



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado

Facultad de Ingeniería de Sistemas e Informática

Unidad de Posgrado

**Desafíos de la ingeniería de requisitos en el desarrollo
de software ágil: un enfoque en la mejora del producto**

TESIS

Para optar el Grado Académico de Magíster en Ingeniería de
Sistemas e Informática con mención en Ingeniería de Software

AUTOR

Oscar BENITO PACHECO

ASESOR

Dr. Percy Edwin DE LA CRUZ VÉLEZ DE VILLA

Lima, Perú

2025



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Benito, O. (2025). *Desafíos de la ingeniería de requisitos en el desarrollo de software ágil: Un enfoque en la mejora del producto*. [Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Oscar Benito Pacheco.
Tipo de documento de identidad	DNI
Número de documento de identidad	06242587
URL de ORCID	https://orcid.org/0000-0003-1157-7764
Datos de asesor	
Nombres y apellidos	Percy Edwin De la Cruz Vélez de Villa
Tipo de documento de identidad	DNI
Número de documento de identidad	08583141
URL de ORCID	https://orcid.org/0000-0002-4943-7620
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Santiago Domingo Moquillaza Henriquez.
Tipo de documento	DNI
Número de documento de identidad	08280889
Miembro del jurado 1	
Nombres y apellidos	Ciro Rodriguez Rodriguez .
Tipo de documento	DNI
Número de documento de identidad	06020241
Miembro del jurado 2	
Nombres y apellidos	Hugo Froilán Vega Huerta.
Tipo de documento	DNI
Número de documento de identidad	06147737CO / 9955249956

Miembro del jurado 3	
Nombres y apellidos	Luis Angel Guerra Grados.
Tipo de documento	DNI
Número de documento de identidad	15644548
Datos de investigación	
Línea de investigación	No aplica
Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento.
Ubicación geográfica de la investigación	Edificio: Universidad Nacional Mayor de San Marcos País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Calle: Av. Carlos Germán Amezaga #375 Latitud: -12.05611 Longitud: -77.08444
Año o rango de años en que se realizó la investigación	Julio 2023 - agosto 2025
URL de disciplinas OCDE	Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

**ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE
MAGÍSTER EN INGENIERÍA DE SISTEMAS E INFORMÁTICA CON MENCIÓN EN
INGENIERÍA DE SOFTWARE**

A los diecisiete (17) días del mes de diciembre de 2025, siendo las 4:00 pm., se reunieron en el Auditorio Alfredo Celso Alva Bravo, el Jurado de Tesis conformado por los siguientes docentes:

Mg. Santiago Domingo Moquillaza Henriquez (Presidente)
Dr. Ciro Rodriguez Rodriguez (Miembro)
Mg. Luis Angel Guerra Grados (Miembro)
Dr. Hugo Froilán Vega Huerta (Miembro)
Dr. Percy Edwin De la Cruz Vélez de Villa (Miembro Asesor)

Se inició la Sustentación invitando al candidato a Magíster **OSCAR BENITO PACHECO**, para que realice la exposición oral de la tesis para optar el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software, siendo la Tesis intitulada:

**“DESAFÍOS DE LA INGENIERÍA DE REQUISITOS EN EL DESARROLLO DE SOFTWARE
ÁGIL: UN ENFOQUE EN LA MEJORA DEL PRODUCTO”.**

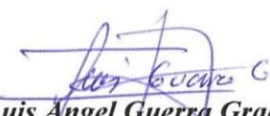
Concluida la exposición, los miembros del Jurado de Tesis procedieron a formular sus preguntas que fueron absueltas por el graduando; acto seguido se procedió a la evaluación correspondiente, habiendo obtenido la siguiente calificación:

Excelente (A) *Muy bueno*

Por tanto, el presidente del Jurado, de acuerdo con el Reglamento General de Estudios de Posgrado, otorga al Bachiller **OSCAR BENITO PACHECO** el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software.

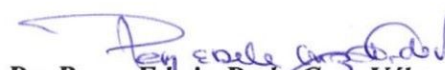
Siendo las 5:30 horas, el presidente del Jurado de Tesis, da por concluido el acto académico de Sustentación de Tesis.


Mg. Santiago Domingo Moquillaza Henriquez
(Presidente)


Mg. Luis Angel Guerra Grados
(Miembro)


Dr. Ciro Rodriguez Rodriguez
(Miembro)


Dr. Hugo Froilán Vega Huerta
(Miembro)


Dr. Percy Edwin De la Cruz Vélez de Villa
(Miembro Asesor)



UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, Percy Edwin De la Cruz Vélez de Villa en mi condición de asesor acreditado con Dictamen N°00429-2023-UPG-VDIP-FISI/UNMSM de la tesis, cuyo título es ***Desafíos de la Ingeniería de Requisitos en el Desarrollo de Software Ágil: Un Enfoque en la Mejora del Producto***, presentado por el bachiller don: *Oscar Benito Pacheco*, para optar el grado de, Maestría en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software. CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 11% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del grado.

Lima, 9 de abril de 2025

Firma del Asesor

Nombres y Apellidos del asesor: Percy Edwin De la Cruz Vélez de Villa

DNI 08583141

DEDICATORIA

A mi familia, especialmente a mis hijos: Jesús Sebastián, Olga Verónica y Sofía Milagros; a mis estudiantes y a mis colegas, quienes me incentivaron a dar un paso más en mi desarrollo profesional e intelectual

AGRADECIMIENTO

Al Magister Percy De la Cruz Vélez de Villa, una persona muy comprometida con la educación, sinónimo de apoyo incondicional, gracias a él pude concretar la investigación. A los colegas de los diferentes grupos de investigación donde participo quienes aportaron con sus observaciones críticas y aportes al presente trabajo que ya ve la luz de una realidad.

INDICE

DEDICATORIA.....	II
AGRADECIMIENTO.....	III
INDICE.....	IV
ÍNDICE DE TABLAS.....	VI
ÍNDICE DE ILUSTRACIONES.....	VII
RESUMEN.....	VIII
ABSTRACT.....	IX
1 INTRODUCCIÓN.....	1
1.1 Situación Problemática.....	1
1.2 Formulación del Problema.....	2
1.2.1 Problema general.....	2
1.2.2 Problemas específicos.....	3
1.3 Justificación.....	3
1.3.1 Justificación teórica.....	4
1.3.2 Justificación práctica.....	4
1.4 Objetivos.....	5
1.4.1 Objetivo general.....	5
1.4.2 Objetivos específicos.....	5
1.5 Estructura de la Tesis.....	6
2 MARCO TEÓRICO.....	7
2.1 Marco Filosófico de la Investigación.....	7
2.1.1 Paradigma epistemológico.....	7
2.1.2 Fundamentos teóricos.....	8
2.1.3 Perspectiva ontológica.....	8
2.1.4 Perspectiva axiológica.....	8
2.1.5 Perspectiva interdisciplinaria.....	8
2.1.6 Metodología de la investigación.....	8
2.1.7 Conclusiones esperadas.....	9
2.2 Desarrollo de Software Tradicional.....	9
2.2.1 Modelo en cascada.....	9
2.2.2 Ingeniería de requisitos tradicional.....	10
2.3 Desarrollo de Software Ágil.....	10
2.3.1 Concepto proceso ágil.....	11
2.3.2 Valores principales del desarrollo de software ágil.....	11
2.3.3 Principios básicos del desarrollo de software ágil.....	12
2.4 Métodos Ágiles en el Ambiente de desarrollo de Software Ágil.....	14
2.4.1 El enfoque Scrum.....	14
2.4.2 El enfoque programación extrema (XP).....	15
2.5 El rol del Cliente en el Desarrollo de Software Ágil.....	19
2.6 Ingeniería Requisitos.....	20
2.6.1 Ingeniería de requisitos tradicionales.....	20
2.6.2 Prácticas de ingeniería de requisitos ágiles.....	24
2.6.3 Documentación en ingeniería de requerimientos ágiles.....	26
3 METODOLOGÍA.....	30
3.1 Diseño y Realización de la Investigación.....	30

3.2	Revisión Sistemática de la Literatura (RSL).....	31
3.3	Identificación de la Necesidad de una Revisión	32
3.4	Preguntas de Investigación Relacionadas	32
3.5	Estrategia de Búsqueda	34
3.5.1	Palabras clave	34
3.5.2	Fuentes de búsqueda	35
3.5.3	Selección de estudio.....	36
3.5.4	Evaluación de la calidad del estudio.....	37
3.5.5	Extracción y síntesis de datos	38
3.6	Entrevista a Profesionales en el Ambiente de Desarrollo Ágil.....	38
4	RESULTADOS Y ANÁLISIS	43
4.1	Resultados de la Revisión Sistemática de la Literatura (RSL)	43
4.2	Resultados de RSL	46
4.2.1	Pregunta de investigación 1	46
4.2.2	Pregunta de investigación 2	47
4.2.3	Pregunta de investigación 3	51
4.2.4	Pregunta de investigación 4	54
4.2.5	Análisis metodológico de resultados.....	55
4.3	Entrevistas, Resultados y Análisis	57
4.3.1	Perfiles de los entrevistados.....	57
4.3.2	Caracterización del proyecto	59
4.3.3	Caracterización del equipo.....	60
4.3.4	Contribuciones	61
5	DISCUSIÓN DE RESULTADOS	64
5.1	Desafíos identificados en Entornos Ágiles	66
5.2	Prácticas en el Proceso de Ingeniería de Requisitos Ágiles	68
5.3	Artefactos de la Práctica en el Proceso de Ingeniería de Requisitos Ágiles.....	70
6	EXHORTACIÓN SOBRE LA INVESTIGACIÓN	74
6.1	Amenazas de la Revisión Sistemática de la Literatura	74
6.1.1	Búsqueda y selección de estudios	74
6.1.2	Extracción y síntesis de datos	75
6.2	Amenazas sobre las Entrevistas	75
6.2.1	Validez de constructo.....	75
6.2.2	Escasez de entrevistados	75
7	CONCLUSIONES Y TRABAJOS FUTUROS	76
7.1	Conclusiones.....	76
7.1.1	Preguntas de investigación revisadas.....	76
7.2	Trabajos Futuros	80
	BIBLIOGRAFIA	81
	APÉNDICES	86
	Apéndice 1 Matriz de consistencia.....	1
	Apéndice 2 Matriz de operacionalización	2
	Apéndice 3 Fuentes incluidas en la RSL	1
	Apéndice 4 Guía de entrevista.....	3

ÍNDICE DE TABLAS

Tabla 1	Problemas específicos.....	3
Tabla 2	Justificación teórica	4
Tabla 3	Justificación práctica	4
Tabla 4	Objetivos específicos.....	5
Tabla 5	Mapeo de objetivos específicos / Problemas específicos	5
Tabla 6	Escenario metodológico de la investigación.....	31
Tabla 7	Etapas de la revisión sistemática.	32
Tabla 8	Preguntas de investigación	33
Tabla 9	Palabras claves.....	35
Tabla 10	Fuentes de búsqueda.....	35
Tabla 11	Criterios de inclusión.....	36
Tabla 12	Criterios de exclusión	36
Tabla 13	Lista de verificación de evaluación de calidad	37
Tabla 14	Extracción de datos.....	38
Tabla 15	Criterios de filtro de las fuentes.....	44
Tabla 16	Tipo de contribuciones	45
Tabla 17	Metodologías	46
Tabla 18	Perfil de profesionales entrevistados	58
Tabla 19	Resumen de los desafíos y prácticas en IRA.....	79

ÍNDICE DE ILUSTRACIONES

Ilustración 1	Metodología en tradicional (en cascada)	10
Ilustración 2	Modelo típico de un proceso de desarrollo de Scrum.....	15
Ilustración 3	Fases del ciclo de vida de un proyecto XP	18
Ilustración 4	Procesos y resultado del estudio de selección.....	45

RESUMEN

La implementación de las diferentes actividades en la disciplina de la ingeniería de requisitos en el desarrollo de un proyecto informático consiste en: obtener, analizar, documentar y verificar las necesidades de parte del usuario estas son por sí mismas de vital importancia en todas las metodologías de desarrollo del producto software, para el éxito del proyecto. Cabe destacar que las prácticas de la ingeniería de requisitos en los ambientes de desarrollo ágil, presentan restricciones y con ello varios desafíos en la industria del software, ayudan a abordar las dificultades de las propuestas convencionales y evitan costos innecesarios. Sin embargo, debe destacarse, la no existencia de una separación clara de estas actividades en sus procesos individuales, y que las prácticas que se implementan se utilizan muy de cerca durante el ciclo de desarrollo y dan bienvenida a los requisitos cambiantes en su proceso de desarrollo. Utilizando las normas y patrones en la teoría de la Revisión Sistemática de la Literatura, el estudio adaptó las pautas de Kitchenham propuestas para la disciplina de la Ingeniería de Software. Los resultados obtenidos han permitido identificar y conocer el estado actual de las técnicas de la ingeniería de requisitos implícitas o explícitamente de las prácticas que se vienen implementando al respecto en los ambientes de desarrollo ágil. Por tanto, esta contribución identifica desafíos, prácticas y resultados y a la vez que sirvan en la industria del software.

Palabras claves: Ingeniería de software, metodologías de desarrollo de software, ingeniería de requisitos, metodologías ágiles.

ABSTRACT

The implementation of the different activities in the discipline of requirements engineering in the development of a computer project consists of: obtaining, analyzing, documenting and verifying the needs of the user, these are in themselves of vital importance in all methodologies of development of the software product, for the success of the project. It should be noted that requirements engineering practices in agile development environments present restrictions and thus several challenges in the software industry, helping to address the difficulties of conventional proposals and avoiding unnecessary costs. However, it should be highlighted that there is no clear separation of these activities into their individual processes, and that the practices that are implemented are used very closely during the development cycle and welcome changing requirements in their development process. The results obtained have made it possible to identify and understand the current state of requirements engineering techniques, implicitly or explicitly, of the practices that are being implemented in this regard in agile development environments. Therefore, this contribution identifies challenges, practices and results that serve the software industry.

Keywords: Software engineering, software development methodologies, requirements engineering, agile methodologies.

1 INTRODUCCIÓN

La rápida evolución de la tecnología y las demandas del mercado han puesto de manifiesto la necesidad de adaptarse en el desarrollo de software. La calidad del producto es a menudo comprometida por diversas razones, incluyendo la falta de apoyo de la alta dirección y la insuficiente planificación de proyectos. En este contexto, surge la necesidad de explorar los desafíos de la ingeniería de requisitos en las metodologías ágiles, que han ganado relevancia en la última década. Esta tesis se centra en identificar problemas específicos y prácticas para mejorar la calidad del software, proponiendo que la implementación de enfoques ágiles puede facilitar una mayor flexibilidad, colaboración y satisfacción del cliente, asegurando así el éxito en los proyectos de desarrollo de software.

1.1 Situación Problemática

En medio de una sociedad cambiante y como consecuencia, el mundo evoluciona cada segundo; En este contexto las transformaciones ocurren muy rápido y continuamente; las empresas deben responder rápidamente a este fenómeno, y las empresas de desarrollo de software no son una excepción, por lo tanto, es perentorio alinearse a este contexto de cambio.

El producto software materia del estudio generalmente carece de calidad, algunos de los problemas son debido a la falta de apoyo decidido de la alta dirección en la ejecución de su desarrollo, a la carencia de planificación y gestión en proyectos informáticos, desconocimiento de técnicas y métodos de desarrollo de software, expectativas de los usuarios fuera de la realidad, los requerimientos del usuario generalmente son incompletos y están en el tiempo tienden a cambiar, poco involucramiento de parte del usuario, en muchos casos la falta de recursos profesionales. Todo ello resumiendo en proyectos en fracaso por falta de calidad, costos en exceso y concluidos fuera del cronograma preestablecido.

En consecuencia, se hace perentorio plantear alternativas que se supere este conjunto de problemas, entendiéndose que el desarrollo del hardware en capacidad de almacenamiento, velocidad de procesamiento se han incrementado considerablemente, al igual que el desarrollo de los medios de comunicación.

Los requerimientos de los usuarios en los proyectos de desarrollo de software evolucionan rápidamente; factores como el avance de la tecnología, las demandas del mercado, los productos de las competencias, etc. pueden hacer que un proyecto que apenas esté terminado ya esté desactualizado; los requisitos de un usuario evolucionan dentro del ciclo de vida de un proyecto, por tanto es una necesidad que se debe dar atención a ello para lograr el éxito de los mismos, por tanto, no atenderlo significa concluir en la disconformidad del usuario.

El desarrollo de software es un proceso complejo, elaborado en contextos bastante distantes de la realidad, con el proceso de transformación que se vive. Las metodologías tradicionales de desarrollo de software como en cascada (Royce, 1970) proponen principalmente un desarrollo en fases con un enfoque estático. Su visión del proyecto es vertical y la mayoría de ellos están estructurados principalmente de forma secuencial; aunque enfoques como el modelo espiral (Boehm & Turner, 2003) o el proceso unificado racional (Rational Rose, 1998) intentaron resolver los problemas del método en cascada y presentan algunas variaciones en las fases tradicionales (Abbas et al., 2008). Estos enfoques tradicionales realizan un conjunto de tareas de "pre-construcción" donde se realiza el análisis del problema en sí mismo, en esta etapa el objetivo es lograr una comprensión completa y una definición inequívoca del problema; De esta forma, se puede diseñar, implementar y desplegar una solución adjunta a la definición estática del problema.

A pesar de que esta forma de pensar es probablemente la más estructurada, pero debe resolverse la realidad cambiante en todos los contextos y por consiguiente en el ciclo de vida de un proyecto, y es así que surge las metodologías ágiles de desarrollo de software y vienen tomando protagonismo a partir de este nuevo milenio.

1.2 Formulación del Problema

1.2.1 Problema general

¿Cuáles son los desafíos de la ingeniería de requisitos en las metodologías ágiles en pro del éxito del proyecto “producto software”?

1.2.2 Problemas específicos

Para el trabajo de investigación, luego de haber expuesto la problemática se identifican los problemas específicos detallados en la Tabla 1.

Tabla 1

Problemas específicos

Problemas específicos	Descripción
PE.1	¿Qué entienden los profesionales por ingeniería de requisitos en el contexto de un proyecto ágil?
PE.2	¿Qué desafíos con respecto a ingeniería de requisitos se encuentran en proyectos ágiles?
PE.3	¿Qué prácticas / enfoques se siguen para superar estos desafíos?
PE.4	¿Qué efectos producen estas prácticas / enfoques en el proyecto

1.3 Justificación

La ingeniería de requisitos es una actividad fundamental en la elaboración del producto software, en todas las metodologías del desarrollo de software, por tanto, el no dar la debida atención a esta disciplina, se tiene como consecuencia un producto de mala calidad y el proyecto califica como fracaso para el cliente (Sommerville, 2016).

Por otra parte, la vertiginosa competencia y los procesos comerciales complejos vienen impulsando a las empresas de Tecnologías de Información (TI) a obtener una ventaja competitiva al reducir los plazos de entrega. Por lo tanto, las empresas buscan cada vez más formas de ser más receptivas y flexibles en su metodología y requisitos. Las prácticas de desarrollo ágil se han vuelto ampliamente aceptadas como una clase efectiva de enfoques para la gestión de proyectos con el fin de tener una entrega rápida y exitosa del software y de alta calidad. Usualmente el trabajo de los equipos que adoptan una metodología ágil es flexible y cambiante. En comparación con los procesos de software tradicionales, el desarrollo ágil está menos centrado en documentos y más orientado al código, adaptativo en lugar de predictivo y orientado a las personas en lugar de a los procesos (Beck et al., 2001).

Las prácticas en la Ingeniería Requisitos (IR) tradicionales como: observaciones, entrevistas, talleres y fuerte colaboración en equipo es de un hacer cotidiano en los métodos ágiles basados en la iteración. Del mismo modo, las prácticas de IR como el involucramiento del cliente, priorización de requisitos, el modelado de requisitos, la documentación de los requisitos (Cao & Ramesh, 2008a) (Boness et al., 2008), también se han considerado como propuesta para ser utilizado con los métodos ágiles.

En consecuencia, implementar buenas prácticas en la disciplina de la ingeniería de requisitos reduce las deficiencias en producir productos software de baja calidad en las organizaciones de TI dedicadas a este rubro.

1.3.1 Justificación teórica

La Ingeniería de Requisitos Agiles (IRA), es un enfoque moderno para la gestión y desarrollo del producto software que se basa en principios ágiles. Estos promueven la flexibilidad, la colaboración y la adaptabilidad a lo largo del ciclo de vida del desarrollo del software. Teóricamente se justifica por lo siguiente Tabla 2:

Tabla 2

Justificación teórica

Ítem	Descripción	Fuente
1	Adaptación al cambio	Principios y
2	Retroalimentación continua	
3	Entrega incremental y valor temprano	
4	Colaboración y comunicación	
5	Enfoque en el cliente	Valores
6	Reducción del desperdicio	
7	Flexibilidad en la planificación	Ágiles

Por tanto, la IRA se justifica teóricamente por su capacidad de adaptarse a cambios, ofrecer un mayor valor al cliente a lo largo del desarrollo, fomentar la comunicación y colaboración, reducir el riesgo de entregar software que no cumple con las expectativas del usuario. Estos principios ayudan a crear un enfoque más efectivo y eficiente para el desarrollo del producto software en un entorno más cambiante y competitivo.

1.3.2 Justificación práctica

La justificación práctica de la IRA lo que se resumen en la Tabla 3:

Tabla 3

Justificación práctica

Ítem	Descripción	Fuente
1.-	Flexibilidad para los cambios	Practicas de los enfoques ágiles
2.-	Rápida entrega de valor	
3.-	Minimización de riesgos	
4.-	Mayor satisfacción del cliente	
5.-	Colaboración continua	
6.-	Mejora la calidad del producto software	
7.-	Alineación con el negocio	
8.-	Mayor participación del equipo y del usuario	

1.4 Objetivos

1.4.1 Objetivo general

Identificar los desafíos de la ingeniería de requisitos en las metodologías ágiles para el logro de proyectos exitosos en el desarrollo del producto software.

1.4.2 Objetivos específicos

Tabla 4

Objetivos específicos

Ídem	Descripción
OE1	Precisar el entendimiento de los profesionales en entornos ágiles por ingeniería de requisitos
OE2	Establecer qué desafíos con respecto a ingeniería de requisitos de software se encuentran en proyectos ágiles.
OE3	Identificar qué prácticas / enfoques (prácticas, métodos, técnicas) se siguen para superar estos desafíos.
OE4	Identificar qué efectos provocan estas prácticas / enfoques en el proyecto.

Es posible verificar un mapa entre los objetivos y los problemas específicos en la siguiente Tabla 5.

Tabla 5

Maapeo de objetivos específicos / Problemas específicos

Objetivos específicos	Problemas específicos
OE.1: Precisar el entendimiento de los profesionales en entornos ágiles por ingeniería de requisitos en el contexto de un proyecto ágil.	PE.1: ¿Qué entienden los profesionales ágiles por ingeniería de requisitos en el contexto de un proyecto ágil?
OE.2: Establecer qué desafíos con respecto a ingeniería de requisitos de software se encuentran en proyectos ágiles.	PE.2: ¿Qué desafíos con respecto a ingeniería de requisitos de software se encuentran en proyectos ágiles?
OE.3: Identificar qué prácticas / enfoques (prácticas, métodos, técnicas) se siguen para superar estos desafíos.	PE.3: ¿Qué prácticas / enfoques se siguen para superar estos desafíos?
OE.4: Identificar qué efectos provocan estas prácticas / enfoques en el proyecto.	PE.4: ¿Qué efectos provocan estas prácticas / enfoques en el proyecto?

Nota. Esta tabla muestra la correlación ente objetivos específicos y problemas específicos

Los objetivos se abordan a través del análisis de la información obtenida respondiendo a los cuatro problemas específicos anteriores y la discusión permitida sobre las contribuciones más importantes y cómo los hallazgos empíricos apoyan o niegan los resultados de la Revisión sistemática de la Literatura (RSL).

El presente trabajo se propone aportar en este sentido cubriendo de alguna manera la brecha de conocimiento, en este sentido se dispone a llevar a cabo la investigación de las evidencias publicadas que están disponible en las diferentes fuentes académicas sobre las prácticas de IR adoptadas en la industria del software, la academia y los retos que enfrentan los equipos desarrolladores en el ambiente ágil.

1.5 Estructura de la Tesis

El presente trabajo de tesis está estructurado de la siguiente manera: El capítulo II (Antecedentes y trabajos relacionados) se describen los antecedentes de los principales temas relacionados. Esta sección contiene un resumen del trabajo relacionado en Ingeniería de Requisitos (IR) en los ambientes de Desarrollo de Software Ágil (DSA).

El diseño de la investigación se presenta en el Capítulo III, Diseño de la Revisión Sistemática de la Literatura (RSL). Esta sección expone el método empleado para realizar esta investigación junto con su motivación. Los resultados y la discusión se exponen en el capítulo IV, se dan los resultados obtenidos luego de realizar la revisión se realiza un análisis y síntesis de los datos con su respectiva interpretación. El capítulo V provee los detalles y discusión sobre los resultados encontrados. En el capítulo VI. En el capítulo se exponen las conclusiones y trabajos futuros al respecto.

2 MARCO TEÓRICO

En un entorno de constante cambio, el desarrollo de software enfrenta desafíos significativos, especialmente en la ingeniería de requisitos dentro de metodologías ágiles. Este marco filosófico de investigación establece fundamentos teóricos imprescindibles para abordar estos problemas. Centrándose en un paradigma constructivista, se sostiene que el conocimiento se construye a través de la interacción entre los profesionales del software y su entorno. Se analizan múltiples aspectos, desde la teoría de la ingeniería de requisitos hasta los marcos de trabajo ágiles como Scrum y Programación Extrema (XP). Además, se considera la gestión eficaz de proyectos, subrayando la importancia de los requisitos en la creación de software. En este contexto, la investigación se orienta a comprender los obstáculos existentes, ofreciendo recomendaciones prácticas para mejorar la calidad del producto final y la satisfacción del cliente, sentando así las bases para futuras prácticas en el desarrollo de software ágil.

2.1 Marco Filosófico de la Investigación

El marco filosófico o epistemológico es una parte fundamental de un proyecto de investigación, ya que establece las bases teóricas y filosóficas sobre las cuales se construirá el estudio. En el contexto de un tema como "Desafíos de la Ingeniería de Requisitos en el Desarrollo de Software Ágil: Un Enfoque en la Mejora del Producto", el marco epistemológico proporciona la orientación y la justificación para la investigación.

