proyectos de desarrollo, operación o mantenimiento de software.	No disponer de la experiencia de al menos 3 proyectos en el desarrollo, operación o mantenimiento de software.
Disponer de tiempo y manifestar el deseo de participar en la validación del proceso IPARES.	Desconocer el uso de prácticas de arquitectura en el desarrollo de los proyectos de desarrollo, operación o mantenimiento de software.

Con base en los criterios de inclusión y exclusión, se recurrió a las experiencia, conocimiento y referencia de los profesores del programa de Ingeniería de Sistemas para establecer los candidatos al rol de validadores. Se logró identificar un conjunto de 6 validadores, de los cuales después de haber aplicado los criterios de inclusión y exclusión, se seleccionaron 3.

2.4.2 Preparación del instrumento

En esta etapa se selecciona y diseña el instrumento que permitirá recopilar los datos de los informantes para la validación del proceso IPARES. El diseño del instrumento se realiza con base en un conjunto de indicadores que posibilitaran realizar la medición. Posteriormente, se hace una validación del instrumento mediante juicio de expertos. Finalmente, se crea el escenario en el cual se desarrollará validación.

Para la elaboración del instrumento, se seleccionó el protocolo realizado por (Guerrero y Hernández, 2023) como se muestra a continuación:

- Definición de objetivos

El objetivo de la encuesta es identificar aspectos positivos, por mejorar e incorporar al proceso

IPARES. El instrumento se diseña para ser aplicado a profesionales de la industria de software que hacen parte de equipos que hacen uso de métodos agiles en un punto fijo en el tiempo.

- Elaboración del instrumento

La elaboración del instrumento incluye la definición de los tipos preguntas y respuestas a incluir, el formato, organización y longitud del cuestionario (Kitchenham & Pfleeger, 2008). Las preguntas fueron de tipo cerrado y para profundizar la respuesta, se establecen preguntas abiertas. Para el caso de las preguntas cerradas se introdujo respuestas de selección múltiple utilizando una escala Likert como esquema de valoración subjetivo.

El cuestionario evalúo 48 ítems y el tiempo estimado para su diligenciamiento fue de 20 minutos. El instrumento se organizó a través de cuatro secciones. La primera sección corresponde con las Generalidades donde se pone a consideración del encuestado las indicaciones para el diligenciamiento del instrumento y la aceptación del consentimiento informado para el tratamiento de los datos. La segunda corresponde con la Información Sociodemográfica, donde se formulan preguntas para recolectar información sociodemográfica de los validadores. La tercera corresponde con los Aspectos Positivos de IPARES, donde se busca identificar desde la percepción de los validadores cuales son los elementos favorables y positivos la incorporar prácticas de arquitectura en un equipo Scrum. La cuarta hace referencia a los Aspectos por mejorar de IPARES, donde se busca establecer desde la percepción de los validadores cuales son los elementos susceptibles de ser cambiados o transformados con el fin de favorecer la aceptación del proceso IPARES en los equipos que hacen uso de Scrum. Finalmente, en la última sección se presenta los Aspectos por Incorporar a IPARES, donde se busca determinar qué elementos y/o acciones no se han tenido en cuenta en el proceso IPARES que sería interesante incluir.

2.4.3 Aplicación del instrumento

En esta etapa, se coloca en operación el escenario planificado en la etapa de preparación con el fin de que los validadores conozcan el contexto de la problemática y la estrategia de intervención creada. Posteriormente, se aplica el instrumento diseñado con el fin de recopilar las percepciones

de los validadores sobre el proceso IPARES.

La aplicación del instrumento se realizó a los validadores resultantes del proceso de aplicar los criterios de aceptación, los validadores fueron:

- Ingeniero Diego Paredes, Director Unidad de Negocio Humano, Empresa Soporte Lógico, experto en arquitectura de software.
- Ingeniero Álvaro Martínez, Líder técnico, Empresa PlanupSoft, experto en arquitectura de software.
- Ingeniera Sandra Marcela Guerrero Calvache, Docente ocasional tiempo Completo, Institución Universitaria Pascual Bravo, experta en arquitectura de software.

El instrumento tuvo aplicación en el mes de noviembre de 2024, entre las fechas del 2 de noviembre de 2024 y 20 de noviembre del 2024 con reuniones individuales programadas en esas fechas.

Cada encuesta tuvo una duración aproximada de 20 a 30 minutos.

Durante el desarrollo las reuniones mencionadas, se inició con una presentación que contextualizó la problemática a resolver. Posteriormente, se llevó a cabo una exposición detallada del proceso IPARES, explicando las etapas relacionadas con el análisis del dominio del problema, el diseño, la descripción y la evaluación de la arquitectura de software. Una vez concluida la socialización del proceso, se proporcionó acceso al instrumento diseñado para que los validadores pudieran realizar las evaluaciones correspondientes, basándose en la información presentada.

2.4.4 Análisis de los resultados

En esta etapa, se organiza e interpreta los datos recopilados con el instrumento para establecer aspectos positivos, por mejorar e incorporar en el proceso IPARES. Finalmente, se sintetiza los hallazgos mediante una representación gráfica con el fin de facilitar su comprensión.

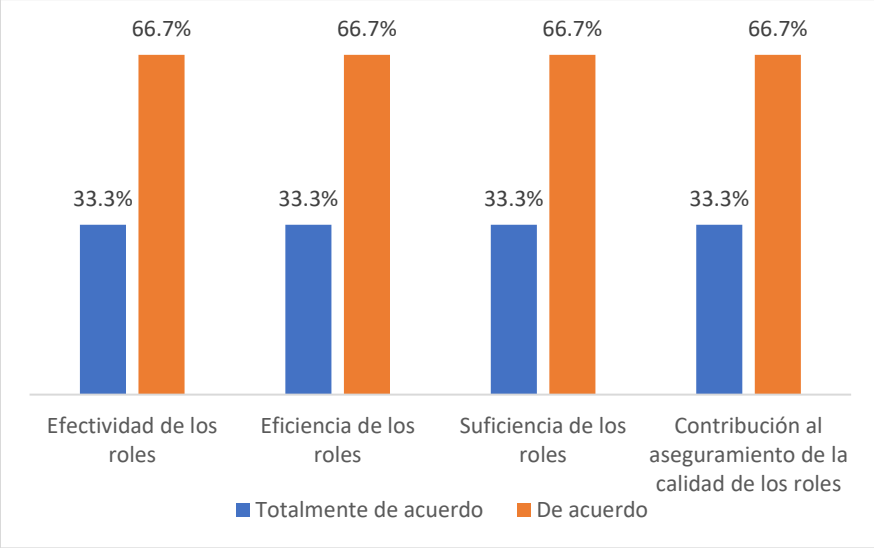
Los resultados se presentan por cada proceso planteado en la propuesta IPARES, es decir, para analizar del dominio del problema, describir y diseñar la arquitectura de software; y evaluar la arquitectura de software.

