Análisis sistemático de los métodos de evaluación de software y su aplicabilidad en entornos actuales de desarrollo de software

Autor(es)

Alex Fabian Suarez Caviedes

Rubén Darío Torres Gómez

Asesor

Harol Emilio Cabrera Meza

Universidad Nacional Abierta y a Distancia – UNAD

Escuela Ciencias de la Educación ECEDU

Ingeniería de Sistemas

2026

Análisis sistemático de los métodos de evaluación de software y su aplicabilidad en entornos actuales de desarrollo de software

Autor(es)

Alex Fabian Suarez Caviedes

Rubén Darío Torrez Gómez

Nota de Aceptación

"Monografía presentada como requisito parcial para optar al título de Ingeniería de Sistemas, bajo la orientación del profesor Harol Emilio Cabrera Meza."

Universidad Nacional Abierta y a Distancia – UNAD

Escuela Ciencias de la Educación ECEDU

Ingeniería de Sistemas

2026

Resumen

La presente monografía examina de manera sistemática los métodos contemporáneos de evaluación de software y su pertinencia en escenarios actuales de desarrollo, partiendo de la preocupación creciente por garantizar calidad en entornos tecnológicos dinámicos; a través de una revisión documental de estudios publicados entre 2020 y 2025, el trabajo organiza y clasifica enfoques vinculados con DevOps, metodologías ágiles, métricas automatizadas y evaluación de requerimientos, lo que permite reconocer distintos niveles de validación empírica y contextos de aplicación; además, se identifican fortalezas como la integración de métricas en procesos de integración continua y la detección temprana de defectos, aunque también se evidencian limitaciones relacionadas con la escasa estandarización y la débil medición de atributos no funcionales; en este sentido, el análisis ofrece una mirada crítica que articula fundamentos teóricos y hallazgos recientes, proponiendo líneas de mejora orientadas a fortalecer la adaptabilidad y la solidez empírica de los modelos, así como su integración coherente en prácticas de desarrollo cada vez más automatizadas y colaborativas.

Palabras clave: Ingeniería del software, control de calidad, desarrollo de programas, evaluación, seguridad informática.

Abstract

The monograph systematically examines contemporary software evaluation methods and their relevance in current development environments, emerging from the growing concern for ensuring quality in dynamic technological contexts; through a documentary review of studies published between 2020 and 2025, the work organizes and classifies approaches related to DevOps, agile methodologies, automated metrics, and requirements evaluation, which makes it possible to identify different levels of empirical validation and diverse application settings; it also highlights strengths such as the integration of metrics into continuous integration processes and the early detection of defects, while acknowledging limitations associated with limited standardization and the weak measurement of non-functional attributes; in this way, the analysis offers a critical perspective that connects theoretical foundations with recent findings, suggesting lines of improvement aimed at strengthening the empirical robustness and contextual adaptability of evaluation models, as well as their coherent integration into increasingly automated and collaborative development practices.

Keywords: Software engineering, quality control, program development, evaluation, computer security.

Tabla de Contenido

Lista de Tablas

Introducción

En el escenario contemporáneo, marcado por un desarrollo tecnológico vertiginoso y por transformaciones constantes en los entornos digitales, la calidad del software adquiere un lugar central en la garantía de confiabilidad, seguridad y desempeño de los sistemas; a medida que las organizaciones integran soluciones informáticas en procesos estratégicos, se vuelve imprescindible adoptar criterios de evaluación más rigurosos y coherentes con dichas exigencias, ya que los errores pueden tener impactos operativos y reputacionales significativos; además, las expectativas de los usuarios y los marcos regulatorios incrementan la presión por asegurar productos sólidos y verificables; en este marco, la presente investigación se orienta a examinar y clasificar diversos métodos de evaluación de la calidad del software, considerando su sustento empírico, sus límites y su pertinencia en distintos contextos de aplicación.

La evolución del desarrollo de software ha estado estrechamente vinculada con transformaciones en las formas de evaluar la calidad, ya que los modelos tradicionales, como el ciclo de vida en cascada, fueron concebidos bajo una lógica secuencial que situaba el control en momentos específicos del proceso; con el paso del tiempo y ante contextos más cambiantes, surgieron enfoques ágiles e iterativos como DevOps y DevSecOps, los cuales integran la evaluación de manera continua y articulada con el trabajo diario del equipo; este desplazamiento metodológico ha impulsado la creación de nuevas métricas, herramientas y marcos conceptuales que demandan análisis rigurosos y contrastación empírica para valorar su pertinencia y consistencia en escenarios reales.

En coherencia con este panorama, la monografía se propone examinar críticamente diversos métodos contemporáneos de evaluación de la calidad del software, apoyándose en la revisión y análisis de fuentes empíricas aplicadas en distintos entornos de desarrollo; a partir de

una lectura sistemática se identificaron patrones recurrentes, enfoques predominantes y tensiones persistentes que permiten comprender mejor la configuración actual del campo; además, el estudio dialoga con teorías previas y marcos conceptuales consolidados, lo que amplía la interpretación de los hallazgos y favorece una mirada reflexiva que articula antecedentes, prácticas recientes y desafíos aún abiertos en la evaluación del software.

Uno de los criterios centrales del análisis fue la relevancia práctica de los métodos estudiados, entendida como la capacidad de las propuestas para sostener coherencia conceptual y, al mismo tiempo, responder de manera efectiva a condiciones operativas diversas; esta perspectiva permitió valorar con mayor cuidado el alcance real de cada enfoque, distinguiendo aquellos trabajos que permanecen en el plano teórico de los que evidencian aplicaciones concretas en contextos específicos; a partir de esta diferenciación se configuró una mirada crítica sobre el grado de madurez de los métodos revisados, lo que facilitó comprender hasta qué punto sus planteamientos pueden trasladarse a escenarios reales sin perder consistencia ni viabilidad.

De forma complementaria, se realizó un examen detallado de las limitaciones señaladas por los propios autores, lo que abrió un espacio de reflexión sobre vacíos persistentes en la literatura reciente; entre los aspectos más recurrentes se encuentran la limitada validación en entornos ágiles, la atención insuficiente a atributos no funcionales y las dificultades para integrar ciertos modelos en ciclos de desarrollo acelerados; estas observaciones permiten advertir tensiones metodológicas que aún no han sido resueltas y, a la vez, orientan posibles líneas de investigación orientadas a fortalecer la solidez empírica y la adaptabilidad de los enfoques de evaluación de la calidad del software.

Con el propósito de facilitar la comprensión y comparación de los hallazgos, se diseñaron matrices que organizan la información de acuerdo con enfoques metodológicos, contextos de

aplicación, niveles de validación empírica y limitaciones identificadas; esta sistematización permitió ordenar la diversidad de propuestas revisadas y ofrecer una herramienta clara para quienes buscan seleccionar o ajustar métodos de evaluación en proyectos concretos; al mismo tiempo, dichas tablas brindan una visión amplia del campo, ya que hacen visibles tendencias recurrentes y carencias persistentes, lo que contribuye a orientar la reflexión académica y la toma de decisiones profesionales con mayor fundamento y criterio contextual.

Por su parte, la discusión se configura como un espacio de articulación entre los resultados obtenidos y el marco teórico que sustenta la investigación, pues en ella se contrastan los métodos emergentes con modelos clásicos y antecedentes conceptuales relevantes; este ejercicio comparativo permite valorar con mayor equilibrio si las propuestas recientes representan avances significativos o si, por el contrario, prolongan líneas ya establecidas bajo nuevas denominaciones; además, el análisis pone en evidencia cómo han evolucionado los criterios de calidad, desplazándose desde enfoques centrados en etapas específicas hacia perspectivas más integradas y automatizadas, acordes con dinámicas contemporáneas de desarrollo.

Las conclusiones recogen los hallazgos más significativos del estudio y ofrecen una lectura equilibrada de los avances y debilidades identificados en los métodos analizados, lo que permite dimensionar con mayor claridad su alcance real en contextos contemporáneos; a partir de este balance se plantean orientaciones para investigaciones futuras, entre ellas el fortalecimiento de la validación empírica en escenarios diversos, el diseño de métricas automatizables que contemplen atributos no funcionales y la adaptación flexible de los modelos a entornos con dinámicas particulares; estas recomendaciones buscan encauzar el trabajo

académico y profesional hacia prácticas más rigurosas y coherentes con las exigencias actuales del desarrollo de software.

La justificación del trabajo se sostiene en la creciente demanda de confiabilidad en productos digitales que hoy respaldan servicios críticos en ámbitos como salud, educación, industria y seguridad, lo que implica asumir la calidad como un compromiso permanente y no como una verificación ocasional; en este escenario, evaluar el estado del arte con sentido crítico se convierte en una tarea necesaria para reconocer fortalezas, advertir limitaciones y promover propuestas que articulen fundamentos teóricos con prácticas tecnológicas pertinentes; de este modo, la investigación busca contribuir a la consolidación de enfoques más sostenibles y eficaces, acordes con la responsabilidad que implica desarrollar software en sociedades cada vez más dependientes de sistemas digitales.

Desde esta perspectiva, la monografía aporta tanto en el plano académico como en el profesional, ya que ofrece un análisis comparativo actualizado que dialoga con debates recientes y, al mismo tiempo, proporciona herramientas conceptuales útiles para la toma de decisiones en procesos de aseguramiento de calidad; el estudio no se limita a describir métodos, sino que propone una reflexión fundamentada sobre su aplicabilidad y sus condiciones de implementación, lo que favorece una comprensión más situada del campo; en un entorno caracterizado por cambios constantes, sostener una mirada crítica respaldada por evidencia empírica resulta clave para orientar el desarrollo hacia modelos más consistentes y centrados en las necesidades reales de quienes interactúan con el software.

Contextualización del Tema

El desarrollo de software se ha consolidado como un soporte fundamental para la operación de diversos sectores, dado que organizaciones de distinta naturaleza dependen de aplicaciones capaces de ofrecer estabilidad y adaptarse a escenarios cambiantes; a medida que estas herramientas se integran en procesos cotidianos y estratégicos, la preocupación por su calidad adquiere mayor relevancia, pues un error puede afectar tanto la experiencia del usuario como la continuidad de actividades esenciales; en consecuencia, surge un interés creciente por comprender cómo se evalúa la calidad del software y qué criterios permiten valorar su desempeño con coherencia y responsabilidad en contextos reales.

En este marco, la evaluación asume un carácter estratégico, ya que permite anticipar dificultades y verificar que el producto responda a las expectativas del entorno en el que será implementado; sin embargo, la variedad de enfoques disponibles plantea desafíos al momento de elegir el método más adecuado, puesto que cada propuesta prioriza dimensiones distintas de la calidad; además, las decisiones suelen estar condicionadas por factores como los recursos disponibles, la cultura organizacional o la metodología de trabajo adoptada; por ello, se vuelve necesario contar con criterios más claros que faciliten comparaciones consistentes y orienten la selección de enfoques pertinentes.

A esta situación se suman los cambios recientes en el desarrollo de software, marcados por la adopción de arquitecturas distribuidas, prácticas de integración continua y herramientas automatizadas que transforman la forma en que se construyen y despliegan los sistemas; tales dinámicas demandan evaluaciones más flexibles y oportunas, ya que los modelos tradicionales no siempre responden con la agilidad requerida por equipos que trabajan en ciclos breves; en este sentido, resulta pertinente revisar la evolución de los métodos de evaluación y valorar su

capacidad para ajustarse a entornos contemporáneos caracterizados por la variabilidad y la innovación constante.

Reconocer el alcance de la evaluación de software implica asumir que no se trata únicamente de comprobar si un sistema funciona de acuerdo con lo esperado, sino de valorar dimensiones como la seguridad, el rendimiento, la usabilidad y la mantenibilidad, atributos que influyen de manera directa en su aceptación y permanencia en el tiempo; en este sentido, cada decisión tomada a lo largo del desarrollo repercute en la calidad global del producto, lo que evidencia su carácter transversal y la necesidad de enfoques integrales que articulen distintas etapas y perspectivas; por ello, revisar cómo se han definido y aplicado los métodos de evaluación permite advertir fortalezas, identificar vacíos y comprender mejor los factores que condicionan su adopción en la práctica profesional.

Este panorama confirma que el análisis de los métodos de evaluación responde a una demanda concreta tanto de la industria como del ámbito académico, dado que la selección adecuada de un enfoque incide de manera directa en el éxito de los proyectos y en la consolidación de procesos más consistentes; además, abordar el tema desde una perspectiva sistemática facilita ordenar la diversidad de propuestas existentes y ofrecer una comprensión más estructurada del campo; a partir de esta mirada es posible reflexionar sobre los cambios que han experimentado las prácticas evaluativas y valorar por qué resulta necesario mantener una revisión crítica y actualizada frente a contextos tecnológicos en constante transformación.

Justificación

La evaluación de la calidad en el desarrollo de software ocupa actualmente un lugar determinante, dado que los sistemas que respaldan servicios en salud, finanzas o educación deben operar con altos niveles de confiabilidad y continuidad; por ello, revisar la manera en que se seleccionan y aplican los métodos disponibles resulta necesario, ya que cada enfoque parte de supuestos distintos y condiciona la interpretación del desempeño del producto; en este sentido, Camacho Quintero (2023) sostiene que una evaluación rigurosa permite establecer si el software responde realmente a las expectativas del entorno y de los usuarios, lo que refuerza la necesidad de examinar con atención la consistencia conceptual y la aplicabilidad práctica de los modelos utilizados.

De igual manera, el interés por analizar estos métodos se explica por las discrepancias que la literatura ha evidenciado respecto a las métricas y procedimientos más pertinentes en escenarios reales, situación que ha generado dispersión conceptual y dificultades para comparar resultados entre proyectos; según Callejas-Cuervo y Alarcón-Aldana (2017), aunque existen marcos ampliamente reconocidos, su implementación suele adaptarse de manera fragmentada a contextos específicos, lo que puede desdibujar su alcance original; en consecuencia, se vuelve pertinente organizar y revisar críticamente estas propuestas, a fin de comprender su verdadero impacto y ofrecer criterios más claros para su aplicación en distintos entornos de desarrollo.

La diversidad de enfoques se amplía cuando se consideran métodos centrados en atributos internos del software, propuestas orientadas a dimensiones externas y modelos sustentados en estándares internacionales, panorama que aporta riqueza conceptual al campo aunque también genera incertidumbre al momento de elegir la alternativa más adecuada en cada contexto; esta decisión se vuelve más compleja cuando no se conocen con claridad los

fundamentos ni las limitaciones de cada propuesta; en este sentido, Castaño (2021) explica que las métricas internas pueden resultar útiles para examinar estructuras y componentes, aunque su pertinencia depende del propósito evaluativo y del tipo de proyecto, lo que evidencia la necesidad de criterios sólidos que respalden una selección metodológica consciente y fundamentada.