2.1.1 Paradigma epistemológico

Este estudio se basa en un paradigma epistemológico constructivista, que postula que el conocimiento es construido activamente por los individuos a través de la interacción con su entorno y su interpretación de la realidad. En el contexto de la ingeniería de requisitos en el desarrollo de software ágil, esto significa que el conocimiento sobre los desafíos y las estrategias de mejora se construye a través de la experiencia práctica de los profesionales y su reflexión sobre las prácticas ágiles.

2.1.2 Fundamentos teóricos

1.- Teoría de la ingeniería de requisitos: Se exploraron conceptos fundamentales de la ingeniería de requisitos, como la captura de requisitos, la especificación, la verificación y la validación.

2.- Enfoque ágil: Se analizaron los principios y valores del enfoque ágil en el desarrollo de software, incluyendo Scrum, Kanban, eXtreme Programming (XP) y otras metodologías ágiles.

3.- Gestión de proyectos de software: Se consideraron las teorías y mejores prácticas en la gestión de proyectos de software, ya que los desafíos de la ingeniería de requisitos están intrínsecamente vinculados a la gestión eficaz de proyectos.

2.1.3 Perspectiva ontológica

En términos ontológicos, este estudio parte de la premisa de que los desafíos en la ingeniería de requisitos son una realidad inherente en el desarrollo de software ágil. Se asume que existen obstáculos y problemas que deben ser identificados, comprendidos y abordados para lograr una mejora efectiva del producto.

2.1.4 Perspectiva axiológica

La investigación se guía por valores como la mejora continua, la calidad del producto y la satisfacción del cliente. Se considera que la identificación y superación de desafíos en la ingeniería de requisitos contribuyen a la creación de software de mayor calidad y a la entrega de valor al cliente de manera más efectiva.

2.1.5 Perspectiva interdisciplinaria

Dado que la ingeniería de requisitos en el desarrollo de software ágil es un campo multidisciplinario, un marco epistemológico sólido podría incorporar elementos de varias disciplinas, como la informática, la psicología organizacional, la gestión de proyectos y la sociología.

2.1.6 Metodología de la investigación

Se utilizó una metodología cualitativa que incluye entrevistas y análisis documental. Esto permitirá explorar en profundidad los desafíos específicos que enfrentan los equipos de

desarrollo ágil en la ingeniería de requisitos y las estrategias que utilizan para mejorar sus prácticas.

2.1.7 Conclusiones esperadas

Se espera que este estudio contribuya a una comprensión más sólida de los desafíos de la ingeniería de requisitos en el desarrollo de software ágil y ofrezca recomendaciones prácticas para abordar estos desafíos con el objetivo de mejorar la calidad del producto y la satisfacción del cliente.

Este marco epistemológico proporciona una base sólida para la investigación sobre los desafíos de la ingeniería de requisitos en el desarrollo de software ágil, alineando la investigación con una perspectiva constructivista y enfocándose en la mejora continua del producto como objetivo.

2.2 Desarrollo de Software Tradicional

Los enfoques de desarrollo de software tradicional se basan en una concepción predictiva y estructurada del desarrollo de software; Se sostiene en la planificación de todos los procesos que se identifican para su desarrollo, desde el inicio hasta la finalización de un proyecto.

Este tipo de metodologías de desarrollo de software son las más recomendadas para dominios estables bien conocidos, donde se ha tenido éxitos, ya que permiten resolver una visión completa del problema, lo que permite la consideración de soluciones alternativas y posibilita la creación de estimaciones precisas en términos de costos, esfuerzo y tiempo. Las metodologías de desarrollo de software basadas en planes más populares son el modelo en cascada.

2.2.1 Modelo en cascada

El modelo en cascada (Royce, 1970) es un proceso secuencial planificado para el desarrollo de software; Se divide en diferentes etapas que estructuran el proceso de creación de software. Es interesante señalar que la primera vez que Royce propuso este modelo, no se lo denominó "Cascada". La base del modelo es que cada etapa comienza cuando termina su predecesor, y toma como entrada la salida de su predecesor. Estas fases se establecen en la ilustración 1.

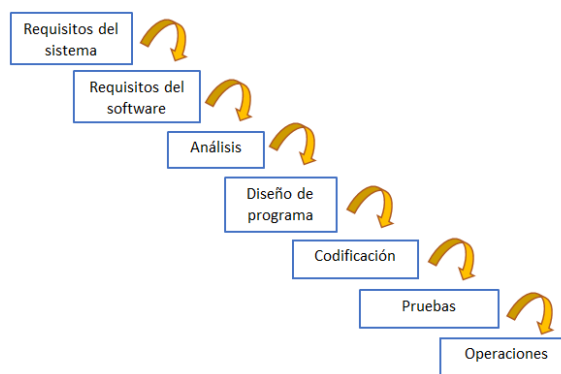
2.2.2 Ingeniería de requisitos tradicional

Viene a ser la primera actividad por desarrollar del ciclo de vida del desarrollo de un producto software en la propuesta predictiva se considera gestionar los requisitos.

La gestión de requisitos es el proceso de gestionar un deseo o una necesidad simple. Los requisitos exitosos son la clave para entregar productos de alta calidad. Para ello se identifican las siguientes actividades (Sommerville, 2016).

Ilustración 1

Metodología en tradicional (en cascada)



Nota. (Royce, 1970)

- Obtención: Identificar fuentes; identificar requisitos
- Análisis: Clasificar los requisitos; modelo; arquitectura de alto nivel; asignar requisitos a los componentes; negociar requisitos
- Especificación: Documento de definición de requisitos; especificación de requerimientos de Software; estándares de documentos; calidad del documento.
- Validación: revisiones; prototipos; modelado; prueba de definición
- Gestión: trazabilidad; atributos; cambio / evolución

2.3 Desarrollo de Software Ágil

El problema de la ingeniería de software que ha venido en denominarse las crisis del software desde fines de la década del 70 del siglo pasado, radica en el cómo debe organizarse las actividades propias del desarrollo de software con el objetivo de hacer entregas oportunas y a costos planificadas al inicio de los proyectos, como consecuencia de ello desde inicio del 2000 surgen las metodologías ágiles y se acentúan su práctica en la actualidad en la industria del software.

2.3.1 Concepto proceso ágil

Pudiendo entender que ágil es sinónimo de eficaz y se tiene por virtud el de dirigirse fácilmente hacia nuevas direcciones. El proceso debe ser ligero para que sea posible dirigirlo fácilmente hacia las nuevas direcciones. La suficiencia del proceso permite permanecer en el juego. La agilidad, por ejemplo, se puede medir, con el número de miembros del equipo. Un equipo de 30 o 40 personas no será tan ágil como un equipo de cuatro o seis personas (Williams y Tomayko, 2002). La agilidad en su núcleo significa ser más capaz de promover una respuesta rápida a los entornos cambiantes, los cambios en los requisitos del usuario, los plazos acelerados del proyecto y ese tipo de cosas (Erickson et al., 2005).

(Williams y Tomayko, 2002) describen que los métodos básicos de desarrollo de software como, por ejemplo, los basados en planes no pudieron responder al cambio que tanto la tecnología como el entorno empresarial mantuvieron durante el proyecto. Además, los clientes se volvieron cada vez más incapaces de expresar sus necesidades al comienzo del proyecto. Para cumplir con estos requisitos, Williams y Cockburn afirman que la agilidad se trata de retroalimentación y cambio.

2.3.2 Valores principales del desarrollo de software ágil

Los métodos ágiles son un enfoque importante para la ingeniería de software y la idea principal de los métodos ágiles es maximizar el valor comercial de crear nuevos productos de software. Los métodos ágiles son una alternativa eficiente a los estándares de ingeniería de sistemas tradicionales y al desarrollo basado en documentos debido a su forma centrada en el cliente para desarrollar productos (Rico, 2010).

El Manifiesto Ágil fue publicado por un grupo de profesionales del software en 2001(Beck et al., 2001). Según el manifiesto, El desarrollo de Software Ágil tiene cuatro valores principales:

Individuos e interacciones sobre procesos y herramientas

Este valor significa que los desarrolladores están facultados para formar equipos auto gestionado. Además, los desarrolladores deben ser programadores informáticos expertos y altamente motivados, y trabajaron juntos para resolver problemas complejos (Rico, 2010). Además, los clientes/usuarios y los desarrolladores deben trabajar juntos todos los días durante el proyecto (Beck et al., 2001).

Software de trabajo sobre documentación completa

El segundo valor significa que una de las prioridades más altas dentro de los métodos ágiles es producir software de funcionamiento. Además, esto significa que todos los recursos se centran en producir el software en intervalos de dos a cuatro semanas. Esta es la forma de maximizar el valor comercial porque el cliente está pagando por el software funcionando, no por la documentación (Rico, 2010). Un producto de software funcionando es también una medida primaria del progreso realizado por el equipo de desarrollo (Beck et al., 2001). Abrahamsson et al. (2004) señalan que se supone que los desarrolladores deben mantener el código lo más simple posible, de modo que la carga de documentación pueda reducirse a un nivel apropiado.

Colaboración del cliente sobre negociación de contrato

La colaboración con el cliente representa uno de los principales aspectos del desarrollo ágil de software (Paetsch et al., 2003). En cierto sentido, esto implica que los desarrolladores consultan a los clientes acerca de sus requerimientos. Además de preguntar, escuchar e interactuar con los clientes son también formas efectivas de identificar sus necesidades (Rico, 2010).

Responde al cambio sobre el seguimiento a un plan

Los métodos ágiles aceptan cambios en los requisitos incluso en las fases finales del desarrollo (Beck et al., 2001). Esto implica que el proceso de desarrollo debe planificarse con el mínimo esfuerzo necesario y repetirse frecuentemente hasta lograr la plena satisfacción del cliente. Las necesidades del cliente evolucionan y se modifican a lo largo del proyecto, y la planificación iterativa, sin requerir grandes inversiones en documentación, permite alcanzar de manera clara el mayor valor comercial (Rico, 2010).

2.3.3 Principios básicos del desarrollo de software ágil

Según Cockburn (Williams y Tomayko, 2002), una parte fundamental para alcanzar la agilidad en el desarrollo de software consiste en identificar los “puntos clave” que caracterizan un desarrollo efectivo. El proyecto debe orientarse lo más cerca posible a dichos puntos clave o principios. A continuación, se presentan los cinco principios propuestos por Cockburn.

Dos a ocho personas en una habitación

Este principio es para la comunicación. Cuando un pequeño equipo de proyecto está sentado en la misma sala, puede hacer preguntas breves y las preguntas pueden responderse muy rápidamente. Además, las ideas de diseño y el plan del proyecto se mantienen en la pizarra y todos pueden tenerlo a la vista.

Expertos del negocio en el lugar de desarrollo

Es muy útil tener un experto en uso disponible en todo momento porque el tiempo de retroalimentación es lo más corto posible. El equipo de desarrollo puede obtener una comprensión más profunda de las necesidades y hábitos de los usuarios. Además, los desarrolladores pueden probar más ideas y de ese modo aumentar la calidad del producto final.

Incrementos de una iteración

Este principio también es necesario para garantizar la retroalimentación rápida. Ahora, la retroalimentación se concentra tanto en el producto como en el proceso de desarrollo en sí. El desarrollo incremental es perfecto para proporcionar puntos de retroalimentación que permite que los requisitos y el proceso se reparen rápidamente. La pregunta más importante es qué tan largos deberían ser los incrementos de entrega. No solo hay una respuesta correcta a esta pregunta, sino que las personas parecen ser capaces de concentrarse mejor en su esfuerzo durante aproximadamente tres meses.

Pruebas de regresión completamente automatizadas

Las pruebas de regresión totalmente automatizadas pueden ser pruebas unitarias o funcionales o ambas. El uso de pruebas de regresión automatizadas tiene dos ventajas principales. Primero, el código se puede modificar libremente o se pueden mejorar o reemplazar módulos completos porque las pruebas automatizadas descubrirán posibles errores. En segundo lugar, los propios desarrolladores pueden disfrutar de una mejor calidad de vida porque no necesitan pensar en la posibilidad de que alguien haya alterado su código.

Desarrolladores experimentados

En una situación ideal, el equipo de desarrollo está formado solo por desarrolladores experimentados porque pueden ser muchas veces más efectivos que los inexpertos. Cuando el equipo se compone solo de desarrolladores experimentados, el número de

miembros del equipo puede reducirse drásticamente. Esto es, por supuesto, difícil de conseguir y en caso de que haya desarrolladores sin experiencia en el equipo, puede ser útil llevar a cabo una capacitación eficaz.

Abrahamsson et al. (2004) ofrecen un resumen claro acerca de las características que definen a un método de desarrollo ágil. Según su perspectiva, el desarrollo de software puede considerarse ágil cuando se lleva a cabo de manera incremental, cooperativa, directa y adaptativa.

2.4 Métodos Ágiles en el Ambiente de desarrollo de Software Ágil

Existen diversos enfoques de métodos ágiles. Entre los más difundidos en la actualidad se encuentran Scrum, Programación Extrema (XP), el Desarrollo de Sistemas Dinámicos (DSD), el Desarrollo Basado en Funciones (DFD) y la familia Crystal. De estos, Scrum y Programación Extrema son considerados los métodos más utilizados (Rico, 2010).

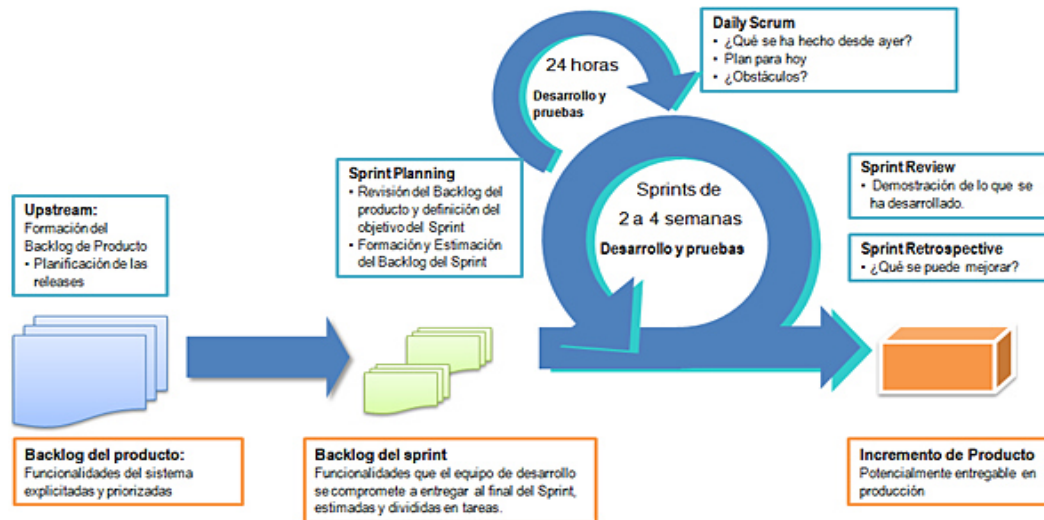
2.4.1 El enfoque Scrum.

Este enfoque se basa en un modelo de control de proceso empírico en lugar del modelo de control de proceso tradicional definido. Debido a esa diferencia, Scrum funciona bien en el contexto de proyectos de desarrollo de sistemas (Schwaber, 1997). Scrum como marco de desarrollo consiste en equipos Scrum y sus roles asociados, como Scrum Master y Product Owner, eventos, artefactos y reglas. Scrum no tendrá éxito y no se puede usar sin ningún componente dentro del marco establecido. Los eventos en Scrum se utilizan para crear regularidad y minimizar la necesidad de reuniones innecesarias, estos eventos son: Sprint, Sprint Planning, Daily Scrum, Sprint Review y Sprint Retrospective. Los artefactos en el Scrum representan trabajo o valor y están diseñados para maximizar la información clave. De esa manera, todos tienen la misma comprensión de un artefacto. Estos artefactos son el Product Backlog y el Sprint Backlog (Cho, 2009; Sutherland, 2013).

Los eventos y artefactos mencionados anteriormente se muestran en el modelo básico de un proceso de desarrollo de Scrum en la Ilustración 2.

Ilustración 2

Modelo típico de un proceso de desarrollo de Scrum



Nota. (Schwaber, 1997)

2.4.2 El enfoque programación extrema (XP)

Fue introducido en 1998 por Kent Beck, Ron Jeffries, Ward Cunningham basado en la experiencia del proyecto C3 (Beck, 2004).

El enfoque de programación extrema (XP) fue introducido en 1998 por Kent Beck, Ron Jeffries y Ward Cunningham, basado en la experiencia del proyecto C3 (Beck, 2004). XP constituye una metodología ágil que se centra en la adaptación al cambio y la mejora continua, promoviendo ciclos de desarrollo cortos, una participación activa del cliente y prácticas técnicas rigurosas como la integración continua, la programación en parejas y el desarrollo guiado por pruebas.

Filosofía y alcance

Extreme Programming (XP) es una disciplina ágil ligera enfocada en prácticas estrictas de ingeniería de software. Su objetivo principal es permitir el desarrollo exitoso de software en contextos caracterizados por requisitos poco claros o en constante cambio (Beck, 2004). XP implementa un sistema de prácticas que busca mejorar la calidad del software y responder ágilmente a las necesidades cambiantes de los clientes y del negocio. Entre sus prácticas fundamentales se incluyen la recopilación de requisitos informales de clientes in situ, la organización de equipos de programación en parejas, el diseño simple, la refactorización continua, la integración y las pruebas continuas, así como lanzamientos

frecuentes en ciclos cortos de desarrollo. Estos elementos favorecen la productividad y establecen puntos de control periódicos en los que pueden ser incorporados nuevos requisitos del cliente.

XP resulta especialmente adecuado para proyectos que requieren equipos pequeños o medianos que comparten el mismo entorno físico, favoreciendo la colaboración intensa y la rápida adaptación a los cambios.

Características y beneficios

Tamaño del equipo: de 2 a 12 miembros y preferiblemente co-ubicado Duración de la iteración: generalmente de 1 a 3 semanas.

Ciclo de vida de un proyecto XP

Un ciclo de vida ideal del proyecto XP contiene las siguientes seis fases (Beck et al., 2001):

1. Fase de exploración

En la fase de preproducción de Extreme Programming (XP), los clientes redactan las historias de usuario que la primera versión del proyecto debe implementar. Simultáneamente, los programadores exploran distintas tecnologías y experimentan con ellas para seleccionar la más adecuada para el desarrollo, además de evaluar las opciones arquitectónicas para el sistema. Esta fase concluye cuando el equipo adquiere la confianza necesaria para avanzar a las siguientes fases de producción, es decir, cuando considera posible finalizar el programa conforme a los objetivos establecidos. Su duración puede variar desde unas pocas semanas, en casos donde el equipo ya está familiarizado con el dominio y las tecnologías, hasta algunos meses, en proyectos con tecnologías o dominios desconocidos.

2. Fase de planificación

En la fase de planificación de Extreme Programming (XP), se lleva a cabo el denominado “juego de planificación” (Planning Game), durante el cual se priorizan las historias de usuario. En este proceso, los desarrolladores y los clientes colaboran para acordar cuáles serán las historias implementadas en el primer lanzamiento o release. Si el equipo se ha preparado adecuadamente durante la fase de exploración, esta etapa puede completarse en tan solo unos días. La planificación XP establece los objetivos, los plazos

previstos y detalla los pasos a seguir, permitiendo así al equipo organizar su trabajo de forma iterativa y adaptativa.

3. Iteraciones a la fase de lanzamiento

La versión se divide en diferentes iteraciones, generalmente de una a cuatro iteraciones. En esta fase, el cliente es responsable de elegir las historias que permitirán la creación de la "arquitectura completa" del sistema, incluso en forma esquelética. El cliente también es responsable de seleccionar cada historia de iteración y completar las pruebas funcionales que deben ejecutarse al final de cada una. Una vez que finaliza la última iteración, el sistema está listo para entrar en producción.

4. Fase de producción

Este es el juego final del lanzamiento; Esta fase se puede describir como la fase de implementación de XP. El objetivo aquí es certificar que el sistema está listo para entrar en producción, por lo que las pruebas juegan un papel clave en la Fase de Producción. En esta fase, la evolución del sistema se ralentiza y el equipo está más preocupado por abordar los riesgos a la hora de decidir si un cambio debe incluirse en la versión actual. Cuando finaliza la prueba, el sistema está listo para entrar en producción.

5. Fase de mantenimiento

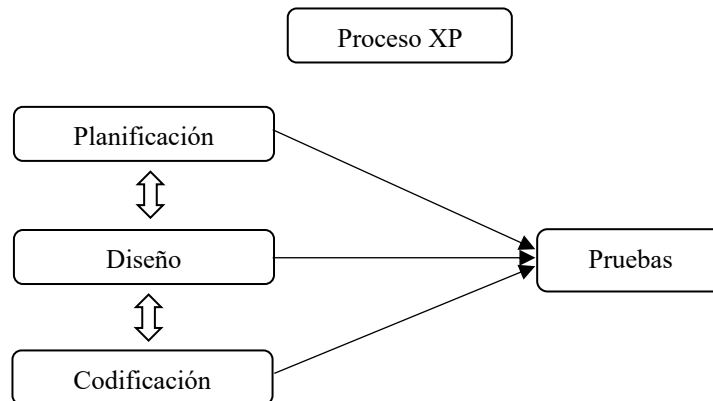
Esta fase es el estado normal de un proyecto XP; Esto se debe a que durante el mantenimiento el sistema sigue evolucionando con el tiempo. La fase de mantenimiento abarca todas las anteriores, en cuanto se lanzan nuevas versiones. Cabe señalar que, cuando un sistema está en producción, cada cambio debe ser evaluado de manera más cuidadosa y conservadora, por lo que la gestión del cambio es muy útil en esta fase.

6. Fase de muerte

Esta fase aparece cuando el cliente no tiene más historias para implementar. Este es el momento adecuado para crear documentación del sistema si es necesario.

Ilustración 3

Fases del ciclo de vida de un proyecto XP



Valores XP

- Comunicación
- Simplicidad
- Comentarios
- Coraje

Prácticas XP

1. Juego de planificación
2. Pequeños lanzamientos
3. Metáfora del sistema
4. Diseño simple
5. Desarrollo guiado por pruebas
6. Refactorización
7. Integración continua
8. Propiedad del código colectivo
9. Programación en pares
10. Estándares de codificación
11. Cliente en el sitio
12. Semana de 40 horas

Roles y responsabilidades de XP

- **XP Coach:** guía al equipo para seguir el proceso XP.

- **Cliente XP:** escribe historias, pruebas funcionales y establece la prioridad de implementación.
- **Administrador de XP:** configura el entorno del programador y actúa como administrador local.
- **Programador de XP:** escribe pruebas y código.
- **XP Tracker:** realiza un seguimiento de las iteraciones, como el administrador de programas / proyectos, y proporciona comentarios sobre la precisión de las estimaciones y rastrea el progreso de las iteraciones.
- **XP Tester:** ayuda al cliente a escribir pruebas funcionales y ejecuta pruebas funcionales y mantiene herramientas de prueba.
- **Consultor XP:** Un miembro externo que guía al equipo para resolver problemas. El gerente toma decisiones.

2.5 El rol del Cliente en el Desarrollo de Software Ágil

En el desarrollo de software ágil, el cliente o usuario desempeña un papel central. La satisfacción, la colaboración y el compromiso con el cliente son tres de los nueve factores fundamentales para el éxito en proyectos ágiles, siendo el compromiso especialmente relevante para lograr resultados óptimos (Misra et al., 2009). Además, la colaboración con el cliente es uno de los valores principales del desarrollo ágil de software. Según los participantes en el taller sobre la participación del cliente en programación extrema reportado por Misra et al. (2009), las principales tareas asignadas al cliente incluyen la redacción de historias de usuario, la priorización de requisitos, la definición de pruebas de aceptación y la validación continua del producto.

1. Para comprender las necesidades de los clientes, mantener un contacto regular con los usuarios finales y equilibrar los intereses potencialmente conflictivos de los usuarios finales.
2. Para hablar con los desarrolladores, aclarar las solicitudes de funciones cuando sea necesario y comprender algunas de las preocupaciones técnicas que los desarrolladores tienen.
3. Para especificar pruebas funcionales para historias de usuarios (requisitos), y verificar que estas pruebas se ejecuten correctamente.
4. Participar en la planificación de iteraciones y lanzamientos.

5. Para mantener un buen contacto con la gerencia, explicar el progreso y justificar el tiempo que pasa con el equipo de desarrollo.

Corral et al. (2013) señalan que, en la práctica de Extreme Programming (XP), el cliente in situ debe poseer ciertos atributos fundamentales. Primero, el cliente debe estar disponible y responder oportunamente a las preguntas del equipo de desarrollo, ya que los retrasos en las respuestas pueden afectar negativamente el avance del proyecto. Además, el cliente debe ser un experto en el dominio o en las reglas del negocio, contando con la capacidad de responder a cualquier consulta técnica o funcional; para ello, resulta esencial que comprenda en profundidad cómo debe funcionar la aplicación. Finalmente, el cliente debe poseer la autoridad para tomar decisiones y comprometerse con ellas, lo que permite una gestión ágil de los cambios en los requisitos (Corral et al., 2013).

Por ejemplo, estudios como el de Sillitti citados en Corral et al. (2013) y Martin et al. (2009), han evidenciado que los clientes experimentan una presión significativamente mayor que los desarrolladores u otros participantes en el proyecto, debido al nivel de responsabilidad y exposición que exige su rol dentro de la metodología XP.

2.6 Ingeniería Requisitos

A continuación, se hace una breve presentación sobre la ingeniería de requisitos en el proyecto de software y proporciona un modelo para el proceso tradicional de ingeniería de requisitos. Luego las prácticas de la ingeniería de requisitos ágiles se analizan con más detalle.

2.6.1 Ingeniería de requisitos tradicionales

La ingeniería de requisitos proporciona el mecanismo necesario para comprender lo que quiere el cliente, analizar la necesidad, evaluar la viabilidad del sistema, negociar una solución, especificar la solución, validar la especificación y gestionar los requisitos a lo largo del ciclo de vida del proyecto. Su objetivo es asegurar que el sistema sea especificado para satisfacer las necesidades y expectativas del cliente, tal como destaca Pressman (Ernst et al., 2014; Suryn, 2014).