El instrumento construido plantea un conjunto de preguntas orientadas a validar el proceso IPARES utilizando una escala Likert con 5 opciones que capturan la percepción (Totalmente de acuerdo, De acuerdo, Neutro, En desacuerdo, y Totalmente en desacuerdo). Para identificar los aspectos positivos en la validación, se optó por calcular la frecuencia observada en las respuestas que se seleccionaron como: Totalmente de acuerdo y De acuerdo. Los aspectos por mejorar corresponden con la frecuencia observada en las respuestas: Neutro, En desacuerdo y Totalmente desacuerdo. Con base en el planteamiento anterior se procede a presentar los resultados de la validación a partir de las respuestas obtenidas.

2.4.4.1 Analizar el dominio del problema. Al indagar por los roles, como se puede observar en la Figura 221, se logró identificar que, el 100% de los validadores consideran que el proceso IPARES para analizar el dominio del problema propone roles suficientes, efectivos, eficientes y que contribuyen al aseguramiento de la calidad.

Figura 21

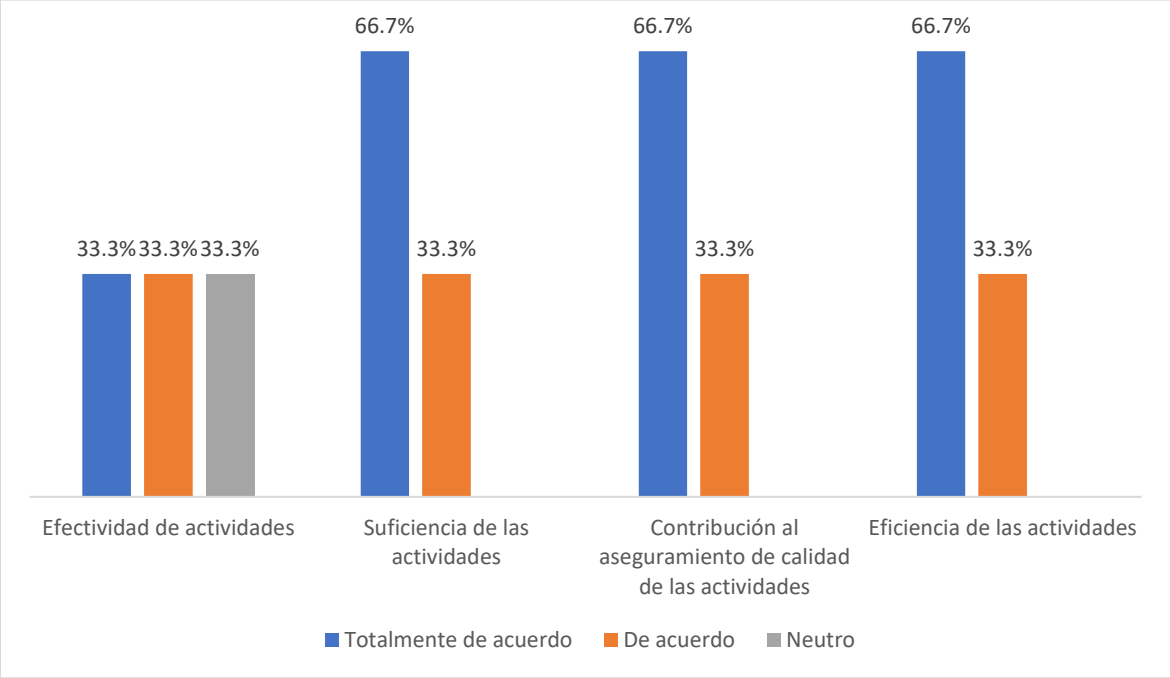
Porcentaje de respuestas sobre los roles de analizar del dominio del problema



También al indagar por las actividades, como se puede observar en la Figura 22, se logró identificar el 100% de los validadores que consideran que el proceso IPARES para analizar el dominio del problema propone actividades efectivas, eficientes, suficientes y que contribuyen al aseguramiento de la calidad