A esta situación se suma el hecho de que numerosos métodos presentan validaciones empíricas limitadas, dado que han sido probados principalmente en contextos académicos o experimentales sin incorporar plenamente las condiciones propias de entornos productivos; esta brecha puede generar dudas sobre su aplicabilidad cuando se enfrentan escenarios con mayores exigencias técnicas o equipos de trabajo diversos; al respecto, Mera Paz (2016) señala que las pruebas en laboratorio permiten identificar comportamientos relevantes, aunque no siempre reproducen las dinámicas reales del desarrollo, circunstancia que refuerza la importancia de revisar críticamente la evidencia disponible para estimar con mayor precisión el grado de madurez de los modelos de evaluación existentes.

Otra razón que respalda este estudio se vincula con las transformaciones recientes del ecosistema tecnológico, puesto que prácticas como la integración continua, el despliegue automatizado y el uso de servicios distribuidos demandan métodos de evaluación más flexibles y sensibles a cambios constantes; en consecuencia, las arquitecturas actuales requieren enfoques capaces de adaptarse a ciclos de desarrollo más breves y a variaciones rápidas del entorno operativo; según Rodríguez Sinisterra et al. (2021), los modelos tradicionales no siempre ofrecen respuestas adecuadas frente a estas dinámicas, lo que evidencia una brecha que merece ser examinada con detenimiento para reconocer alternativas metodológicas acordes con las nuevas exigencias tecnológicas.

En este contexto, el análisis sistemático se presenta como una estrategia pertinente para organizar información dispersa y ofrecer una visión estructurada del estado actual de los métodos de evaluación; mediante este tipo de aproximación es posible identificar coincidencias, tensiones e inconsistencias en la literatura, así como comprender los fundamentos que sostienen cada propuesta y valorar su posible utilidad en escenarios reales; Díaz y Peralta Luján (2022) señalan que los modelos de evaluación, cuando se contextualizan adecuadamente, aportan criterios valiosos para mejorar la calidad del software, lo que refuerza la necesidad de integrar resultados y tendencias de manera clara y metodológicamente transparente.

La relevancia de este trabajo también se refleja en su posible incidencia en la toma de decisiones dentro de organizaciones que requieren procesos de evaluación confiables y coherentes con sus objetivos estratégicos, dado que la ausencia de métodos adecuados puede repercutir en la gestión del riesgo, la asignación de recursos y la sostenibilidad de los proyectos; desde esta perspectiva, Serna (2011) señala que evaluar el software implica considerar dimensiones que influyen directamente en su funcionamiento y en las expectativas de los usuarios, lo que pone de relieve la importancia de contar con criterios sólidos para valorar la calidad y sostener procesos eficientes que minimicen fallos operativos.

De igual manera, la investigación aporta al ámbito académico al identificar vacíos y limitaciones que abren nuevas líneas de trabajo orientadas a fortalecer o replantear los métodos existentes, aspecto que resulta clave en un campo sujeto a transformaciones constantes; Esterkin y Pons (2017) explican que los procesos de evaluación deben ajustarse a las dinámicas del desarrollo contemporáneo, lo que invita a examinar cómo los marcos actuales responden a prácticas emergentes y qué modificaciones requieren para mantener su pertinencia; esta reflexión

contribuye a consolidar bases teóricas más consistentes y a estimular propuestas que acompañen la evolución de la ingeniería de software.

En conjunto, estas razones evidencian la pertinencia de desarrollar un análisis sistemático que permita comprender con mayor claridad la diversidad de métodos disponibles, la consistencia de sus fundamentos y los desafíos asociados a su aplicación en contextos reales; organizar de manera rigurosa esta información favorece la construcción de un panorama estructurado que facilite a estudiantes, profesionales e investigadores seleccionar enfoques acordes con sus necesidades y sustentar decisiones con argumentos sólidos; además, esta mirada integrada contribuye a orientar mejoras en los procesos de evaluación, ya que articula reflexión académica con exigencias prácticas; de este modo, el estudio responde tanto a inquietudes teóricas como a demandas de la industria, que requiere herramientas confiables y actualizadas para desenvolverse en entornos tecnológicos cada vez más complejos y dinámicos.

Planteamiento del Problema

El crecimiento sostenido de la industria del software ha reafirmado su presencia en sectores que dependen de sistemas estables y seguros, circunstancia que convierte la evaluación de la calidad en un proceso clave para garantizar continuidad operativa y confianza en los resultados, tal como señalan Pressman y Maxim (2020); pese a ello, la forma en que se desarrolla dicha evaluación continúa mostrando una marcada diversidad de prácticas, dado que no siempre se aplican criterios homogéneos ni se cuenta con procedimientos que permitan interpretar los resultados con suficiente claridad; además, cuando el proceso se apoya más en experiencias previas que en marcos formales, como advierte OpenWebinars (2025), se debilita la consistencia y se dificulta la comparación entre productos desarrollados en contextos distintos.

A este panorama se suma la necesidad de comprender con mayor precisión cómo se define la calidad del software y de qué manera se valora el cumplimiento de requisitos que integran expectativas técnicas y funcionales, aspectos que no siempre aparecen explícitos en los documentos de diseño; Andrade de Casañas (2006) resalta la importancia de modelos que permitan interpretar la calidad desde una perspectiva integral, lo que pone en evidencia que la ausencia de criterios bien delimitados puede propiciar análisis fragmentados y apreciaciones subjetivas del desempeño; esta ambigüedad conceptual incide directamente en la selección de herramientas de evaluación y condiciona decisiones relevantes a lo largo del ciclo de desarrollo.

La amplia variedad de métodos disponibles ha favorecido evaluaciones con enfoques diversos, algunos orientados a examinar aspectos estructurales del código y otros centrados en dimensiones externas vinculadas con la experiencia del usuario, lo que configura un campo dinámico y en constante ajuste; en este contexto, Castaño (2021) señala que las métricas internas permiten aproximarse a elementos como la complejidad o la cohesión del software, aunque su

pertinencia depende del propósito analítico y de las características del proyecto; así, esta diversidad aporta riqueza conceptual, aunque también plantea desafíos metodológicos, dado que cada enfoque se sustenta en supuestos distintos y puede producir resultados divergentes cuando se analiza un mismo sistema o se intentan generalizar conclusiones.

A ello se suma la preocupación por la solidez empírica de los métodos, ya que numerosos estudios se han desarrollado en contextos controlados o con muestras reducidas que limitan la posibilidad de valorar su efectividad en escenarios más amplios; según Jústiz-Núñez et al. (2014), las experiencias en laboratorios de calidad ofrecen información relevante, aunque no siempre transferible a entornos de producción donde intervienen variables como el volumen de usuarios o la complejidad arquitectónica; esta situación pone de relieve la necesidad de validaciones más rigurosas y comparativas que permitan estimar con mayor precisión la capacidad de los modelos para anticipar comportamientos críticos o detectar fallos significativos.

La amplia variedad de métodos disponibles ha propiciado evaluaciones con enfoques diversos, algunos orientados al análisis estructural del código y otros enfocados en dimensiones externas vinculadas con la experiencia del usuario, lo que configura un panorama heterogéneo y en permanente ajuste; en este contexto, Castaño (2021) señala que las métricas internas permiten aproximarse a la complejidad o cohesión del software, aunque su aplicabilidad depende del propósito analítico y de las particularidades del proyecto; así, esta diversidad evidencia la riqueza del campo, aunque también introduce desafíos metodológicos, puesto que cada enfoque se sustenta en supuestos distintos y puede arrojar resultados divergentes cuando se examina un mismo producto o se intenta extender conclusiones a sistemas de mayor alcance.

Por otra parte, la solidez empírica de los métodos constituye una preocupación significativa, dado que numerosos estudios se han desarrollado en contextos controlados o con

muestras reducidas que limitan la posibilidad de valorar su efectividad en escenarios más amplios; Jústiz-Núñez et al. (2014) describen experiencias en laboratorios de calidad donde las pruebas ofrecen información relevante, aunque no siempre transferible a entornos de producción en los que influyen variables como el volumen de usuarios o la complejidad arquitectónica; esta situación pone de manifiesto la necesidad de validaciones más rigurosas y comparativas que permitan estimar con mayor precisión la capacidad de los modelos para identificar fallos relevantes o anticipar comportamientos inesperados.

A la complejidad metodológica se añaden las transformaciones tecnológicas que han modificado de manera significativa la forma en que se desarrolla software, dado que prácticas como la integración continua, los despliegues automatizados y el uso de servicios en la nube demandan modelos de evaluación más flexibles y sensibles al cambio; Rodríguez Sinisterra et al. (2021) señalan que la calidad adquiere nuevas implicaciones cuando se vincula con sistemas distribuidos o arquitecturas críticas para la seguridad, ya que las evaluaciones deben considerar variables contextuales antes poco visibles; en este escenario, los métodos tradicionales muestran limitaciones frente a entornos dinámicos, lo que plantea la necesidad de enfoques capaces de acompañar ciclos de desarrollo más breves y configuraciones técnicas en constante evolución.

Asimismo, resulta pertinente examinar la relación entre modelos de calidad, procesos y productos generados durante el desarrollo, puesto que una mirada fragmentada puede derivar en valoraciones parciales; Callejas-Cuervo y Alarcón-Aldana (2017) sostienen que los modelos deben comprenderse como sistemas que articulan múltiples dimensiones del software, lo que implica que la evaluación no puede restringirse a un único atributo o etapa del ciclo de vida; desde esta perspectiva, conviene revisar cómo las métricas se integran a los procesos y cómo

influyen en la percepción global de la calidad, especialmente cuando intervienen equipos multidisciplinarios o metodologías ágiles que distribuyen responsabilidades de forma diferente.

Otra preocupación señalada en la literatura se vincula con el lugar que ocupan la seguridad y la protección de la información dentro de los procesos de evaluación, dimensiones que han adquirido mayor visibilidad a medida que los sistemas gestionan datos sensibles y operan en entornos expuestos; Mellado Fernández et al. (2010) sostienen que la calidad no puede desligarse de la seguridad, dado que un producto vulnerable compromete su confiabilidad y la integridad de la información que administra; en consecuencia, la evaluación debe ampliarse hacia prácticas más rigurosas que integren factores técnicos y organizacionales, especialmente cuando se trata de sistemas críticos o con alta exposición pública.

Por otra parte, diversos estudios advierten la distancia existente entre los modelos formales y su aplicación cotidiana, pues en muchos equipos de desarrollo prevalecen decisiones basadas en la intuición o en rutinas heredadas que dificultan la adopción de lineamientos estructurados; Pacienzia (2015) identifica esta brecha en proyectos formativos donde los procesos de evaluación dependen en gran medida de la experiencia de los actores involucrados; esta situación afecta la consistencia de los resultados y limita la posibilidad de consolidar aprendizajes a partir de evidencia acumulada, lo que refuerza la necesidad de fortalecer la articulación entre teoría y práctica en la evaluación del software.

Ante este conjunto de tensiones, se hace evidente la necesidad de desarrollar análisis sistemáticos que permitan ordenar información dispersa y construir una visión comparativa de los enfoques vigentes; Díaz y Peralta Luján (2022) subrayan la importancia de revisar y clasificar los métodos disponibles para comprender cómo sus características dialogan con las exigencias actuales del desarrollo de software; en este sentido, contar con una síntesis rigurosa favorece la

identificación de tendencias, coincidencias y vacíos presentes en la literatura, al tiempo que ofrece criterios más claros para orientar decisiones en proyectos concretos; así, la organización estructurada del conocimiento no solo fortalece la reflexión académica, sino que también aporta herramientas pertinentes para evaluar la calidad del software en contextos diversos y cambiantes.

Pregunta Problema

¿Cuáles son los métodos de evaluación de software descritos en la literatura científica, cómo han sido validados empíricamente y qué limitaciones presentan en su aplicación dentro de los entornos contemporáneos de desarrollo de software?

Objetivos

Objetivo General

Realizar una revisión sistemática de la literatura científica para identificar, clasificar y analizar los métodos de evaluación de software, con el fin de determinar su validez empírica y las limitaciones que inciden en su aplicabilidad en los contextos actuales de desarrollo.

Objetivos Específicos

Identificar los métodos de evaluación de software descritos en fuentes científicas especializadas, clasificándolos en sus enfoques, características y contextos de aplicación.

Analizar la evidencia empírica asociada a la efectividad y validez de los métodos de evaluación identificados a partir de los estudios incluidos en la revisión sistemática.

Analizar críticamente las fortalezas, limitaciones y vacíos de los métodos de evaluación de software, con el fin de identificar tendencias y proponer mejoras para su aplicación en los contextos actuales de desarrollo.

Delimitación del Estudio

El estudio se orienta a examinar los métodos de evaluación de software desde una perspectiva documental, sustentada en los principios metodológicos de la investigación científica. Hernández Sampieri et al. (2014) señalan que toda investigación requiere establecer límites claros que definan su alcance y aseguren coherencia interna. En este sentido, la delimitación principal consiste en la consulta exclusiva de fuentes académicas con revisión por pares que presenten marcos teóricos consistentes y aplicaciones verificables. Se excluyen documentos sin control editorial o de carácter comercial, con el propósito de garantizar que el análisis se apoye en literatura científica validada y metodológicamente fundamentada dentro del campo de la ingeniería de software.

Una segunda delimitación se relaciona con el tipo de métodos incluidos en la revisión. Siguiendo los lineamientos para revisiones sistemáticas en ingeniería de software propuestos por Kitchenham y Charters (2007), se consideran únicamente estudios que describan de manera explícita sus fundamentos conceptuales, procedimientos metodológicos y criterios de aplicación. Se excluyen aproximaciones que presenten descripciones superficiales o insuficientemente detalladas, ya que su falta de precisión dificultaría la clasificación comparativa y el análisis estructurado. Esta decisión metodológica permite garantizar que los métodos examinados puedan analizarse bajo criterios homogéneos, facilitando la identificación de patrones, convergencias y divergencias dentro de la literatura especializada.