El proceso general de la ingeniería de requisitos tradicional incluye cuatro subprocesos de alto nivel: (1) estudio de factibilidad, donde se evalúa la utilidad comercial del sistema; (2) obtención y análisis de requisitos, que permite descubrir y clarificar necesidades; (3) especificación de requisitos, en la que los requisitos se formalizan en algún estándar; y (4) validación, que verifica que los requisitos definan exactamente lo que el cliente desea, según lo explicado por Sommerville (2016). Además, los requisitos deben gestionarse adecuadamente, ya que pueden cambiar a lo largo de la vida útil del sistema (Suryn, 2014).

Estudio de factibilidad: Para todos los sistemas nuevos, el proceso de ingeniería de requisitos debe comenzar con un estudio de factibilidad. La idea de un estudio de viabilidad es determinar si vale la pena continuar con la ingeniería de requisitos y el proceso de desarrollo del sistema o no. Cuando se realiza un estudio de factibilidad, se incluye una evaluación de la información, recopilación de información y redacción de informes. A continuación, se presentan algunos ejemplos de posibles preguntas que se deben poner en el contexto de la recopilación y evaluación de información Sommerville (2016).

1. ¿Cómo haría frente la organización si este sistema no se implementara?
2. ¿Cuáles son los problemas con los procesos actuales y cómo un nuevo sistema ayudaría a aliviar estos problemas?
3. ¿Qué contribución directa hace el sistema a los objetivos y requisitos del negocio?
4. ¿Se puede transferir información hacia y desde otros sistemas organizacionales?
5. ¿Requiere el sistema tecnología que no se haya utilizado previamente en la organización?
6. ¿Qué debe ser compatible con el sistema y qué no debe ser compatible?

Obtención y análisis de requisitos: En esta actividad, los ingenieros de software colaboran con los clientes y usuarios finales para comprender el dominio de la aplicación, los servicios requeridos, el rendimiento necesario y otras expectativas relacionadas con el sistema (Sommerville, 2016). Sin embargo, existen dificultades que pueden entorpecer la recolección de requisitos, por lo que es fundamental abordar esta tarea de manera estructurada y organizada (Suryn, 2014). El proceso ideal contempla los siguientes pasos: (1) obtener los objetivos generales del sistema, (2) recopilar información sobre el trabajo y los problemas actuales, (3) identificar los problemas detallados a resolver, (4) analizar

posibles soluciones y (5) traducir los problemas y oportunidades en requisitos claros y documentados (Lauesen, 2016).

Diversas actividades y técnicas pueden emplearse para la obtención y análisis de requisitos. Ejemplos comunes incluyen entrevistas, escenarios, casos de uso, observación, cuestionarios, grupos focales, lluvia de ideas, creación de prototipos y negociación entre las partes interesadas (Sommerville, 2016; Lauesen, 2016). Las entrevistas resultan especialmente útiles para obtener conocimiento sobre el trabajo presente y los desafíos existentes, aunque es recomendable utilizar técnicas adicionales para validar y complementar la información recolectada.

Una vez reunidos los requisitos, se procede a su análisis: en esta etapa se clasifican y agrupan en subconjuntos relacionados, se examinan los conflictos y la coherencia entre los requisitos, y se priorizan en función de las necesidades del cliente. Además, es fundamental involucrar a todas las partes interesadas para discutir y resolver posibles discrepancias en las prioridades, así como identificar y analizar los riesgos asociados a cada requisito (Suryan, 2014).

Especificación de requisitos: La especificación de requisitos puede interpretarse de diferentes maneras según el contexto del software. Para algunos, implica la elaboración de un documento escrito; para otros, puede consistir en un modelo gráfico o en una colección de escenarios de uso. Según Lauesen (2016), una buena especificación de requisitos debe cumplir ocho criterios de calidad: debe ser correcta, completa, inequívoca, consistente, clasificada por importancia y estabilidad, modificable, verificable y rastreable.

No obstante, durante el desarrollo de una especificación de requisitos es esencial mantener cierta flexibilidad. El tipo de especificación más adecuado puede depender del tamaño y la complejidad del software. Por ejemplo, para sistemas grandes, resulta conveniente optar por documentos escritos detallados que incluyan descripciones en lenguaje natural y modelos gráficos; para productos más pequeños, bastaría con escenarios de uso bien definidos (Suryan, 2014).

Los criterios de calidad mencionados se encuentran alineados con los estándares internacionales, como IEEE 830-1998, que establecen principios para una especificación clara, verificable y gestionable a lo largo del ciclo de vida del software.

Validación de requerimientos: La validación de requisitos tiene como objetivo principal demostrar que los requisitos definidos representan correctamente lo que el cliente desea para el sistema. Este proceso verifica cualidades como la validez, consistencia, integridad, realismo y verificabilidad de los requisitos (Sommerville, 2016; Tobergte y Curtis, 2013). Pressman (Tobergte y Curtis, 2013) destaca que la validación se refiere a la calidad de los requisitos, asegurando que reflejen fielmente las necesidades del cliente. Además, es fundamental validar que las partes interesadas estén satisfechas y que los riesgos principales sean considerados y gestionados adecuadamente (Lauesen, 2016).

Las técnicas más comunes en la validación de requisitos incluyen las revisiones formales y la creación de prototipos (Sommerville, 2016). Durante las revisiones, un equipo compuesto por desarrolladores, clientes y analistas examina detenidamente la especificación de requisitos para identificar errores o inconsistencias (Lauesen, 2016). Pressman destaca que las revisiones técnicas formales son un mecanismo primordial para validar los requisitos, involucrando a diversas partes interesadas en el proceso (Suryn, 2014). En estas sesiones, los revisores reportan problemas observados pero no discuten posibles soluciones en ese momento. Por otro lado, las pruebas mediante prototipos permiten validar que los requisitos sean realistas y significativos, facilitando la visualización y prueba de funcionalidades antes de su desarrollo completo (Lauesen, 2016).

Gestión de requerimientos: La gestión de requisitos es esencial en desarrollo de software, ya que los cambios empresariales, organizativos y tecnológicos inevitablemente provocan modificaciones en los requisitos iniciales del sistema (Sommerville, 2016). Durante el proceso de desarrollo, algunos requisitos pueden resultar incorrectos, aparecen nuevas necesidades, ciertos requisitos se consideran demasiado costosos o complejos de implementar, o se acuerdan alternativas más simples entre las partes interesadas (Lauesen, 2016). La gestión de requisitos constituye un conjunto de actividades y metodologías enfocadas en controlar y administrar estos cambios eficazmente; cualquier propuesta de cambio debe ser analizada y evaluada en cuanto a su impacto en el proyecto, el alcance, el costo y los recursos disponibles.

Tanto Sommerville (2016) como Pressman (Suryn, 2014) destacan la importancia de la trazabilidad de los requisitos dentro de las actividades de gestión. La trazabilidad permite documentar, rastrear y controlar el historial de cada requisito, facilitando la toma de decisiones informadas cuando el sistema es complejo o de gran escala. Mantener un

seguimiento eficiente es clave para garantizar la alineación entre los requisitos implementados y las expectativas actuales de los clientes y las partes interesadas.

Las herramientas y procesos modernos de gestión de requisitos, incluyendo plataformas colaborativas y sistemas automatizados, mejoran la visibilidad, la trazabilidad y el control de los cambios, contribuyendo así a la estabilidad, la adaptabilidad y el éxito final del proyecto.

2.6.2 Prácticas de ingeniería de requisitos ágiles

2.6.2.1 Interacción entre el equipo de desarrollo y el cliente.

Asegurar la accesibilidad del cliente o su presencia en el sitio es un punto esencial en todos los enfoques ágiles (Paetsch et al., 2003). Sin embargo, aunque el cliente esté involucrado durante todo el ciclo de desarrollo, no siempre se cuenta con la garantía de que todos los usuarios o representantes del cliente tengan los conocimientos y antecedentes adecuados. En desarrollo ágil, se recomienda evitar la existencia de capas intermedias de comunicación entre el equipo de desarrollo y el cliente, favoreciendo el diálogo directo para reducir significativamente los malentendidos (Corral et al., 2013).

El objetivo de la ingeniería de requisitos ágil es traducir eficazmente las ideas del cliente al equipo de desarrollo, priorizando la comunicación efectiva por encima de la documentación extensa. Cuando existen vías directas e informales de comunicación, ya no resultan necesarias aprobaciones formales extensas ni documentación detallada que puede volverse obsoleta rápidamente al evolucionar los requisitos. Así, la mayoría de las organizaciones recurren a técnicas ágiles y sencillas, como las historias de usuario, para definir los requisitos del cliente (Ramesh et al., 2010).

La participación activa de todo el equipo de desarrollo, utilizando el lenguaje del cliente, también reduce los riesgos de ambigüedad. Si los requisitos son complejos, se recomienda descomponerlos con la ayuda del cliente en elementos más manejables (Corral et al., 2013). La interacción continua entre cliente y equipo de desarrollo ofrece múltiples ventajas:

- El cliente puede guiar el proyecto hacia nuevas direcciones, especialmente cuando los requisitos evolucionan debido a cambios en el entorno o mejor entendimiento de la solución (Ramesh et al., 2010).

- La comunicación directa minimiza la transferencia errónea de conocimiento y ayuda a construir relaciones de confianza (Misra et al., 2009).
- El cliente se integra como un miembro activo del equipo, priorizando tareas y ayudando a que los requerimientos críticos sean atendidos primero.

No obstante, la representación continua del cliente es difícil de garantizar en la mayoría de los casos. La ausencia de interacción intensiva puede originar requisitos inadecuados o mal definidos. Además, cuando existen múltiples grupos de clientes, es complejo lograr consensos; la literatura recomienda reducir el número de representantes a una sola persona experta en el dominio, capaz de tomar decisiones clave y priorizar requisitos (Corral et al., 2013).

2.6.2.2 Ingeniería de requisitos iterativos.

En los métodos ágiles, las funcionalidades se liberan en ciclos pequeños y frecuentes, lo que permite al equipo de desarrollo obtener retroalimentación constante y en tiempo real del cliente (Corral et al., 2013). Al inicio del proyecto, el equipo se hace una idea general de las funciones críticas del software, evitando invertir un esfuerzo excesivo en la ingeniería de requisitos desde el comienzo, debido a la alta volatilidad de los requisitos, el conocimiento incompleto de las tecnologías empleadas o la dificultad de los clientes para articular sus necesidades sin tener un producto visible.

Tras una breve identificación inicial de requisitos, la ingeniería ágil de requisitos continúa a lo largo de cada ciclo de desarrollo. Al comienzo de cada iteración, el cliente y el equipo revisan y detallan las características que se implementarán, adaptando y expandiendo los requisitos de manera gradual y continua (Ramesh et al., 2010; Bjarnason et al., 2011). Este enfoque iterativo facilita que los requisitos sean gestionados de manera activa durante todo el proceso, contribuyendo a cerrar brechas de comunicación dentro del equipo de desarrollo y entre el negocio y los técnicos (Bjarnason et al., 2011).

La iteración permite mantener actualizadas las especificaciones y mejora la relación con el cliente, quien aporta retroalimentación constante y ayuda a evitar el desperdicio en características innecesarias (Ramesh et al., 2010; Suryn, 2014; Bjarnason et al., 2011). De este modo, la ingeniería de requisitos en entornos ágiles se convierte en un proceso viviente, flexible y alineado con las necesidades reales del negocio y de los usuarios.

2.6.2.3 Priorización de requisitos.

En el desarrollo ágil, las funcionalidades de mayor prioridad se implementan de manera temprana para maximizar el valor comercial generado para el cliente (Suryan, 2014). Este proceso de priorización debe repetirse frecuentemente durante todo el ciclo de desarrollo, ya que el proyecto evoluciona, surgen nuevos requisitos y aumenta la comprensión colectiva de las necesidades (Paetsch et al., 2003). Ramesh et al. (2010) recomiendan que los requisitos sean priorizados al inicio de cada iteración, asignando juntos el cliente y el equipo de desarrollo las prioridades, de modo que las características más importantes se aborden primero. Este enfoque se centra fundamentalmente en el valor comercial para el cliente.

Corral et al. (2013) presentan un proceso de cuatro pasos para la priorización de requisitos en métodos ágiles:

1. El equipo de desarrollo estima el tiempo requerido para implementar cada funcionalidad.
2. El cliente determina la prioridad de negocio para cada funcionalidad.
3. El equipo de desarrollo evalúa un factor de riesgo asociado a cada funcionalidad.
4. El equipo y el cliente deciden conjuntamente qué características serán implementadas en la iteración.

Al repetir la obtención de requisitos y este proceso de priorización en cada ciclo, resulta posible identificar y filtrar aquellos requisitos que no aportan suficiente valor para el cliente. Esta priorización favorece que el equipo de desarrollo comprenda el razonamiento detrás de cada requisito y reduzca la posibilidad de incluir historias que no sean urgentes ni generen un beneficio claro.

No obstante, existe el riesgo de que, si la priorización no considera también factores de negocio o requisitos no funcionales, el sistema final no cubra aspectos que pueden ser críticos para el éxito operativo de la solución (Inayat et al., 2015).

2.6.2.4 Requisitos no funcionales.

Los requisitos no funcionales son restricciones que definen cómo debe operar el sistema, centrándose en atributos de calidad como el rendimiento, la seguridad, la usabilidad, la confiabilidad, la escalabilidad y la mantenibilidad (Hester et al., 2013; Ramesh et al., 2010). Estos criterios determinan aspectos esenciales del sistema tales como la velocidad de respuesta, el tiempo de disponibilidad, el manejo de grandes volúmenes de datos, la

facilidad con la que puede mantenerse o ampliarse, y la protección frente a amenazas externas.

En metodologías ágiles, los requisitos no funcionales suelen recibir menor atención que los funcionales, principalmente porque los clientes tienden a enfocarse en la funcionalidad principal y pueden tener dificultades para expresar restricciones técnicas o de calidad necesarias antes de usar el sistema (Ramesh et al., 2010; Misra et al., 2009). De manera habitual, los clientes suelen destacar ciertas necesidades de facilidad de uso, interfaz gráfica o seguridad, mientras que otras restricciones como escalabilidad o rendimiento quedan implícitas o invisibles (Paetsch et al., 2003; Inayat et al., 2015).

Por este motivo, el equipo de desarrollo debe guiar y ayudar al cliente a identificar necesidades no funcionales latentes. Reuniones específicas en las primeras etapas del desarrollo, lideradas conjuntamente por clientes y responsables técnicos, facilitan la exploración y clarificación de estos requisitos (De Lucia y Qusef, 2010; Corral et al., 2013). No obstante, la continua interacción entre cliente y equipo agiliza la respuesta ante carencias: tras cada iteración, el cliente puede probar el sistema; si detecta problemas relacionados con requisitos no funcionales, el equipo puede adaptarlo en la siguiente entrega (Corral et al., 2013).

2.6.3 Documentación en ingeniería de requerimientos ágiles

En las prácticas ágiles, la documentación se minimiza y los requisitos no siempre están documentados (Bjarnason et al., 2011). Sin embargo, algunos enfoques ágiles recomiendan usar documentos de requisitos, pero su extensión depende de la decisión de los equipos de desarrollo. Además, se debe considerar el tamaño del equipo al planificar la documentación. Se considera que el equipo ágil es más rentable y productivo cuando se evita el exceso de documentación. La documentación mínima también aumenta las posibilidades de mantener el documento actualizado cuando se cambia el software (Paetsch et al., 2003).

Los clientes a menudo solicitan al equipo que produzca documentación antes de que se resuelva, pero el alcance de esta documentación es muy limitado y se concentra en los aspectos centrales del sistema. Aunque el modelado se utiliza como parte de la ingeniería de requisitos ágiles, la mayoría de los modelos no formaron parte de la documentación persistente del sistema (Paetsch et al., 2003).

2.6.3.1 Historias de usuarios.

Una historia de usuario describe la funcionalidad que un usuario o un cliente del software o sistema mantienen valioso. Este enfoque es conocido como la fórmula de las “tres C” de las historias de usuario:

- Card (tarjeta): donde se sintetiza la funcionalidad.
- Conversation (conversación): donde se desarrolla el entendimiento y se aclaran los detalles.
- Confirmation (confirmación): donde se validan los criterios de aceptación para la historia de usuario

Según Cohn (2009) precisa que, si bien la descripción escrita puede contener el texto de la historia, la información detallada se resuelve en las conversaciones y se documenta en las pruebas. La descripción escrita puede referirse a una pequeña pieza de funcionalidad y estar escrita en una hoja de papel. Estos papeles se cuelgan en un tablón de anuncios (Corral et al., 2013). Una historia de usuario no contiene muchos detalles y, por lo tanto, una pregunta básica es dónde están los detalles. Cohn (2009) responde a esta pregunta y dice que muchos de los detalles de los requisitos se pueden expresar como historias adicionales. Es preferible tener más historias que algunas que son demasiado grandes.

2.6.3.2 Desafíos de documentación mínima.

En general, los métodos ágiles suelen ser criticados por producir una cantidad insuficiente de documentación. Esta carencia puede ocasionar problemas dentro de los equipos, ya que la documentación es fundamental para compartir y preservar información entre los miembros (Paetsch et al., 2003). La falta de documentación adecuada puede desembocar en fallos de comunicación provocados por factores como la rotación de personal, cambios rápidos en los requisitos, ausencia de representantes clave del cliente o el aumento en la complejidad de la aplicación (Ramesh et al., 2010).

Cuando se produce una falla en la comunicación, surgen múltiples problemas: se dificulta la escalabilidad del software, su evolución a lo largo del tiempo y la incorporación eficiente de nuevos integrantes al equipo de desarrollo (Ramesh et al., 2010). Los nuevos miembros, en particular, se enfrentan a la dificultad de integrarse rápidamente al proyecto

y, si deben depender constantemente de otros para responder a sus dudas, el proceso de aprendizaje puede ralentizar considerablemente el ritmo de trabajo (Paetsch et al., 2003).

3 METODOLOGÍA

La investigación desarrollada es de tipo cualitativa el cual permite el entendimiento de los fenómenos explorándolos en todas las dimensiones profundizando en sus enfoques, apreciaciones y significados (Sampieri et al., 2006). Así mismo Marshall y Rossman (2006) recomienda la aplicación de este enfoque cuando el tema a investigar que se pretende abordar no ha sido explorado y existen estudios al respecto.

3.1 Diseño y realización de la investigación

Como consecuencia del problema del presente trabajo de tesis considerado, el enfoque de la investigación cualitativa, tomando como base la teoría fundamentada de diseño sistemático. En tal sentido, se pretende generar conocimiento que responda al planteamiento del problema.

De lo expresado, la presente investigación denota las características que a continuación se describe:

- Por su finalidad la Investigación es Básica, por la finalidad de la investigación de crear conocimiento nuevo respecto a los desafíos que se evidencia por una mejor calidad del producto software.
- El diseño es no experimental, dado que la investigación sigue una estrategia para obtener información mediante la observación de una situación existente y no provocada intencionalmente por el investigador.
- Nivel es descriptiva, por el nivel de investigación, explica los conceptos y variables definidos por la ingeniería de software, específicamente en el proceso de Ingeniería de requisitos.
- Transversal, por tener un alcance temporal, la investigación se realiza en un solo periodo de tiempo.

A continuación, se expone la metodología de investigación utilizada para alcanzar los objetivos de esta investigación. En esta sección, se detalla el proceso de Revisión

Sistemática de la Literatura (RSL) en torno al tema de Ingeniería de Requisitos en el ámbito del desarrollo de software ágil, cumpliendo con los protocolos y pautas establecidos para llevar a cabo revisiones sistemáticas de la literatura en Ingeniería de Software (Kitchenham y Charters, 2007).

En primer lugar, la definición del problema a resolver fue fundamental en la ejecución del presente estudio de tesis. Esto permitió establecer el objetivo general de la investigación y sus objetivos específicos. En esta fase inicial, se orientó a la formulación y refinamiento de las preguntas de investigación, que son las que realmente motivan el trabajo de tesis. Según Marconi y Lakatos (2002), las variables de una investigación pueden considerarse independientes o dependientes. Las variables independientes determinan las condiciones que afectan a las variables dependientes. En el caso de este estudio, las variables independientes son las prácticas y desafíos en el proceso de ingeniería de requisitos y las metodologías ágiles, mientras que la variable dependiente es la mejora en la calidad del producto software. El escenario metodológico del estudio se presenta en la Tabla 6.

Tabla 6

Escenario metodológico de la investigación.

Escenario	Descripción
Método del enfoque	Inductivo
Naturaleza de los datos	Cualitativos
Sobre el alcance	Revisión Sistemática de la Literatura
Método de procedimiento	Temática de análisis y síntesis
Variable independiente (X)	Desafíos y prácticas en Ingeniería de Requisitos y Metodología ágil
Variable dependiente (Y)	Obtención de productos software de calidad

3.2 Revisión Sistemática de la Literatura (RSL)

La referencia principal utilizada para llevar a cabo la Revisión Sistemática de la Literatura son las **Directrices para realizar revisiones sistemáticas de la literatura en ingeniería de software** de Kitchenham y Charters (2007) y Okesola et al. (2019). Las razones para usar estas pautas son que están respaldadas por cientos de citas, son ampliamente utilizadas y además fueron la recomendación específica de nuestra asesoría para realizar la revisión. Estas pautas establecen que la RSL se lleve a cabo en tres etapas principales (véase la Tabla 7), cada una de ellas compuesta por varias actividades, que son las siguientes:

Tabla 7*Etapas de la revisión sistemática*

1. Planificación	2. Realizar investigaciones	3. Informes / Difusión
▪ Identificación de la necesidad de una revisión. ▪ Especificando las preguntas. ▪ Desarrollo de un protocolo de revisión ▪ Evaluación del protocolo de revisión	▪ Identificación de la investigación. ▪ Búsqueda. ▪ Selección de estudios primarios. ▪ Estudio de aseguramiento de la calidad. ▪ Extracción y monitoreo de datos. ▪ Análisis y síntesis de datos.	▪ Extrayendo y discutiendo los resultados. ▪ Redactar el informe. ▪ Informe de formato. ▪ Evaluando el informe.

Las siguientes subsecciones contienen los detalles sobre la adaptación de estas directivas para llevar a cabo las RSL.

3.3 Identificación de la necesidad de una revisión

Según Kitchenham y Charters (2007), la revisión sistemática de la literatura es un medio para identificar, evaluar e interpretar toda la investigación disponible relevante para una pregunta de investigación en particular, un área temática o un fenómeno de interés. Este trabajo de investigación requiere conocer el estado del arte sobre los problemas relacionados con las prácticas de la ingeniería de requisitos en el proceso de desarrollo de software en proyectos ágiles y las soluciones propuestas en cada contexto particular. Por lo tanto, una revisión sistemática de la literatura se considera una metodología de investigación adecuada para reunir todo el material relevante sobre este tema.

3.4 Preguntas de investigación relacionadas

En el momento de la formulación del problema se identifica un problema general que se constituye como la pregunta de investigación, la cual motiva la realización de la RSL. Esta es:

¿Cuáles son los desafíos de la ingeniería de requisitos en las metodologías ágiles para el logro de proyectos exitosos en el desarrollo de productos de software?

Resulta imposible abordar completamente esta pregunta de investigación general dentro del alcance total del Trabajo de Investigación, debido a la amplitud de los campos de estudio relacionados. Dado que no se pretende reducir el alcance de la investigación (por ejemplo, limitándola a metodologías específicas), es muy útil, e incluso necesario, descomponer este objetivo general en objetivos específicos que faciliten la obtención de una respuesta integral.

Así, la pregunta de investigación se divide en las siguientes cuatro preguntas específicas, que orientan la Revisión Sistemática de la Literatura (RSL). Estas preguntas se resumen en la Tabla 8.

Tabla 8

Preguntas de investigación

P. investigación	Descripción
PI.1	¿De qué manera interpretan y ponen en práctica los profesionales ágiles la ingeniería de requisitos en proyectos ágiles, y cómo impactan esas interpretaciones en los resultados y desafíos enfrentados durante el ciclo de vida del proyecto?
PI.2	¿Cómo abordan y gestionan los profesionales ágiles los desafíos en la ingeniería de requisitos de software dentro de proyectos ágiles, y qué estrategias se demuestran efectivas para superar estos obstáculos en la práctica organizacional?
PI.3	¿Cómo seleccionan, adaptan y valoran los equipos ágiles las prácticas para superar los principales desafíos de la ingeniería de requisitos, y cuáles son los factores contextuales que determinan la efectividad de dichas prácticas en distintos entornos organizacionales?
PI.4	¿Cómo impactan las prácticas implementadas para superar los desafíos de la ingeniería de requisitos en proyectos ágiles en el éxito, la calidad y la satisfacción de los clientes dentro de dichos proyectos?

Las descripciones de las preguntas de investigación que orientan la RSL se presentan a continuación, con el fin de hacer más comprensible y claro su propósito:

PI 1: Como señalan los trabajos relacionados, es necesario comprender las percepciones de los profesionales del desarrollo ágil respecto a la Ingeniería de Requisitos. Esta pregunta busca responder a ese problema: entender qué piensan los profesionales ágiles sobre la Ingeniería de Requisitos constituye el primer paso para descubrir qué esperan cuando se habla de la “obtención de requerimientos o necesidades por parte del cliente o usuario”.

PI 2: Esta pregunta de investigación profundiza en las percepciones de los profesionales ágiles, con el objetivo de identificar los problemas que surgen en los proyectos a partir de los conflictos propios de la Ingeniería de Requisitos en entornos ágiles. Es importante señalar que dichos problemas pueden relacionarse tanto con la ejecución de las actividades de obtención de requerimientos en un contexto ágil, como con la ausencia de estas.

PI 3: Esta pregunta forma parte del núcleo central del presente trabajo de investigación. Su propósito es descubrir las prácticas o enfoques empleados para integrar el desarrollo ágil con la Ingeniería de Requisitos. La respuesta esperada consiste en una colección de métodos o estrategias que permitan dicha integración.

PI 4: Esta pregunta se orienta a evaluar las prácticas propuestas, identificando los beneficios y desventajas que estas generan en los proyectos de desarrollo.