Figura 22

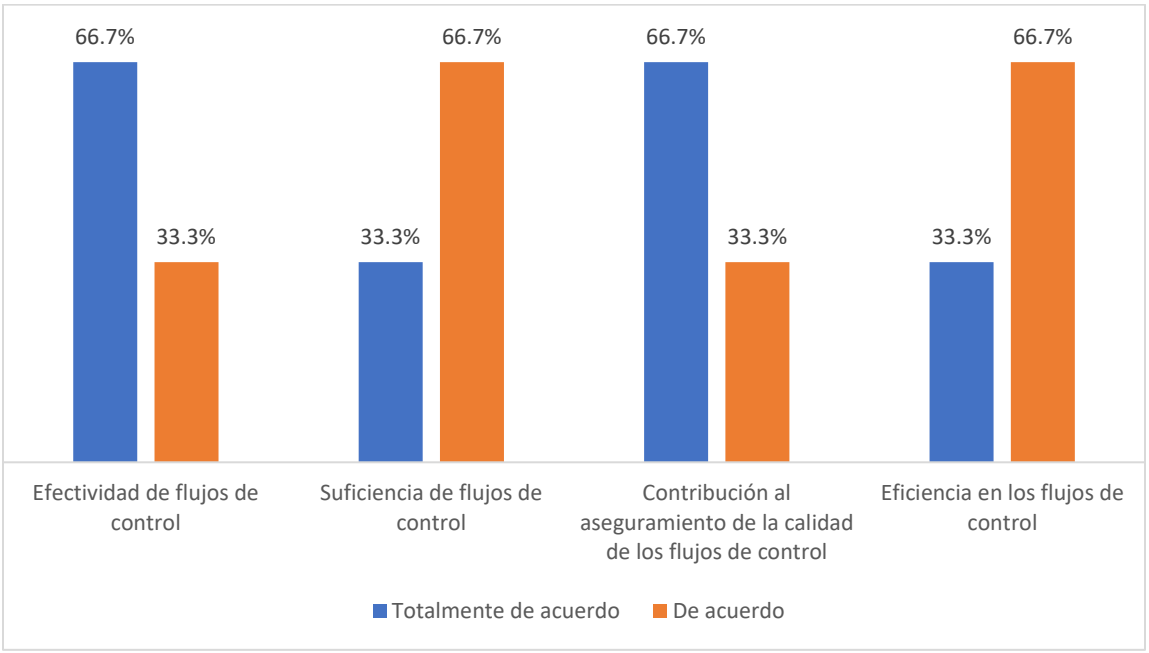
Porcentaje de respuestas sobre las actividades de analizar del dominio del problema



Al indagar por los flujos de control, como se puede observar en la Figura 23, se logró identificar que, el 100% de los validadores consideran que el proceso IPARES para analizar el dominio del problema propone flujos de control eficientes, efectivos, suficientes y que contribuyen al aseguramiento de la calidad.

Figura 23

Porcentaje de respuestas sobre los flujos de control de analizar del dominio del problema



El proceso para analizar el dominio del problema, como se evidencia en la Figura 24, para el 100% de los validadores no resulta costoso de implementar en un equipo de desarrollo pequeño que utilice métodos ágiles.

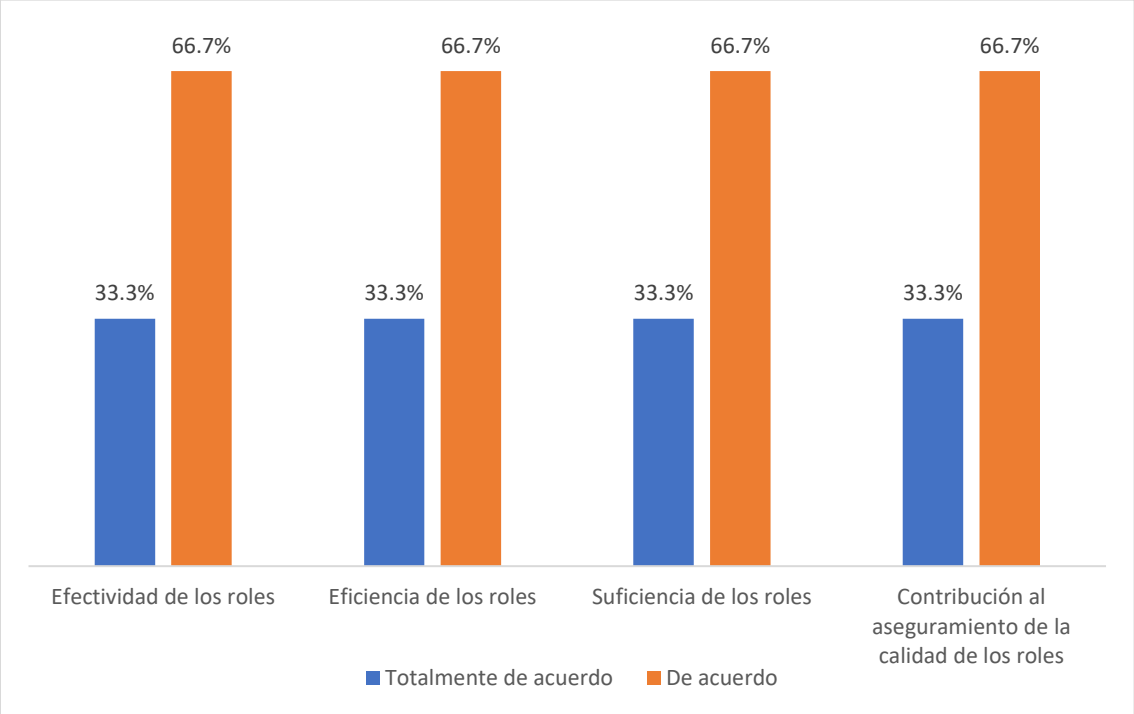
Figura 24

Porcentaje de respuestas de costo para analizar el dominio del problema

2.4.4.2 Describir y deseñar la arquitectura de software. Al revisar los roles, como se muestra en la Figura 25, se identificó que el 100% de los validadores consideran que el proceso IPARES, orientado a describir y diseñar la arquitectura de software, propone roles efectivos, suficientes, eficientes y además que aportan significativamente al aseguramiento de la calidad.

Figura 25

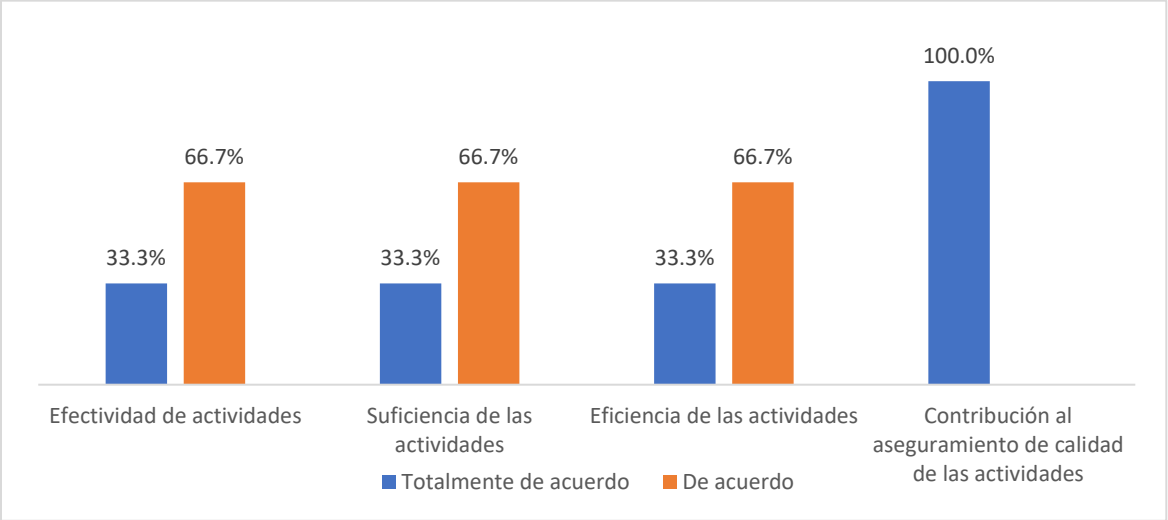
Porcentaje de respuestas sobre los roles de describir y diseñar la arquitectura de software