El alcance temporal del estudio comprende publicaciones producidas en los últimos diez años, con el fin de asegurar actualidad y pertinencia frente a los contextos actuales de desarrollo de software. Esta ventana temporal permite analizar propuestas que dialogan con transformaciones recientes del sector, tales como metodologías ágiles, integración continua,

automatización de pruebas y arquitecturas distribuidas. No obstante, cuando resulta necesario para la comprensión conceptual, se reconocen antecedentes teóricos previos que fundamentan la evolución histórica de los modelos de calidad. De este modo, se equilibra la actualización del análisis con el sustento teórico indispensable para interpretar adecuadamente las tendencias identificadas.

Asimismo, la revisión se limita a estudios que presenten algún tipo de validación empírica, ya sea mediante experimentos controlados, estudios de caso o análisis comparativos. Esta delimitación responde al propósito de examinar la evidencia disponible sobre la aplicabilidad real de los métodos de evaluación en situaciones concretas. Sin embargo, el estudio no pretende replicar dichas validaciones ni emitir juicios definitivos sobre la efectividad absoluta de cada modelo. Su intención es analizar críticamente los resultados reportados, identificar fortalezas y limitaciones metodológicas, y reconocer el grado de respaldo empírico existente en la literatura reciente.

Otra delimitación importante consiste en que la investigación no desarrolla un nuevo marco de evaluación ni propone herramientas originales. El estudio mantiene un carácter descriptivo y analítico orientado a organizar, clasificar y examinar críticamente los métodos existentes. Esta decisión permite concentrar los esfuerzos en la interpretación de la evidencia disponible y en la identificación de tendencias dentro del campo, evitando ampliar el alcance hacia fases experimentales o de diseño metodológico. De esta manera, se preserva la coherencia interna del trabajo y se garantiza que las conclusiones se fundamenten exclusivamente en el análisis sistemático de la literatura científica revisada.

La revisión se restringe además a métodos aplicables en entornos de desarrollo general, excluyendo aquellos diseñados exclusivamente para dominios altamente especializados como

software médico regulado, sistemas aeroespaciales o aplicaciones militares. Si bien estos sectores cuentan con marcos normativos rigurosos y prácticas específicas de evaluación, su inclusión ampliaría considerablemente el alcance del estudio y dificultaría la síntesis comparativa. La decisión adoptada permite concentrarse en modelos con potencial de aplicación transversal, cuya pertinencia pueda analizarse en diversos contextos organizacionales sin depender de regulaciones sectoriales altamente particulares.

Del mismo modo, el estudio no aborda la evaluación del software desde perspectivas centradas en gestión organizacional, análisis económico o modelos de madurez de procesos, aunque reconoce que estos factores influyen indirectamente en la calidad del producto. La revisión se enfoca específicamente en métodos orientados a valorar atributos del software, ya sean internos, externos o relacionados con su interacción con el usuario. Esta delimitación temática contribuye a mantener una línea analítica clara y evita la dispersión conceptual hacia áreas que requieren marcos teóricos distintos y tratamientos metodológicos independientes.

Los criterios de búsqueda y selección se definieron de manera explícita, considerando pertinencia temática, claridad metodológica, actualidad y aporte empírico. Este procedimiento sigue recomendaciones ampliamente aceptadas para la realización de revisiones sistemáticas en ingeniería de software, lo que favorece la transparencia y reproducibilidad del proceso. Cada documento fue evaluado según su coherencia interna, descripción metodológica y relevancia para los objetivos planteados. Este proceso de filtrado garantiza que la muestra final de estudios analizados responda a criterios uniformes y verificables, fortaleciendo así la validez científica del trabajo.

En cuanto al alcance geográfico, el estudio no se limita a una región específica, dado que los métodos de evaluación de software se desarrollan y aplican en contextos internacionales

diversos. No obstante, la investigación no pretende comparar diferencias culturales o normativas entre países, sino identificar tendencias generales y rasgos conceptuales comunes presentes en la literatura. Esta delimitación permite ofrecer una visión amplia del estado actual del campo sin desviar el análisis hacia estudios comparativos regionales que excederían los objetivos planteados.

Por ello, el propósito del estudio se circunscribe al análisis conceptual y crítico de los métodos identificados, por lo que sus resultados no deben interpretarse como lineamientos prescriptivos ni como recomendaciones definitivas para su implementación. La investigación busca proporcionar un panorama organizado que facilite la comprensión del estado actual de la evaluación de software y contribuya a la reflexión académica. Con esta delimitación se establece un marco claro sobre lo que el estudio aborda y aquello que queda fuera de su alcance, garantizando coherencia entre objetivos, metodología y resultados.

Marco Teórico

Ciclo de Vida del Software

El ciclo de vida del software puede representarse mediante distintos modelos que organizan las etapas de desarrollo de manera específica. Entre los más reconocidos en la literatura se encuentran el modelo en cascada, el modelo en V, el modelo espiral, el desarrollo incremental y el prototipado. Cada uno propone una secuencia particular de actividades y define momentos distintos para la verificación, validación y control de calidad. Estos modelos no solo estructuran el proceso técnico, sino que también condicionan la forma en que se aplican los métodos de evaluación del software. Comprender sus características generales resulta fundamental para analizar posteriormente la pertinencia de los enfoques evaluativos identificados en esta investigación.

Tabla 1

Ciclo de vida del software

Modelo	Características principales	Enfoque de calidad
Cascada	Secuencial y estructurado	Evaluación al final del proceso
Modelo en V	Validación paralela al desarrollo	Verificación en cada fase
Espiral	Iterativo con gestión de riesgos	Evaluación progresiva
Incremental	Desarrollo por módulos funcionales	Validación por entregas
Prototipado	Construcción preliminar para refinar requisitos	Retroalimentación temprana

Nota. Elaboración propia.

Los modelos de ciclo de vida del software establecen la estructura bajo la cual se planifican, desarrollan y validan los sistemas. Cada modelo responde a necesidades específicas del proyecto y determina el momento en que se realizan las actividades de control y evaluación de calidad. Por ejemplo, el modelo en cascada organiza el proceso de manera lineal, mientras que

el modelo en V introduce mecanismos explícitos de verificación en cada fase. En contraste, el modelo espiral y el incremental permiten evaluaciones progresivas que reducen riesgos de forma temprana. Estas diferencias influyen directamente en cómo se planifican pruebas, validaciones y controles de calidad, aspecto clave para comprender la aplicabilidad de los métodos evaluados en esta monografía.

El ciclo de vida del software reúne las etapas que permiten concebir, construir y mantener un producto digital, y su comprensión resulta esencial para valorar la calidad que se obtiene en cada fase del desarrollo. Las prácticas asociadas a estas etapas influyen directamente en la manera en que se documenta, prueba y despliega un sistema, lo que convierte al ciclo de vida en un eje de organización del trabajo. Soto Durán y Reyes Gamboa (2007) explican que los modelos basados en procesos aportan una estructura que guía al equipo y facilita la identificación de actividades críticas, idea que permite entender por qué este concepto se mantiene vigente dentro de la ingeniería de software.

Otro aspecto importante del ciclo de vida es su capacidad para adaptarse a diferentes metodologías y contextos, ya que no todos los proyectos requieren el mismo nivel de formalización o la misma secuencia de actividades. Pacienzia (2015) describe cómo la elección de un enfoque influye en la forma en que se planifican las tareas, se gestionan los recursos y se controlan los riesgos, lo que sugiere que cada organización interpreta el ciclo de vida en función de su cultura y necesidades. Esta flexibilidad permite que el concepto sea aplicable tanto en entornos estructurados como en desarrollos más dinámicos.

El papel de las pruebas dentro del ciclo de vida adquiere especial relevancia, ya que constituyen la etapa que permite verificar el comportamiento del software antes de su implementación en escenarios reales. Serna (2011) argumenta que las pruebas deben integrarse

desde el inicio del proceso, puesto que su función no se limita a detectar fallas al final, sino que ayuda a prevenir errores y a mejorar la comprensión de los requisitos. Esta visión amplía el significado del ciclo de vida y lo vincula con prácticas que fortalecen la calidad del producto, además de favorecer la toma de decisiones informadas durante el desarrollo.

La validación y verificación son también elementos clave, pues permiten contrastar el producto con los requisitos y garantizar que su diseño cumpla con las expectativas establecidas. Mera Paz (2016) señala que estas actividades deben abordarse en distintos momentos del ciclo, ya que cada una aporta información complementaria sobre el funcionamiento del software. Esta perspectiva revela que el ciclo de vida no es una secuencia rígida, sino un proceso que integra prácticas iterativas que ayudan a mejorar el producto conforme avanza su construcción. La variedad de técnicas disponibles demuestra la importancia de seleccionar las más adecuadas según los objetivos del proyecto.

Además, el ciclo de vida del software incluye actividades relacionadas con el mantenimiento y la evolución del sistema, elementos que adquieren relevancia a medida que el software se utiliza en entornos cambiantes. Rodríguez Sinisterra et al. (2021) describen cómo los sistemas deben evaluarse continuamente para garantizar su seguridad, especialmente cuando atienden funciones críticas o manejan información sensible. Esta necesidad de ajustes permanentes evidencia que el ciclo de vida no finaliza con la entrega del producto, sino que continúa mientras el software siga en operación, lo que resalta la importancia de contar con metodologías que acompañen la evolución del sistema.

El ciclo de vida también está influido por los procesos de documentación, ya que estos permiten asegurar que cada etapa cuente con información clara que respalde las decisiones tomadas. Cárdenas Castellanos (2017) explica que la calidad de los productos intermedios

depende en gran medida de la precisión con que se registren los hallazgos y procedimientos, puesto que esta documentación facilita la comunicación entre el equipo y reduce la incertidumbre en fases posteriores. Al integrar esta práctica a lo largo del ciclo de vida, se favorece la coherencia del desarrollo y se consolidan criterios que permiten una evaluación más completa del producto.

Otra dimensión relevante del ciclo de vida se relaciona con la selección de modelos que orientan la construcción del software, ya que estos establecen la secuencia y el tipo de actividades necesarias para avanzar. Esterkin y Pons (2017) indican que los enfoques dirigidos por modelos permiten entender de manera estructurada cómo se organizan los artefactos y cómo se transforman durante el desarrollo, lo que contribuye a optimizar el proceso y facilitar su análisis. Esta perspectiva demuestra que los modelos del ciclo de vida no solo guían la práctica, sino que también ofrecen herramientas conceptuales para evaluar la calidad en cada etapa.

Asimismo, el ciclo de vida se convierte en un punto de referencia para analizar los métodos de evaluación de software, ya que cada uno se aplica en momentos distintos y responde a atributos específicos del producto. Díaz y Peralta Luján (2022) muestran que los modelos de calidad permiten establecer criterios que pueden vincularse con etapas del ciclo como el diseño, la implementación o la prueba, lo que facilita determinar qué aspectos deben evaluarse y cómo interpretarlos. Esta relación entre métodos y etapas evidencia que comprender el ciclo de vida resulta indispensable para aplicar modelos de evaluación de manera pertinente y coherente con las necesidades del proyecto.

La interacción entre componentes y artefactos dentro del ciclo de vida también influye en la calidad final, pues cada uno desempeña un papel en la arquitectura del sistema. Carvallo et al. (2008) plantean que la calidad de los componentes depende de decisiones que se toman en fases

tempranas, lo que sugiere que una planificación adecuada del ciclo puede evitar retrabajos y reducir errores. Esta relación refuerza la idea de que el ciclo de vida no solo organiza el proceso, sino que contribuye a la coherencia técnica del software, ya que integra actividades que responden tanto a requisitos funcionales como a criterios de diseño.

Para culminar, el ciclo de vida puede entenderse como un marco que articula prácticas, herramientas y decisiones que orientan el desarrollo del software, permitiendo valorar su calidad desde perspectivas complementarias. OpenWebinars (2025) indica que los modelos que guían estas actividades ayudan a establecer expectativas claras sobre el producto, lo que respalda una gestión más organizada y enfocada en resultados verificables. Esta visión convierte al ciclo de vida en un elemento central del análisis de métodos de evaluación, ya que su estructura permite comprender cómo se relacionan los enfoques teóricos con las actividades prácticas del desarrollo.

Metodologías Tradicionales y Ágiles

La forma de gestionar proyectos de software ha evolucionado a partir de metodologías tradicionales que organizan el trabajo de manera secuencial y altamente estructurada, lo que permite seguir un orden claro desde la definición de requisitos hasta la entrega final. Estas metodologías surgieron como una respuesta a la necesidad de controlar proyectos complejos y reducir la incertidumbre, aunque su carácter rígido puede generar dificultades en entornos cambiantes. Pacienzia (2015) describe que estos enfoques se apoyan en una planificación detallada que busca anticipar riesgos y asegurar entregables definidos, lo que permite comprender por qué se mantienen en organizaciones que requieren estabilidad y documentación extensa.

El interés por metodologías ágiles surgió cuando la industria comenzó a enfrentar necesidades de adaptación más rápida, especialmente en proyectos donde los requisitos evolucionan de manera continua. Estos enfoques proponen ciclos iterativos de trabajo y una colaboración constante entre los actores involucrados, lo que favorece ajustes tempranos y una mayor capacidad de respuesta frente a imprevistos. Cárdenas Castellanos (2017) observa que la adopción de prácticas ágiles transforma la dinámica del equipo, ya que promueve una comunicación más directa y una gestión que prioriza la entrega de valor en periodos cortos, lo que explica su creciente popularidad en contextos con alta variabilidad.

Al comparar ambos enfoques, resulta evidente que cada uno responde a modelos de trabajo distintos, determinados por la estabilidad del proyecto y la naturaleza de los requisitos. Las metodologías tradicionales permiten un control preciso del proceso, mientras las ágiles facilitan la adaptación continua, lo que plantea desafíos para las organizaciones al momento de decidir cuál enfoque adoptar. Camacho Quintero (2023) señala que esta decisión debe considerar la relación entre los objetivos del proyecto y la capacidad del equipo para responder a cambios frecuentes, lo que evidencia que no existe un modelo universal, sino prácticas que deben seleccionarse a partir del contexto operativo.

Otro aspecto central en este marco es la manera en que ambas metodologías influyen en la calidad del software, ya que el enfoque elegido determina cómo se planifican las pruebas y cómo se controla la evolución del producto. En entornos tradicionales, las pruebas suelen ubicarse en etapas específicas, mientras que los métodos ágiles promueven evaluaciones continuas que permiten detectar problemas con mayor anticipación. Serna (2011) explica que esta diferencia incide en la forma en que se interpretan los resultados y en la manera en que se

documentan los avances, lo que demuestra que la metodología no solo organiza tareas, sino que también orienta la perspectiva desde la cual se evalúa la calidad.