Responder estas cuatro preguntas permitirá abordar la pregunta general de investigación planteada al inicio de esta sección. La respuesta se construye a partir de la comprensión del concepto de Ingeniería de Requisitos de Software desde la perspectiva de los profesionales ágiles, la identificación de los problemas derivados de la presencia o ausencia de actividades de Ingeniería de Requisitos en proyectos ágiles, las prácticas o métodos aplicados para integrar ambos enfoques y los efectos que estas medidas producen en los proyectos.

3.5 Estrategia de Búsqueda

3.5.1 Palabras clave

El enfoque propuesto para buscar material relevante provocó que se utilizaran cadenas de búsqueda en bases de datos científicas; se requirió un esfuerzo considerable para filtrar la información obtenida. Esto permitió definir un conjunto de palabras clave que facilitan la distinción entre material relevante e irrelevante.

El primer paso para establecer esta lista consistió en identificar los conceptos esenciales presentes en las preguntas de investigación que orientan la RSL.

Los conceptos clave de las preguntas de investigación corresponden a las palabras en mayúsculas (estas preguntas fueron formuladas originalmente en inglés). Se reconoce que dos de ellos —ágil e ingeniería de requisitos de software— son especialmente relevantes, mientras que los demás (proyecto, desafío, práctica, efecto) representan conceptos fundamentales que deben entenderse en el contexto adecuado. De este modo, dichos conceptos permitieron sintetizar el objetivo de la revisión sistemática de literatura.

Una vez que los conceptos clave fueron claros, se creó un conjunto de palabras clave relacionadas para ayudar en el proceso de revisión, que identificaron material relevante según se muestra en la Tabla 9.

Tabla 9*Palabras claves*

Concepto clave	Palabra clave
Agile	Agile, Scrum, Extreme Programing, XP, Dynamic Software development Method, DSDM, Crystal, Feature Driven Development, FDD, Test-Driven Development, TDD, Pair Programing, Metaphor, Lean Development, Adaptative Software Development, ASD.
Requirements engineering	Requirement Engineering, Gathering Requirement, Requirement Engineering Agile, Requirement Engineering practices, Requirement Engineering challenge agile, lifecycle development, development agile, development Waterfall, Methodology, Discipline.
Projects	Empirical study, case study, Industrial case, experiment, experience
Challenges	Problem, issue, interaction
Practices	Recommendation, solution, integration, hybrid, process
Effects	Benefits, drawbacks

Este conjunto de palabras claves ayudó a reconocer qué documentos podrían ser relevantes a primera vista, nuevamente, es importante señalar que obligatoriamente debe estar presente una de las palabras claves relacionadas con Ágil o Ingeniería de Requerimientos, las otras palabras clave (proyecto, desafíos, prácticas, efectos) son útiles para identificar los documentos que pueda responder a nuestras preguntas de investigación.

3.5.2 Fuentes de búsqueda

En la sección se mencionó que, para hacer la búsqueda más completa, se revisó el resultado de la búsqueda preliminar que se realizó para la propuesta de tesis. Esa búsqueda se realizó en las siguientes bases de datos Tabla 10

Tabla 10*Fuentes de búsqueda*

Base de datos electrónicos	Artículos buscados	Búsqueda aplicada
ACM Digital library IEEE Xplore Springer Link ISI Web of Knowledge ScienceDirect Scopus Google scholar	Journal, workshop y artículos de conferencias y/o congresos en la disciplina de Ingeniería de Software y Ciencias de la Computación, Sistemas de Información	Texto completo, para evitar perder cualquiera de los documentos que no incluyen nuestras palabras clave de búsqueda en títulos o resumen, pero que son relevantes para el objeto de nuestra revisión

A continuación, se muestra un ejemplo de una búsqueda realizada en las bases de datos electrónicas: Software AND (agile OR agility OR scrum OR "XP" OR "extreme programing" OR fdd OR "Feature-Driven development" OR "Feature-Driven" OR tdd OR

"Test Driven Development" OR kanban) AND (" Requirement Engineering " OR " Requirement " or " user story " OR "Feature").

Se compuso la cadena de búsqueda en cada base de datos manualmente en función de la funcionalidad de búsqueda que ofrece esa base de datos. Se trato en la búsqueda, en cada base de datos como un proceso de aprendizaje y experimentación.

3.5.3 Selección de estudio

Criterios de selección de estudios Inclusión y Exclusión

La inclusión o exclusión de un trabajo se decidió tomando como base los siguientes criterios los cuales son resumidos en la siguiente Tabla 11 y Tabla 12 respectivamente.

Tabla 11

Criterios de inclusión

Clave	Descripción
I-1	Los documentos deben estar redactados en inglés.
I-2	Los documentos deben ser trabajos revisados por pares.
I-3	Los documentos deben estar disponibles en texto completo.
I-4	Los documentos deben estar relacionados con ingeniería de software, Ingeniería de Requisitos de software o desarrollo de software ágil, metodologías de desarrollo de software.
I-5	Los documentos deben contener información que aborde cualquiera de las preguntas de investigación planteadas.
I-6	En caso de que un documento se haya publicado dos veces o más, la versión más reciente del documento se incluirá en el estudio.

Tabla 12

Criterios de exclusión

Clave	Descripción
E-1	Documentos redactados en otros idiomas diferente del inglés.
E-2	Documentos no disponibles en texto completo.
E-3	Documentos que no se relacionan con el desarrollo de software.
E-4	Documentos no relevantes para las preguntas de investigación.
E-5	Documentos que, a pesar de estar relacionados con el desarrollo ágil, no mencionan cuestiones de Ingeniería de Requisitos en absoluto.
E-6	Documentos que no son estudios primarios.
E-7	Se excluirán los documentos duplicados.

Procedimiento de selección de estudios

Dado que una característica fundamental de la RSL es su repetibilidad, en esta sección se describe detalladamente el procedimiento seguido para la selección de estudios. El proceso de selección de artículos se llevó a cabo mediante un enfoque adaptado de Cruzes y Dyb (2011)[41], el cual divide dicho procedimiento en distintas fases.

Fase 1 - Título e inclusión de palabras clave. En esta etapa, se seleccionaron aquellos estudios que están dentro del alcance de acuerdo con su título y palabras clave. Aquí se detectaron y eliminaron los documentos repetidos.

Fase 2 - Inclusión abstracta. Esta etapa seleccionó aquellos documentos que eran relevantes para las preguntas de investigación propuesta.

Fase 3 - Lectura rápida: Una lectura rápida de los documentos completos que permiten identificar palabras y conceptos clave que son relevantes para los propósitos de la Revisión sistemática de literatura. Esta lectura de rápida también ayudó a adquirir la inmersión requerida en el tema en sí para familiarizarse con la profundidad y amplitud de la evidencia.

Fase 4 - Lectura exhaustiva y conclusiones. Consistió en leer el documento completo para detectar duplicidades internas, artículos irrelevantes y simplemente selecciona aquellos con contenido relevante para los propósitos del estudio. Esta lectura exhaustiva también se considera útil para comprender completamente los conceptos que cada artículo intenta exponer e identificar secciones relevantes del texto, que ayudaron en las fases posteriores de la RSL, específicamente la fase de extracción de datos.

3.5.4 Evaluación de la calidad del estudio

Una vez que se obtuvo un conjunto de documentos preseleccionados que cumplen con los criterios de inclusión / exclusión establecida anteriormente la calidad de los estudios se evaluó mediante el siguiente cuestionario de calidad ver Tabla 13.

Tabla 13

Lista de verificación de evaluación de calidad

Clave	Criterio	Evaluación (*)
C-1	¿Están los objetivos del estudio claramente descritos?	Si/no/parcialmente
C-2	¿Se describen claramente los hallazgos?	Si/no/parcialmente
C-3	¿Se refiere el estudio a una metodología ágil específica?	Si/no/parcialmente
C-4	¿El estudio se refiere a problemas específicos relacionados con la arquitectura?	Si/no/parcialmente
C-5	¿El estudio describe las medidas tomadas para diseñar la arquitectura en el proyecto ágil?	Si/no/parcialmente
C-6	¿Se explican los efectos en el proyecto general?	Si/no/parcialmente
C-7	¿Se describe adecuadamente el método de recopilación de datos?	Si/no/parcialmente

(*) La escala para la evaluación es: Sí = 2 Parcialmente = 1 No = 0

Inicialmente, los documentos incluidos en la RSL deben obtener una marca de calidad mínima de 6 puntos de acuerdo con la escala anterior. Este criterio se basa en la creencia

de que cada estudio incluido debe cumplir al menos parcialmente con 2/3 de los criterios de calidad establecidos para la evaluación de los estudios seleccionados.

3.5.5 Extracción y síntesis de datos

Extracción de datos

Una revisión sistemática de la literatura requiere llevar a cabo un proceso formal de extracción de datos que explique cómo se extrae la información de las fuentes seleccionadas. En este estudio, se ha utilizado un método específico para extraer datos que consiste en una adaptación de la técnica de lectura estructurada expuesta por (Cruzes & Dyb, 2011) y (Kitchenham & Charters, 2007). El procedimiento ha consistido en realizar cuatro pasos principales de acuerdo a lo resumido en la siguiente Tabla 14.

Tabla 14

Extracción de datos

Clave	Descripción
Ext-1	Leer el documento para obtener una descripción general.
Ext-2	Leer el documento buscando la siguiente información: Resultados probados y descripción del contexto
Ext-3	Resalte y numere la información relevante encontrada.
Ext-4	Transfiera detalles clave a los formularios de datos codificados.

Síntesis de datos. Una revisión sistemática de la literatura requiere llevar a cabo una síntesis de los datos recopilados (Kitchenham y Charters, 2007). Es fundamental considerar la naturaleza de los datos obtenidos; en este caso, se trata de datos cualitativos, por lo que se realizó una síntesis cualitativa.

Esta característica resulta esencial para la elección del método y la temática en la etapa de síntesis de datos de la revisión sistemática de la literatura. En la sección siguiente, “Revisión sistemática de la literatura: resultados y análisis”, se evidencia que la mayoría de las contribuciones presentan enfoques específicos adaptados a cada contexto.

Existen diversos métodos para la síntesis cualitativa de los datos. Por ejemplo, Cruzes y Dyb (2011) ofrecen un resumen de los métodos disponibles, de los cuales la “Síntesis Temática” fue considerada adecuada para los propósitos de esta investigación.

3.6 Entrevista a Profesionales en el Ambiente de Desarrollo Ágil

Se debe precisar que esta sección se realizó una vez que finalizó la RSL; gran parte del razonamiento se muestra a continuación.

Motivación. La revisión sistemática de la literatura (RSL) descrita previamente aporta información relevante sobre el concepto de ingeniería de requisitos de software en proyectos ágiles, los desafíos enfrentados por esta disciplina, las prácticas empleadas para superarlos y los efectos derivados en los proyectos. Sin embargo, la mayoría de las fuentes relevantes incluidas se fundamentan en opiniones de expertos o teorías aún por validar. Como recomienda Abrahamsson et al. (2004), “se requiere más trabajo empírico”.

En ausencia de investigación empírica, los resultados de la RSL pueden resultar limitados en cuanto a validez y aplicabilidad, especialmente frente a la utilidad real en la industria del software. Por tal motivo, se considera pertinente complementar la revisión sistemática con entrevistas semiestructuradas que permitan acceder a la situación real del fenómeno estudiado.

Para llevar a cabo esta tarea se identificaron dos opciones principales: (a) emplear las entrevistas como fase de validación de los resultados obtenidos en la RSL, solicitando la opinión de profesionales ágiles sobre dichos hallazgos; o (b) abordar las entrevistas como un estudio independiente, estableciendo posteriormente una comparación con los resultados de la RSL. Este segundo enfoque es considerado más adecuado, ya que minimiza posibles sesgos y permite a los entrevistados compartir libremente sus experiencias, ideas y percepciones sin verse influidos por los resultados previos de la revisión.

Para este estudio, resulta más apropiada la realización de entrevistas semiestructuradas individuales (Kvale, 1994), en lugar de grupales, dado que el objetivo no es generar debates o controversias, sino aprender sobre los enfoques y estrategias implementadas por profesionales de la industria para abordar los desafíos de la ingeniería de requisitos en proyectos ágiles. Además, la entrevista individual permite un mayor control sobre el desarrollo de la sesión y facilita la obtención de información precisa y relevante.

El tipo de entrevista seleccionada para este estudio es la entrevista semiestructurada (Díaz-Bravo et al., 2013). Esta técnica de investigación cualitativa combina preguntas previamente elaboradas con preguntas abiertas, permitiendo al investigador explorar en profundidad áreas específicas que surgen durante la conversación. La estructura flexible de la entrevista semiestructurada se ajusta a la naturaleza del estudio. Aunque la información obtenida en la revisión sistemática de la literatura (RSL) facilita anticipar

algunos ejes temáticos, la flexibilidad es indispensable para indagar en detalles contextuales y experiencias únicas (Seaman, 1999).

Las entrevistas semiestructuradas permiten obtener información detallada sobre experiencias, creencias y perspectivas de los participantes, ya que favorecen un entorno conversacional donde el investigador puede seguir nuevas ideas según se presentan. En este tipo de entrevistas, el investigador diseña una guía temática con preguntas orientadoras, pero dispone de la libertad para ajustar el orden o profundizar en determinados temas conforme avanza la conversación (Flick, 2014).

Esta metodología resulta especialmente adecuada cuando se busca facilitar la expresión natural de perspectivas y percepciones por parte de los participantes, tal como es el caso de los profesionales ágiles considerados expertos en este trabajo de investigación. Además, las entrevistas semiestructuradas individuales ofrecen mayor control y profundidad en el abordaje de los testimonios, en comparación con dinámicas grupales, maximizando la riqueza y relevancia de la información obtenida.

Selección de participantes

- Perfil ideal del entrevistado

Una cuestión clave en la realización de estas entrevistas es saber quién es interrogado. Este aspecto ha sido una de las razones decisivas para elegir las entrevistas semiestructuradas como metodología de investigación para obtener información de entornos de organizaciones dedicadas al desarrollo de software.

- Búsqueda de entrevistados

Las medidas tomadas para encontrar candidatos para ser entrevistados han sido principalmente dos:

- Contactos proporcionados por La Asociación Peruana de Software (*APESOFT*, n.d.).
- Contactos personales: contactos personales que actualmente trabajan en proyectos ágiles en las empresas desarrolladoras de software.

La etapa de entrevistas se realizó en Setiembre de 2022.

Protocolo de entrevistas

- Estrategia de entrevista

Como se mencionó, las entrevistas buscan proporcionar una comprensión más profunda sobre las percepciones de los profesionales en ambientes de desarrollo ágil, encontrando lo que realmente sucede en las empresas desarrolladoras de software, complementando los resultados de la RSL. Desde este punto de vista, está claro que las preguntas de investigación que orientaron las entrevistas son las mismas que las que orientaron la RSL. Las pautas para crear el procedimiento para llevar a cabo las entrevistas se han obtenido de (Carolyn B Seaman, 1999). Estas pautas recomiendan la creación de una guía de entrevista; Esto consiste en una lista de preguntas abiertas que ayudan a organizar la reunión. El guion general para la entrevista se puede estructurar en tres fases diferentes:

- Caracterización del entrevistado.
- Caracterización del proyecto.
- Preguntas de investigación

Caracterización del entrevistado. En esta sección, el objetivo es crear un perfil del entrevistado para poder clasificar la información encontrada, y también para facilitar la realización de la entrevista. Puede ser interesante considerar información como experiencia total en proyectos de ingeniería de software, experiencia en ambientes de desarrollo ágil, metodologías utilizadas en el pasado, rol en el equipo, etc.

Caracterización del proyecto. En esta sección se busca definir el contexto en el que se aplica la metodología ágil. La información relevante aquí sería:

Complejidad del proyecto: tamaño, conocimiento del dominio, etc.

Metodología ágil utilizada: cultura, adopción, etc.

Equipo: tamaño, experiencia, roles, etc.

Cliente: implicación en el equipo, demandas, etc.

Preguntas de investigación. Esta fase, las más abiertas y amplias de las tres fases, se realizó preguntas destinadas a resolver las preguntas de investigación que impulsaron la RSL. De esta forma, es más fácil establecer una comparación entre la información obtenida del SLR y los testimonios del entrevistado.

Guía de entrevista

Aquí se elabora la guía de las entrevistas, creado para mantener la coherencia y el orden en la conversación, aunque se le permitirá hacer preguntas fuera de este guion (ver Apéndice - Guía de entrevista),

Realización de entrevistas

Participantes anónimos. Para que el entrevistado se sienta más cómodo compartiendo detalles sobre su trabajo en las empresas, se garantiza el anonimato de las respuestas proporcionadas.

Duración de la entrevista. La intención es que la duración aproximada de las entrevistas sea de 30 minutos cada una. Esto se debe a que es más probable que un entrevistado cansado proporcione información menos relevante. La guía de la entrevista también ayuda a mantener el enfoque en los temas relevantes y, de ese modo, evitar extender innecesariamente la charla.

Recolección de información. La recopilación de la información de las entrevistas se habilitó de la siguiente manera.

- Grabación de entrevistas: para facilitar el proceso de recopilación de datos, las entrevistas se grabaron utilizando un teléfono móvil o capturando la pantalla y la voz, en caso de videoconferencias.
- Transcripción de entrevistas: las grabaciones de las entrevistas no se transcribirán completamente a palabras escritas; en cambio, se tomaron notas breves sobre el contenido de las entrevistas. Las citas de los entrevistados que sean especialmente significativas fueron el único contenido que se transcribió literalmente.

4 RESULTADOS Y ANÁLISIS

En la investigación sobre los desafíos de la Ingeniería de Requisitos en el desarrollo de software ágil, la revisión sistemática de la literatura (RSL) juega un papel crucial en la recopilación de información relevante y actualizada. En esta sección, se presentan los resultados obtenidos tras un exhaustivo proceso de búsqueda y selección de fuentes académicas, que han permitido consolidar un cuerpo de conocimientos significativo sobre el tema. Comenzando por una revisión detallada de las referencias citadas por Inayat et al. (2015), se identificaron un total de 2375 fuentes potenciales. A medida que se aplicaron filtros de evaluación, se fueron descartando aquellas que no cumplían con criterios de relevancia, lo que culminó en 37 documentos seleccionados para un análisis profundo. La exhaustiva selección de literatura relevante no solo refuerza la fundamentación teórica del estudio, sino que también ofrece conocimientos valiosos sobre la manera en que los profesionales ágiles abordan la ingeniería de requisitos y los desafíos relacionados en sus proyectos.

4.1 Resultados de la Revisión Sistemática de la Literatura (RSL)

En esta sección se presentan los resultados de la búsqueda y extracción de información de fuentes y bases de datos que están disponibles y se constituyen relevantes para el objetivo de la investigación.

Como se mencionó en el capítulo anterior, la estrategia para localizar fuentes relevantes ha consistido en revisar todas las referencias y citas de (Inayat et al., 2015). La RSL, que se ha tomado como punto de partida, y se repite sucesivamente el proceso para encontrar todo el material relevante disponible.

Esta estrategia ha proporcionado 2375 fuentes potenciales para analizar, que son una cantidad considerable de contribuciones relacionadas con el tema, aunque debe tenerse en cuenta que algunas de estas fuentes de interés iniciales se repiten.

La primera etapa del procedimiento de selección consistió en revisar el Título y las Palabras clave de cada contribución tratando de encontrar material relevante, también en este paso se han eliminado las fuentes duplicadas; En total, se han eliminado 1949.

Después de revisar los títulos y las palabras clave, el objetivo de la revisión ha sido el resumen de las fuentes de interés. El nivel de relevancia requerido en este paso ha sido que el resumen está relacionado no solo con el área o campo de estudio, objeto de la investigación, sino también con alguna de las preguntas de investigación que orientaron la RSL, luego, se han eliminado 232 fuentes.

En ese momento, era posible asegurar que las fuentes que aún quedaban estuvieran de alguna manera relacionadas con el objetivo de la investigación. Siguiendo las recomendaciones de (Cruzes & Dyb, 2011), se realizó una lectura ligera para sumergirse en el tema en sí mismo e identificar aspectos o ideas clave que son relevantes para responder las preguntas de investigación. El paso lectura rápida eliminó 105 fuentes más.

Al final del proceso se realiza una lectura en profundidad del resto de fuentes. Este paso tiene como objetivo descubrir información que pueda ser relevante para preguntas de investigación específicas; se trata de encontrar fuentes que proporcionen material relevante para ayudar a responder las preguntas de investigación. Al mismo tiempo, se aplican estrictamente los criterios de inclusión y exclusión definidos en el protocolo de la RSL. En este paso, se eliminaron 52 fuentes. Al final se obtuvieron 37 documentos seleccionados.

En la Tabla 15 se puede encontrar una descomposición detallada de las fuentes de interés encontradas y los filtros sucesivos aplicados para eliminar fuentes irrelevantes. Todo el algoritmo del procedimiento de selección se resume gráficamente en la Ilustración 4, que muestra los procesos y el resultado del estudio de selección:

Tabla 15

Criterios de filtro de las fuentes

	Citaciones	Referencias	Resultados
Fuentes potenciales	991	1384	2375
Filtro / Títulos	869	1080	1949
Filtro / Resumen	63	169	232
Filtro / Lectura Rápida	28	77	105
Filtro / Lectura Exhaustiva	17	35	52
Pre-Seleccionado	14	23	37

Todo el algoritmo del procedimiento de selección se resume gráficamente en la figura 5.

Caracterización de las fuentes incluidas

Es posible establecer diferentes clasificaciones para caracterizar todo el conjunto de contribuciones incluidas en la RSL; esto se debe a la diversidad y la amplitud de los

campos de estudio de este trabajo de investigación, la tabla 16 resume los tipos de contribuciones.

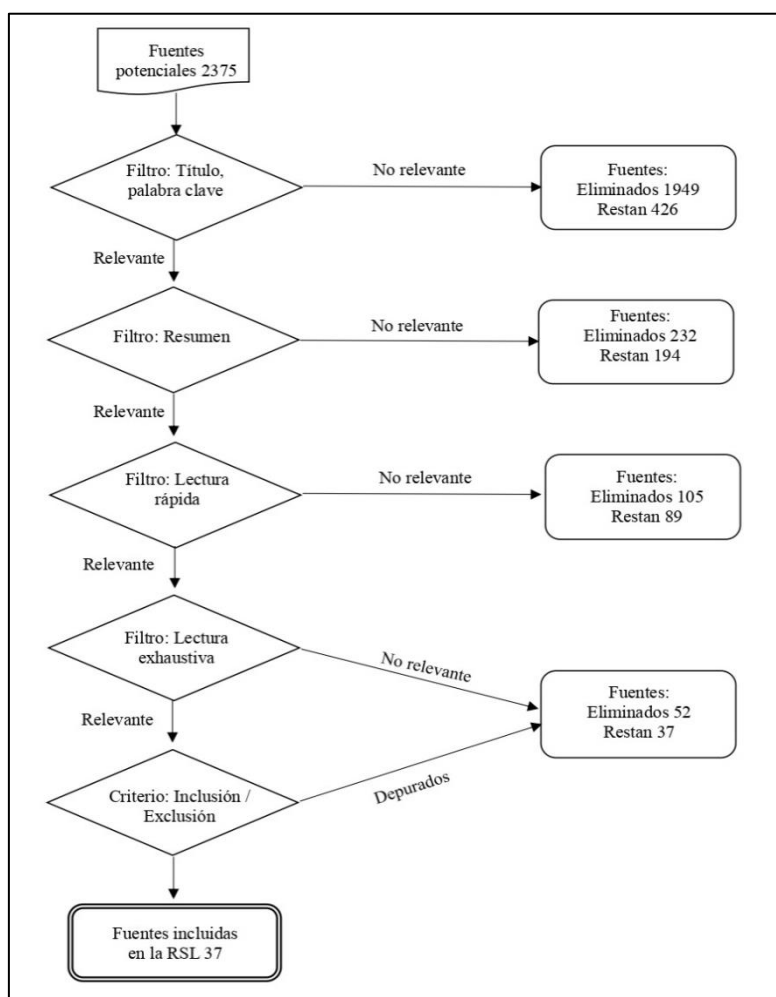
Tabla 16

Tipo de contribuciones

Tipo de dato	Número fuentes	Porcentual
Estudios empíricos	6 fuentes	16%
Opinión de expertos	18 fuente	49%
Contribuciones mixtas	13 fuentes	35%

Ilustración 4

Procesos y resultado del estudio de selección



Metodología relacionada

Es necesario especificar que la misma contribución podría estar relacionada con más de una metodología o enfoque en el contexto ágil, el cual se resume en la tabla 17.

Tabla 17*Metodologías*

Metodologías	Número de fuentes	Distrib. porcentual
Ningún método específico	13 fuentes	35%
Programación extrema (XP):	9 fuentes	24%
Scrum	8 fuentes	21%
Desarrollo de Software adaptativo	2 fuentes	5%
Cristal	2 fuentes	5%
Desarrollo Dirigido por Funciones	1 fuente	3%
Desarrollo Dirigido por pruebas	1 fuente	3%
Proceso unificado ágil	1 fuente	3%

4.2 Resultados de RSL

Es pertinente anotar que diferentes metodologías ágiles presentaron diferentes desafíos en relación con la ingeniería de requisitos; los enfoques para resolver esos desafíos y problemas también serán diferentes, por lo que serán las consecuencias de implementar esas medidas.

Revisando las preguntas de investigación de manejo de esta RSL, ha sido posible encontrar los siguientes resultados.

4.2.1 Pregunta de investigación 1

¿De qué manera interpretan y ponen en práctica los profesionales ágiles la ingeniería de requisitos en proyectos ágiles, y cómo impactan esas interpretaciones en los resultados y desafíos enfrentados durante el ciclo de vida del proyecto?

Los resultados de la RSL muestran que no existe ningún concepto especial de ingeniería de requisitos que pueda extraerse de las fuentes incluidas, esto se justifica por lo que en las metodologías ágiles no se tiene los roles diferenciados para cada miembro del equipo, por lo que todos comparten las diversas funciones. Esta es probablemente la mayor escasez en los resultados de de la Literatura; No ha sido posible encontrar mucha información sobre el concepto de ingeniería de requisitos con nombre propio para metodologías ágiles.