Asimismo, al analizar las actividades, como se muestra en la Figura 26, se identificó que el 100% de los validadores consideran que el proceso IPARES, enfocado a describir y diseñar la arquitectura de software, propone actividades efectivas, suficientes, eficientes y que aportan al aseguramiento de la calidad

Figura 26

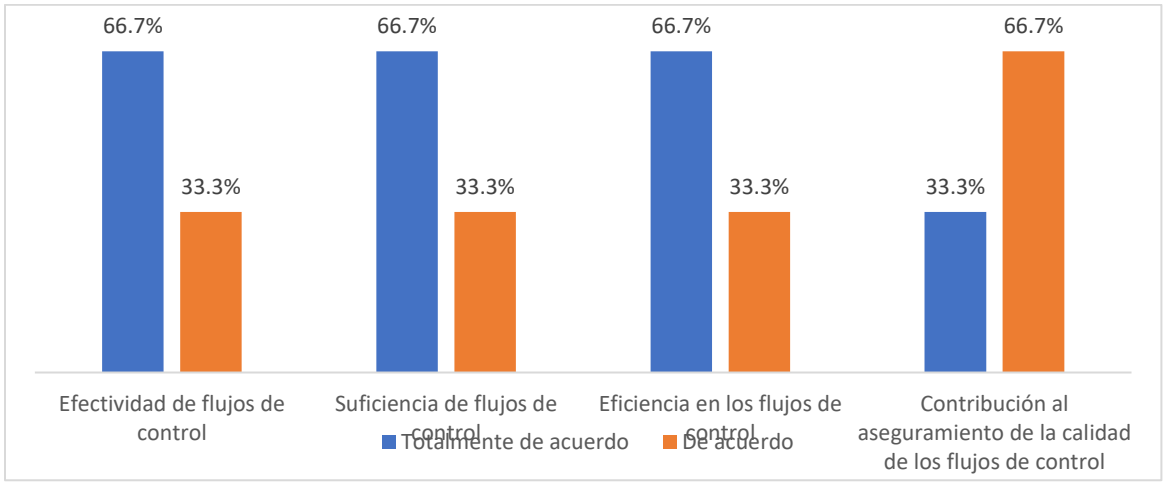
Porcentaje de respuestas sobre las actividades de describir y diseñar la arquitectura de software



De igual forma, al evaluar los flujos de control, como se muestra en la Figura 27, se identificó que el 100% de los validadores consideran que el proceso IPARES, orientado a describir y diseñar la arquitectura de software, presenta flujos de control suficientes, eficientes, efectivos, y que contribuyen significativamente al aseguramiento de la calidad.

Figura 27

Porcentaje de respuestas de flujos de control de describir y diseñar la arquitectura de software



El proceso para describir y diseñar la arquitectura de software, como se evidencia en la Figura 28, para el 100% de los validadores no resulta costoso de implementar en un equipo de desarrollo pequeño que utilice métodos ágiles.

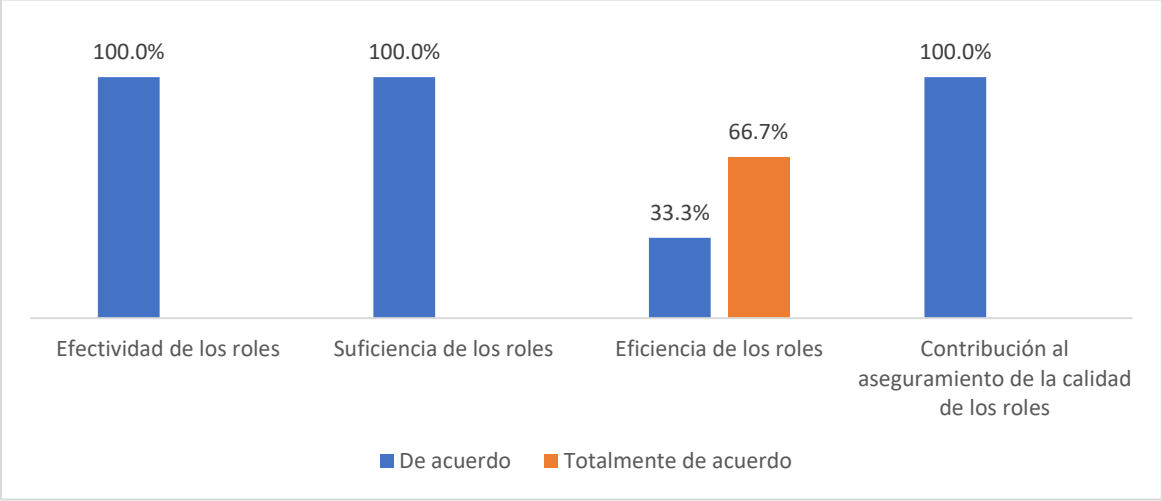
Figura 28

Porcentaje de respuestas de costo para describir y diseñar la arquitectura de software

2.4.4.3 Evaluar la arquitectura de software. Al examinar los roles, como se observa en la Figura 29, se determinó que el 100% de los validadores coinciden en que el proceso IPARES, enfocado en evaluar la arquitectura de software, define roles efectivos, suficientes, eficientes y que contribuyen de manera notable al aseguramiento de la calidad.