La integración de estándares y modelos de calidad en ambas metodologías también marca diferencias importantes, ya que las aproximaciones tradicionales suelen alinearse con marcos más estructurados, mientras las ágiles requieren interpretaciones flexibles que acompañen su dinámica de trabajo. Callejas-Cuervo y Alarcón-Aldana (2017) muestran que los modelos de calidad pueden adaptarse a distintos enfoques, aunque requieren ajustes que respondan a la forma en que se produce el software, lo que sugiere que las metodologías no deben entenderse únicamente como secuencias de actividades, sino como entornos que condicionan la aplicación de criterios evaluativos. Esta adaptación continua fortalece la coherencia entre prácticas y resultados.

A modo de cierre, el análisis de metodologías tradicionales y ágiles permite comprender cómo la industria ha buscado equilibrar estabilidad y flexibilidad, especialmente en un contexto donde los sistemas deben responder a exigencias cambiantes con tiempos de entrega cada vez más reducidos. OpenWebinars (2025) señala que la gestión de la calidad exige integrar prácticas que acompañen el ciclo de desarrollo sin generar cargas innecesarias, lo que evidencia que ambas metodologías ofrecen aportes valiosos que pueden combinarse según las particularidades de cada proyecto. Esta perspectiva abre la posibilidad de análisis híbridos que permitan aprovechar fortalezas de ambos enfoques.

Tabla 2

Principales metodologías tradicionales y ágiles

Tipo	Metodología	Características principales	Relación con la calidad
Tradicional	Cascada	Planificación rígida y secuencial	Control formal y documentado

Tipo	Metodología	Características principales	Relación con la calidad
Tradicional	RUP	Iterativo con fuerte documentación	Evaluación estructurada
Ágil	Scrum	Iteraciones cortas (sprints)	Evaluación continua
Ágil	Kanban	Flujo continuo de tareas	Control visual del proceso
Ágil	XP	Programación extrema y pruebas constantes	Calidad integrada al desarrollo

Nota. Elaboración propia.

Las metodologías tradicionales y ágiles representan enfoques distintos para la gestión del desarrollo de software. Las tradicionales, como cascada o RUP, privilegian la planificación detallada, la documentación extensa y el control secuencial del proceso. Por su parte, las metodologías ágiles, como Scrum, Kanban y Extreme Programming (XP), promueven ciclos iterativos, colaboración constante y adaptación continua a cambios en los requisitos. Estas diferencias influyen directamente en la manera en que se integra la evaluación de calidad, ya que en los enfoques tradicionales suele concentrarse en fases definidas, mientras que en los ágiles se incorpora de forma continua durante el desarrollo. Comprender estas variaciones resulta fundamental para analizar la aplicabilidad de los métodos de evaluación en los contextos actuales de desarrollo.

Gestión de Riesgos

La gestión de riesgos constituye un componente esencial dentro de los proyectos de software, ya que permite anticipar problemas que podrían afectar el alcance, el desempeño o la calidad del producto. Su relevancia se relaciona con la complejidad creciente de los sistemas y con la necesidad de tomar decisiones informadas en cada etapa del desarrollo. Mellado Fernández et al. (2010) mencionan que la identificación temprana de amenazas técnicas y operativas facilita proteger la integridad del sistema, lo que demuestra que la gestión de riesgos

no se limita a un ejercicio preventivo, sino que incide en el modo en que se evalúa la seguridad y la confiabilidad del software.

En los procesos de desarrollo, la gestión de riesgos se integra como una práctica que acompaña la planificación, el diseño y las pruebas, permitiendo ajustar estrategias según evoluciona el proyecto. Esta visión dinámica requiere considerar factores como cambios en los requisitos, dificultades técnicas o disponibilidad del equipo, elementos que influyen en la estabilidad del proceso. Cárdenas Castellanos (2017) afirma que el análisis de riesgos favorece una mejor comprensión del proyecto, especialmente cuando los equipos deben decidir qué actividades priorizar y qué medidas implementar para evitar retrasos o fallas críticas, lo que evidencia su efecto directo sobre la organización del trabajo.

Otro aspecto clave de la gestión de riesgos es su relación con la calidad del software, ya que los riesgos no solo afectan la funcionalidad del sistema, sino también atributos como usabilidad, rendimiento o mantenibilidad. Díaz y Peralta Luján (2022) explican que los modelos de calidad ayudan a identificar riesgos asociados a estos atributos, lo que permite que el equipo valore qué pruebas realizar y qué ajustes realizar durante el ciclo de vida. De esta manera, la gestión de riesgos se convierte en una herramienta que conecta la evaluación técnica con las expectativas del usuario, lo que contribuye a mejorar el producto desde una perspectiva integral.

El papel de las pruebas dentro de la gestión de riesgos también es determinante, ya que estas permiten revelar comportamientos inesperados que podrían comprometer el funcionamiento del software en condiciones reales. Jústiz-Núñez et al. (2014) muestran que los laboratorios de calidad permiten detectar riesgos asociados al desempeño y la estabilidad, lo que permite corregir errores antes de la implementación. Esta dinámica refuerza la idea de que la gestión de riesgos no es una actividad aislada, sino un proceso continuo que articula

verificaciones, documentación y análisis de resultados, lo que aporta información clave para la toma de decisiones.

La seguridad de la información representa otra dimensión central dentro de la gestión de riesgos, especialmente en sistemas que manejan datos sensibles o en aquellos que deben operar sin interrupciones en entornos críticos. Rodríguez Sinisterra et al. (2021) plantean que evaluar la seguridad permite reconocer vulnerabilidades que pueden pasar desapercibidas en otras etapas del desarrollo, lo que resalta la importancia de adoptar métodos robustos que acompañen esta labor. En este sentido, la gestión de riesgos contribuye a prevenir incidentes que afectan la confianza del usuario y la continuidad de los servicios, lo que la convierte en una práctica estratégica dentro de proyectos de software.

La gestión de riesgos también se relaciona con la elección de metodologías de desarrollo, ya que cada enfoque propone formas distintas de identificar, monitorear y mitigar las amenazas que surgen durante el proceso. Pacienzia (2015) indica que los modelos tradicionales suelen abordar los riesgos desde la planificación inicial, mientras que las metodologías ágiles los tratan de forma iterativa, ajustando las acciones según se presentan los cambios. Esta distinción revela que la gestión de riesgos debe adaptarse al estilo de trabajo de cada equipo y que no existe un único método aplicable a todos los contextos, por lo que su efectividad depende del equilibrio entre estructura y flexibilidad.

Así, la gestión de riesgos se posiciona como un componente transversal que articula decisiones técnicas, organizacionales y operativas, lo que demanda un análisis constante del entorno y del avance del proyecto. OpenWebinars (2025) destaca que una visión integral de los riesgos favorece una gestión más coherente del ciclo de vida, ya que permite anticipar situaciones que pueden comprometer la calidad del software y orientar prácticas que reduzcan su

impacto. Esta perspectiva confirma que la gestión de riesgos constituye un elemento indispensable para asegurar resultados confiables en entornos contemporáneos caracterizados por altas exigencias y cambios continuos.

Tabla 3

Tipos de amenazas en proyectos de software y riesgos asociados

Tipo de amenaza	Riesgos más comunes	Impacto en la calidad
Técnicas	Fallos en arquitectura, errores de integración, deuda técnica acumulada	Disminución de mantenibilidad y rendimiento
Operativas	Retrasos en cronograma, rotación de personal, mala comunicación	Afectación en cumplimiento y coherencia del producto
De requisitos	Cambios frecuentes, ambigüedad, requisitos incompletos	Defectos funcionales y retrabajos
De seguridad	Vulnerabilidades, accesos no autorizados, pérdida de datos	Compromiso de confiabilidad e integridad
De desempeño	Bajo rendimiento, escalabilidad limitada	Insatisfacción del usuario
Organizacionales	Falta de liderazgo, gestión inadecuada	Descoordinación y baja calidad documental

Nota. Elaboración propia.

La gestión de riesgos en proyectos de software implica identificar y clasificar las amenazas que pueden comprometer el desarrollo y la calidad del producto. Estas amenazas pueden ser técnicas, operativas, organizacionales o relacionadas con la seguridad y los requisitos. Cada categoría presenta riesgos específicos que afectan atributos clave como confiabilidad, rendimiento o mantenibilidad. Por ejemplo, los riesgos técnicos pueden generar fallos estructurales, mientras que los cambios constantes en los requisitos pueden provocar retrabajos y defectos funcionales. La clasificación sistemática de estas amenazas permite establecer estrategias de mitigación más precisas y facilita la selección de métodos de evaluación

adecuados. Comprender esta tipología resulta fundamental para analizar la pertinencia de los modelos de calidad revisados en esta investigación.

Buenas Prácticas

Las buenas prácticas en el desarrollo de software buscan orientar a los equipos hacia procesos más confiables, organizados y coherentes con los objetivos del proyecto, lo que favorece la calidad del producto final. Estas prácticas suelen abarcar planificación, documentación y mecanismos de control que permiten mantener claridad sobre el trabajo realizado en cada etapa. Según Soto Durán y Reyes Gamboa (2007), la adopción de procesos bien definidos facilita que los equipos comprendan qué actividades deben ejecutarse y con qué criterios, de modo que el desarrollo avance con mayor estabilidad y orden. Esta visión resalta la importancia de contar con lineamientos que fortalezcan la gestión del ciclo de vida.

La claridad en la definición de requisitos constituye otra buena práctica relevante, ya que permite evitar ambigüedades que podrían traducirse en errores durante el diseño o la implementación. Cuando los requisitos se documentan de forma precisa y comprensible, se reduce la posibilidad de interpretaciones divergentes entre los integrantes del equipo. Femmer et al. (2017) explican que la identificación temprana de inconsistencias en los requisitos mejora la calidad del software, puesto que los problemas detectados en etapas iniciales resultan menos costosos y más fáciles de corregir. Esta práctica reafirma el papel central que tiene la comunicación en la reducción de riesgos y en la planificación del trabajo.

Las pruebas sistemáticas representan un componente fundamental dentro de las buenas prácticas, ya que permiten evaluar el comportamiento del software y reconocer posibles fallas antes de su implementación en escenarios reales. Serna (2011) considera que las pruebas deben integrarse a lo largo del desarrollo, no solo como una etapa final, lo que favorece una

retroalimentación continua y un ajuste oportuno del producto. Esta integración contribuye a mejorar la calidad del software y a fortalecer la toma de decisiones, pues los resultados obtenidos aportan información que orienta las acciones posteriores del equipo. Así, las pruebas se convierten en un mecanismo de apoyo constante.

Otra buena práctica está vinculada con la documentación, ya que esta actúa como un registro que facilita el seguimiento del proyecto y crea un soporte para futuras modificaciones. Cárdenas Castellanos (2017) menciona que una documentación bien elaborada ayuda a mantener la coherencia del desarrollo, especialmente en proyectos de larga duración o con rotación de personal. Esta práctica permite que los equipos comprendan el propósito y funcionamiento de cada componente, lo que favorece intervenciones más precisas y disminuye la posibilidad de errores generados por desconocimiento del historial del proyecto. De este modo, la documentación contribuye a la sostenibilidad del software.

La evaluación de la usabilidad también forma parte de las buenas prácticas, ya que se orienta a garantizar que el software responda adecuadamente a las necesidades de las personas que lo utilizarán. Este tipo de evaluación requiere conocer las expectativas y dificultades del usuario, lo que permite realizar ajustes que mejoren la experiencia de interacción. Díaz y Peralta Luján (2022) advierten que la usabilidad constituye un atributo clave en contextos educativos y organizacionales, donde el éxito del software depende de su capacidad para ser comprendido y utilizado con facilidad. Esta perspectiva subraya la importancia de integrar criterios centrados en el usuario durante el proceso de evaluación.

Las buenas prácticas también incluyen la revisión de componentes y artefactos, dado que ellos conforman la base estructural del software y su calidad influye directamente en el comportamiento general del sistema. Carvallo et al. (2008) sostienen que evaluar estos elementos

permite identificar inconsistencias en etapas tempranas del diseño, lo que favorece la cohesión entre partes y reduce errores de integración. Esta práctica fortalece la arquitectura del software y contribuye a que el desarrollo avance de manera más organizada, ya que cada componente se analiza desde una perspectiva funcional y estructural. Su incorporación resulta especialmente útil en proyectos complejos.

La gestión de riesgos representa otra dimensión clave dentro de las buenas prácticas, puesto que anticipa posibles fallas y permite tomar decisiones que reduzcan su impacto. Mellado Fernández et al. (2010) explican que la identificación de amenazas técnicas y operativas facilita proteger el sistema y establecer medidas preventivas que fortalezcan la seguridad. Esta visión destaca que la gestión de riesgos debe integrarse a lo largo del desarrollo, ya que los problemas pueden surgir en cualquier momento del ciclo de vida y requieren una respuesta oportuna. Con ello, se promueve un enfoque que combina prevención, seguimiento y mejora continua.

La cultura de mejora continua se reconoce como una buena práctica transversal, ya que impulsa a los equipos a revisar sus procesos y ajustar estrategias según los aprendizajes obtenidos. OpenWebinars (2025) menciona que esta cultura permite identificar oportunidades para optimizar tiempos, reducir errores y fortalecer la calidad del software. Su implementación requiere disposición para reflexionar sobre el trabajo realizado y adoptar mecanismos que permitan evaluar y perfeccionar las prácticas existentes. Esta perspectiva ayuda a mantener un desarrollo más consciente, flexible y orientado al crecimiento profesional y organizacional.

La planificación estructurada y la definición clara de procesos constituyen una buena práctica esencial en proyectos de desarrollo de software. Por ejemplo, en un proyecto de implementación de un sistema académico, establecer desde el inicio un cronograma detallado, roles definidos y entregables por fase permite reducir ambigüedades y mantener control sobre el

avance. Cuando el equipo documenta decisiones técnicas y criterios de aceptación, se facilita la supervisión y el seguimiento del cumplimiento de objetivos. Esta práctica no solo organiza el trabajo, sino que también fortalece la trazabilidad entre requisitos, diseño e implementación, permitiendo detectar desviaciones tempranas y mejorar la coherencia del producto final.

La correcta definición y validación temprana de requisitos representa otra buena práctica determinante para la calidad del software. Por ejemplo, en el desarrollo de una plataforma de comercio electrónico, la identificación precisa de requisitos funcionales —como métodos de pago o gestión de inventarios— evita retrabajos posteriores derivados de interpretaciones erróneas. Si durante la fase inicial se realizan revisiones cruzadas con los interesados y se detectan inconsistencias, el equipo puede ajustar el diseño antes de iniciar la programación. Esta práctica reduce costos, mejora la comunicación entre desarrolladores y clientes, y contribuye a minimizar defectos estructurales que afectarían la experiencia del usuario en etapas posteriores.