Sin embargo, para los métodos eXtreme Programming (XP), Scrum que aplican técnicas tradicionales de IR en sus procesos de desarrollo, ellos utilizan diferentes técnicas clásicas. En la RSL los autores utilizan la combinación de varias técnicas en el proceso

de obtención de requisitos como: entrevistas, lluvia de ideas, priorización, análisis, modelado, caso de uso, escenarios, entrevistas.

En la RSL si se encuentra información sobre artefactos de obtención de requerimientos, por ejemplo, mediante las tarjetas denominadas “historia de usuario” en donde se resume la necesidad por parte del usuario y se adjunta a este el criterio de verificación, [S22] [S23] [S24].

4.2.2 Pregunta de investigación 2

¿Cómo abordan y gestionan los profesionales ágiles los desafíos en la ingeniería de requisitos de software dentro de proyectos ágiles, y qué estrategias se demuestran efectivas para superar estos obstáculos en la práctica organizacional?

Para alcanzar una visión global de los desafíos que surgen con respecto a la Ingeniería de Requisitos en ambientes de desarrollo ágil, se describen los siguientes considerandos antes de y después del desarrollo.

4.2.2.1 Delimitación de idoneidad.

Las propuestas de desarrollo ágil no funcionan para todo tipo de proyecto; Esta es una opinión compartida entre la mayoría de las fuentes incluidas en la RSL [S04] [S10][S13]. Existen ciertas condiciones que restringen a priori la aplicabilidad de los principios ágiles; estos se pueden clasificar de la siguiente manera:

La propuesta de desarrollo ágil no es adecuada para proyectos de gran envergadura

En muchos estudios precisan que la estimación del presupuesto y el cronograma es un desafío para las organizaciones que siguen métodos ágiles. Aunque la práctica de métodos ágiles permite la valoración inicial de un proyecto, no es posible hacer estimaciones iniciales debido a requisitos volátiles, planificación dinámica y diseño (Sarkan et al., 2011) [S18]. El alcance de un proyecto se basa en historias de usuarios conocidas y algunas de estas pueden descartarse en próximas iteraciones. Por lo tanto, las estimaciones a veces se ven significativamente afectadas [S05] [S18].

La propuesta ágil no se recomienda para proyectos que implican tiempo e inversión considerablemente alto, en este tipo de proyectos, muy a menudo los clientes requieren una documentación completa del sistema debido a las grandes dimensiones del proyecto, por lo que parece claro que las metodologías ágiles no son adecuadas para este propósito (Zaki & Moawad, 2010)(Guha et al., 2011) [S13] [S27].

Conocimiento del dominio y complejidad del proyecto

Aunque no todas las fuentes mencionan explícitamente este desafío, se puede deducir la importancia de la experiencia en el campo en el contexto ágil. Cuando un equipo tiene experiencia en un área específica, su dominio permite la posibilidad de crear soluciones incrementales (Farid & Mitropoulos, 2012) [S09], siempre que el equipo posea una comprensión profunda del problema.

Algunos autores señalan los problemas que puede tener el uso de ágil en proyectos complejos y/o críticos (Boehm & Turner, 2003) [S04]. Esto sucede porque en entornos ágiles la mayoría de las decisiones se toman muy rápido y se aprenden sobre la marcha, sin tener en cuenta todas las situaciones posibles que un sistema puede enfrentar.

Tamaño del equipo

Esto es extremadamente importante; Las metodologías ágiles requieren una gran coordinación y comunicación entre el equipo. Trabajando con equipos muy grandes, incluso los equipos distribuidos pueden significar un obstáculo inevitable para la correcta aplicación de los principios ágiles. (Lehman & Sharma, 2011)[S10].

4.2.2.2 Desafíos del proyecto.

Esta categoría de desafíos se compone de problemas que aparecen durante la ejecución del proyecto, no antes de él, como en la sección anterior. Estas dificultades son obvias para el equipo o el cliente y pueden significar esfuerzos muy costosos para resolverlas sobre la marcha (Sillitti et al., n.d.)(Yu & Sharp, 2011)[S03] [S07] [S09] [S10] [S13][S20] [S21].

Falta de calidad, gestión profesional y compromisos contractuales: Esto se refiere al hecho de que muy a menudo en proyectos ágiles, la calidad desde las perspectivas de ISO y/o CMMI no se considera relevante; el tiempo, el presupuesto y la funcionalidad llaman la atención del equipo [S09] y el software resultante no presenta la calidad requerida (Jeon et al., 2015) [S19] [S28].

Respecto a los compromisos y limitaciones contractuales a ello se suma la volatilidad de los requisitos son importantes al no permitir cambios en los requisitos después de la firma del contrato; los cambios pueden causar un aumento en el costo y, a veces, el fracaso del proyecto, por lo tanto, se deben tomar medidas legales para evitar tal situación y manejar adecuadamente la naturaleza flexible de la ingeniería de requisitos ágiles. Sin embargo,

estos problemas pueden resolverse mediante un pago fijo por lanzamiento, que protege las inversiones y evita la volatilidad de los requisitos. También es probable que los contratos de pago fijo reduzcan los riesgos específicos del proyecto a nivel de proveedor en proyectos externalizados basados en el desarrollo ágil [S26]. Del mismo modo, la eliminación de requisitos imprecisos e incorrectos basados en cambios requiere un costo adicional y un esfuerzo para gestionar los esfuerzos inútiles. Esta situación se puede manejar incrementando la comunicación e involucrando a los clientes para evitar la reelaboración posterior.

4.2.2.3 Desafíos del equipo.

Esta familia contiene desafíos que afectan directamente al equipo, ya sea en su percepción de las tareas que están llevando a cabo, su comunicación (malentendidos, discusiones, dificultades para establecer comunicación) o en la dependencia o su experiencia para realizar un trabajo efectivo [S9] [S18].

Confianza en la experiencia: Esto refleja que las metodologías ágiles dependen demasiado de la experiencia de las personas que componen el equipo, no del método en sí. Esto puede conducir a una dependencia de las personas que podría significar grandes problemas en el caso de que esas personas no estén en el equipo [S9] [S18]

Problemas de comunicación del equipo: El rol del gestor del proyecto es un vehículo muy útil para permitir la comunicación entre el equipo; Todos los miembros del equipo pueden compartir una visión común del software consultando el diseño del sistema, por ejemplo, en XP se trata como una metáfora y contribuye a tener una visión. Quitar este vehículo puede llevar a perder la perspectiva común del sistema [S9].

4.2.2.4 Desafíos del cliente.

Estos desafíos comparten la característica común de que todos involucran directamente al cliente; ya sea durante el proyecto [S1] o cuando se entrega el software.

El cliente en el lugar de desarrollo es asumido y defendida en los ambientes ágiles. Sin embargo, esta suposición a menudo no es realista, ya que los estudios empíricos confirmaron que la disponibilidad del cliente y el acceso son un desafío general [S29] [S18]. Si bien ningún estudio discute que los cambios en los requisitos pueden ser determinados directamente por el cliente para acelerar el proceso, la disponibilidad del cliente se considera desafiante y muy escasa, debido a muchos factores desde una

perspectiva comercial, como el tiempo, costo y carga de trabajo del representante del cliente (Racheva et al., 2014) [S17].

Problemas de comunicación con el cliente: Además de proporcionar una descripción completa del sistema, La ingeniería de requisitos es básica para permitir una comunicación fluida con el cliente a fin de garantizar la obtención de las necesidades del cliente respecto al software que se está desarrollando. La falta de ingeniería de requisitos hace que esta comunicación sea mucho más complicada y aparecen interpretaciones no siempre coincidentes entre el equipo y el cliente [S01].

La incapacidad y el acuerdo del cliente son los dos problemas principales, aparte de la disponibilidad, según [S26]. La incapacidad del cliente se refiere a la incompetencia del cliente en términos de toma de decisiones y conocimiento completo del dominio, y el acuerdo del cliente se trata del consenso de más de un grupo de clientes involucrado en un proyecto [S16] [S18]. El desacuerdo entre los grupos de clientes afecta el rendimiento, especialmente en ciclos cortos de desarrollo [S18].

Falta de documentación: Sin la especificación del Software, el cliente no tiene ningún documento de referencia que le permita comprender completamente su sistema; en cambio, existe una dependencia total de las personas que crearon el sistema para conocerlo (Pichler & Wahler, 2006)[S16]. Si esto sucede, el cliente podría encontrarse en una situación en la que no conoce su propio sistema y no tiene ninguna herramienta para obtener este conocimiento.

Por tanto, la documentación mínima es un desafío vital que los métodos ágiles representan para los equipos de desarrollo [S05] [S18]. Cada vez que hay un lapso de comunicación debido a cambios repentinos en los requisitos, la falta de disponibilidad de representantes apropiados del cliente, la complejidad del proyecto y la falta de documentación, surgen múltiples problemas. Además, si se supone que los requisitos deben comunicarse a los clientes en ubicaciones geográficas distribuidas y no colocadas o en el sitio, se vuelve engorroso abordar tal situación con poca o ninguna documentación (Goetz, 2002)(Maiti et al., 2018) [S30] [S36]. En algunos casos, el espacio de trabajo de un equipo, es decir, sala u oficina, no es lo suficientemente grande como para acomodar a todos los miembros; en ese caso, la comunicación verbal de los requisitos es insuficiente.

4.2.3 Pregunta de investigación 3

¿Cómo seleccionan, adaptan y valoran los equipos ágiles las prácticas para superar los principales desafíos de la ingeniería de requisitos, y cuáles son los factores contextuales que determinan la efectividad de dichas prácticas en distintos entornos organizacionales?

4.2.3.1 Enfoques propuestos.

Contrariamente a la intuición inicial de que la mejor manera de combinar la Ingeniería de requisitos y Desarrollo de Software Ágil sería modificar los procesos Ágiles para apoyar las tareas obtención de necesidades por parte del cliente, los resultados de la RSL han demostrado que agregar la actividad explícitamente a la agilidad es una posibilidad, pero otra posibilidad es hacer que la actividad de ingeniería de requisitos sea más ágil, y también hay otra opción que consiste en olvidar todo y desarrollar una metodología completamente nueva que combine los dos mundos: el dinamismo de lo ágil con la disciplina de la ingeniería de requisitos de software y los enfoques tradicionales.

Enfoque ágil e ingeniería de requisitos: Este enfoque abarca prácticas que se agregan a un proyecto ágil para permitir la actividad de ingeniería de requisitos. Consiste en variar las actividades que generalmente se llevan a cabo en el proyecto ágil con prácticas tradicionales que se pueden insertar y usar tratando de afectar lo menos posible a la agilidad del proceso general. Las fuentes incluidas en la RSL que siguen el enfoque ágil con ingeniería de requisitos son: [S6] [S7] [S8] [S10].

Las prácticas asociadas a este enfoque son:

- Realizar análisis de impacto en cada iteración.
- Iteración cero.
- Cliente involucrado en tareas identificación de necesidades.
- Roles dedicados.
- Uso del conocimiento.
- Ingeniería de Requisitos basada en la experiencia.
- Mejorar la calidad.

4.2.3.2 Categorización de prácticas.

Anteriormente, se presentaron las prácticas agrupándolas atendiendo al enfoque general al que pertenecen. Pero puede surgir una categorización diferente siguiendo el espíritu de

la clasificación de los desafíos; esto está atendiendo a la naturaleza de las prácticas propuestas. En la medida en que se haya proporcionado una descripción de cada práctica en la subsección anterior, no se repetirá nuevamente aquí.

Prácticas ágiles propias: En este caso, estas prácticas se conciben de procesos o principios a partir de enfoques ágiles. Esto consiste en transformar procedimientos ágiles para hacerlos compatibles con el trabajo de IR. Estas prácticas son. (Haugset & Stålhane, 2012)(Mahmud & Veneziano, 2011)[S9] [S10] [S11] [S14] [S18] [S22] [S26].

- Iteración cero
- Priorización de los requisitos
- Proceso iterativo e incremental

Prácticas de comunicación: En la práctica, la mayoría de los equipos ágiles tienen sustitutos o clientes sustitutos para desempeñar el papel de un cliente real (por ejemplo, propietarios de productos [S26]). Otras organizaciones implementan la práctica del "desarrollador en el sitio" moviéndose un representante del desarrollador en el sitio del cliente (Martakis, 2015)[S15] [S17].

Estas prácticas se centran en mejorar la comunicación; persiguen habilitar el trabajo obtención de requisitos a través de la mejora de la comunicación entre el equipo y con el cliente. [S12] [S22] [S23] [S24]

Estas prácticas son.:

- Reuniones de revisión (Retrospectiva).
- Cliente involucrado en tareas identificación de necesidades.
- Uso de herramientas de comunicación.
- El scrum master o el coach como habilitador de comunicación.

Prácticas de calidad: En un enfoque similar a las prácticas de comunicación, este tipo de prácticas está orientado a mejorar la calidad del software creado, y en ese contexto se necesita unas reuniones de retrospectiva (Boness et al., 2008) [S09] [S16] [S18] [S29] [S35].

Estas prácticas son:

- Mejorar la calidad.
- Los atributos de calidad.

4.2.3.3 Definición de prácticas.

Esta práctica analiza las consecuencias de los cambios introducidos en cada iteración para identificar riesgos y abordarlos (Klaus Pohl, 2015) [S13] [S18] [S34].

Iteración cero: Esta práctica consiste en incluir una iteración “Cero” (anterior a las comunes) para llevar a cabo un análisis general de los proyectos, crear diseños preliminares y establecer el contexto general del proyecto [S09] [S10] [S12] [S13] [S16] [S18].

Reuniones de evaluación: Esta práctica consiste en incluir la evaluación en las reuniones diarias; De esta forma, se convierte en una práctica conjunta de todo el equipo. Este enfoque combina el trabajo de consenso con el principio ágil de "trabajar juntos" (Jun et al., 2010)[S12] [S26] [S32].

Uso de conocimiento: Esto se refiere a la confianza en la experiencia de los miembros del equipo y también en la captura de conocimiento para su posterior reutilización o compra de soluciones predefinidas que satisfagan las necesidades del proyecto (Hruschka, 2018)(Cao & Ramesh, 2008b)(Sohaib & Khan, 2010) [S33] [S37].

Mejorar la calidad: Esta práctica consiste en remarcar la importancia de la calidad al momento de medir el éxito de un proyecto; también se refiere a medidas específicas tomadas para cuidar aspectos de calidad del software [S9] [S16] [S18] [S29].

Proceso iterativo e incremental: Esto indica que las tareas de obtención e identificación de necesidades se abordan varias veces y, a veces, de forma incremental; esto es, enfocándose progresivamente en dominios más grandes. Cabe señalar que estas dos características pueden aparecer juntas o separadas, esto es, de formas iterativas, incrementales o ambas al mismo tiempo [S5] [S6] [S7] [S8] [S16] [S17] [S19] [S21].

Uso de herramientas de comunicación: Hace referencia al uso de herramientas como un medio para mejorar la comunicación del equipo (usando un wiki, chat, github). Esta práctica puede significar costos adicionales para adquirir e implementar estas herramientas [S9].

Dividir equipos: Esta práctica está indicada para superar las limitaciones relacionadas con los grandes equipos y los problemas de comunicación que esto conlleva. Por lo general, la división del equipo va junto con la descomposición del problema o con el trabajo concurrente [S13].

Enfoque híbrido: Esta práctica intenta superar las dificultades que encuentran las metodologías ágiles en algunos contextos (por ejemplo, en dominios desconocidos). El enfoque consiste en seguir una metodología híbrida tradicional-ágil para cumplir con el dominio y al mismo tiempo aplicar principios ágiles cuando sea posible [S9].

Evaluación y caracterización del proyecto: Esta práctica consiste en evaluar y analizar el proyecto atendiendo a cinco factores (personal, dinamismo, cultura, tamaño y criticidad) [S04].

4.2.4 Pregunta de investigación 4

¿Cómo impactan las prácticas implementadas para superar los desafíos de la ingeniería de requisitos en proyectos ágiles en el éxito, la calidad y la satisfacción de los clientes dentro de dichos proyectos?

A continuación, se describe, después de conocer los principales desafíos relacionados con la ingeniería de requisitos en los proyectos de desarrollo de software ágil y las prácticas utilizadas para superarlos, nos toca verificar qué efectos pueden tener estas prácticas en el proyecto.

4.2.4.1 Principios ágiles mejorados.

Esta familia de efectos comparte la característica de que todos están contenidos en los principios ágiles detrás del manifiesto ágil [15]: principios como "más comunicación", "entregas rápidas", "participación del cliente", etc. están presentes aquí [S03] [S6] [S9] [S22].

Menos documentación: Se producen menos documentos [S9]. Los métodos ágiles reemplazan la documentación de requisitos convencionales con historias de usuario precisas y orientadas al "objetivo del usuario" [S31].

Se mejora la comunicación tanto entre el equipo como con los clientes [S9] [S26].

El análisis y el diseño creado es menos complejo, más comprensible y más fácil de manejar [S03] [S9] [S22] [S26].

La agilidad del proyecto no se ve afectada por las prácticas introducidas para identificar requisitos [S28] [S31].

Mayor participación del cliente: Las prácticas hacen que el cliente esté más involucrado en las tareas del proyecto, de esta manera, el cliente está más consciente de lo que se ha estado haciendo en el proyecto [S25] [S26].

4.2.4.2 Efectos negativos.

Este grupo hace referencia a algunos inconvenientes que el uso de ciertas prácticas para superar el "problema identificación y obtención de requisitos en el ambiente ágil" puede causar en el proyecto [S03] [S26].

Dependencia de experiencia del equipo: Las prácticas introducidas se basan en la experiencia del equipo. Esto plantea un problema; Si no hay documentación de referencia del sistema, cada vez que se desea cambiar algo, se necesita el equipo que creó el sistema, ya que son los únicos que poseen el conocimiento requerido sobre cómo es el sistema. Esta dependencia puede ser un problema muy importante para cualquier organización [S03].

4.2.5 Análisis metodológico de resultados

Para fines prácticos, se ha considerado como una contribución interesante incluir un mapeo de los resultados encontrados, atendiendo a la metodología ágil en particular para XP y Scrum.

Se ha mencionado que hay algunas fuentes que no hacen referencia a un método específico, y otras que se centran en una o varias metodologías ágiles específicas. En esta sección, se presenta una clasificación de los desafíos, prácticas y efectos causados agrupados por metodología.

4.2.5.1 Programación extrema (XP).

Desafíos detectados

- Confianza en la experiencia [S18].
- Falta de reutilización [S13].
- Se produce software no confiable [S13].
- Falta de calidad [S19].
- Falta de flexibilidad [S26].
- Falta de integridad conceptual [S27].

Prácticas implementadas

- Proceso iterativo e incremental [S19] [S21].
- Iteración cero [S13] [S18].
- Análisis y Diseño preliminar de alto nivel [S13] [S21].
- Esfuerzo de integración [S13] [S21].
- Roles dedicados [S19].
- Priorización de los requisitos [S13].
- Realizar análisis de impacto en cada iteración [S13] [S18].
- Cliente involucrado en tareas de IR [S18] [S25].
- Reuniones de revisión [S26].
- Dividir equipos [S13].
- Uso de herramientas de comunicación [S26].
- Mejorar la calidad [S18].
- Los atributos de calidad [S18].
- Uso del conocimiento [S33].

Efectos observados

Positivo:

- Agilidad mantenida [S28].
- Reducción del tiempo de desarrollo [S28].
- Reusabilidad mejorada [S28].
- Mayor participación del cliente [S25] [S26].
- Reducción del tiempo de comercialización [S25].
- Comunicación mejorada [S26].
- Calidad mejorada [S22]

Negativo:

- Dependencia de los equipos que implementan [S26]

4.2.5.2 Scrum.

Desafíos detectados

- Falta de calidad [S28].
- Problemas de mantenimiento [S28].
- Problemas de comunicación del equipo [S28].

- Confianza en la experiencia [S32].
- Se produce software no confiable [S32].
- Agile no es adecuado para grandes proyectos [S32].

Prácticas implementadas

- Proceso iterativo e incremental [S30] [S31] [S32].
- Diseño preliminar de alto nivel [S12].
- Priorización de los requisitos [S30] [S31].
- Roles dedicados [S26] [S32].
- Mejorar la calidad [S28].
- Iteración cero [S12].
- Los atributos de calidad mejoran [S28].
- Cliente involucrado en tareas identificación y obtención de requisitos [S22] [S23].

Efectos observados

Positivo:

- Agilidad mantenida [S29].
- Tiempo de desarrollo reducido [S29].
- Calidad mejorada [S29].

4.3 Entrevistas, Resultados y Análisis

4.3.1 Perfiles de los entrevistados

Esta sección presenta los resultados obtenidos a partir de las entrevistas realizadas con profesionales de entornos de desarrollo ágil. Para los fines de este trabajo de tesis, se priorizó la calidad por encima de la cantidad al seleccionar la muestra de participantes. El proceso de selección se desarrolló de manera intencional y basada en criterios estrictos de experiencia profesional y formación académica, con el objetivo de asegurar la relevancia y profundidad de los testimonios recopilados.

En total, participaron quince profesionales considerados “expertos” en desarrollo de software ágil e ingeniería de software. Todos los entrevistados cuentan con el calificativo de programadores senior y poseen títulos de magíster o doctorado, además de una reconocida trayectoria en la industria del software. La selección de esta muestra permitió

acceder a perspectivas y prácticas fundadas en la experiencia directa, facilitando el análisis en profundidad de los desafíos, enfoques y soluciones adoptados en contextos ágiles de alto nivel de exigencia.

La estrategia metodológica adoptada —centrada en la selección intencionada de informantes clave con amplia experiencia y formación especializada— asegura la validez interna de los datos cualitativos, pues se centra en voces reconocidas y capacitadas para reflexionar críticamente sobre la problemática objeto de estudio. De este modo, los hallazgos aquí expuestos ofrecen una visión autorizada y matizada acerca de la ingeniería de requisitos y su gestión en proyectos ágiles contemporáneos.

Se garantizó el anonimato de los entrevistados en el momento de realizar las entrevistas, por lo que cada entrevistado recibirá un código: E1, E2, E3, E4, E5, E6, E7, E8, ...E15.

Como se mencionó en la sección caracterización del entrevistado, el perfil de los diferentes participantes es interesante para ajustar la información proporcionada en el contexto apropiado, y también fue muy útil durante la entrevista para formular preguntas adicionales que son familiares a ellos. El perfil de los entrevistados se puede ver en tabla 18.

Tabla 18

Perfil de profesionales entrevistados

	E8	E9	E10	E11	E12	E13	E14	E15
Experiencia en Ingeniería de Software	27 años	16 años	15 años	10 años	8 años	7 años	4 años	14 años
Experiencia en desarrollo ágil	12 años	10 años	9 años	5 años	4 años	5 años	3 años	7 años
Enfoque ágil en uso	Scrum, XP	Scrum	Scrum	XP	Scrum, Kanban	Scrum, TDD	XP	XP/SCRUM
Puesto que ocupa en la actualidad	Desarrollador, trato / cliente	Desarrollador, Scrum Master	Scrum Master	PMI-ágil	Desarrollador, Scrum Master	Desarrollador, Tester	Coach ágil, consultor	Scrum master, consultor
Experiencia empresa actual con enfoque ágil	10 años	5 años	7 años	5 años	3 años	5 años	3 años	10 años
Aporte de la academia en el enfoque ágil	Autodidacta	Autodidacta cursos extrauniversitarios	Autodidacta	Autodidacta	Autodidacta	Autodidacta	Autodidacta	Auto-didacta, extra-universitario

Como se nota en el perfil que se muestra en la tabla anterior, la mayoría de los entrevistados tienen una alta experiencia trabajando en Ingeniería de Software, y también, una experiencia considerable en trabajar con Metodologías Ágiles (el menos experimentado ha estado trabajando durante 3 años); Considerando esto, sus experiencias pueden considerarse más valiosas para los propósitos del presente trabajo de tesis. No obstante, debe destacarse de la tabla sobre el aporte de la academia en cuanto

conocimiento que se difunde al respecto es cual es deficiente, todos los entrevistados manifiestan ser autodidactas al respecto.

Además, sus antecedentes también son interesantes. Todos tienen diferentes antecedentes con similitudes, es decir, todos han trabajado con Scrum; Programación Extrema no se usa de la forma esperada, otras Metodologías y Prácticas Ágiles como TDD, BDD o Kanban, o Lean. Esta diversidad de técnicas utilizadas por los entrevistados da credibilidad a sus testimonios en la medida en que se originan en diferentes experiencias y contextos. Esta diversidad de orígenes también puede hablar positivamente sobre la generalización de la información obtenida.

4.3.2 Caracterización del proyecto

Como se ha dicho en la sección anterior, las experiencias de los entrevistados se originan en diferentes contextos. A pesar de esto, se ha considerado útil caracterizar el proyecto actual en el que trabajan los entrevistados, ya que los resultados de la RSL mostraron una preocupación por las condiciones de aplicabilidad de la filosofía de desarrollo ágil. Esta sección muestra las características de los contextos donde se usa el concepto Ágil.

En general, los participantes estaban trabajando en proyectos medianos; este punto fue bastante difícil de aclarar, ya que las referencias dadas por los participantes fueron bastante escasas y vagas. Ninguno de ellos expresó claramente que estaban trabajando en proyectos grandes; sin embargo, la mayoría de los entrevistados eran propensos a transmitir la idea de trabajar en proyectos de mediano tamaño.

Lo mismo ocurrió con respecto a la complejidad del sistema, les resultó bastante difícil evaluar el grado de complejidad de sus sistemas. La mayoría de los sistemas construidos eran sistemas de información gerencial; en consecuencia, la mayoría de ellos expresaron que consideraban sus proyectos como proyectos de complejidad media.

En este estudio también fue interesante conocer la experiencia que el equipo tenía en el dominio, por lo que se preguntó a los entrevistados sobre experiencias previas en el mismo tipo de sistemas. Con una excepción, todos trabajaron en proyectos conocidos por el equipo, incluso con sistemas heredados. Estos participantes trabajaron en equipos cuyo cliente era su propia empresa. La excepción mencionada anteriormente fue el único de los entrevistados cuyo equipo trabajó para clientes externos.

4.3.3 Caracterización del equipo

Conocer los antecedentes de cada entrevistado y las características generales de cada proyecto es muy útil, pero la imagen del contexto en el que se aplica el desarrollo de software ágil no estaría completa sin obtener información sobre las características del equipo.