Figura 29

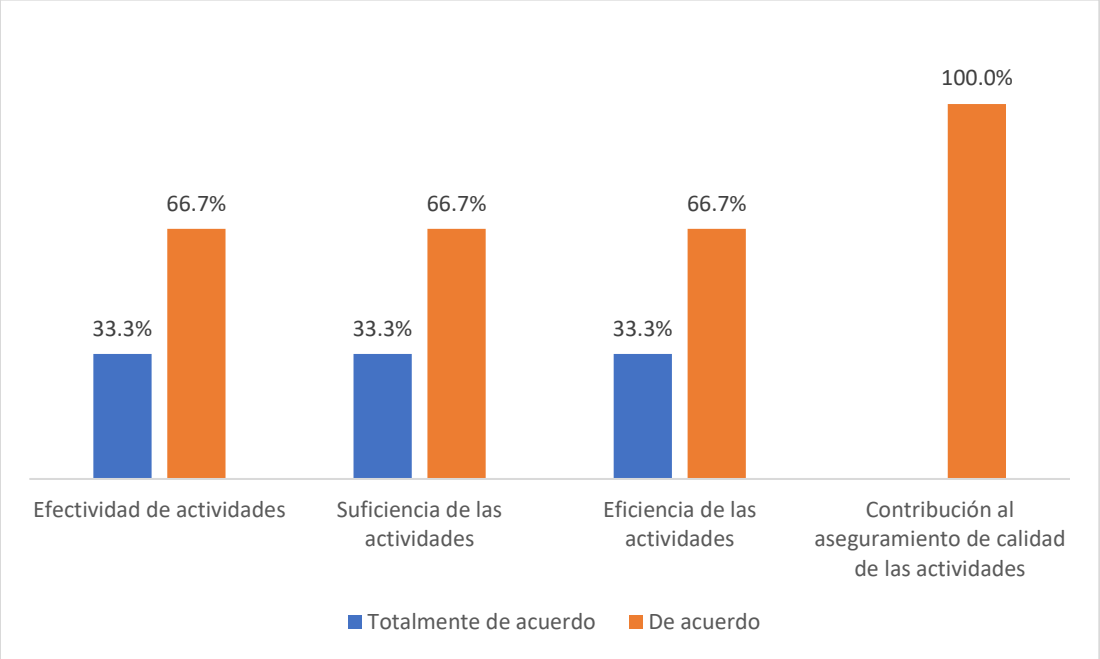
Porcentaje de respuestas sobre los roles de evaluar la arquitectura de software



De igual manera, al evaluar las actividades, como se observa en la Figura 30, se encontró que el 100% de los validadores opinan que el proceso IPARES, orientado a evaluar la arquitectura de software, establece actividades suficientes, efectivas, eficientes y que contribuyen de forma significativa al aseguramiento de la calidad.

Figura 30

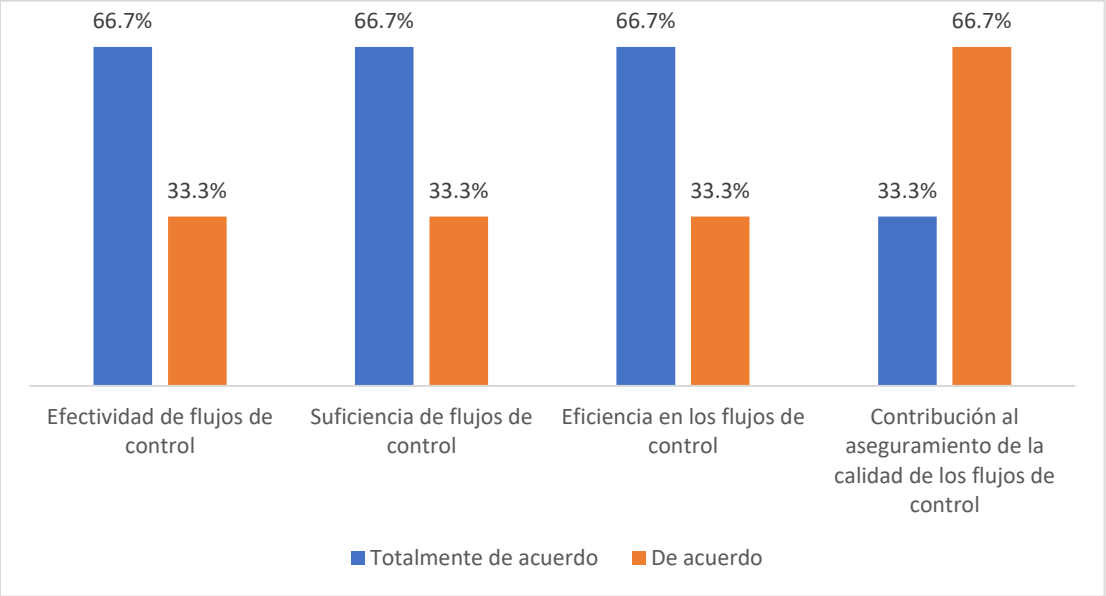
Porcentaje de respuestas sobre las actividades de evaluar la arquitectura de software



De manera similar, al analizar los flujos de control, como se ilustra en la Figura 31, se concluyó que el 100% de los validadores consideran que el proceso IPARES, enfocado en evaluar la arquitectura de software, ofrece flujos de control adecuados, efectivos y eficientes, que aportan significativamente al aseguramiento de la calidad.

Figura 31

Porcentaje de respuestas de flujos de control de evaluar la arquitectura de software



El proceso para describir y diseñar la arquitectura de software, como se evidencia en la Figura 32, para el 100% de los validadores no resulta costoso de implementar en un equipo de desarrollo pequeño que utilice métodos ágiles.

Figura 32

Porcentaje de respuestas de costo para evaluar la arquitectura de software

2.4.4.4 Aspectos positivos, aspectos por mejorar y aspectos por incorporar. Al realizar la aplicación del instrumento de recolección de información se tuvieron en cuenta tres aspectos fundamentales, aspectos positivos, aspectos a mejorar y aspectos a incorporar, las respuestas por parte de nuestros tres validadores fueron las que se muestran a continuación en la Tabla 32.