La integración continua de pruebas constituye igualmente una práctica clave para asegurar estabilidad y confiabilidad. En un proyecto de aplicación móvil, por ejemplo, implementar pruebas automatizadas en cada actualización permite identificar fallos de rendimiento o compatibilidad antes de publicar nuevas versiones. Esta estrategia facilita una retroalimentación constante y reduce el riesgo de liberar productos con errores críticos. Además, cuando las pruebas se combinan con revisiones de código y análisis de seguridad, se fortalece la calidad técnica del sistema y se mejora la capacidad del equipo para tomar decisiones informadas durante el ciclo de desarrollo. De esta manera, las pruebas se convierten en un mecanismo preventivo y no solo correctivo.

Sostenibilidad del Software

La sostenibilidad del software se entiende como la capacidad de un sistema para mantenerse funcional, comprensible y adaptable a lo largo del tiempo, lo que implica evaluar no solo su funcionamiento inmediato, sino también las condiciones que permitirán su evolución futura. Esta perspectiva reconoce que un software sostenible requiere procesos claros, documentación estable y decisiones de diseño que faciliten modificaciones. Vega Zepeda (2010) menciona que la sostenibilidad depende de la coherencia entre procesos y productos, ya que los equipos deben considerar los impactos de cada decisión sobre la vida útil del sistema. Esta visión ubica la sostenibilidad como un componente transversal del desarrollo.

La sostenibilidad también se relaciona con la calidad interna del software, pues atributos como mantenibilidad, modularidad o legibilidad influyen en la capacidad del sistema para adaptarse a cambios o integrarse con nuevas tecnologías. Carvallo et al. (2008) explican que la calidad de los componentes condiciona la posibilidad de realizar ajustes sin generar efectos inesperados en otras partes del sistema, lo que resalta la importancia de diseñar estructuras claras que favorezcan intervenciones posteriores. De esta manera, la sostenibilidad se vincula con decisiones técnicas tempranas que pueden fortalecer o limitar la evolución del software.

Otro elemento importante para la sostenibilidad se encuentra en la gestión de pruebas, ya que estas permiten verificar que los cambios realizados no afecten el funcionamiento del sistema, lo que garantiza su estabilidad a largo plazo. Mera Paz (2016) señala que las pruebas continuas contribuyen a detectar inconsistencias que podrían acumularse con el tiempo, situación que pondría en riesgo la capacidad del software para mantenerse operativo. Esta práctica se integra como un mecanismo de preservación que acompaña el ciclo de vida, pues asegura que cada modificación se realice con una comprensión clara de sus consecuencias.

La documentación también desempeña un papel decisivo en la sostenibilidad, ya que constituye el medio por el cual el conocimiento del sistema se transmite entre equipos y se preserva a lo largo de su ciclo de vida. Cárdenas Castellanos (2017) indica que una documentación precisa facilita la continuidad del proyecto y evita pérdidas de información técnica cuando cambia el personal, lo que reafirma su valor como insumo para el mantenimiento. Esta práctica permite que las decisiones previas se comprendan correctamente y que las futuras intervenciones se realicen con mayor seguridad y coherencia.

La sostenibilidad, además, implica considerar factores de seguridad que afectan la permanencia del sistema en contextos de riesgo. Rodríguez Sinisterra et al. (2021) explican que la detección de vulnerabilidades contribuye a mantener la integridad del software a medida que evoluciona, lo que demuestra que la seguridad no se limita a un análisis puntual, sino que constituye una dimensión que debe acompañar la evaluación continua. Este enfoque permite que el software se mantenga protegido frente a amenazas emergentes y garantiza una mayor confianza en su funcionamiento a largo plazo.

Asimismo, la sostenibilidad se relaciona con la evaluación de la usabilidad, ya que un sistema que resulta difícil de comprender o utilizar suele requerir revisiones constantes, lo que afecta la eficiencia del mantenimiento. Díaz y Peralta Luján (2022) destacan que la usabilidad influye en la aceptación del software, de modo que su análisis temprano reduce la necesidad de ajustes posteriores y prolonga la vida útil del sistema. Esta relación muestra que la sostenibilidad no depende únicamente de aspectos técnicos, sino también de la experiencia del usuario y de la facilidad con que logra interactuar con el producto.

La integración de prácticas de mejora continua también contribuye a la sostenibilidad, puesto que promueve revisiones periódicas que permiten actualizar procesos y corregir

deficiencias. OpenWebinars (2025) sostiene que la calidad se fortalece cuando los equipos revisan y ajustan sus métodos para responder a nuevas exigencias del entorno, lo que beneficia el mantenimiento a largo plazo. Esta cultura de revisión constante ayuda a que el software se conserve vigente y funcional, ya que evita la acumulación de problemas que podrían comprometer su estabilidad en el futuro.

Por último, la sostenibilidad del software debe entenderse como un equilibrio entre decisiones técnicas, organizacionales y operativas que permiten que el sistema permanezca útil en contextos cambiantes. Callejas-Cuervo y Alarcón-Aldana (2017) mencionan que los modelos de calidad ayudan a identificar atributos que inciden en esta permanencia, lo que facilita evaluar qué tan preparado está un software para acompañar la evolución de su entorno. Esta perspectiva integra tanto el diseño como el mantenimiento, lo que demuestra que la sostenibilidad es un proceso continuo que se construye a lo largo de todo el ciclo de vida.

Estado del Arte

Evolución del Desarrollo de Software

El desarrollo de software ha experimentado una transformación constante desde sus primeras prácticas hasta los enfoques contemporáneos que responden a necesidades crecientes de flexibilidad y precisión. En sus inicios, la construcción de software se caracterizaba por procesos improvisados que carecían de una estructura formal, lo que dificultaba la gestión de la complejidad. Briano (2025) explica que la disciplina evolucionó a partir de la necesidad de organizar el trabajo mediante modelos que permitieran planificar, controlar y documentar las actividades, lo que abrió camino a los primeros ciclos de desarrollo estructurados orientados a reducir incertidumbres inherentes al proceso.

Con el tiempo, la industria adoptó modelos más formales que permitieron separar el desarrollo en etapas claras y secuenciales, lo que fortaleció la capacidad para anticipar problemas y coordinar al equipo. Esta transición posibilitó un mayor control del proyecto y facilitó la estandarización de prácticas. La Universidad Abierta y a Distancia de México (2025) señala que estos modelos respondieron al objetivo de estructurar los flujos de trabajo para garantizar que el producto cumpliera requisitos previamente establecidos, lo que consolidó un enfoque lineal que predominó durante varias décadas en diversos tipos de organizaciones.

La comprensión de la evolución del desarrollo también requiere reconocer el papel que desempeñaron los sistemas de control de versiones, los cuales permitieron gestionar cambios de manera más ordenada a medida que el software se volvía más complejo. Tello-Leal y Marín (2012) mencionan que estas herramientas contribuyeron a organizar la colaboración entre desarrolladores y a documentar la historia del proyecto desde una perspectiva técnica, lo que impactó directamente en la calidad del producto. Con ello, el desarrollo comenzó a percibirse

como un proceso que necesitaba mecanismos formales para registrar decisiones y asegurar la coherencia del código.

A medida que la industria avanzó, surgió la necesidad de contar con una visión más amplia sobre los modelos utilizados en la práctica, lo que motivó la aparición de revisiones y análisis que buscaban identificar tendencias y características comunes. Desde una mirada sistemática, Jafari y Motlagh (2021) resaltan que la evolución de las metodologías responde a la búsqueda de criterios técnicos que permitan evaluar la pertinencia de cada enfoque según las particularidades del proyecto. Esta perspectiva evidencia que el desarrollo de software no es estático, sino un campo que ajusta sus prácticas conforme emergen nuevas tecnologías y necesidades organizacionales.

El crecimiento del software en sectores críticos transformó el panorama y generó demandas adicionales relacionadas con seguridad, mantenibilidad y escalabilidad, lo que impulsó enfoques centrados en proteger el ciclo de vida del sistema. En este contexto, Fox (2023) describe la transición desde modelos clásicos hacia propuestas que incorporan prácticas específicas para mitigar riesgos y fortalecer el diseño seguro, lo que dio origen a ciclos de desarrollo orientados a garantizar integridad y estabilidad. Esta evolución muestra cómo las preocupaciones contemporáneas redefinieron el papel del desarrollo en entornos donde los fallos pueden tener consecuencias significativas.

El análisis de la evolución también debe considerar que, con la consolidación de tecnologías distribuidas y arquitecturas más complejas, surgieron nuevas demandas que cuestionaron la rigidez de los modelos tradicionales. La reconstrucción histórica evidencia que los cambios tecnológicos suelen motivar transformaciones metodológicas, lo que se observa en los ajustes que las organizaciones realizaron para responder a entornos cambiantes. El

documento de la Universidad Abierta y a Distancia de México (2025) señala que el diseño arquitectónico adquirió un papel crucial en esta transición, ya que permitió comprender cómo estructurar sistemas capaces de adaptarse a escenarios dinámicos.

Así, la evolución del desarrollo de software ha estado marcada por una continua búsqueda de equilibrio entre estructura y adaptación, lo que ha generado un ecosistema metodológico diverso que reúne enfoques secuenciales, iterativos y orientados a seguridad. Esta diversidad refleja la necesidad de contar con modelos que respondan a problemas específicos y que puedan ajustarse con facilidad a distintos contextos. En este sentido, la revisión realizada por Jafari y Motlagh (2021) evidencia que la evolución no solo implica adoptar nuevos métodos, sino también reinterpretar prácticas existentes para mantener su vigencia en escenarios tecnológicos cada vez más complejos.

Surgimiento de Metodologías

El surgimiento de metodologías de desarrollo respondió a la necesidad de otorgar mayor orden y previsibilidad a proyectos que, con el tiempo, se volvieron más complejos y difíciles de gestionar. En sus primeras etapas, los equipos de software enfrentaban problemas de coordinación y control, lo que motivó la creación de modelos que dieran estructura al proceso. Briano (2025) explica que estos modelos establecieron actividades claramente definidas que permitieron organizar la secuencia de trabajo, lo que contribuyó a mejorar la comprensión del producto y a disminuir la improvisación característica de los primeros proyectos. De esta manera, se consolidaron enfoques basados en planificación detallada.

El modelo en cascada figura entre las metodologías más representativas de esos primeros intentos de sistematización, pues proponía una progresión lineal que facilitaba la documentación y el seguimiento. Este enfoque se extendió rápidamente por su claridad y su facilidad para

implementarlo en organizaciones con estructuras jerárquicas. La Universidad Abierta y a Distancia de México (2025) menciona que este tipo de modelos buscaba asegurar una transición ordenada entre etapas, de modo que los errores pudieran identificarse de acuerdo con su ubicación en el ciclo. Su popularidad se mantuvo por décadas, especialmente en proyectos con requisitos estables.

Con el avance de la industria, comenzaron a surgir cuestionamientos sobre la rigidez de los modelos secuenciales, especialmente cuando los equipos debían adaptarse a cambios frecuentes. En respuesta a estas limitaciones, aparecieron metodologías que proponían ciclos iterativos y mayor involucramiento del cliente, lo que transformó la manera de entender el proceso de desarrollo. Jafari y Motlagh (2021) indican que estas metodologías se apoyan en criterios técnicos que buscan flexibilidad y retroalimentación temprana, permitiendo ajustes continuos que resultan más compatibles con contextos dinámicos. Este cambio abrió paso a prácticas más colaborativas.

El surgimiento de enfoques ágiles marcó un punto crucial en la historia del desarrollo de software, ya que introdujo principios basados en comunicación directa, iteraciones cortas y una atención constante al usuario. Estos enfoques emergieron como respuesta directa a la lentitud y poca adaptabilidad de los modelos tradicionales. En su revisión, Chipe Ortiz (2024) explica que las metodologías ágiles incorporan prácticas que permiten reorganizar prioridades con rapidez y entregar valor de manera continua, lo que facilitó su adopción en organizaciones que enfrentaban demandas cambiantes. Su carácter flexible impulsó una transformación profunda en la gestión de proyectos.

Las metodologías ágiles se consolidaron aún más con la proliferación de estudios que analizaban su eficacia en distintos entornos, especialmente en startups y equipos pequeños que

necesitaban ajustar sus procesos al ritmo del mercado. Una tesis de la Universidad de Oulu muestra que estas metodologías permiten gestionar proyectos con recursos limitados y alta incertidumbre, lo que las convierte en una opción atractiva para empresas que deben responder con rapidez a nuevas oportunidades (Software Development Methodologies in Startups, 2018). Esta evidencia impulsó su reconocimiento académico y profesional, lo que amplió su aplicación en diversos sectores.

Sin embargo, la consolidación de metodologías ágiles no reemplazó por completo los modelos tradicionales, sino que motivó una reflexión más amplia sobre las condiciones en las que cada enfoque resulta pertinente. Un análisis publicado en el *International Journal of Interactive Mobile Technologies* señala que las metodologías ágiles requieren culturas organizacionales que faciliten la comunicación constante y la toma de decisiones descentralizada (Agile Software Development, 2020). Esta observación muestra que la efectividad de una metodología no depende solo de su estructura interna, sino también del entorno en el que se implementa, lo que favoreció enfoques híbridos.

A partir de estas discusiones, comenzaron a surgir propuestas integradoras que buscan combinar la estabilidad de los modelos tradicionales con la adaptabilidad de los enfoques ágiles. En una revisión reciente, un estudio de carácter crítico analiza los malentendidos comunes sobre ambas metodologías y propone interpretaciones que permitan aprovechar sus fortalezas (A Comprehensive Review, 2025). Esta perspectiva evidencia que el surgimiento de metodologías no constituye un proceso lineal, sino un campo en constante revisión donde se ajustan prácticas según las necesidades técnicas y organizacionales. Así, el estudio histórico revela una evolución en diálogo permanente con las transformaciones del sector.

Tendencias Actuales

Las tendencias actuales del desarrollo de software se vinculan con la creciente necesidad de ofrecer soluciones escalables y adaptables a entornos cambiantes, lo que ha impulsado la adopción de nuevas tecnologías y prácticas orientadas a optimizar el ciclo de vida del producto. Una de las transformaciones más visibles se encuentra en la expansión de la computación en la nube, pues ofrece flexibilidad y capacidad de crecimiento acorde con las demandas de los usuarios. Valencia-Arias et al. (2024) señalan que la investigación reciente destaca la importancia de la nube para soportar aplicaciones modernas, lo que demuestra su papel central en la arquitectura de sistemas contemporáneos.