La situación más típica representada por los participantes fue que un equipo pequeño ubicados físicamente en un mismo ambiente donde se lleva a cabo el proyecto; A pesar de esto, algunos de los participantes dijeron que sus equipos colaboran con otros equipos que estaban en otras ciudades. En cuanto al tamaño de los equipos, todos trabajaron en equipos que no superaron las 10 personas, por lo que trabajaron en unidades de tamaño pequeño.

Para establecer la colaboración con los equipos distribuidos, utilizan herramientas de comunicación: teléfonos, videoconferencias, repositorios en línea, pizarras compartidas, etc. Estas herramientas no son necesarias para trabajar en el equipo de ubicación conjunta, donde la conversación cara a cara es lo más destacado medio de comunicación; junto con pizarras para organizar visualmente el trabajo.

La mayoría de estos equipos nacieron como ágiles y, en opinión de los participantes, esto es mucho mejor, porque pasar de una forma tradicional de trabajar hacia un enfoque ágil es muy complicado. Uno de los entrevistados, cuyo equipo hizo esta transición, señaló que "fue una verdadera pesadilla", porque las personas no se adaptaban bien al cambio de rutinas, y las dudas surgían continuamente. En su opinión, es mucho mejor constituir un equipo ágil desde cero, con protocolos de trabajo predefinidos, en lugar de cambiarlos poco a poco.

Como se señaló en la sección anterior, muchos de los participantes trabajan en equipos cuyo cliente es la empresa; el papel del cliente para ellos es bastante interactivo, en la medida en que se encuentran en el mismo lugar, y bastante completo, en el sentido de que la empresa no exige una gran cantidad de documentación.

En el caso del participante que trabajó para clientes externos, expresó que su empresa tenía reglas estrictas para los clientes, algunas de estas reglas son: no hay documentación en papel, el cliente colabora al final de cada iteración proporciona retroalimentación al equipo, y la influencia de los clientes debe ser limitada.

4.3.4 Contribuciones

Toda la información anterior era interesante para establecer el contexto en el que se aplicaba el desarrollo de software ágil, ahora se exponen los elementos de información que están directamente relacionados con las preguntas de investigación.

Pregunta de investigación 1

¿De qué manera interpretan y ponen en práctica los profesionales ágiles la ingeniería de requisitos en proyectos ágiles, y cómo impactan esas interpretaciones en los resultados y desafíos enfrentados durante el ciclo de vida del proyecto?

En cuanto al concepto de ingeniería de requisitos de software, las respuestas proporcionadas son muy variadas. Cada participante entiende la ingeniería de requisitos de la manera más conveniente para sus intereses.

E3, explicó que en su empresa el concepto de ingeniería de requisitos era "La obtención e identificación general de necesidades de todo el sistema", y también que la importancia de este enfoque es extremadamente alta para su equipo.

El participante E9, estar de acuerdo con E3, en que la ingeniería de requisitos de software es extremadamente importante, el participante E1, autodefinido como un defensor radical extremo de las metodologías ágiles, se mostró "ofendido" por la sugerencia de que el enfoque ágil no propone ingeniería de requisitos, lo definió como "una idea falsa, ágil propone ingeniería de requisitos, pero no de una manera tradicional". Su concepto de ingeniería de requisitos de software era que "No debería haber documentación escrita extensa" y concluyo en "elaborar las historias de usuarios correctamente".

Esta opinión coincide con la que E12 declaró: "Nunca pensamos en la ingeniería de requisitos como algo completamente importante, no hablamos de ingeniería de requisitos; Para nosotros, la ingeniería de requisitos es una cosa evolutiva, no es estático. No está documentado en ningún lado, no hay diagramas UML ni nada de eso".

Pregunta de investigación 2

¿Cómo abordan y gestionan los profesionales ágiles los desafíos en la ingeniería de requisitos de software dentro de proyectos ágiles, y qué estrategias se demuestran efectivas para superar estos obstáculos en la práctica organizacional?

Con respecto a la aplicabilidad del desarrollo ágil de software, la mayoría de los participantes coincidieron en que existen limitaciones importantes que hacen que la

agilidad no se pueda aplicar a todos los dominios, pero algunos especificaron que esto es aplicable a cualquier paradigma de desarrollo de software.

E2 también señaló algunas de las limitaciones de su forma de trabajar, en este caso las limitaciones se referían a la flexibilidad del sistema al momento de modificar algún elemento del sistema, porque en su equipo "No tenemos un sistema flexible estructura que permite la escalabilidad del sistema. Se vive pensando en el cambio inmediato, no planificamos ningún cambio que no sabemos si va a suceder". Este entrevistado también admitió: "Esta forma de trabajar depende completamente de la experiencia del equipo que creó el sistema".

E14 simplemente dijo: "Un proyecto ágil es como cualquier otro proyecto, si no tiene ningún soporte de ingeniería de requisitos, tendrá todos los problemas que cualquier otro tipo de proyecto tendría".

E3 señaló: "Es muy serio no realizar ningún trabajo ingeniería de requisitos en el proyecto ágil". Dijo que los problemas relacionados con la ingeniería de requisitos no aparecen de inmediato, en los primeros sprints; Los problemas aparecen cuando los requisitos cambian, incluso cuando se descubren nuevas características, que el cliente ni siquiera había pensado en ellos. En general, los problemas principales están "relacionados con problemas de escalabilidad".

Como una reflexión interesante, E8 afirmó que el problema clave con la adaptación del trabajo de ingeniería de requisitos en las metodologías ágiles es "¿cómo se puede combinar una estructura rígida como ingeniería de requisitos tradicional con los requisitos cambiantes a lo largo del tiempo? Ese es la clave de todas las preguntas".

Pregunta de investigación 3

¿Cómo seleccionan, adaptan y valoran los equipos ágiles las prácticas para superar los principales desafíos de la ingeniería de requisitos, y cuáles son los factores contextuales que determinan la efectividad de dichas prácticas en distintos entornos organizacionales?

E1 enfatizó fuertemente que la práctica clave para desarrollar la ingeniería de requisitos era desarrollarla de forma iterativa e incremental, describió un proceso en el que lo importante era alcanzar lo antes posible una primera versión para presentar al cliente, y con los comentarios seguir mejorando a medida que aprenden al respecto. Se adopta una iteración cero para permitir el desarrollo del primer prototipo; En esta iteración se crea

un modelo muy pequeño del sistema, entendiendo el modelo no como una documentación estática, sino algo que es posible mostrar al cliente. E1 también declaró que, “más adelante en el desarrollo, todo el equipo participa en la toma de decisiones de ingeniería de requisitos, por lo que se convierte en una práctica conjunta; pero debe tenerse en cuenta que, si alguien tiene una gran experiencia sobre el tipo de sistema que se está construyendo, la opinión de esa persona es la regla ”.

E2 y E11 explicó que en su equipo Scrum, han introducido una fase previa al plan que es anterior a la reunión conjunta de todo el equipo; en este plan previo, algunos miembros responsables del equipo discuten la mejor manera de implementar una historia de usuario, también planean cómo se lleva a cabo la reunión de todos los equipos, y en esa reunión se plantea la solución para su discusión. Entonces, esta etapa de pre - plan funciona como una fase de diseño preliminar para la historia del usuario que se implementa. Una vez que se aprueba la historia del usuario, se desarrolla.

E6 explicó que en su compañía habían adoptado algunas variaciones sobre el ciclo de vida convencional de Scrum; el principal fue la inclusión de un Sprint cero. Este sprint adicional implica "crear un modelo rápido" del sistema a desarrollar.

E8 expresó que "la visión ágil sugiere hacer cosas a pedido: hacer ingeniería de requisitos cuando se necesita ingeniería de requisitos, hacer documentación cuando se necesita documentación, mejorar la calidad cuando la calidad del sistema disminuye, etc." lo que constituye un enfoque de reacción flexible, o en otras palabras: ingeniería de requisitos "a pedido". Algunas de las técnicas utilizadas para desarrollar ingeniería de requisitos dentro del proyecto ágil fueron: documentación ligera, discusión cara a cara y reuniones conjuntas.

Por otro lado, el entrevistado E15 propuso un punto medio entre la identificación de todas las necesidades inicialmente y el que no está precisado, esto consiste en "permitir que la necesidad emerja a medida que el producto evoluciona, ese es el enfoque verdaderamente ágil".

Pregunta de investigación 4

¿Cómo impactan las prácticas implementadas para superar los desafíos de la ingeniería de requisitos en proyectos ágiles en el éxito, la calidad y la satisfacción de los clientes dentro de dichos proyectos?

Refiriéndose a la etapa de pre - plan adoptada por su equipo, E2 señaló que "tener esta ligera fase de análisis del conjunto inicial de necesidades, mejora la reunión del equipo en el sentido de que las personas están un poco influenciadas en la solución potencial, no comienzan desde cero, por lo que la presión es menos y la gente está más relajada y más productiva".

E13 señaló que la introducción de Sprint cero y el modelado al comienzo del proyecto quitan algo de agilidad, pero vale la pena durante el resto del proyecto, porque "obtenemos muchos más beneficios".

5 DISCUSIÓN DE RESULTADOS

Como consecuencia de los resultados obtenidos de ambas fuentes de información, Revisión Sistemática de la Literatura y Entrevistas Semiestructuradas, es posible obtener hallazgos interesantes respecto de los temas que componen el objetivo de este trabajo. Esta sección presenta los hallazgos más importantes, relevantes y sustentados empíricamente de este trabajo de tesis: el concepto general de Ingeniería de requisitos en Proyectos Ágiles no es bien precisado después de revisar la literatura, se identifican, desafíos más comunes relacionados con la Ingeniería de Requisitos y el Desarrollo de Software Ágil, las mejores prácticas recomendadas para superarlos, junto con los posibles efectos causados en el proyecto por la implementación de dichas prácticas.

Como se mencionó en la sección de trabajos relacionados, era muy necesario para los propósitos de este trabajo de tesis (descubrir cómo se pueden combinar Ingeniería de Requisitos de Software y Agilidad). Los resultados obtenidos de la literatura en ese sentido muestran que, cuando la mayoría de los "Profesionales Ágiles" hablan de Ingeniería de Requisitos de Software, se refieren a la necesidad del sistema y que estos progresivamente evolucionan. Aunque no se define así explícitamente, se abstrae

fácilmente de las descripciones de Ingeniería de Requisitos de Software que deben de identificarse determinados requerimientos de alto nivel. Este concepto ha sido corroborado por los testimonios de los “Profesionales Ágiles” entrevistados, algunos de ellos de forma muy elemental mediante por ejemplo el artefacto historias de usuarios.

Existen varias situaciones que han sido consideradas como Desafíos Ágiles relacionados con la Ingeniería de Requisitos en el contexto del presente trabajo, pero que no requieren de ninguna medida especial de práctica para ser resueltas. Este tipo de desafío tiene que ver con las restricciones de idoneidad ágil. Tiene que ver con las características que debe cumplir un determinado proyecto para recomendar el uso de Metodologías Ágiles. Hay varias condiciones (ver Sección 4.2.2.1) que identifican la idoneidad, un “candidato óptimo”, algunas de ellas tienen que ver con el equipo, otras con el cliente, y las más importantes tienen que ver con el proyecto en sí. El desarrollo ágil no se recomienda para proyectos complejos o críticos, y principalmente no se recomienda para proyectos de gran envergadura.

Los resultados de las entrevistas semiestructuradas realizada, respaldan la existencia de estas limitaciones en la aplicabilidad de la filosofía de Desarrollo de Software Ágil, por lo que la primera práctica recomendada a realizar es evaluar las características del proyecto y su contexto para valorar si lo ágil es adecuado o no. Esta cuestión no significa que las metodologías ágiles sean buenas o malos; Los enfoques o metodologías existentes para el desarrollo de software no pueden juzgarse en términos absolutos, sino que es necesario evaluar las circunstancias específicas para elegir la solución más óptima posible.

Hay muchos otros problemas en los proyectos ágiles que están relacionados con la Ingeniería de Requisitos, algunos de ellos se solucionan con la propia inclusión de la Ingeniería de Requisitos en el proyecto, y otros requieren medidas específicas para ser solucionados. Si bien se han encontrado muchos otros desafíos (se pueden consultar en la Pregunta de Investigación 2 – Sección ¿Qué desafíos en materia de Ingeniería de Requisitos de Software se encuentran en proyectos ágiles?), hay varios de ellos que aparecen con mayor frecuencia en la Literatura y que, además, están respaldados empíricamente.

5.1 Desafíos identificados en Entornos Ágiles

Los principales desafíos relacionados con la ausencia de una Ingeniería de Requisitos de Software ágil están relacionados con problemas a mediano plazo. Estos desafíos se pueden especificar como: problemas estructurales, problemas de escalabilidad, problemas de mantenimiento y falta de reutilización. Este tipo de desafíos han sido señalados como los más importantes en las Entrevistas Semiestructuradas realizada.

Documentación mínima. Los métodos ágiles reemplazan la especificación tradicional de requisitos de software con metas de usuario descritas en historias de usuario (Carlson & Matuzic, 2010). Sin embargo, la falta de comunicación durante el proyecto con el cliente, la complejidad del proyecto y la falta de documentación podrían incrementar el número de problemas (Inayat et al., 2015). La falta de documentación se considera la mayor debilidad de la ingeniería de software ágil (Curcio et al., 2018). Además, en equipos distribuidos es necesario tener algún tipo de descripción de requisitos que permitan a los desarrolladores hacer las implementaciones correctas (Daneva, 2013).

Respecto de la documentación de requisitos, existen algunos problemas para usarlos durante el proceso de desarrollo. En consecuencia, es importante encontrar la combinación apropiada de artefactos según el contexto del proyecto y los miembros del equipo de desarrollo de software (Schön et al., 2017).

Disponibilidad del cliente. Aunque los métodos ágiles asumen que el cliente participa en la creación de historias de usuario, el supuesto de la disponibilidad del cliente es poco realista (Ramesh et al., 2010). En la práctica el dueño del producto toma el rol de cliente real (Daneva et al., 2013). Por otra parte, también se ha reportado la dificultad del usuario para describir sus necesidades de procesamiento de información (Mendes et al., 2016).

Imprecisión en las estimaciones de presupuesto y calendarización. Al inicio del proyecto es imposible hacer estimaciones debido a la volatilidad de requisitos y al dinamismo de la planificación y el diseño (Ramesh et al., 2010). Existen riesgos en depender sólo de las historias de usuario para realizar estimación de costos y tiempo (Curcio et al., 2018). Además, los defectos encontrados en los requisitos tienen impacto en la exactitud del tamaño funcional que puede afectar el esfuerzo del proyecto.

Eludir los requisitos no funcionales. La falta de atención a los requisitos no funcionales se considera el reto principal de los métodos ágiles y puede ser la principal razón del

trabajo correctivo (Inayat et al., 2015; Ramesh et al., 2010; Curcio et al., 2018) y de futuros problemas (Mendes et al., 2016).

Falta de poder de decisión del representante del cliente. En ocasiones el representante del cliente no puede tomar decisiones o no tiene conocimiento completo del dominio del problema. Además, en ocasiones el grupo de clientes puede tener dificultades para llegar a acuerdos respecto del proyecto. Como consecuencia, estas características del cliente pueden afectar el rendimiento del equipo de desarrollo de software (Daneva et al., 2013, Ramesh et al., 2010; (Mendes et al., 2016)(Curcio et al., 2018).

Impacto del cambio. El cambio en los requisitos del proyecto puede crear problemas cuando se evalúan las consecuencias de este cambio (Inayat et al., 2015).

Dificultad en la priorización de requisitos. Un enfoque empleado para priorizar los requisitos en cada iteración se basa en considerar el valor de los requisitos para satisfacer las necesidades del cliente. Sin embargo, esta aproximación pone en riesgo al proyecto (Mendes et al., 2016) porque no toma en cuenta otros factores, como los técnicos y la arquitectura del producto de software, así como la planificación de liberaciones de software (Curcio et al., 2018).

Volatilidad de los requisitos. Se refiere a los cambios que ocurren en los requisitos durante el desarrollo del producto de software (Mendes et al., 2016). Los cambios en los requisitos pueden causar un incremento en el costo y algunas veces llevan al fracaso del proyecto (Inayat et al., 2015). Se recomienda tomar medidas legales (e.g., contrato) para manejar apropiadamente la flexibilidad que ofrecen los métodos ágiles. Por ejemplo, los contratos de pago fijo pueden reducir este riesgo específico (Daneva et al., 2013).

Dependencia entre los requisitos. En teoría, las historias de usuario son independientes para que éstas se puedan implementar en cada iteración. Sin embargo, en la práctica es difícil trabajar con este supuesto (Mendes et al., 2016). Por otra parte, la documentación y trazabilidad de los requisitos es vista como un proceso pesado para el equipo ágil porque es difícil crearla y mantenerla (Curcio et al., 2018). Sin embargo, la carencia de gestión de dependencias entre requisitos está relacionada con la falta de satisfacción del cliente y numerosas fallas del software (Curcio et al., 2018).

Problemas para validar los requisitos. La validación de los requisitos puede presentar dificultades debido a las divergencias entre la descripción de la historia de usuario y lo que se ha desarrollado (Mendes et al., 2016). Se ha observado un alineamiento débil entre

los requisitos descritos y su correspondiente validación. Esta situación podría llevar a problemas en liberar el producto a tiempo y con la calidad esperada (Curcio et al., 2018). Además, la consistencia y verificabilidad de las historias de usuario son difíciles de validar (Curcio et al., 2018).

Problemas de comunicación en equipos grandes o distribuidos. Dado que los métodos ágiles promueven la comunicación cara a cara y documentar lo mínimo acerca de los requisitos del software, los problemas de comunicación y coordinación surgen en este tipo de equipos (Curcio et al., 2018).

Dificultad para encontrar desarrolladores con competencias específicas. La ingeniería de requisitos ágil requiere de desarrolladores de software altamente especializados. La alta rotación del personal, característica en estos ambientes de desarrollo, hace que la organización pierda el conocimiento de los requisitos (Curcio et al., 2018).

5.2 Prácticas en el Proceso de Ingeniería de Requisitos Ágiles

No existen medidas específicas que aborden individualmente cada uno de los desafíos; todos ellos son temas que se potenciarán a través de la actividad de Ingeniería de Requisitos Ágil. Entonces, la pregunta relevante es: ¿cómo se pueden realizar las Tareas Ingeniería de Requisitos en un Proyecto Ágil?

Para responder a esta pregunta, la Literatura ha proporcionado un amplio conjunto de prácticas asociadas a los artefactos que fueron adecuándose en cada enfoque. Entre estas prácticas hay algunas de las más recomendadas, y que encontraron respaldo empírico, estas son:

Iteración cero. Esto debe realizarse antes de comenzar a iterar para desarrollar el sistema. Iteración Cero es una técnica muy interesante en la medida en que permite “tomarse un tiempo” para analizar las necesidades proporcionados por el cliente e implementar la técnica anterior (Priorización de necesidades), un ejemplo interesante de adaptación de estas técnicas lo proporcionó uno de los entrevistados, quien señaló una técnica interesante que consiste en incorporar en cada iteración una etapa de planificación previa. Esta etapa consiste en tomar a dos miembros del equipo, permitirles analizar un subconjunto de requerimientos existentes, y estos dos miembros seleccionarían un

apropiado grupo de historias de usuarios, pendientes a implementar a continuación, además, las historias de usuario seleccionadas y el modelo de alto nivel creado se presentan al equipo en una reunión del día a día, donde se discuten y se aceptan o rechazan.

Proceso iterativo e incremental. Consiste en que el proyecto se divide en iteraciones, cada una de las cuales produce un resultado funcional. El proceso se repite varias veces, con cada iteración construyendo sobre la anterior, hasta que se alcanza el resultado final deseado.

.La mayoría de las fuentes consultadas en el presente trabajo de tesis, tanto en la RSL como las entrevistas semiestructuradas, coinciden en indicar que la forma más adecuada de realizar la actividad de Ingeniería de Requisitos en entornos ágiles es haciéndolo de manera iterativa e incremental. Los “profesionales ágiles” consultados indicaron que la Ingeniería de Requisitos emerge progresivamente, a medida que el equipo aprende cómo debe ser el Sistema. Realizar esta tarea requiere dividir los requerimientos disponibles en varias iteraciones (o sprint) y priorizar los requerimientos (sean Historias de Usuario, Requisitos o del Backlogs) parece aconsejable, de esta manera se implementan en el menor tiempo las funcionalidades más urgentes o relevantes demandadas por el cliente. y así, pueden ser probados.

Diseño preliminar de alto nivel. El diseño preliminar de alto nivel, es una etapa del proceso de desarrollo de software en la que se realiza una planificación general del sistema que se va a construir, teniendo en cuenta los requisitos y las funcionalidades principales. Se utiliza para permitir una mayor flexibilidad y adaptación a los cambios que puedan surgir durante el desarrollo.

Priorización de requisitos. Los requisitos se priorizan en función de su valor para el cliente y su impacto en el proyecto. Se utilizan técnicas como el backlog del producto y el análisis de riesgos para determinar qué requisitos deben abordarse primero.

Equipo altamente comprometido. Otro de los grandes desafíos encontrados en la Literatura y en los entrevistados es la Confianza en la Experiencia del Equipo. Este desafío tiene que ver con la dependencia que existe del equipo que construyó el sistema para alcanzar una comprensión profunda de cómo se construye el sistema. La aplicación de las prácticas recomendadas descritas anteriormente para realizar la Ingeniería de

Requisitos debería solucionar el problema. Es necesario considerar contar con “profesionales ágiles” y tengan por habilidades como:

Comunicación efectiva. Es importante tener habilidades de comunicación para poder transmitir las ideas de manera clara y efectiva dentro del equipo.

Trabajo en equipo. La metodología ágil se basa en el trabajo en equipo, por lo que es fundamental saber trabajar en equipo y colaborar con los demás miembros del equipo.

Flexibilidad. La metodología ágil implica cambios constantes y adaptación a nuevas situaciones, por lo que es importante tener habilidades de flexibilidad para poder adaptarse a los cambios.

Orientación al cliente. La metodología ágil se centra en el cliente y sus necesidades, por lo que es importante tener habilidades de orientación al cliente para poder entender sus necesidades y ofrecer soluciones adecuadas.

Resolución de problemas. La metodología ágil implica enfrentar y resolver problemas constantemente, por lo que es importante tener habilidades de resolución de problemas para poder encontrar soluciones efectivas.

5.3 Artefactos de la Práctica en el Proceso de Ingeniería de Requisitos Ágiles

Los artefactos surgen como consecuencia de una práctica o actividad y se utilizan para comunicar, elaborar, validar y especificar los requisitos en entornos de desarrollo de software (Schön et al., 2017). Algunos de estos artefactos se describen a continuación:

Historias de usuario. Las historias de usuario se desarrollan como especificaciones de los requisitos del cliente (Inayat et al., 2015). Estas historias facilitan la comunicación y la comprensión general de los requisitos entre los involucrados en el proceso de desarrollo (Daneva et al., 2013). A diferencia de los métodos tradicionales de especificación de requisitos de software, las historias de usuario trasladan el énfasis desde la documentación exhaustiva y detallada hacia la interacción directa y frecuente con el cliente (Carlson & Matuzic, 2010). Además, las historias de usuario destacan las metas del usuario final, expresando de manera concisa la perspectiva del usuario y centrándose en lo que realmente se necesita realizar (Carlson & Matuzic, 2010).

Métodos de validación. Una historia de usuario representa una funcionalidad valiosa para el usuario o cliente de un software. No se limita únicamente a la descripción de una necesidad o deseo del cliente, sino que también engloba la conversación entre los

participantes para refinar los detalles, así como la definición de las pruebas y la documentación necesarias para determinar el cumplimiento de la historia (Dimitrijevic et al., 2015). Este enfoque se denomina modelo 3C: Card (tarjeta), Conversation (conversación) y Confirmation (confirmación), el cual estructura las historias de usuario en tres componentes esenciales para su gestión en entornos ágiles (Agile Alliance, 2019).

El modelo INVEST puede emplearse para evaluar la calidad de una historia de usuario. Dicho modelo establece que cada historia debe ser: Independiente, Negociable, Valiosa para los usuarios o clientes, Estimable, Pequeña y Comprobable (Agile Alliance, 2019b).

Otras técnicas complementarias incluyen la presentación y evaluación de prototipos, la realización de reuniones con los clientes y la aplicación de pruebas de aceptación, con el objetivo de asegurar la correcta interpretación y validación de los requisitos definidos en la historia de usuario.

Prototipos. El prototipo es un modelo de la aplicación del software que soporta la comunicación y la evaluación de alternativas (Schön et al., 2017). La creación de prototipo se percibe como una manera directa de revisar la especificación de requisitos con los clientes para obtener una validación oportuna de éstos (Inayat et al., 2015). La construcción del prototipo inicia con los requisitos que han sido seleccionados para abordarse en la siguiente iteración (De Lucia y Qusef, 2010). En particular, aquellos que se entienden completamente y que tienen alta prioridad. Sin embargo, este enfoque puede ser muy costoso cuando se requiere el desarrollo de múltiples prototipos de alta fidelidad (Inayat et al, 2015).

Casos de uso. Un caso de uso describe un conjunto de secuencia de acciones que ejecuta un sistema para producir un resultado observable, de valor para un actor (Booch, et al., 2006). Un actor representa u rol que es desempeñado por una persona, sistema o dispositivo (Booch et al., 2006 p. 247).

Escenario. Un escenario es una secuencia específica de acciones que ilustra un comportamiento particular (Booch et al. 2006). Describe textualmente la interacción entre el usuario y el sistema en un contexto particular (Schön et al., 2017).

Tarjeta de historia. La tarjeta de historia es la representación tangible, en texto, de una historia de usuario (Schön et al., 2017). Las tarjetas de historia también pueden capturar estimados, quienes trabajaron en la historia y metas. Otras propuestas de su contenido

consideran ocho aspectos: quién, qué, por qué, sin ignorar, mientras sea agradable tenerlo, dentro de, con una prioridad de, y cuyo impacto (Schön et al., 2017).

Persona. Persona es una descripción de una persona ficticia que representa a un conjunto de potenciales usuarios del sistema (Schön et al., 2017).

Visión. La visión es una representación abstracta de la meta general que guía el desarrollo de producto (Schön et al., 2017).

Product backlog. La gestión de requisitos se realiza a través de gestionar el backlog del proyecto (Inayat et al. 2015). El backlog contiene las tarjetas que describen cada historia de usuario.