Tabla 32

Aspectos positivos, aspectos por mejorar y aspectos a incorporar

	Aspectos positivos	**Aspectos por mejorar**	**Aspectos por incorporar**
PS1 - Análisis del dominio del problema	Identificación de requerimientos arquitectónicos por parte del Product Owner (PO) para integración en el Product Backlog (PB)	Eliminar los id que tienen las prácticas porque tiendes a confundir	Hacer una puesta en común de conceptos sobre arquitectura de software en el equipo Scrum
	Involucrar en las tareas de arquitectura a los developers	Capacitar al equipo de desarrollo en temas de arquitectura de software de forma previa	
	La identificación de los roles		
PS2 - Describir y diseñar la arquitectura de software	Diseño de la arquitectura de manera colaborativa	Eliminar los id que tienen las prácticas porque tiendes a confundir	Unificar los conceptos sobre diseño en AS en los integrantes del equipo
	Establecimiento de prácticas acorde al papel que un arquitecto de software puede desempeñar en el proceso software	El conocimiento previo de los integrantes del equipo Scrum sobre AS	

	Aspectos positivos	**Aspectos por mejorar**	**Aspectos por incorporar**
PS3 - Evaluación de la arquitectura de software	Establecimiento de prácticas acorde al papel que un arquitecto de software puede desempeñar en el proceso software	Sugiero considerar los eventos de Scrum para especificar exactamente en qué momento se ejecutan las actividades o prácticas.	Analizar la viabilidad de incluir los stakeholders dentro de la evaluación, más si son estos los que aprueban el producto final.
	Evaluar la arquitectura de manera colaborativa	El conocimiento previo de los integrantes del equipo Scrum sobre AS	Unificar los conceptos sobre diseño en AS en los integrantes del equipo

El análisis de las respuestas muestra que los procesos evaluados en el cuadro destacan por fomentar la colaboración y la implementación de buenas prácticas. En particular, la identificación de requisitos arquitectónicos, la participación de los desarrolladores y el enfoque colaborativo en el diseño y evaluación de la arquitectura son aspectos positivos que fortalecen la integración y alineación dentro de los equipos Scrum. Además, el establecimiento de prácticas claras acorde al rol del arquitecto de software asegura que las responsabilidades estén bien definidas, lo cual beneficia al equipo.

Sin embargo, se identifican áreas que requieren mejora, como la confusión generada por el uso de identificadores en las prácticas, lo cual afecta su claridad y comprensión. Otro punto relevante es la falta de conocimiento previo sobre arquitectura de software por parte de los integrantes del equipo Scrum, lo que podría limitar la efectividad de los procesos. Además, en la evaluación de la arquitectura, no se especifican claramente los momentos en los que las actividades o prácticas deben ejecutarse dentro de los eventos de Scrum, lo que podría dificultar su integración con metodologías ágiles.

Por último, se señala la importancia de unificar los conceptos sobre arquitectura y diseño de software entre los miembros del equipo para mejorar la comunicación y evitar malentendidos.

También se destaca la necesidad de analizar la inclusión de los stakeholders en la evaluación de la arquitectura, ya que su participación puede contribuir a garantizar que los objetivos del producto final estén alineados con las expectativas del cliente. Abordar estas áreas de mejora fortalecería significativamente la implementación de los procesos IPARES en equipos de desarrollo.

2.4.5 Síntesis de los resultados

La validación del proceso de Incorporación de Prácticas de Arquitectura en un Equipo Scrum (IPARES) fue desarrollada en cuatro etapas principales.

En la primera etapa de selección de validadores se establecieron criterios de inclusión y exclusión para garantizar la idoneidad de los validadores, entre los criterios destacaron la experiencia profesional de los validadores (mínimo 3 años de experiencia), el uso de metodologías agiles (SCRUM) y conocimiento de prácticas de arquitectura de software. De un grupo inicial de 6 candidatos se seleccionaron 3 validadores expertos que cumplieron con estos criterios.

En la siguiente etapa se diseñó un cuestionario basado en el protocolo propuesto por (Guerrero y Hernández, 2023), organizado en 4 secciones:

- **Generalidades**: Se Informa sobre el contexto del cuestionario.
- **Información sociodemográfica**: Recolección de datos básicos de los validadores.
- **Aspectos positivos de IPARES**: Identificación de las fortalezas de los procesos.
- **Aspectos por mejorar e incorporar de IPARES**: Identificación de elementos a optimizar y nuevos a considerar.

El instrumento incluyó 48 ítems entre los cuales se destacan preguntas cerradas (escala Likert) y preguntas abiertas, el instrumento tuvo un tiempo promedio de diligenciamiento de 20 minutos por cada validador.

El instrumento fue aplicado en reuniones individuales con cada uno de los validadores entre el 2 y 20 de noviembre de 2024. Durante las sesiones, se realizó la presentación del contexto del

problema, los detalles del proceso IPARES y los roles, actividades y flujos de control del análisis del dominio del problema, describir y diseñar la arquitectura de software y evaluar la arquitectura de software.

Los datos recolectados fueron agrupados en tres dimensiones clave por cada uno de los procesos de IPARES

- **Analizar el dominio del problema**: Los roles, actividades y flujos de control fueron percibidos como efectivos, suficientes, eficientes y que contribuían al aseguramiento de la calidad por el 100% de los validadores, además el costo no es considerado elevado para equipos pequeños agiles.

- **Describir y diseñar la arquitectura de software:** los roles, actividades y flujos de control recibieron una valoración positiva del 100% de los validadores en términos de eficiencia, efectividad, suficiencia y contribución al aseguramiento de calidad, de igual forma el costo no es considerado elevado para equipos pequeños agiles.

- **Evaluar la arquitectura de software:** los role, actividades y flujos de control son eficientes, efectivos, suficientes y contribuyen al aseguramiento de la calidad según el 100% de los validadores, también los validadores determinaron que el costo no es elevado para equipos pequeños agiles.