La incorporación de la nube ha permitido que los equipos trabajen con arquitecturas más distribuidas, lo que influye en el diseño y la implementación de soluciones educativas, empresariales y gubernamentales. Campoverde-Molina et al. (2021) muestran que, en el ámbito educativo, las arquitecturas web actuales se apoyan en servicios distribuidos para garantizar disponibilidad y rendimiento, lo que evidencia que las tendencias tecnológicas no solo transforman el desarrollo, sino también la forma en que se conciben los sistemas. Este enfoque amplía la visión sobre cómo las aplicaciones deben responder a necesidades de acceso permanente y alto tráfico.

Otra tendencia destacada es la integración creciente de inteligencia artificial en distintas fases del desarrollo, lo que transforma tanto la construcción como la evaluación del software. Herramientas basadas en IA facilitan actividades como la generación de código, el análisis automático de errores y la optimización del rendimiento. Velasco et al. (2025) indican que la literatura reciente evidencia un aumento sostenido en el uso de estas herramientas para apoyar decisiones técnicas, lo que sugiere que la IA se está consolidando como un recurso que

complementa el trabajo humano y permite acelerar procesos que antes requerían intervenciones manuales.

El surgimiento de prácticas cloud native también ha influido en la manera en que se construyen y despliegan aplicaciones, ya que estas promueven el uso de contenedores, microservicios e infraestructuras elásticas que facilitan la integración y el mantenimiento. Giraldo-Hurtado (2023) explica que estas herramientas permiten una gestión más eficiente del despliegue, especialmente en proyectos donde es necesario realizar actualizaciones continuas sin interrumpir el servicio. Este enfoque fomenta una cultura de automatización que transforma tanto la organización interna del desarrollo como la relación entre los distintos componentes del sistema.

El interés por los microservicios ha crecido debido a su capacidad para dividir aplicaciones complejas en unidades más pequeñas y manejables, lo que facilita el mantenimiento y la ampliación del producto. En una tesis de la Universidad del Valle se describe la importancia de definir adecuadamente la granularidad de los servicios para asegurar coherencia funcional y evitar sobrecargas en la arquitectura (Modelo Inteligente de Especificación de Granularidad, 2020). Esta línea de investigación sugiere que los microservicios no solo representan una tendencia técnica, sino también un desafío que requiere criterios claros para su implementación en entornos reales.

El desarrollo contemporáneo también incorpora prácticas de integración continua y despliegue continuo, que permiten automatizar pruebas, verificaciones y entregas de software en ciclos rápidos. Estas prácticas generan un flujo constante de retroalimentación que mejora la calidad y reduce los tiempos de entrega. Campoverde-Molina et al. (2021) mencionan que la automatización se ha convertido en un componente clave para responder a la demanda de

actualizaciones frecuentes en plataformas digitales, lo que muestra cómo las tendencias actuales buscan armonizar velocidad y estabilidad a lo largo del proceso de desarrollo.

En conjunto, estas tendencias revelan un ecosistema de desarrollo cada vez más orientado hacia la flexibilidad, la automatización y la inteligencia adaptativa, elementos que redefinen tanto la arquitectura como los procesos de trabajo. Velasco et al. (2025) sostienen que estas transformaciones no solo incorporan nuevas herramientas, sino que modifican los modos de interacción entre equipos y tecnologías, generando un enfoque más colaborativo y distribuido. Con ello, el desarrollo de software se afianza como una actividad que evoluciona continuamente, guiada por innovaciones que buscan responder a desafíos técnicos y organizacionales en constante cambio.

Marco Metodológico

Tipo de Investigación

El estudio se estructura como una investigación de carácter documental, pues se orienta a revisar, clasificar y analizar información proveniente de literatura científica especializada que aborda los métodos de evaluación de software. Este tipo de investigación permite organizar el conocimiento disponible y construir interpretaciones fundamentadas sobre el estado actual del tema. Hernández Sampieri et al. (2014) indican que los estudios documentales resultan apropiados cuando se pretende comprender un fenómeno a partir de fuentes existentes, ya que permiten integrar hallazgos dispersos y elaborar análisis sistemáticos que favorecen la construcción de argumentos sólidos adaptados al propósito del estudio.

Enfoque y Método

El enfoque adoptado es cualitativo, dado que se busca interpretar conceptos, tendencias y perspectivas que emergen de los estudios examinados, lo que permite comprender el fenómeno desde una visión amplia y flexible. Hernández Sampieri et al. (2014) señalan que el enfoque cualitativo se caracteriza por su capacidad para analizar información de manera inductiva y contextual, lo que facilita identificar patrones conceptuales y relaciones entre categorías. El método seleccionado es la revisión sistemática, ya que esta estrategia ofrece una ruta ordenada para localizar, evaluar y sintetizar evidencia científica, favoreciendo procesos de análisis rigurosos que permiten construir conclusiones coherentes con la literatura revisada.

Técnicas de Recolección de Información

La recolección de información se lleva a cabo mediante un proceso sistemático de búsqueda, selección y organización de literatura científica, recurriendo a repositorios académicos y bases de datos indexadas como Scopus, IEEE Xplore, Redalyc, SciELO y Google Scholar. Se

definen los criterios de inclusión como: publicaciones entre 2020 y 2025, documentos revisados por pares, enfoque en métodos de evaluación de software y disponibilidad del texto completo. Se excluyeron trabajos comerciales, artículos sin validación empírica clara, y documentos que no aportaran elementos conceptuales o metodológicos relevantes.

Tabla 4

Criterios de inclusión y exclusión

Criterios de Inclusión	Criterios de Exclusión
Documentos publicados entre 2020 y 2025.	Publicaciones anteriores a 2020 o sin vigencia frente a prácticas contemporáneas.
Artículos, tesis y libros con revisión por pares.	Documentos comerciales o sin revisión científica.
Fuentes disponibles en repositorios reconocidos (Scopus, IEEE Xplore, Redalyc, SciELO, Google Scholar).	Documentos sin acceso al texto completo o fuentes duplicadas.
Estudios que abordan métodos de evaluación de software con enfoque técnico, conceptual o empírico.	Trabajos que mencionan métodos de evaluación de forma superficial o sin detalle metodológico.
Trabajos con validación empírica (experimentos, estudios de caso, análisis comparativos).	Estudios sin evidencia empírica verificable.
Documentos redactados en español o inglés.	Publicaciones en idiomas distintos al español o inglés.
Estudios aplicables a contextos generales de desarrollo de software.	Investigaciones centradas exclusivamente en dominios altamente especializados (médico, militar, aeroespacial).
Enfoque centrado en atributos del software (funcionalidad, mantenibilidad, usabilidad, etc.).	Estudios con enfoque exclusivamente organizacional, económico o de gestión del talento.

Nota. Elaboración propia.

La recolección documental, entendida como un proceso que articula la búsqueda intencionada de conocimiento con el análisis riguroso de fuentes académicas, ha ofrecido un terreno fértil para comprender la diversidad de enfoques utilizados en la evaluación del software,

especialmente cuando estos se aplican en entornos caracterizados por la constante transformación tecnológica; esta práctica ha facilitado no solo el acceso a marcos conceptuales relevantes, sino también el contraste de experiencias empíricas que han permitido observar cómo se comportan dichos métodos más allá de su formulación teórica, lo cual resulta crucial para construir interpretaciones contextualizadas y pertinentes, como lo advierten Hernández Sampieri et al. (2014), quienes reconocen el valor de consolidar evidencia desde fuentes confiables; en este sentido, el análisis de 16 documentos ha favorecido la construcción de una mirada integradora, donde las perspectivas técnicas, metodológicas y aplicadas se entrelazan, abriendo espacio para identificar tensiones, patrones y desafíos que aún persisten en la práctica actual.

Criterios de Análisis

El análisis de la información se desarrolla a partir de una lectura detallada que permite identificar categorías relacionadas con la evolución de las metodologías, su validación y sus limitaciones para el uso contemporáneo. Este proceso incluye comparar enfoques, reconocer patrones y sintetizar los aportes de cada estudio. Según Hernández Sampieri et al. (2014), el análisis cualitativo se caracteriza por organizar la información en categorías que emergen tanto de la teoría como de los hallazgos, lo que permite interpretar el contenido con mayor profundidad. De esta manera, los criterios adoptados buscan mantener coherencia entre los objetivos del estudio y la evidencia encontrada.

Delimitación Temática

La delimitación temática de este estudio se centra en el análisis de los métodos de evaluación de software, su validación empírica y las limitaciones que presentan para su aplicación en entornos contemporáneos de desarrollo. Este enfoque permite acotar el objeto de estudio a un conjunto específico de prácticas y criterios utilizados para valorar la calidad del

software, dejando por fuera aspectos relacionados con gestión de proyectos, programación o diseño arquitectónico. Hernández Sampieri et al. (2014) explican que toda investigación requiere una delimitación precisa que oriente la búsqueda de información y evite dispersión teórica, de modo que el análisis pueda desarrollarse con claridad y coherencia respecto a los objetivos planteados.

Análisis y Discusión

Análisis de Resultados

Clasificación de Enfoques en Métodos de Evaluación de Software

La revisión sistemática realizada permitió identificar una serie de estudios clave que abordan distintos métodos y enfoques para la evaluación de la calidad del software en contextos contemporáneos. Estos estudios, publicados entre 2020 y 2025, muestran una clara evolución desde modelos tradicionales hacia propuestas integradas con prácticas modernas como DevOps, integración continua, automatización de métricas y evaluación de atributos no funcionales. A continuación, se presenta una matriz de clasificación que resume los enfoques centrales y la contribución principal de cada fuente:

Tabla 5

Clasificación de métodos de evaluación de software según enfoque, características y contexto de aplicación

Fuente	Enfoque del método	Características principales	Contexto de aplicación	Tipo de validación
Zhang & Zhang (2024)	DevSecOps	Métricas cuantitativas en microservicios	Arquitecturas basadas en microservicios	Estudio empírico
Haindl (2021)	DevOps	Evaluación del valor funcional	Desarrollo continuo	Análisis aplicado
Najmi & El-Dosuky (2025)	Detección temprana de defectos	Evaluación automatizada	Integración continua	Experimental
Hanief et al. (2025)	Ágil	Calidad en entornos iterativos	Equipos ágiles	Revisión sistemática
Olsson et al. (2022)	Requerimientos	Requisitos de calidad	Ingeniería de requisitos	Estudio empírico
Gobov & Zuieva (2025)	Requerimientos	Atributos de calidad	Fase de análisis	Investigación aplicada

Fuente	Enfoque del método	Características principales	Contexto de aplicación	Tipo de validación
Olivero et al. (2023)	DevOps	Integración de calidad en pipeline	CI/CD	Estudio de caso
Terrón-Macías et al. (2025)	Deuda técnica	Identificación y clasificación	Mantenimiento evolutivo	Revisión
Syed-Mohamad et al. (2025)	Mantenibilidad	Métricas continuas	Desarrollo incremental	Empírico
Rahman et al. (2024)	Ágil/Tradicional	Comparación de defectos	Waterfall vs Ágil	Comparativo
Domínguez-Acosta & García-Mireles (2021)	DevOps	Actividades de aseguramiento	Entornos DevOps	Estudio aplicado
Delgado Jojoa et al. (2024)	Factores humanos	Meticulosidad y calidad	Equipos de desarrollo	Cuantitativo
Poth et al. (2020)	Ágil	Calidad del trabajo en equipo	Scrum	Empírico
Koneru (2021)	DevSecOps	Seguridad en pipelines CI/CD	Integración continua	Experimental
Klotins et al. (2022)	Evaluación continua	Análisis costo-beneficio	Ingeniería continua	Estudio longitudinal
Czekster (2024)	DevSecOps	Evaluación continua de riesgos	Seguridad de software	Empírico

Nota. Elaboración propia.

La primera categoría corresponde a los métodos basados en DevOps y DevSecOps, caracterizados por integrar actividades de evaluación directamente en los pipelines de desarrollo continuo. Zhang y Zhang (2024) proponen métricas cuantitativas orientadas a microservicios en entornos cloud, con validación empírica en arquitecturas distribuidas. Koneru (2021) incorpora herramientas de análisis de seguridad como SAST, DAST y SCA dentro de procesos CI/CD, demostrando experimentalmente la efectividad de la detección temprana de vulnerabilidades. Asimismo, Czekster (2024) plantea un modelo de evaluación continua de riesgos en contextos

DevSecOps, resaltando la necesidad de monitoreo permanente. En conjunto, estos estudios comparten características de automatización, integración continua y validación práctica en entornos reales de desarrollo.

Un segundo grupo reúne estudios centrados en métricas cuantitativas y evaluación automatizada de atributos técnicos. Najmi y El-Dosuky (2025) proponen mecanismos de detección temprana de defectos mediante pruebas automatizadas, validados en entornos de integración continua. Syed-Mohamad et al. (2025) abordan la mantenibilidad a través de métricas continuas aplicadas en desarrollos incrementales, demostrando su utilidad mediante análisis empíricos. Klotins et al. (2022) complementan esta perspectiva con un enfoque de costo-beneficio en ingeniería continua, evaluado mediante estudios longitudinales. Estas investigaciones comparten la característica de fundamentarse en mediciones objetivas y análisis sistemáticos, aplicados en contextos donde la automatización y la eficiencia operativa resultan esenciales para sostener la calidad del producto.

La tercera categoría se vincula con metodologías de desarrollo, tanto ágiles como tradicionales, y su impacto en los mecanismos de evaluación. Hanief et al. (2025) presentan una revisión sistemática sobre prácticas de calidad en entornos ágiles, destacando la retroalimentación iterativa como rasgo distintivo. Rahman et al. (2024) comparan defectos en proyectos Waterfall y ágiles mediante análisis empírico, evidenciando diferencias en el tratamiento de requerimientos. Poth et al. (2020) analizan la calidad del trabajo en equipo en Scrum a través de estudios cuantitativos. Estos trabajos muestran que el contexto metodológico condiciona las características de los métodos de evaluación, influyendo en la frecuencia, profundidad y tipo de métricas utilizadas.