Tablero Kanban. Para monitorear el flujo de trabajo durante el proyecto de software, el tablero Kanban puede ser utilizado para visualizar el progreso de un requisito.

Estas prácticas son adaptables a la mayoría de los proyectos ágiles; como se ha indicado al inicio de este apartado, resulta fundamental considerar las particularidades de cada proyecto y su contexto (equipo, experiencia, cliente, etc.) para implementarlas con éxito. Las técnicas descritas proporcionan los beneficios esenciales de la ingeniería de requisitos: escalabilidad, flexibilidad y reutilización.

Tal como se ha mencionado anteriormente, existen desafíos inherentes a la ausencia de una gestión adecuada de los requisitos. Por lo tanto, permitir prácticas que atiendan estos desafíos contribuye a su resolución. Sin embargo, hay otros retos que requieren medidas especiales: uno de los más frecuentes en la literatura es la falta de calidad en el software producido. Esto ocurre porque la atención del equipo suele centrarse en entregar software funcional a tiempo y dentro del presupuesto, en ocasiones reduciendo el énfasis en atributos de calidad que satisfagan por completo al cliente.

Una práctica recomendada para superar este desafío es considerar la calidad como un criterio fundamental de éxito; por lo tanto, no basta con entregar el software a tiempo o dentro del presupuesto, sino que también es necesario asegurar atributos como confiabilidad, eficiencia, seguridad, entre otros. No obstante, la calidad es un concepto abstracto, cuya interpretación puede variar según el proyecto. La mejor forma de garantizar la alineación con las expectativas del cliente es involucrarlo activamente en las tareas de ingeniería de requisitos, obteniendo retroalimentación continua sobre el sistema.

La participación del cliente y la evaluación constante de los requerimientos simplifican la comunicación, fomentan una mayor implicación de cliente y mejoran la calidad general del sistema. Sin embargo, un riesgo importante señalado por uno de los entrevistados es que una participación excesiva puede conducir a prácticas de micromanagement sobre el equipo de desarrollo. Para evitar esto, es necesario definir estrictamente las tareas del cliente y mantener una gestión activa de la colaboración.

Finalmente, otro desafío significativo identificado tanto en la literatura como por algunos entrevistados es la confianza en la experiencia del equipo. Este reto apunta a la dependencia del conocimiento tácito sobre el sistema, el cual puede dificultar la transferencia de información y la continuidad del proyecto. La aplicación rigurosa de las prácticas recomendadas para la ingeniería de requisitos ayuda a mitigar este problema al formalizar el conocimiento y facilitar su transmisión.

Como consideración final, el conflicto entre la concepción estática de la ingeniería de requisitos y la naturaleza dinámica del desarrollo ágil se resuelve mediante la adaptación y el análisis del contexto del proyecto. El equilibrio entre ambas filosofías proporciona ventajas sustanciales cuando se personalizan las prácticas a las características y expectativas específicas de cada equipo y cliente. En general, los hallazgos más relevantes de la literatura cuentan con respaldo en los testimonios obtenidos en las entrevistas semiestructuradas, aunque no de manera universal.

6 EXHORTACIÓN SOBRE LA INVESTIGACIÓN

En esta sección se presentan las principales advertencias a la validez de la Revisión Sistemática de la Literatura y las Entrevistas Semiestructuradas. Estos son:

6.1 Amenazas de la Revisión Sistemática de la Literatura

6.1.1 Búsqueda y selección de estudios

6.1.1.1 Identificación de estudios primarios relevantes.

La primera amenaza que se consideró en la Revisión Sistemática de la Literatura fue decidir la mejor manera de asegurar la inclusión de todo el material relevante (o al menos la mayor cantidad de contribuciones posible).

Teniendo en cuenta la amplitud de los campos de estudio de este trabajo de tesis, y el hecho de que existe una revisión de la literatura (Inayat et al., 2015) con un enfoque similar; el método elegido para realizar la revisión fue la “técnica de bola de nieve” (Greenhalgh & Peacock, 2005).

El proceso que propone este enfoque se realizó una y otra vez, hasta que sólo se encontraron fuentes repetidas, por lo que esto se consideró como la prueba de que se cubrió el principal cuerpo de conocimiento disponible sobre el tema.

Pese a esto, también se revisaron los resultados obtenidos durante la elaboración de la propuesta de tesis. Esta búsqueda “manual” adicional se realizó en las bases de datos científicas más relevantes y ayuda a garantizar que también se cubra alguna fuente relevante.

6.1.1.2 Sesgo de publicación.

Como señalaron (Kitchenham y Charters, 2007), el sesgo de publicación aborda la cuestión de que es más probable que se publiquen resultados positivos que resultados negativos. Este tema no es relevante para esta revisión sistemática de la literatura. Este estudio no pregunta si esto es bueno o malo, solo pretende determinar cómo integrar las mejores prácticas con la ingeniería de requisitos y la flexibilidad. Como se analizará en

la conclusión, existen formas muy diferentes de afrontar la situación, y los enfoques que son eficaces en un contexto pueden no serlo en otro. A pesar de esto, se desarrolló un protocolo detallado para determinar los criterios de inclusión/exclusión y calidad.

6.1.2 Extracción y síntesis de datos

6.1.2.1 Confiabilidad.

Esta amenaza a la validez está relacionada con la estabilidad de los datos en el tiempo. Por distintas circunstancias, la duración del estudio se ha extendido durante cuatro años, durante este lapso de tiempo es posible que se hayan publicado nuevas contribuciones y que no estén incluidas en la Revisión Sistemática de la Literatura.

6.2 Amenazas sobre las Entrevistas

6.2.1 Validez de constructo

Según Wohlin et al. (2012), la validez de constructo se ocupa de comprobar si las medidas son adecuadas para el concepto que se estudia. En este estudio esto se ha abordado tomando dos decisiones sobre la metodología de investigación elegida para esta etapa de la tesis.

La primera fue decidir realizar entrevistas donde se conozca el perfil del entrevistado y se demuestre su “autoridad” sobre el tema. Y la segunda fue realizar Entrevistas Semiestructuradas, creando una guía de entrevista que impulse la conversación, pero permitiendo la formulación de preguntas adicionales para aclarar la información brindada y asegurar su relación con los conceptos de los estudios.

6.2.2 Escasez de entrevistados

Uno de los mayores problemas en la realización de las entrevistas fue encontrar Voluntarios que participen en el Estudio. Las causas de tal escasez se originan en las fechas de realización de las entrevistas, julio de 2020, período de pandemia (COVID 19) para la mayoría de los trabajadores que pudieron ser incluidos en el estudio.

Este problema cuestiona la confiabilidad de las evidencias empíricas extraídas, sin embargo, debido a las tendencias comunes mostradas en los testimonios de los entrevistados, la información obtenida de las entrevistas Semiestructuradas se ha considerado útil para sustentar y complementar la información extraída de la Literatura.

7 CONCLUSIONES Y TRABAJOS FUTUROS

Esta sección incluye el resumen del presente trabajo de investigación al revisar las preguntas de investigación y establecer una comparación implícita entre los resultados obtenidos de la RSL y las entrevistas semi estructuradas, adicionalmente formulamos posibles futuros trabajos a desarrollar.

7.1 Conclusiones

7.1.1 Preguntas de investigación revisadas

7.1.1.1 PI-1: ¿Qué entienden los profesionales ágiles sobre ingeniería de requisitos en el contexto de un proyecto ágil?

No hay un concepto especial al respecto en la literatura. La Ingeniería de Requisitos, como procesos es un tema subestimado en el ambiente ágil y de definición no precisa; proporciona una cantidad tan amplia de mecanismos para descubrir los requisitos de un proyecto en desarrollo. Pero esta es una reflexión general que se puede hacer para cualquier proyecto, no solo para proyectos ágiles.

Centrándose en la concepción ágil de la Ingeniería de Requisitos, la literatura muestra que la mayoría de los profesionales ágiles entienden la ingeniería de requisitos de software como la actividad de identificación de las necesidades del sistema. La posición de los profesionales en el ambiente ágiles respecto a la actividad de ingeniería de requisitos de software es similar a la que se encuentra en la literatura clásica incluso en sus prácticas, se evita la documentación copiosa, en su lugar se sustituye por crear historias de usuarios conjuntamente con los criterios de validez de los mismos.

7.1.1.2 PI- 2: ¿Qué desafíos con respecto a la ingeniería de requisitos de software se encuentran en proyectos ágiles?

Respecto a esta pregunta se debe precisar, el primer problema que debe abordarse es el problema de la idoneidad del proyecto. ¿Dónde es apropiado usar Desarrollo de software Ágil? Los resultados de la RSL muestran un conjunto de condiciones que delimitan la

aplicabilidad del desarrollo ágil. Estas limitaciones están relacionadas con el tamaño del proyecto y el equipo, la complejidad del proyecto y su criticidad, el dominio del equipo en el problema que enfrentan y la garantía en cuanto a su estabilidad en el proyecto.

Los equipos pequeños, los proyectos no complejos de tamaño medio y los proyectos de tamaño mediano son las condiciones óptimas para elegir el desarrollo de software ágil; Estas características generalmente están respaldadas tanto por los hallazgos de SLR. Sin embargo, es necesario señalar que se han encontrado algunas divergencias entre las dos fuentes de información con respecto a la cuestión de la ubicación de los miembros del equipo. En los resultados de SLR se señala que la implementación ágil en equipos distribuidos puede ser un obstáculo inevitable, pero los entrevistados proporcionaron evidencia de que esto, estrictamente hablando, no es cierto; Este obstáculo se puede evitar utilizando la herramienta de comunicación adecuada, por lo que el desarrollo ágil es factible en entornos distribuidos.

Se puede precisar que es ventajosa la experiencia en el dominio, es deseable en un equipo ágil para tomar decisiones más rápidas y efectivas. Por otro lado, la desventaja de la experiencia tiene que ver con la confianza que la organización tiene en la experiencia de los miembros del equipo que construyen el producto software para alcanzar una comprensión profunda. Este tipo de situación puede llevar a una dependencia de los individuos.

7.1.1.3 PI-3: ¿Qué prácticas se siguen para superar estos desafíos?

La revisión sistemática de literatura ha producido una gran cantidad de prácticas que se han utilizado para resolver el problema de combinar el desarrollo de software ágil y la ingeniería de requisitos.

La primera conclusión importante obtenida es que existen varios enfoques para hacer tal combinación, estos son: adaptar metodologías ágiles para permitir la realización de actividades de Ingeniería de Requisitos; importar principios ágiles en las fases de desarrollo tradicionales, de modo que las tareas ingeniería de requisitos se vuelvan más ágiles; o cree una nueva metodología combinando las dos filosofías desde cero.

Las prácticas abstractas pueden usarse en el contexto de estos tres enfoques, pero la flexibilidad debe ser de importancia a la hora de resolver cada situación particular. No hay soluciones absolutas, solo diferentes adaptaciones que dependen a un contexto particular.

A partir de las prácticas identificadas, es posible resumir que en el desarrollo ágil, la Ingeniería de requisitos debe evolucionar con el tiempo, a medida que se desarrolla el ciclo de vida del proyecto. Para evolucionar el proyecto y considerando la ingeniería de requisitos es muy importante tener en cuenta los comentarios de los clientes, lo que proporciona un conocimiento más profundo del sistema y ayuda a solucionar problemas que el equipo podría no haber detectado.

La comunicación es esencial para permitir la toma de decisiones; Algunas prácticas útiles que se identificó de las fuentes empíricas incluidas en el RSL son las reuniones de revisión, evaluación o los llamados retrospectivas para la toma de decisiones conjuntas y el uso de herramientas de comunicación, especialmente en aquellos equipos ágiles que trabajan distribuidos.

Sin embargo, estas prácticas no pueden clasificarse como prácticas útiles o inútiles, se da una clasificación interesante atendiendo a su naturaleza. Algunas prácticas mejoran la agilidad, a un grupo de ellas manifiestan preocupación por mejorar la comunicación o la calidad del proyecto y otras requieren de un esfuerzo costoso para implementarse.

7.1.1.4 PI-4: ¿Qué efectos provocaron en el proyecto las prácticas que superan estos desafíos?

Una conclusión importante con respecto a esta pregunta de investigación es que no es posible hablar estrictamente sobre los efectos, ya que evaluar un efecto requeriría analizar métricas específicas después y antes de tomar medidas específicas. La información encontrada sobre las consecuencias de los enfoques propuestos en cada caso ha sido principalmente aleatoria y más basada en percepciones percibidas de las prácticas.

Una noción de ingeniería de requisitos, incluso si consiste en un modelo ligero, iterativo e incremental, puede proporcionar una dirección al equipo y hacer que se sientan más confiados y seguros del trabajo que están haciendo.

En la tabla 19 resumimos lo encontrado en la revisión sistemática de la literatura y las prácticas que se implementan en los diferentes enfoques ágiles propuestos.

Tabla 19*Resumen de los desafíos y prácticas en IRA*

Desafíos	Método investigación /Autor(es)	Prácticas
Problemas con el costo y la estimación del cronograma; descuido de los requisitos no funcionales; acceso y participación del cliente; priorización en una sola dimensión; verificación inadecuada de requisitos; documentación mínima	Caso de estudio múltiple (16 compañías) Ramesh, Cao, Baskerville (Ramesh et al., 2010)	Proceso iterativo e incremental [S19] [S21]. Análisis y Diseño preliminar de alto nivel [S13] [S21]. Cliente involucrado en tareas de IR [S18] [S25]. Iteración cero [S13] [S18].
Planificación de la agilidad; débiles prioridades de los requisitos; estimaciones de esfuerzo débil; Problemas de calidad; sistema completado tarde; capturando la innovación; falta de requisitos documentados; rol de cliente-proxy; asegurando la competencia (IRA, VV); equipos motivadores para el trabajo de requisitos; requisitos débiles al inicio	Caso de estudio Bjarnason, Wnuk, Regnell (Bjarnason et al., 2011)	Reuniones de evaluación (Retrospectiva) Roles dedicados [S19]. Priorización de los requisitos [S13]. Realizar análisis de impacto en cada iteración [S13] [S18].
Documentación mínima; disponibilidad del cliente; presupuesto y tiempo de estimación; descuidar los requisitos no funcionales (RNF); Incapacidad y acuerdo del cliente; limitaciones contractuales; los requisitos cambian y su evaluación	Revisión Sistemática de la Literatura Inayat, Salim, Marczak, Daneva, Shamshirband (Inayat et al., 2015)	Reuniones de revisión [S26]. Dividir equipos [S13]. Uso de herramientas de comunicación [S26]. Mejorar la calidad [S18].
Problemas con los representantes del cliente o del cliente; insuficiencia del formato de la historia del usuario; dificultades en la priorización de los requisitos; creciente deuda técnica; confianza en el conocimiento de los requisitos tácitos; estimaciones de esfuerzo imprecisas	Estudio de mapeo Heikkilä, Damian, Lassenius, Paasivaara (Heikkilä et al., 2017)	Juego de planificación. Pequeños lanzamientos Metáfora del sistema Mejorar la calidad [S18]. Los atributos de calidad [S18]. Uso del conocimiento [S33]
Priorización de requisitos; identificación de requisitos no funcionales; falta de información; volatilidad de los requisitos; definición de requisitos; dependencia entre requisitos; predicción de los impactos de los cambios; dependencia del usuario; comunicación y colaboración con los usuarios; validación de requisitos	Un estudio retrospectivo Mendes, Thiago Souto De Farias, Mário André F. Mendonça, Manoel Soares, Henrique Frota Kalinowski, Marcos Spínola, Rodrigo Oliveira (Mendes et al., 2016)	Ingeniería de requisitos iterativos Priorización de requisitos Esfuerzo de integración [S13] [S21]. Cliente involucrado en tareas de IR [S18] [S25]. Priorización de los requisitos [S13].

7.2 Trabajos Futuros

Luego de los resultados de la RSL, parece claro que se requiere mucho más trabajo empírico sobre el tema. Solo 6 de las 37 fuentes incluidas eran estudios empíricos, lo que significa un 16% del total.

Parece aconsejable realizar experimentos controlados para evaluar los efectos causados por las técnicas propuestas y junto con las percepciones del equipo. El objetivo principal debe ser crear prácticas adaptativas que puedan usarse dependiendo del proyecto bajo ciertas condiciones.

Algunas de las métricas interesantes que podrían usarse para la evaluación de las prácticas podrían ser: tiempo (esto podría ser medir diferentes procesos, o sprints, o todo el proceso de desarrollo), esfuerzo requerido para corregir defectos, sentimientos de los miembros del equipo (estrés, presión, felicidad, etc.), desviaciones en el presupuesto, satisfacción del cliente, etc.

Considerando, estas métricas deben tomarse tanto después como antes de introducir las prácticas propuestas para medir su impacto, y en aquellos casos en que esto no sea posible, las medidas deberían tomarse mediante la comparación entre proyectos con características similares.

Desde el punto de vista de la industria del software, crear un cuerpo de conocimiento que evalúe la selección de una metodología específica será dependiendo del equipo, el proyecto y las características del cliente los cuales podría significar un gran ahorro en términos de tiempo y costo.

Bibliografia

- Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical roots of agile methods: Where did “Agile thinking” come from? *Lecture Notes in Business Information Processing, 9 LNBIP*, 94–103. https://doi.org/10.1007/978-3-540-68255-4_10
- Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., & Salo, O. (2004). Mobile-D: An agile approach for mobile application development. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, 174–175. <https://doi.org/10.1145/1028664.1028736>
- APESOF. (n.d.). apesoft.org
- Beck, K. (2004). What is XP? In Addison-Wesley (Ed.), *Extreme Programming Explained: Embrace Change* (Segunda ed).
- Beck, K., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S., van Bennekum, A., Hunt, A., Schwaber, K., Cockburn, A., Jeffries, R., Sutherland, J., Cunningham, W., Kern, J., Thomas, D., Fowler, M., & Marick, B. (2001). Manifesto for Agile Software Development. *The Agile Alliance*, 12–14. <http://agilemanifesto.org/>
- Bjarnason, E., Wnuk, K., & Regnell, B. (2011). A case study on benefits and side-effects of agile practices in large-scale requirements engineering. *Proceedings of the 1st Agile Requirements Engineering Workshop, AREW'11 - In Conjunction with ECOOP'11*. <https://doi.org/10.1145/2068783.2068786>
- Boehm, B., & Turner, R. (2003). Observations on balancing discipline and agility. *Proceedings of the Agile Development Conference, ADC 2003*, 32–39. <https://doi.org/10.1109/ADC.2003.1231450>
- Boness, K., Harrison, R., & Liu, K. (2008). Goal sketching: An Agile Approach to Clarifying Requirements. *International Journal On Advances in Software, 1*. <http://www.iariajournals.org/software/>
- Cao, L., & Ramesh, B. (2008a). Agile requirements engineering practices: An empirical study. *IEEE Software*, 25(1), 60–67. <https://doi.org/10.1109/MS.2008.1>
- Cao, L., & Ramesh, B. (2008b). *Agile Requirements Engineering Practices*. 60–67. www.computer.org/software
- Carlson, D., & Matuzic, P. (2010). Practical agile requirements engineering. *13th Annual Systems Engineering Conference*. https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2010/systemengr/WednesdayTrack1_11106Matuzik.pdf
- Carolyn B Seaman. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4), 557–572.
- Cho, J. (2009). a Hybrid Software Development Method for Large-Scale Projects: Rational Unified Process With Scrum. *Issues In Information Systems*, 340–348. https://doi.org/10.48009/2_iis_2009_340-348
- Cohn, M. (2009). *User Stories Applied: For Agile Software Development* (I. Pearson Education (Ed.)).
- Corral, L., Silliti, A., & Succi, G. (2013). Lecture Notes in Computer Science (including

- subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface. *Agile Software Development Processes for Mobile Systems: Accomplishment, Evidence and Evolution*, 8093 LNCS(July 2016). <https://doi.org/10.1007/978-3-642-40276-0>
- Cruzes, D. S., & Dyb, T. (2011). Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, 53(5), 440–455. <https://doi.org/10.1016/j.infsof.2011.01.004>
- Daneva, M., Van Der Veen, E., Amrit, C., Ghaisas, S., Sikkil, K., Kumar, R., Ajmeri, N., Ramteerthkar, U., & Wieringa, R. (2013). Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software*, 86(5), 1333–1353. <https://doi.org/10.1016/j.jss.2012.12.046>
- De Lucia, A., & Qusef, A. (2010). Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, 2(3), 212–220. <https://doi.org/10.4304/jetwi.2.3.212-220>
- Díaz-bravo, L., Torruco-garcía, U., Martínez-hernández, M., & Varela-, M. (2013). *La entrevista , recurso flexible y dinámico*. 2(7), 162–167.
- Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management*, 16(4), 88. <https://doi.org/10.4018/jdm.2005100105>
- Ernst, N. A., Borgida, A., Jureta, I. J., & Mylopoulos, J. (2014). Agile requirements engineering via paraconsistent reasoning. *Information Systems*, 43, 100–116. <https://doi.org/10.1016/j.is.2013.05.008>
- Farid, W. M., & Mitropoulos, F. J. (2012). NORMATIC: A visual tool for modeling non-functional requirements in agile processes. *Conference Proceedings - IEEE SOUTHEASTCON*, 978. <https://doi.org/10.1109/SECon.2012.6196989>
- Flick, U. (2014). *El diseño de Investigación Cualitativa*.
- Goetz, R. (2002). How Agile Processes Can Help in Time-Constrained Requirements Engineering. *International Workshop on Time Constrained Requirements Engineering*. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.280&rep=rep1&type=pdf>
- Greenhalgh, T., & Peacock, R. (2005). Effectiveness and efficiency of search methods in systematic reviews of complex evidence: Audit of primary sources. *British Medical Journal*, 331(7524), 1064–1065. <https://doi.org/10.1136/bmj.38636.593461.68>
- Guha, P., Shah, K., Prasad Shukla, S. S., & Shweta Singh. (2011). Incorporating Agile With mda Case Study: Online Polling System. *International Journal of Software Engineering & Applications*, 2(4), 83–96. <https://doi.org/10.5121/ijsea.2011.2408>
- Haugset, B., & Stålhane, T. (2012). Automated acceptance testing as an agile requirements engineering practice. *Proceedings of the Annual Hawaii International Conference on System Sciences, May 2014*, 5289–5298. <https://doi.org/10.1109/HICSS.2012.127>
- Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017).

- Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. *Empirical Software Engineering*, 22(6), 2892–2936. <https://doi.org/10.1007/s10664-016-9491-z>
- Hester, P. T., Operandi, M., & Adams, K. M. (2013). *Determining Stakeholder Influence Using Input-Output Modeling*. April 2015. <https://doi.org/10.1016/j.procs.2013.09.282>
- Hruschka, P. (2018). *Handbook of RE @ Agile According to the IREB Standard*. October, 1–95.
- Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 51, 915–929. <https://doi.org/10.1016/j.chb.2014.10.046>
- Jeon, S., Han, M., Lee, E., & Lee, K. (2015). *Quality Attribute driven Agile Development*. August 2011. <https://doi.org/10.1109/SERA.2011.24>
- Jun, L., Qiuzhen, W., & Lin, G. (2010). Application of agile requirement engineering in modest-sized information systems development. *Proceedings - 2010 2nd WRI World Congress on Software Engineering, WCSE 2010*, 2(1), 207–210. <https://doi.org/10.1109/WCSE.2010.105>
- Kitchenham, B. A., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering (Software Engineering Group, Department of Computer Science, Keele *Technical Report EBSE 2007- 001. Keele University and Durham University Joint Report*, January.
- Klaus Pohl, C. R. R. N. (2015). *Requirements Engineering Fundamentals*.
- KVALE, S. (1994). Journal of Phenomenological Psychology. *Institute of Psychology, Aarhus University, Denmark*, 2, 147–173.
- Lauesen, S. (2016). *Styles and techniques*. January 2002.
- Lehman, T. J., & Sharma, A. (2011). Software development as a service: Agile experiences. *Proceedings - 2011 Annual SRII Global Conference, SRII 2011*, 749–758. <https://doi.org/10.1109/SRII.2011.82>
- Mahmud, I., & Veneziano, V. (2011). Mind-mapping: An effective technique to facilitate requirements engineering in agile software development. *14th International Conference on Computer and Information Technology, ICCIT 2011*, Iccit, 157–162. <https://doi.org/10.1109/ICCITechn.2011.6164775>
- Maiti, R. R., Krasnov, A., & Wilborne, D. M. (2018). Agile Software Engineering & the Future of Non-Functional Requirements. *Journal of Software Engineering Practice*, 2(1), 1–8. <http://jsep.info/index.php/jsep/article/view/9/7>
- MARconi, M.A; Lakatos, E. M. (2002). *Fundamentos de Metodologia Científica*.
- Marshall, C., & Rossman, G. (2006). *Designing_qualitative_research.pdf*.
- Martakis, A. (2015). *Handling Requirements Dependencies in Agile Projects : A Focus Group with Agile Software Development Practitioners*. May 2013. <https://doi.org/10.1109/RCIS.2013.6577679>
- Martin, A., Biddle, R., & Noble, J. (2009). XP customer practices: A grounded theory. *Proceedings - 2009 Agile Conference, AGILE 2009*, June 2014, 33–40.