En el instrumento se agregaron preguntas abiertas para encontrar aspectos positivos, aspectos por mejorar y aspectos por incorporar, los validadores resaltaron para los aspectos positivos la identificación de requisitos y la definición clara de los roles en los procesos IPARES, para los aspectos a mejorar, los validadores determinaron que se debe incrementar el conocimiento del equipo sobre la arquitectura de software antes de aplicar el proceso y especificar los momentos en los que se ejecutan las actividades en los eventos de scrum, y por último en los aspectos por incorporar, los validadores resaltaron la incorporación del rol de stakeholders para la evaluación de la arquitectura de software, además de unificar los conceptos de diseño de arquitectura de software entre los miembros del equipo scrum.

3. Conclusiones

El mapeo sistemático de literatura efectuado en cuatro fuentes de información (*ACM Digital Library, IEEExplorer Digital Library, Scopus* y *Springer*) nos permitió efectuar los criterios de inclusión y exclusión definidos en la Tabla 7 a 491 artículos extraídos mediante las cadenas de búsqueda definidas en la Tabla 5, de estos se lograron identificar 12 artículos de los cuales se extrajeron 40 prácticas de arquitectura de software que fueron categorizadas en 8 grupos, destacando entre ellos análisis y documentación y gestión y planificación como los más relevantes.

Se definieron 14 competencias esenciales asociadas a las 40 practicas, siendo las competencias las más influyentes: Diseñar una propuesta arquitectónica utilizando métodos, técnicas y patrones de diseño para establecer la dirección técnica del proyecto, documentar un diseño arquitectónico utilizando artefactos utilizando artefactos(formatos, diagramas y visualizaciones) basados en estándares y herramientas para garantizar la comunicación efectiva y evaluar una arquitectura de software utilizando un marco estructurado, principios de diseño arquitectónico, patrones de diseño y herramientas para validar el diseño técnico. ya que abarcan un mayor número de actividades críticas.

El modelado de procesos y la notación BPMN brindaron claridad para poder estructurar y alinear las actividades, roles y flujos de control con objetivos organizacionales, aunque su aplicación en Scrum requiere ajustes para abordar su naturaleza iterativa y ágil

Se logro integrar el 27% de las practicas propuestas en las actividades Scrum, destacándose la priorización de requerimientos no funcionales, la definición de objetivos arquitectónicos claros, la documentación detallada y la evaluación de riesgos arquitectónicos.

El proceso IPARES fue evaluado positivamente por los validadores expertos en arquitectura de software, quienes resaltaron la eficiencia, efectividad, suficiencia y la contribución al aseguramiento de calidad de los roles, actividades y flujos de control en las tres etapas principales: analizar el dominio del problema, describir y diseñar la arquitectura de software y evaluar la arquitectura de software.

Sin embargo, en el proceso de validación solo se pudo realizar a tres personas expertas en arquitectura de software, por lo que no se pudo realizar el proceso de validación dentro de equipos scrum o con más expertos en arquitectura.

Entre los aspectos a mejorar, destacan la necesidad de aumentar el conocimiento del equipo scrum sobre arquitectura de software y definir con mayor claridad los momentos de aplicación de las actividades dentro de los eventos de scrum

Los resultados obtenidos en este estudio demuestran que es posible incorporar las prácticas de un arquitecto de software en un equipo Scrum que desarrolla proyectos medianamente complejos. La identificación de prácticas clave, la definición de competencias necesarias y la validación del proceso IPARES han evidenciado que estas actividades pueden integrarse de manera eficiente y efectiva, alineándose con los principios ágiles.

Si bien se identificaron áreas de mejora, como la necesidad de capacitar al equipo en arquitectura de software y ajustar ciertos conceptos a la naturaleza ágil del marco, los resultados confirman que esta integración es factible y representa un aporte significativo al desarrollo de proyectos de software en contextos reales.

4. Recomendaciones

Incorporar el rol de stakeholders para evaluar la arquitectura de software y unificar conceptos en el diseño de arquitectura entre los miembros del equipo scrum.

Aunque el proceso fue validado en un entorno experimental, se sugiere implementar su uso en empresas de desarrollo de software que trabajen bajo Scrum en proyectos reales. Esto permitirá obtener datos más representativos y medir su efectividad en un contexto empresarial.

Se recomienda ampliar el estudio a otros marcos agiles, aunque el enfoque principal es Scrum, se puede evaluar la aplicabilidad de las practicas propuestas en otros marcos agiles como Kanban o SAFe (Scaled Agile Framework). Esto ampliara el alcance y relevancia del modelo.

Estas recomendaciones no solo buscan perfeccionar el alcance del documento, si no también contribuir al avance de conocimiento y la práctica en la integración de arquitectura de software en metodologías agiles de desarrollo.

Referencias bibliográficas

Abrahamsson, P., Ali Babar, Muhammad & Kruchten, Philippe (2010). Agility And Architecture: Can They Coexist (n.d.). *IEE Computer Society*, *16*,*22*.

Angelov, S., Meesters, M., & Galster, M. (2016). Architects in scrum: What challenges do they face? *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9839 LNCS*(November), 229–237. https://doi.org/10.1007/978-3-319-48992-6_17

Babar, M. A., Brown, A. W., & Mistrík, I. (Eds.). (2014). *Agile software architecture: Aligning agile processes and software architectures*. Newnes.

Barón, A. (2019). Modelo para la Definición Unificada de la Práctica como Constructo Teórico en Ingeniería de Software. Universidad Nacional de Colombia.

Bass, L., Clements, P., & Kazman, R. (2013). Software Architecture in Practice Second Edition Third Edition. In *Communication*. https://www.oreilly.com/library/view/software-architecture-in/9780132942799/

Cadavid, A. N., Martínez, J. D. F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, *11*, 33.

CollabNet VersionOne. (2019). The 13th annual STATE OF AGILE Report - 2019. *Digital.Ai*, *13*, 16. https://stateofagile.com/#ufh-i-613553418-13th-annual-state-of-agile-report/7027494

D. Tofan, M. Galster, and P. Avgeriou, “Difficulty of Architectural Decisions – A Survey with Professional Architects,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), no. 2, 2013, pp. 192–199.