Otra línea relevante corresponde a los estudios enfocados en requerimientos y atributos de calidad. Olsson et al. (2022) identifican debilidades en la definición empírica de requisitos de calidad, lo que afecta su posterior evaluación. Gobov y Zuieva (2025) plantean la integración temprana de atributos como rendimiento y seguridad en la fase de análisis, respaldando su propuesta mediante investigación aplicada. Delgado Jojoa et al. (2024) incorporan factores humanos, como la meticulosidad del equipo, en el análisis de atributos como mantenibilidad. Estos enfoques comparten la característica de situar la evaluación en etapas iniciales del ciclo de vida, ampliando el contexto de aplicación hacia dimensiones técnicas y organizacionales.

La categoría de deuda técnica representa un enfoque particular orientado al mantenimiento evolutivo. Terrón-Macías et al. (2025) desarrollan una revisión sistemática que clasifica tipos de deuda, herramientas de identificación y efectos acumulativos en proyectos de software. Su contribución se centra en demostrar, mediante análisis comparativo, cómo la acumulación de deuda impacta atributos como escalabilidad y mantenibilidad. Este enfoque se aplica especialmente en contextos donde los proyectos evolucionan de manera prolongada y requieren monitoreo constante. La característica distintiva de esta categoría es su orientación longitudinal, ya que evalúa la calidad no como un evento puntual, sino como un proceso continuo de control estructural.

De este modo, el conjunto de estudios analizados permite identificar una clasificación coherente de métodos de evaluación según su enfoque, características técnicas y contexto de aplicación. Las categorías identificadas incluyen DevOps/DevSecOps, métricas automatizadas, metodologías de desarrollo, requerimientos de calidad y deuda técnica. Cada grupo presenta formas específicas de validación, ya sea experimental, empírica o mediante revisión sistemática. Esta diversidad evidencia que la evaluación de software no responde a un modelo único, sino a

enfoques complementarios que deben seleccionarse según el entorno operativo. La sistematización presentada facilita el análisis crítico posterior sobre fortalezas, limitaciones y oportunidades de mejora en contextos actuales de desarrollo.

Efectividad y Validez Empírica

La validez empírica constituye un eje central en la discusión sobre la efectividad de los métodos de evaluación de software, en la medida en que permite valorar hasta qué punto los modelos propuestos han sido contrastados mediante evidencia verificable en escenarios concretos de aplicación. Desde esta perspectiva, evaluar un método implica examinar los procedimientos utilizados para su comprobación, los contextos en los que fue implementado y la coherencia entre resultados y objetivos declarados. Tal como señalan Domínguez-Acosta y García-Mireles (2021), la calidad de un enfoque depende en gran parte de la trazabilidad entre práctica y evidencia, lo cual refuerza la necesidad de estudios que documenten procesos reales y resultados observables.

Al profundizar en los niveles de respaldo empírico, se advierte que algunos trabajos se apoyan en diseños experimentales controlados que permiten medir con mayor precisión el desempeño de métricas o herramientas específicas. En este sentido, Zhang y Zhang (2024) validan indicadores orientados a microservicios dentro de entornos DevSecOps productivos, integrando pruebas en pipelines de integración y despliegue continuo que generan datos cuantificables sobre defectos y tiempos de respuesta. Esta aproximación otorga solidez a sus hallazgos, dado que la medición se realiza sobre sistemas en funcionamiento y bajo condiciones técnicas delimitadas, lo cual favorece la replicabilidad del estudio.

De manera complementaria, la experimentación controlada también ha sido empleada para evaluar herramientas de detección temprana de defectos, incorporando métricas como

sensibilidad y especificidad para estimar la capacidad predictiva de los modelos propuestos. Najmi y El-Dosuky (2025) desarrollan un entorno experimental en el que comparan resultados obtenidos bajo distintos escenarios de prueba, lo que les permite establecer diferencias significativas entre configuraciones. A través de este procedimiento se consolida una evidencia basada en datos medibles, aunque el alcance del estudio depende del tamaño de la muestra y del tipo de proyectos considerados en la validación.

En otros casos, el respaldo empírico se construye a partir de análisis estadísticos aplicados a múltiples proyectos de software, lo que amplía el espectro de observación y reduce el sesgo asociado a estudios individuales. Rahman et al. (2024) examinan defectos documentados en diversos repositorios, identificando patrones que permiten relacionar prácticas de desarrollo con resultados de calidad. Al trabajar con datos provenientes de contextos distintos, el estudio fortalece la confiabilidad de sus conclusiones, puesto que la evidencia no se restringe a un único entorno organizacional sino que integra información heterogénea susceptible de comparación.

Ahora bien, junto a estos trabajos con mayor rigor experimental, se ubican investigaciones aplicadas en entornos reales que prescinden de un diseño experimental formal, aunque conservan un alto valor contextual. Koneru (2021) describe la implementación de herramientas SAST, DAST y SCA en pipelines DevSecOps operativos, evidenciando mejoras en tiempos de detección de vulnerabilidades. Aunque el estudio no presenta comparaciones estadísticas amplias, aporta información situada sobre la viabilidad técnica de estas prácticas, lo que resulta relevante para organizaciones que buscan adoptar modelos similares en escenarios productivos.

Por otra parte, el estudio longitudinal desarrollado por Haindl (2021) ofrece una aproximación que combina datos cuantitativos y cualitativos dentro de un equipo DevOps

específico, permitiendo observar transformaciones en prácticas de calidad a lo largo del tiempo. Este tipo de diseño fortalece la validez interna, ya que documenta procesos de adopción y ajuste de herramientas en contextos reales, aunque su alcance externo puede verse limitado por la particularidad del caso analizado. Sin embargo, la profundidad del seguimiento proporciona una comprensión detallada de dinámicas organizacionales asociadas a la evaluación del software.

En relación con metodologías cualitativas, también se encuentran investigaciones que recurren a entrevistas y análisis documental para identificar prácticas consolidadas en proyectos reales. Domínguez-Acosta y García-Mireles (2021) exploran percepciones de profesionales sobre calidad de software, articulando evidencias provenientes de distintas fuentes. Esta estrategia metodológica permite captar dimensiones que difícilmente emergen en estudios puramente cuantitativos, aunque exige criterios rigurosos de análisis para evitar interpretaciones subjetivas. En consecuencia, la validez se apoya en la coherencia entre categorías construidas y datos recogidos.

Otro nivel de validación se configura mediante revisiones sistemáticas y mapeos de literatura que integran resultados de investigaciones previas, aportando una visión panorámica del estado del arte. Hanief et al. (2025) sintetizan más de cincuenta estudios sobre calidad en metodologías ágiles, identificando tendencias y vacíos recurrentes. Este tipo de aproximación no valida directamente un modelo propio, aunque consolida conocimiento acumulado y orienta futuras investigaciones, siempre que los criterios de selección y análisis de fuentes se encuentren claramente explicitados.

De manera similar, Olsson et al. (2022) analizan un conjunto amplio de investigaciones empíricas relacionadas con requerimientos de calidad, lo que permite identificar patrones metodológicos y áreas de mayor desarrollo. Al integrar múltiples estudios, la revisión adquiere

un carácter secundario de validación que depende de la robustez de las investigaciones originales. Por consiguiente, la efectividad de sus conclusiones está condicionada por la calidad de las fuentes incluidas, aspecto que subraya la importancia de procedimientos sistemáticos y transparentes en la construcción del corpus analizado.

Asimismo, Terrón-Macías et al. (2025) examinan herramientas y tipos de deuda técnica reportados en la literatura, estableciendo relaciones entre frecuencia de uso y contextos de aplicación. Aunque su estudio no implica experimentación directa, contribuye a ordenar el panorama conceptual y práctico del tema, ofreciendo criterios para evaluar la madurez de ciertas propuestas. Este tipo de trabajo amplía la comprensión colectiva del campo y facilita decisiones fundamentadas sobre adopción de herramientas, siempre que se reconozcan las limitaciones inherentes a la evidencia recopilada.

En contraste con los estudios empíricos consolidados, algunos trabajos se orientan principalmente a la estructuración conceptual de atributos de calidad sin aplicación directa en proyectos reales. Gobov y Zuieva (2025) proponen un marco desde la ingeniería de requisitos que organiza dimensiones relevantes, aunque la ausencia de pruebas en entornos productivos reduce el nivel de evidencia práctica. A pesar de ello, su aporte resulta pertinente para delimitar categorías analíticas que podrían ser contrastadas posteriormente mediante estudios aplicados, lo que demuestra la complementariedad entre teoría y práctica.

Por último, al considerar el conjunto de investigaciones revisadas, se observa una heterogeneidad significativa en cuanto a niveles de validación y estrategias metodológicas empleadas, lo cual evidencia un campo en proceso de consolidación. Delgado Jojoa et al. (2024) introducen variables psicológicas mediante encuestas y reportes propios, aportando una perspectiva innovadora aunque con validación técnica moderada. Esta diversidad sugiere que, si

bien existen enfoques con respaldo sólido en datos medibles, persiste la necesidad de ampliar estudios aplicados en entornos productivos que fortalezcan la replicabilidad y la confianza en los métodos de evaluación de software.

Análisis Crítico de Fortalezas, Limitaciones y Vacíos

El análisis crítico de las dieciséis fuentes seleccionadas permite identificar patrones relevantes en torno a las fortalezas, limitaciones y vacíos presentes en los métodos de evaluación de software. Más allá de describir enfoques individuales, este apartado busca interpretar cómo estos métodos responden a los contextos actuales de desarrollo y qué tan sólidos resultan en términos metodológicos. La revisión evidencia una evolución hacia modelos más automatizados e integrados al ciclo de vida, aunque también revela inconsistencias en la validación empírica y en la estandarización de métricas. La diversidad de enfoques encontrados confirma que el campo se encuentra en consolidación, con avances significativos en áreas como DevOps y seguridad, pero con desafíos persistentes en la medición integral de atributos de calidad.

Una de las principales fortalezas identificadas es la integración progresiva de la evaluación dentro de procesos de desarrollo continuo. Estudios como los de Zhang y Zhang (2024) y Koneru (2021) demuestran que es posible incorporar métricas y herramientas de seguridad directamente en pipelines CI/CD, reduciendo la distancia entre desarrollo y control de calidad. Esta integración favorece la detección temprana de defectos y vulnerabilidades, fortaleciendo la confiabilidad del producto. Asimismo, Rahman et al. (2024) aportan evidencia empírica robusta mediante análisis estadísticos de defectos reales, lo que incrementa la credibilidad de sus conclusiones. Estas investigaciones muestran que la automatización y el uso de datos cuantitativos representan una tendencia consolidada hacia evaluaciones más objetivas y técnicas.

Otra fortaleza relevante se observa en los estudios que combinan análisis empírico con aplicación en entornos reales. Haindl (2021) desarrolla un estudio longitudinal en equipos DevOps, lo que permite evaluar el impacto del método en condiciones operativas auténticas. De manera similar, Klotins et al. (2022) integran métricas económicas y técnicas para analizar el costo-beneficio en ingeniería continua. Estas aproximaciones amplían la comprensión de la calidad más allá de atributos técnicos aislados, incorporando variables organizacionales y financieras. Este enfoque multidimensional fortalece la aplicabilidad práctica de los métodos y demuestra que la evaluación de calidad puede contribuir a la toma de decisiones estratégicas dentro de las organizaciones.

Sin embargo, el análisis también revela limitaciones metodológicas importantes. Una de las más recurrentes es la falta de validación empírica amplia en contextos industriales diversos. Estudios como Gobov y Zuieva (2025) y Olivero et al. (2023) presentan aportes conceptuales valiosos, pero carecen de experimentación directa o aplicación en proyectos reales. Esta ausencia de validación limita la generalización de sus propuestas. Asimismo, algunas investigaciones dependen excesivamente de revisiones documentales o análisis exploratorios sin contrastes prácticos, lo que reduce su capacidad para demostrar efectividad operativa en entornos dinámicos y de alta exigencia técnica.

Otra limitación significativa se relaciona con la integración de los métodos en ciclos de desarrollo rápidos. En contextos ágiles y DevOps, donde las iteraciones son cortas y las decisiones deben tomarse con rapidez, ciertos modelos resultan difíciles de aplicar. Terrón-Macías et al. (2025) señalan que la gestión de deuda técnica requiere herramientas y análisis que no siempre se integran fluidamente en pipelines continuos. Del mismo modo, algunos marcos de evaluación requieren evaluaciones posteriores al desarrollo, lo que contradice la lógica de

retroalimentación inmediata que caracteriza a los entornos contemporáneos. Esta desconexión reduce la viabilidad práctica de algunos métodos en escenarios reales.

La medición de atributos no funcionales constituye uno de los vacíos más notorios en la literatura revisada. A pesar de su relevancia para la calidad global del software, atributos como mantenibilidad, usabilidad o escalabilidad presentan dificultades para ser cuantificados de manera objetiva. Syed-Mohamad et al. (2025) y Olsson et al. (2022) reconocen que muchas métricas disponibles son parciales o dependen de análisis retrospectivos. Esta situación genera una brecha entre la importancia conceptual de estos atributos y su tratamiento técnico efectivo. La ausencia de herramientas automatizadas consolidadas limita la evaluación integral y reduce la posibilidad de comparaciones estandarizadas entre proyectos.

Otro vacío identificado se relaciona con la dependencia de datos subjetivos en ciertos estudios. Investigaciones como las de Delgado Jojoa et al. (2024) y Poth et al. (2020) incorporan variables humanas mediante encuestas y reportes personales, aportando una dimensión sociotécnica valiosa. Sin embargo, la ausencia de correlación directa con métricas técnicas dificulta confirmar el impacto real de estos factores sobre la calidad del producto. Si bien la dimensión humana es innegablemente relevante, la falta de triangulación metodológica reduce la solidez empírica de los hallazgos y plantea la necesidad de combinar enfoques cuantitativos y cualitativos.

En términos de tendencias, la revisión muestra un desplazamiento hacia enfoques automatizados, empíricos y orientados a la integración continua. La mayoría de los estudios recientes prioriza la incorporación de métricas en tiempo real y la evaluación transversal a lo largo del ciclo de vida. No obstante, también se evidencia fragmentación conceptual, ya que no existe una taxonomía unificada para clasificar métodos o atributos evaluados. Esta dispersión

dificulta la comparación entre investigaciones y obstaculiza la construcción de marcos comunes de referencia que permitan consolidar el campo de estudio.

Con base en estas observaciones, se propone fortalecer la investigación futura mediante tres líneas de mejora. En primer lugar, aumentar la validación empírica en entornos industriales reales, preferiblemente mediante estudios longitudinales replicables. En segundo lugar, desarrollar herramientas automatizadas capaces de medir atributos no funcionales en tiempo real dentro de pipelines CI/CD. En tercer lugar, promover la estandarización de métricas y taxonomías que faciliten la comparabilidad entre estudios. Estas acciones permitirían transformar propuestas teóricas en soluciones escalables y aplicables en contextos actuales de desarrollo de software.