<https://doi.org/10.1109/AGILE.2009.68>

- Mendes, T. S., De Farias, M. A. F., Mendonça, M., Soares, H. F., Kalinowski, M., & Spínola, R. O. (2016). Impacts of agile requirements documentation debt on software projects: A retrospective study. *Proceedings of the ACM Symposium on Applied Computing, 04-08-April-2016*(April), 1290–1295. <https://doi.org/10.1145/2851613.2851761>
- Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 82(11), 1869–1890. <https://doi.org/10.1016/j.jss.2009.05.052>
- Okesola, J., Adebisi, M., Okokpujie, K., Odepitan, D., Goddy-Worlu, R., Iheanetu, O., Omogbadegun, Z., & Adebisi, A. (2019). A systematic review of requirement engineering practices in agile model. *International Journal of Mechanical Engineering and Technology*, 10(2), 671–687.
- Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, 2003-Janua*(July), 308–313. <https://doi.org/10.1109/ENABL.2003.1231428>
- Pichler, M., & Wahler, W. (2006). *Agile Requirements Engineering for a Social Insurance for Occupational Risks Organization : A Case Study Agile Requirements Engineering for a Social Insurance for Occupational Risks Organization : A Case Study. July 2014*. <https://doi.org/10.1109/RE.2006.8>
- Racheva, Z., Daneva, M., & Herrmann, A. (2014). *a case study A Conceptual Model of Client-driven Agile Requirements Prioritization : Results of a Case Study. September 2010*. <https://doi.org/10.1145/1852786.1852837>
- Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5), 449–480. <https://doi.org/10.1111/j.1365-2575.2007.00259.x>
- Rational Rose. (1998). *Rational Unified Process Best Practices.pdf*. www.rational.com
- Rico, D. F. (2010). Lean and agile project management: For large programs and projects. *Lecture Notes in Business Information Processing*, 65 LNBIP, 37–43. https://doi.org/10.1007/978-3-642-16416-3_5
- Royce, W. W. (1970). MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS. *IEEE WESCOM*, 1–9.
- Sampieri, R. H., Collado, C. F., & Lucio, P. B. (2006). *Metodología de la Investigación*.
- Sarkan, H. M., Ahmad, T. P. S., & Bakar, A. A. (2011). Using JIRA and redmine in requirement development for Agile methodology. *2011 5th Malaysian Conference in Software Engineering, MySEC 2011*, 408–413. <https://doi.org/10.1109/MySEC.2011.6140707>
- Schön, E. M., Thomaschewski, J., & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards and Interfaces*, 49, 79–91. <https://doi.org/10.1016/j.csi.2016.08.011>
- Schwaber, K. (1997). *SCRUM Development Process. February 1986*, 1994–1995.
- Sillitti, A., Ceschi, M., Russo, B., Succi, G., Sillitti, A., Ceschi, M., Russo, B., & Succi,

- G. (n.d.). *Managing Uncertainty in Requirements : a Survey in Documentation-driven and Agile Companies*.
- Sohaib, O., & Khan, K. (2010). Integrating usability engineering and agile software development: A literature review. *2010 International Conference on Computer Design and Applications, ICCDA 2010*, 2(Iccda), 32–38.
<https://doi.org/10.1109/ICCDA.2010.5540916>
- Sommerville, I. (2016). Software engineering (10th edition). In *Pearson Education Limited*.
- Suryn, W. (2014). Software Quality Engineering: A Practitioner's Approach. In *Software Quality Engineering: A Practitioner's Approach* (Vol. 9781118592).
<https://doi.org/10.1002/9781118830208>
- Sutherland, J. S. K. &. (2013). *The Definitive Guide to Scrum: The Rules of the Game*.
<http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf>
- Tobergte, D. R., & Curtis, S. (2013). Ingeniería de Software un enfoque práctico. In *Journal of Chemical Information and Modeling* (Vol. 53, Issue 9).
- Williams, L., & Tomayko, J. (2002). Agile software development. *International Journal of Phytoremediation*, 21(1), 167–168.
<https://doi.org/10.1076/csed.12.3.167.8613>
- Yu, Y., & Sharp, H. (2011). *Analysing Requirements in a Case Study of Pairing*. July.
<https://doi.org/10.1145/2068783.2068787>
- Zaki, K. M., & Moawad, R. (2010). A hybrid disciplined agile software process model. *INFOS2010 - 2010 7th International Conference on Informatics and Systems*, April 2010.

APÉNDICES

Apéndice 1 Matriz de consistencia

Problemas	Objetivos	Variables	Metodología
Problema principal: ¿Cuáles son los desafíos de la ingeniería de requisitos en las metodologías ágiles para el logro de proyectos exitosos en el desarrollo del producto software por parte de las organizaciones de TI?	Objetivo general: ¿Determinar los Desafíos que enfrenta la ingeniería de requisitos en las metodologías ágiles para el logro de proyectos exitosos en el desarrollo del producto software en las organizaciones de TI?	Variable independiente: Desafíos de la ingeniería de requisitos en entornos ágiles	Enfoque de la investigación: Cualitativo Nivel de Investigación: Básico Tipo de investigación: Correlacional causal
Problemas específicos: 1: ¿Qué entienden los profesionales ágiles por ingeniería de requisitos en el contexto de un proyecto ágil? 2: ¿Qué desafíos con respecto a ingeniería de requisitos de software se encuentran en proyectos ágiles? 3: ¿Qué prácticas / enfoques se siguen para superar estos desafíos? 4: ¿Qué efectos provocan estas prácticas / enfoques en el proyecto?	Objetivo específico: OE 1: Precisar el entendimiento de los profesionales en entornos ágiles por ingeniería de requisitos en el contexto de un proyecto ágil. OE 2: Establecer qué desafíos con respecto a ingeniería de requisitos de software se encuentran en proyectos ágiles. OE 3: Identificar qué prácticas / enfoques (prácticas, métodos, técnicas) se siguen para superar estos desafíos. OE 4: Identificar qué efectos provocan estas prácticas / enfoques en el proyecto.	Variable dependiente: Logro de proyectos exitosos en el desarrollo del producto software	Diseño de investigación: No-experimental Transversal Método: Hipotético deductivo Analítico Técnicas: encuestas Instrumentos: -test de conocimiento Población: muestra: 15

Apéndice 2 Matriz de operacionalización

Variables	Definición conceptual	Definición operacional	Indicadores	Ítems
Variable dependiente: X Desafíos de la ingeniería de requisitos en entornos ágiles	Los principales desafíos de la ingeniería de requisitos de software en entornos ágiles incluyen la definición de requisitos claros y precisos, la gestión de cambios en los requisitos, la comunicación efectiva entre los miembros del equipo y los usuarios, y la adaptación a los cambios constantes del entorno.	Se trata de la identificación y descripción detallada de los problemas que surgen al aplicar la ingeniería de requisitos en un contexto ágil, y cómo se pueden abordar estos desafíos para lograr una entrega exitosa del proyecto.	Frecuencia de interacción usuario/desarrollador Uso herramientas de comunicación, chats. Nivel de participación de los usuarios. Nro. Solicitud de aclaración de req. Por los desarrolladores Nro de cambios de req, en un periodo de tiempo Capacidad de manejo de cambios sin interrupciones	Las reuniones son: diario, semanal, mensual Reuniones: presencial, virtual, chat. Participación es: permanente, eventual, no participa Frecuencia de aclaraciones: frecuentes, poco frecuentes, ninguna Frecuencia de cambios de requisitos: frecuentes, poco frecuentes, ninguna
Variable independiente: Y Logro de proyectos exitosos en el desarrollo del producto software	Se debe considerar: Planificación sólida, equipo competente, comunicación efectiva, gestión de riesgos, iteración y mejora continua, pruebas exhaustivas, cumplimiento de requisitos, participación de los usuarios. Al integrar estos elementos y abordarlos de manera efectiva, se aumenta la probabilidad de lograr el éxito en el desarrollo de productos de software	Cantidad de proyectos de desarrollo de software que se completan dentro del plazo previsto, dentro del presupuesto asignado y que cumplen con los requisitos funcionales y de calidad establecidos. Además, se entrega un producto software funcional, libre de defectos significativos y que satisface las necesidades y expectativas de los usuarios finales	Nro. Proyectos fracaso Re empezar proyecto % proyectos entregados a tiempo % proyectos entregados en el presupuesto inicial Encuesta para saber el nivel de satisfacción del equipo Nro. De veces que los requisitos han sido modificados % de usuarios finales que han empezado a usar el producto	

Apéndice 3 Fuentes incluidas en la RSL

Código	Título	Autor	Año	Tipo de publicación
S01	Communication Patterns of Agile Requirements Engineering.	Abdullah, N. N. B., Honiden, S., Sharp, H., Nuseibeh, B., & Notkin, D	2011	ACM Proceedings of the 1st Workshop on Agile Requirements Engineering
S02	The Inevitable Pain of Software Development, Including of Extreme Programming, Caused by Requirements Volatility University of Waterloo	Berry, D. M.	2002	Proceedings of the International Workshop on Time Constrained Requirements Engineering
S03	A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering.	Bjarnason, E., Wnuk, K., & Regnell, B.	2011	Agile RE 2011. Lancaster, UK: ACM
S04	Observations on balancing discipline and agility	B. Boehm, and R. Turner	2003	<i>Proceedings of the Agile Development Conference, 2003. ADC</i>
S05	Agile Requirements Engineering Practices: An Empirical Study	Cao, L. C. L., & Ramesh, B.	2008	IEEE Software, 25(1),
S06	Practical Agile Requirements Engineering	Carlson, D., & Matuzic, P.	2010	13th Annual Systems Engineering Conference. San Diego, CA.
S07	Agile requirements prioritization in large-scale outsourced system projects: An empirical study	Daneva, M., Van der Veen, E., Amrit, C., Ghaisas, S., Sikkil, K., Kumar, R., Ajmeri	2013	Journal of Systems and Software
S08	Agile requirements engineering via paraconsistent reasoning.	Ernst, N. a., Borgida, A., Jureta, I. J., & Mylopoulos	2014	<u>Information Systems, Volume 43,</u>
S09	NORMATIC: A visual tool for modeling Non-Functional Requirements in agile processes	Farid, W. M., & Mitropoulos, F. J.	2012	proceedings of IEEE Southeastcon,
S10	Software Development as a Service: Agile Experiences	Lehman, Tobin J. Sharma, Akhilesh	2011	<i>Annual SRII Global Conference</i> , pp. 749–758
S11	Automated Acceptance Testing as an Agile Requirements Engineering Practice	Haugset, B., & Stalhane, T.	2012	45th Hawaii International Conference on System Sciences
S12	Application of Agile Requirement Engineering in Modest-Sized Information Systems Development.	Jun et al., 2010	2010	Second World Congress on Software Engineering
S13	A hybrid disciplined Agile software process model	K.M. Zaki, and R. Moawad,	2010	<i>Informatics and Systems INFOS 2010 The 7th International Conference</i>
S14	Mind-mapping: An effective technique to facilitate requirements engineering in agile software development.	Mahmud, I., & Veneziano, V	2011	Proceedings of 14th International Conference on Computer and Information Technology (ICCIT)
S15	Handling Requirements Dependencies in Agile Projects: A Focus Group with Agile Software Development Practitioners.	Martakis, A., & Daneva, M	2013	IEEE Seventh International Conference on Research Challenges in Information Science (RCIS)
S16	Agile Requirements Engineering for a Social Insurance for Occupational Risks Organization: A Case Study.	Pichler, M., Rumetshofer, H., & Wahler, W.	2006	14th IEEE International Requirements Engineering Conference (RE'06)
S17	A Conceptual Model of Client-driven Agile Requirements Prioritization: Results of a Case Study.	Racheva, Z., Daneva, M., & Herrmann, A.	2010	IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)
S18	Agile requirements engineering practices and challenges: an empirical study	Ramesh, B., Baskerville, R., & Cao, L.	2010	Information Systems Journal 2010
S19	Using JIRA and Redmine in requirement development for agile methodology.	Sarkan, H. M., Ahmad, T. P. S., & Bakar, A. A.	2011	Malaysian Conference in Software Engineering
S20	Managing Uncertainty in Requirements: a Survey in	Sillitti, A., Ceschi, M., Russo, B., & Succi, G.	2005	11th IEEE International Software Metrics Symposium (METRICS)

	Documentation driven and Agile Companies			
S21	Analysing requirements in a case study of pairing.	Yu, Y., & Sharp, H.	2011	Workshop on Agile Requirements Engineering. Lancaster, UK.
S22	Extreme Programming Explained: Embrace Change	K. Beck	1999	Addison-Wesley, Reading
S23	Scrum development process	K. Schwaber	1997	<i>Business Object Design and Implementation</i>
S24	User Stories Applied – For Agile Software Development	Cohn M.	2004	Boston: Pearson Education, Inc., 268 p. ISBN 0-321-20568-5
S25	“Mobile-D: An Agile Approach for Mobile Application Development,”	P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo	2004	<i>Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (OOPSLA '04)</i> . ACM, New York, NY, USA, 2004.
S26	Agile Reuirements prioritization in large-scala outsourced system projects: an empirical study	Daneva Maya.	2013	Journal of system and software 2013
S27	Incorporating Agile with MDA Case Study: Online Polling System	P. Guha, K. Shah, S. S. P. Shukla, and S. Singh,	2011	<i>International Journal of Software Engineering & Applications (IJSEA)</i> , vol. 2., no.4, pp. 83-95
S28	Quality Attribute driven Agile Development	S. Jeon, M. Han, E. Lee, and K. Lee,	2011	<i>Ninth International Conference on Software Engineering Research, Management and Applications</i>
S29	Agile requirements engineering for a social insurance for occupational risks organization: A case study.	Pichler, M., Rumetshofer, H., & Wahler, W.	2006	14th IEEE int. Requirement Engineeering. Conference.
S30	How agile processes can help in time-constrained requirements engineering.	Goetz, R.	2002	In Proceedings of the international workshop on time constrained requirements engineering.
S31	Practical agile requirements engineering	Carlson, D., & Matuzic, P.	2010	. In 13 th Annual systems engineering conference
S32	A hybrid Software Development method for large-scale projects: Rational Unified Process with Scrum	. J. Cho,	2009	<i>Issues in Information Systems (Journal IACIS)</i> , vol. 10, no. 2,
S33	Requirements engineering and Agile Development	Hruschka, Peter	2018	International Requirements Engineering Board
S34	Requirements Engineering Fundamentals.	Klaus Pohl, Chris Rupp; Rocky Nook	2011	1. Edition;Paperback; ISBN-13: 978- 1933952819
S35	Goal Sketching: Towards Agile Requirements Engineering.	Boness, K., & Harrison, R	2007	International Conference on Software Engineering Advances 2007
S36	Agile Software Engineering & the Future of Non-Functional Requirements	Maiti, Richard R Krasnov, Aleksandr Wilborne, Deanna Marie	2018	Journal of Software Engineering Practice
S37	Integrating usability engineering and agile software development: A literature review	Sohaib, Osama Khan, Khalid	2010	<i>CHI '08 Extended Abstracts on Human Factors in Computing Systems (CHI EA '08)</i> . ACM, New York, NY, USA

Apéndice 4 Guía de entrevista

Caracterización del entrevistado

1. ¿Cuántos años llevas trabajando en ingeniería de software?
2. ¿Cuántos años llevas trabajando en proyectos ágiles?
3. ¿Con qué metodologías ágiles ha trabajado?
 - a. ¿En el pasado?
 - b. ¿Hoy en día?
4. ¿Cuál es su rol en el equipo?

Caracterización del proyecto

5. Características del proyecto
 - a. ¿Qué tan grande es el proyecto?
 - b. ¿Qué tan crítico es el proyecto?
 - c. ¿Qué tan complejo es el proyecto?
6. ¿Tiene la empresa / equipo experiencia en proyectos similares en la misma área?
7. ¿Cuál es la metodología / enfoque ágil utilizado?
8. ¿Cómo fue la adopción de ágil?
9. Equipo
 - a. ¿Qué experiencia tiene el equipo en ...
 - i. ¿Metodologías ágiles?
 - ii. ¿Ingeniería de requisitos de software?
 - b. ¿Cuál es la organización del equipo? (Roles y responsabilidades...)
 - c. ¿Cómo se habilita la comunicación entre el equipo?
10. Cliente
 - a. ¿Cuáles son las demandas del cliente con respecto a la documentación de ingeniería de requisitos de software?
 - b. ¿Cuál es el rol del cliente en el equipo ágil?
 - c. ¿Cómo se habilita la comunicación con el cliente?

Preguntas de investigación

11. Pregunta de investigación 1
 - a. ¿Qué importancia tiene la ingeniería de requisitos de software en el proyecto ágil?
 - i. Utilidad

ii. Percepción del equipo

- b. Para el equipo, en el contexto ágil; ¿Qué es ingeniería de requisitos de software?

12. Pregunta de investigación 2

- a. ¿Cuáles son los principales problemas de ingeniería de requisitos de software en su equipo?
- b. ¿Qué problemas derivaron del hacer explícitamente o dejar de hacer ingeniería de requisitos de software en el proyecto?
- c. ¿Por qué considera que necesita dar énfasis en la disciplina de la ingeniería de requisitos de software en el proyecto?

13. Pregunta de investigación 3

- a. ¿Cómo se aplica la metodología ágil?
- b. ¿Cómo se hace más ágil la disciplina de ingeniería de requisitos?

14. Pregunta de investigación 4

- a. ¿Cuáles son los beneficios o inconvenientes del enfoque que está siguiendo?

Aporte de la academia

15. La formación universitaria respecto a las metodologías ágiles fue eficiente?

16. Respecto a la disciplina de ingeniería de requisitos ágiles fue explícita?

Apéndice 5 Juicio de expertos

FICHA DE INFORME DE OPINIÓN DE EXPERTOS

TÍTULO DE LA INVESTIGACIÓN: **Desafíos de la ingeniería de requisitos en el desarrollo de software ágil: Un enfoque en la mejora del producto**

I. DATOS GENERALES:

- 1.1 Apellidos y Nombres del Experto : López Córdova Frida Mereyda
 1.2 Grado académico : Doctora en Ingeniería de Sistemas e Informática
 1.3 Cargo e institución donde labora : Universidad Nacional Mayor de San Marcos
 1.4 Nombre del instrumento : Encuesta
 1.5 Autor(es) del Instrumento : Oscar Benito Pacheco

I. ASPECTOS DE VALIDACIÓN:

INDICADORES	CRITERIOS	Excelente 81-100%	Muy Buena 61-80%	Buena 41-60%	Mejorable 21-40%	Deficiente 00-20%
1.CLARIDAD	Lenguaje claro, preciso, sintáctica y semántica son adecuadas.		X			
2.OBJETIVIDAD	Está expresado en conductas observables.			X		
3.ACTUALIDAD	Adecuado al avance de la ciencia y la tecnología.		X			
4.ORGANIZACION	Existe una organización lógica.			X		
5.SUFICIENCIA	Comprende los ítems en cantidad y calidad para obtener la medición		X			
6.INTENCIONALIDAD	Adecuado para valorar aspectos de las estrategias de investigación.			X		
7.CONSISTENCIA	Basado en aspectos teórico-científicos de la investigación.	X				
8.COHERENCIA	Alineamiento de variables, dimensiones e indicadores			X		
9.METODOLOGIA	La estrategia responde al propósito del diagnóstico		X			
10. PERTINENCIA	El instrumento es adecuado para el propósito de la investigación.	X				

II. OPINION CUALITATIVA

Las preguntas permiten al encuestado compartir comentarios detallados y experiencias personales.

III. OPINIÓN DE APLICABILIDAD

(...) El instrumento puede ser aplicado, tal como está elaborado.

(x...) El instrumento debe ser mejorado antes de ser aplicado, y nuevamente validado.

▲ IV. PROMEDIO DE VALORACIÓN:

66

Lima 5 de noviembre del 2024

FICHA DE INFORME DE OPINIÓN DE EXPERTOS

TÍTULO DE LA INVESTIGACIÓN: **Desafíos de la ingeniería de requisitos en el desarrollo de software ágil: Un enfoque en la mejora del producto**

I. DATOS GENERALES:

- 1.6 Apellidos y Nombres del Experto : Hidalgo Pereda Lyses Augusto
 1.7 Grado académico : Magister
 1.8 Cargo e institución donde labora : Universidad Nacional Mayor de San Marcos
 1.9 Nombre del instrumento : Encuesta
 1.10 Autor(es) del Instrumento : Oscar Benito Pacheco

II. ASPECTOS DE VALIDACIÓN:

INDICADORES	CRITERIOS	Excelente 81-100%	Muy Buena 61-80%	Buen a 41-60%	Mejorable 21-40%	Deficiente 00-20%
1. CLARIDAD	Lenguaje claro, preciso, sintáctica y semántica son adecuadas.	x				
2. OBJETIVIDAD	Está expresado en conductas observables.	x				
3. ACTUALIDAD	Adecuado al avance de la ciencia y la tecnología.	x				
4. ORGANIZACIÓN	Existe una organización lógica.	x				
5. SUFICIENCIA	Comprende los ítems en cantidad y calidad para obtener la medición		x			
6. INTENCIONALIDAD	Adecuado para valorar aspectos de las estrategias de investigación.	x				
7. CONSISTENCIA	Basado en aspectos teórico-científicos de la investigación.	x				
8. COHERENCIA	Alineamiento de variables, dimensiones e indicadores	x				
9. METODOLOGÍA	La estrategia responde al propósito del diagnóstico	x				
10. PERTINENCIA	El instrumento es adecuado para el propósito de la investigación.	x				

III. OPINIÓN CUALITATIVA

En términos generales, el instrumento se presenta como **altamente confiable y válido**. Su diseño refleja un trabajo meticuloso y un profundo entendimiento de los requisitos metodológicos para una investigación de calidad.

Se recomienda su uso para la recolección de datos.

IV. OPINIÓN DE APLICABILIDAD

- (x...) El instrumento puede ser aplicado, tal como está elaborado.
(...) El instrumento debe ser mejorado antes de ser aplicado, y nuevamente validado.

V. PROMEDIO DE VALORACIÓN:

92%

Lima 14-11-2024.



Firma del Experto

DNI.Nº...07295349 Teléfono Nº 979 095 145

FICHA DE INFORME DE OPINIÓN DE EXPERTOS

TÍTULO DE LA INVESTIGACIÓN: **Desafíos de la ingeniería de requisitos en el desarrollo de**
 **software ágil: Un enfoque en la mejora del producto**

I. DATOS GENERALES:

- 1.1 Apellidos y Nombres del Experto : Lázaro Guillermo, Juan Carlos
- 1.2 Grado académico : Magister
- 1.3 Cargo e institución donde labora : Docente Investigador RENACYT UNIA Nivel IV
- 1.4 Nombre del instrumento : Encuesta
- 1.5 Autor(es) del Instrumento : Oscar Benito Pacheco

II. ASPECTOS DE VALIDACIÓN:

INDICADORES	CRITERIOS	Excelente 81-100%	Muy Buena 61-80%	Buena 41-60%	Mejorable 21-40%	Deficiente 00-20%
1. CLARIDAD	Lenguaje claro, preciso, sintáctica y semántica son adecuadas.		X			
2. OBJETIVIDAD	Está expresado en conductas observables.		X			
3. ACTUALIDAD	Adecuado al avance de la ciencia y la tecnología.		X			
4. ORGANIZACIÓN	Existe una organización lógica.		X			
5. SUFICIENCIA	Comprende los ítems en cantidad y calidad para obtener la medición		X			
6. INTENCIONALIDAD	Adecuado para valorar aspectos de las estrategias de investigación.		X			
7. CONSISTENCIA	Basado en aspectos teórico-científicos de la investigación.	X				
8. COHERENCIA	Alineamiento de variables, dimensiones e indicadores		X			
9. METODOLOGÍA	La estrategia responde al propósito del diagnóstico	X				
10. PERTINENCIA	El instrumento es adecuado para el propósito de la investigación.		X			

III. OPINIÓN CUALITATIVA

El instrumento de validación presenta una estructura coherente y metodológicamente sólida para evaluar los desafíos de la ingeniería de requisitos en el contexto del desarrollo ágil. Los ítems muestran pertinencia y suficiencia, pues abordan dimensiones relevantes como la gestión del cambio, la colaboración entre los actores del proyecto y la mejora continua del producto. Además, el contenido refleja actualidad científica, integrando conceptos alineados con las tendencias contemporáneas en ingeniería de software y metodologías ágiles. Se evidencia consistencia teórica y coherencia entre las variables, dimensiones e indicadores planteados, lo que refuerza la validez del instrumento para su aplicación en entornos académicos y profesionales. En conjunto, se considera que el instrumento cumple adecuadamente con los criterios de claridad, objetividad y pertinencia para los fines de la investigación.

IV. OPINIÓN DE APLICABILIDAD

☒ El instrumento puede ser aplicado, tal como está elaborado.

☐ El instrumento debe ser mejorado antes de ser aplicado, y nuevamente validado.

V. PROMEDIO DE VALORACIÓN:

84 %

Pucallpa, 13 de noviembre del 2024.



.....
Firma del Experto
.....

DNI.Nº 1012590.Teléfono Nº.913832105.

FICHA DE INFORME DE OPINIÓN DE EXPERTOS

TÍTULO DE LA INVESTIGACIÓN: **Desafíos de la ingeniería de requisitos en el desarrollo de software ágil: Un enfoque en la mejora del producto**

I. DATOS GENERALES:

- 1.1 Apellidos y Nombres del Experto : Candela Arce David Félix
- 1.2 Grado académico : Magister
- 1.3 Cargo e institución donde labora : UNMSM
- 1.4 Nombre del instrumento : Encuesta
- 1.5 Autor(es) del Instrumento : Oscar Benito Pacheco

II. ASPECTOS DE VALIDACIÓN:

INDICADORES	CRITERIOS	Excelente 81-100%	Muy Buena 61-80%	Buena 41-60%	Mejorable 21-40%	Deficiente 00-20%
1.CLARIDAD	Lenguaje claro, preciso, sintáctica y semántica son adecuadas.		X			
2.OBJETIVIDAD	Está expresado en conductas observables.	X				
3.ACTUALIDAD	Adecuado al avance de la ciencia y la tecnología.	X				
4.ORGANIZACION	Existe una organización lógica.	X				
5.SUFICIENCIA	Comprende los ítems en cantidad y calidad para obtener la medición	X				
6.INTENCIONALIDAD	Adecuado para valorar aspectos de las estrategias de investigación.	X				
7.CONSISTENCIA	Basado en aspectos teórico-científicos de la investigación.	X				
8.COHERENCIA	Alineamiento de variables, dimensiones e indicadores	X				
9.METODOLOGIA	La estrategia responde al propósito del diagnóstico	X				
10. PERTINENCIA	El instrumento es adecuado para el propósito de la investigación.	X				

III. OPINION CUALITATIVA

La encuesta debe medir donde están los mayores problemas, que practicas funcionan mayor o peor, que tan maduro es el equipo en gestión de requisitos ágiles, que mejoras son necesarias.

IV. OPINIÓN DE APLICABILIDAD

☒ (X) El instrumento puede ser aplicado, tal como está elaborado.

☐ El instrumento debe ser mejorado antes de ser aplicado, y nuevamente validado.

V. PROMEDIO DE VALORACIÓN:

90 %

Lima 5 de noviembre del 2025



.....
Firma del Experto

DNI N.º 07578149 - Teléfono N.º 997090168