Digital.ai Software, I. (2021). 15th State of Agile Report. *Digital.Ai*, 1–23.

https://stateofagile.com/#

Digital.ai. (2021). 14th Annual State of Agile Report. *Angewandte Chemie International Edition, 6(11), 951–952.*, 2013–2015.

Eloranta, V. P., & Koskimies, K. (2013). Lightweight Architecture Knowledge Management for Agile Software Development. In *Agile Software Architecture: Aligning Agile Processes and Software Architectures*. Elsevier Inc. https://doi.org/10.1016/B978-0-12-407772-0.00007-1

Fernández, S. F., Sánchez, J. M. C., Córdoba, A., & Largo, A. C. (2002). Estadística descriptiva. Esic Editorial.

Galster, M., Angelov, S., Martínez-Fernández, S., & Tofan, D. (2017). Reference architectures and scrum: Friends or foes? *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Part F1301*, 896–901. https://doi.org/10.1145/3106237.3117773

Galster, M., Angelov, S., Meesters, M., & Diebold, P. (2016). A multiple case study on the architect's role in Scrum. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10027 LNCS*(September), 432–447. https://doi.org/10.1007/978-3-319-49094-6_29

Guerrero Calvache, Sandra Marcela (2023) *Modelo de evaluación de productividad de equipo en el desarrollo ágil de software.* Maestría thesis, Universidad de Nariño.

Guerrero-Calvache, M., & Hernández, G. (2023). Un estudio exploratorio de las percepciones de productividad en equipos de software ágil. *TecnoLógicas, 26*(56).

Hernández, G., Martínez, Á., Jiménez, F., Jiménez, R., & Baron, A. (2021, October). Learning factory for the Software Engineering area: First didactic transformation. In 2021 XLVII Latin American Computing Conference (CLEI) (pp. 1-8). IEEE.

Huberth, D., & Garcia, R. (n.d.). *Importante*. *Gestión de Proyectos de Desarrollo de Software*

Jones, C. (2014). Software Engineering Best Practices. In Paper Knowledge. Toward a Media History of Documents. McGraw-Hill.

Kitchenham, B. A., & Pfleeger, S. L. (2008). Personal opinion surveys. In *Guide to advanced empirical software engineering* (pp. 63-92). London: Springer London.

Lopes, S. V. F., & Junior, P. T. A. (2017). Architectural design group decision-making in agile projects. *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, 210–215. https://doi.org/10.1109/ICSAW.2017.24

Mekni, M., Buddhavarapu, G., Chinthapatla, S., & Gangula, M. (2018). Software Architectural Design in Agile Environments. *Journal of Computer and Communications*, *06*(01), 171–189. https://doi.org/10.4236/jcc.2018.61018

Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., & Rueda, J. R. (2018). *Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura*. 635–639. http://sedici.unlp.edu.ar/handle/10915/67795

Number, O. M. G. D., & File, M. R. (2015). *SMSC / 15-12-02 Essence – Kernel and Language for Software Engineering Methods*. *December*.

Paitán, H. Ñ., Mejía, E. M., Ramírez, E. N., & Paucar, A. V. (2014). Metodología de la investigación cuantitativa-cualitativa y redacción de la tesis. Ediciones de la U.

Rizzo Vargas, G. M., Colina Vargas, A. M., & Túa Ollarves, J. J. (2020). Optimización en la Gestión de Registro y Control de Fichas Estudiantiles del Departamento de Bienestar Estudiantil. *INNOVA Research Journal*, *5*(3), 122–134. https://doi.org/10.33890/innova.v5.n3.2020.1358

S. Rekha V. and H. Muccini, “Suitability of Software Architecture Decision Making Methods for Group Decisions,” in Software Architecture - 8th European Conference, ECSA 2014, Vienna, Austria, August 25-29, 2014. Proceedings, 2014, pp. 17–32.

Schwaber, K & Sutherland, J. (2020). The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game. *Scrum Alliance*,

Schwaber, K., & Sutherland, J. (2020). *Scrum Guide V7*. *November*, 133–152.

Subih, M. A., Malik, B. H., Mazhar, I., Izaz-ul-Hassan, Sabir, U., Wakeel, T., Ali, W., Yousaf, A., Bilal-bin-Ijaz, Nawaz, H., & Suleman, M. (2019). Comparison of agile method and scrum method with software quality affecting factors. *International Journal of Advanced Computer Science and Applications*, *10*(5), 531–535. https://doi.org/10.14569/ijacsa.2019.0100569

Submission, R. (2012). *Essence – Kernel and Language for Software Engineering Methods*. *August 1*–207. http://www.omg.org/cgi-bin/doc?ad/2012-08-15/PDF%5Cnhttp://www.omg.org/spec/Essence/

Tamayo, M. (2004). El proceso de la investigación científica. Editorial Limusa.

Velasco-Elizondo, P. (2021). Software Architecture: Developing Knowledge, Skills, and Experiences. In Latin American Women and Research Contributions to the IT Field (pp. 217-239). IGI Global.

Wedemann, G. (2018). Scrum as a method of teaching software architecture. *ACM International Conference Proceeding Series*, 108–112. https://doi.org/10.1145/3209087.3209096

Woods, E. (2015). Aligning Architecture Work with Agile Teams. *IEEE Software*, *32*(5), 24–26. https://doi.org/10.1109/MS.2015.119

Yang, C., Liang, P., & Avgeriou, P. (2016). A systematic mapping study on the combination of

software architecture and agile development. *Journal of Systems and Software*, *111*, 157–184. https://doi.org/10.1016/j.jss.2015.09.028

Yang, C., Liang, P., Avgeriou, P., AlOmar, E. A., Mkaouer, M. W., Ouni, A., Kassab, M., Alhubaishy, A., Benedicenti, L., Alsaqaf, W., Daneva, M., Wieringa, R., Cakir, C., Cetin, F., Savasci, M., & Findik, O. (2018). Software Architecture Documentation in Agile. *ACM International Conference Proceeding Series*, *December*, 0–3.