En síntesis, el análisis crítico confirma que los métodos de evaluación de software han avanzado hacia mayor automatización e integración, pero aún enfrentan desafíos estructurales relacionados con validación, medición integral y estandarización. La coexistencia de fortalezas técnicas y vacíos metodológicos evidencia un campo en transición, donde la consolidación dependerá de la articulación entre investigación empírica rigurosa y aplicabilidad práctica. Este panorama permite identificar tendencias claras y orientar futuras investigaciones hacia modelos más robustos, comparables y adaptados a las exigencias de los entornos contemporáneos de desarrollo.

Un aspecto que cobra especial relevancia en la revisión es la tensión persistente entre la sofisticación técnica de los métodos y su accesibilidad operativa en contextos organizacionales diversos, pues aunque múltiples propuestas incorporan modelos estadísticos complejos, arquitecturas de monitoreo continuo y esquemas avanzados de evaluación de madurez, su despliegue suele apoyarse en infraestructuras consolidadas y en equipos con alta especialización,

lo que, según diversos estudios sobre adopción tecnológica y modelos de madurez, restringe su implementación en pequeñas y medianas organizaciones con limitaciones presupuestarias o capacidades digitales incipientes; en este escenario, la aplicabilidad efectiva de tales enfoques queda condicionada por factores estructurales que trascienden el diseño metodológico, de modo que la solidez no puede valorarse únicamente por el rigor técnico, sino también por su ajuste al contexto, lo cual abre la necesidad de pensar en alternativas escalables y adaptables a distintos niveles de desarrollo organizacional.

Asimismo, el análisis evidencia una creciente inclinación a traducir la calidad en indicadores cuantitativos, tendencia que aporta mayor objetividad y comparabilidad a las evaluaciones, aunque al mismo tiempo puede derivar en miradas parciales cuando las métricas automatizadas se convierten en el eje casi exclusivo de valoración; en efecto, la medición continua de variables observables facilita el control y la trazabilidad de resultados, pero deja en un segundo plano dimensiones menos tangibles como la experiencia de usuario, la cultura de los equipos o la sostenibilidad en horizontes temporales amplios, aspectos que diversos estudios advierten como decisivos para interpretar el desempeño organizacional; bajo esta lógica predominantemente técnica, las evaluaciones corren el riesgo de fragmentarse si no se articulan con marcos comprensivos más amplios, por lo que resulta pertinente impulsar modelos híbridos capaces de integrar datos cuantificables con análisis cualitativos estructurados, favoreciendo así una comprensión más integral de la calidad en entornos complejos.

Otro elemento crítico remite a la transferencia de conocimiento entre la investigación académica y la industria, pues aunque numerosos estudios ofrecen marcos conceptuales rigurosos y resultados obtenidos en entornos controlados, su traducción a prácticas organizacionales concretas suele quedar implícita o poco desarrollada; en consecuencia, la falta

de guías claras de implementación, de estudios replicables en distintos sectores o de comparaciones longitudinales reduce el alcance práctico de varias propuestas y limita su apropiación por parte de actores productivos; esta distancia entre formulación teórica y aplicación efectiva pone en evidencia la necesidad de impulsar investigaciones colaborativas que integren a académicos y profesionales desde etapas tempranas, de modo que los métodos puedan validarse en escenarios reales y ajustarse a dinámicas productivas específicas, ya que la consolidación del campo dependerá en buena medida de una articulación consistente entre evidencia científica y práctica profesional.

En términos epistemológicos, el análisis pone de relieve la convivencia de enfoques sustentados en paradigmas de evaluación distintos, ya que algunos privilegian atributos internos del producto, otros se orientan a los procesos de desarrollo y otros sitúan el foco en resultados organizacionales; esta diversidad da cuenta de la complejidad que atraviesa la noción de calidad del software y, al mismo tiempo, introduce dificultades para articular marcos integradores capaces de vincular tales dimensiones con coherencia; en efecto, la ausencia de un consenso conceptual respecto a qué se entiende por calidad y cómo jerarquizar sus atributos propicia interpretaciones divergentes y metodologías que dialogan de manera parcial; ante este escenario, se vuelve pertinente avanzar hacia modelos de carácter sistémico que articulen producto, proceso y contexto organizacional bajo criterios comparables y verificables, favoreciendo así una comprensión más articulada y consistente del fenómeno.

El examen crítico confirma que la evolución de los métodos de evaluación se encuentra estrechamente ligada a la transformación tecnológica del ecosistema digital contemporáneo, dado que la incorporación de microservicios, inteligencia artificial y arquitecturas distribuidas introduce variables que los modelos tradicionales apenas alcanzaban a considerar; en

consecuencia, los marcos clásicos resultan insuficientes frente a dinámicas de alta variabilidad y despliegue continuo, lo que plantea la necesidad de enfoques evaluativos más flexibles y adaptativos; bajo esta perspectiva, el propósito de identificar tendencias y plantear mejoras adquiere mayor consistencia, pues se advierte que el porvenir de la evaluación de software dependerá de su capacidad para articular automatización avanzada con validación empírica rigurosa y criterios conceptuales compartidos, manteniendo al mismo tiempo atención a las particularidades contextuales que distinguen a cada proyecto y condicionan sus resultados.

Tabla 6

Fortalezas, limitaciones y vacíos identificados

Categoría	Hallazgos principales	Estudios representativos	Implicación
Fortalezas	Integración en CI/CD, automatización, validación cuantitativa	Zhang & Zhang (2024), Rahman et al. (2024)	Mayor objetividad y detección temprana
Validación aplicada	Estudios de caso y análisis longitudinal	Haindl (2021), Klotins et al. (2022)	Aplicabilidad organizacional
Limitación empírica	Falta de pruebas en entornos industriales amplios	Gobov & Zuieva (2025), Olivero et al. (2023)	Baja generalización
Integración limitada	Dificultad en ciclos ágiles rápidos	Terrón-Macías et al. (2025)	Problemas de adopción
Vacíos técnicos	Medición débil de atributos no funcionales	Olsson et al. (2022), Syed-Mohamad et al. (2025)	Evaluación incompleta
Subjetividad	Dependencia de encuestas y percepción	Delgado Jojoa et al. (2024), Poth et al. (2020)	Riesgo de sesgo

Nota. Elaboración propia.

Matriz Comparativa de Resultados

El estudio de las fuentes seleccionadas permitió identificar un conjunto diverso de enfoques, metodologías y niveles de validación empírica en relación con la evaluación de la

calidad del software. Para facilitar la comparación, se ha elaborado una matriz que resume los resultados clave, incluyendo el tipo de enfoque adoptado, el nivel de validación empírica presente, el contexto de aplicación y las limitaciones observadas. Esta matriz permite identificar patrones comunes, fortalezas metodológicas y vacíos en la literatura actual.

Tabla 7

Matriz comparativa de métodos de evaluación de calidad de software

Fuente	Enfoque	Validación empírica	Contexto aplicado	Limitaciones
Zhang & Zhang (2024)	DevSecOps + métricas	Experimentos en microservicios	CI/CD en la nube	Requiere infraestructura avanzada
Haindl (2021)	DevOps + valor de features	Estudio de caso longitudinal	Equipos ágiles	Limitado a entornos específicos
Najmi & El-Dosuky (2025)	Evaluación automatizada	Pruebas con defectos reales	Procesos de QA	Poca cobertura de atributos no funcionales
Hanief et al. (2025)	Revisión sistemática ágil	Revisión de 50 estudios	General	Poca evaluación de herramientas concretas
Olsson et al. (2022)	Requerimientos de calidad	Metaanálisis de estudios empíricos	Requisitos no funcionales	Bajo detalle en herramientas de evaluación
Gobov & Zuieva (2025)	Ingeniería de requerimientos	Teórico	Conceptual	Sin aplicación práctica
Olivero et al. (2023)	DevOps + calidad	Revisión exploratoria	Mapeo preliminar	Sin pruebas empíricas
Terrón-Macías et al. (2025)	Deuda técnica	Revisión sistemática	Equipos DevOps	Baja integración en ciclos rápidos
Syed-Mohamad et al. (2025)	Mantenibilidad	Revisión de métricas + exploración práctica	Análisis de software	Difícil de automatizar
Rahman et al. (2024)	Comparación ágil vs. waterfall	Análisis de defectos reales	Requerimientos funcionales	Limitado a defectos de requisitos

Fuente	Enfoque	Validación empírica	Contexto aplicado	Limitaciones
Domínguez-Acosta & Garcia-Mireles (2021)	Actividades de calidad en DevOps	Estudio de campo	Prácticas en empresas	Enfoque descriptivo, no experimental
Delgado Jojoa et al. (2024)	Psicología en calidad de software	Encuesta + análisis perceptual	Equipos universitarios	Subjetividad, sin validación técnica
Poth et al. (2020)	Trabajo en equipo ágil	Cuestionarios en equipos reales	Proyectos ágiles	Medición indirecta de calidad
Koneru (2021)	Seguridad en CI/CD	Aplicación práctica en pipelines	DevSecOps	Falta de evaluación comparativa
Klotins et al. (2022)	Evaluación costo-beneficio	Estudio empírico con métricas	Ingeniería continua	Bajo soporte para decisiones rápidas
Czekster (2024)	Evaluación de riesgos en DevSecOps	Simulación y análisis conceptual	Gestión de riesgos	Aún sin validación en campo

Nota. Elaboración propia.

La matriz comparativa revela una diversidad de enfoques en el campo de la evaluación de calidad de software, con una clara inclinación hacia propuestas ligadas a DevOps, requerimientos de calidad, y atributos no funcionales. Un primer hallazgo destacable es la variabilidad en la validación empírica: mientras que algunos estudios como los de Zhang y Zhang (2024) o Rahman et al. (2024) presentan validaciones sólidas mediante datos reales, otros como Gobov y Zuieva (2025) u Olivero et al. (2023) permanecen en un nivel teórico o exploratorio.

Los estudios con validación más fuerte suelen aplicar experimentos en entornos productivos o análisis de defectos documentados, lo que refuerza la confiabilidad de sus métodos. Zhang y Zhang (2024), por ejemplo, prueban métricas DevSecOps en despliegues reales de microservicios, lo cual representa un alto grado de aplicación práctica. Asimismo,

Rahman et al. (2024) analizan defectos en proyectos bajo metodologías ágiles y tradicionales, ofreciendo evidencia empírica sobre la relación entre enfoque de desarrollo y calidad.

En contraste, trabajos como el de Hanief et al. (2025), aunque sistemáticos en su revisión, no llegan a aplicar herramientas concretas, lo que limita la capacidad de sus resultados para influir directamente en la práctica. De igual forma, Gobov y Zuieva (2025) centran su aporte en la definición de atributos, sin desarrollar instrumentos para evaluarlos. Esta tendencia refleja una brecha entre la investigación conceptual y su aplicación operativa, un desafío recurrente en la ingeniería de software.

Otra observación clave es la dificultad de muchos modelos para integrarse en ciclos de desarrollo rápidos. Estudios como los de Klotins et al. (2022) o Terrón-Macías et al. (2025) proponen marcos útiles para decisiones estratégicas, pero que requieren análisis profundos o herramientas que no siempre están disponibles en ciclos cortos como los de integración continua. En entornos de alta velocidad, donde las decisiones deben tomarse rápidamente, estos modelos resultan difíciles de adoptar, a menos que se acompañen de automatización o adaptaciones específicas.

Un punto crítico también lo constituyen los atributos no funcionales, especialmente la mantenibilidad, la seguridad y la escalabilidad. Syed-Mohamad et al. (2025) y Olsson et al. (2022) coinciden en señalar que, a pesar de su importancia, estos atributos son difíciles de medir y rara vez se abordan con herramientas automáticas. Los modelos que intentan tratarlos suelen recurrir a análisis retrospectivos o enfoques cualitativos, como en el caso de Delgado Jojoa et al. (2024), quien vincula aspectos psicológicos del equipo con la calidad del producto, aunque sin pruebas técnicas de dicha correlación.

Por otro lado, algunos estudios logran un equilibrio entre conceptualización y validación empírica parcial. Es el caso de Koneru (2021), quien integra herramientas de análisis de seguridad en pipelines reales, aunque sin ofrecer una comparación sistemática de su impacto frente a métodos alternativos. Otro ejemplo es Domínguez-Acosta y García-Mireles (2021), cuyo análisis de actividades de calidad en entornos DevOps recoge prácticas reales mediante entrevistas, pero no incluye métricas técnicas de validación.

Finalmente, la matriz permite identificar una oportunidad de mejora común: la necesidad de estandarización. Muchos estudios aplican enfoques aislados, sin conectarse con taxonomías de calidad ya existentes o sin proponer marcos que puedan ser replicados en otros contextos. Esta dispersión dificulta la construcción de una base empírica robusta y compartida, que permita comparar resultados entre estudios o escalar buenas prácticas a nivel industrial.

En síntesis, la matriz comparativa pone de manifiesto que, si bien existe una amplia gama de propuestas para evaluar la calidad del software, la mayoría enfrenta limitaciones relacionadas con la falta de validación práctica, la dificultad de integración en contextos ágiles o DevOps, y la medición de atributos complejos. Los avances más prometedores provienen de estudios que combinan conceptualización rigurosa con experimentación controlada o aplicación en entornos reales. Para consolidar el campo, se requiere promover más estudios empíricos, desarrollar herramientas automatizadas adaptadas a ciclos rápidos, y establecer marcos comunes de evaluación que sean aplicables en distintas realidades organizacionales.

Discusión

Los hallazgos derivados de la revisión sistemática evidencian una diversidad estructurada de enfoques para la evaluación de la calidad del software, lo cual permite afirmar que el proceso investigativo logró identificar y clasificar métodos conforme al objetivo general planteado. Al

contrastar estos resultados con los referentes teóricos, se observa una evolución respecto a los modelos clásicos descritos por Callejas-Cuervo y Alarcón-Aldana (2017), particularmente en relación con estándares como ISO/IEC 25000, que priorizaban auditorías formales y documentación extensa. En comparación, los enfoques recientes, como los propuestos por Zhang y Zhang (2024), integran métricas automatizadas en entornos DevSecOps, favoreciendo evaluaciones continuas. Esta transición confirma que la literatura científica actual ofrece métodos diferenciables y contextualizados, aunque, como advierte Fox (2023), su adopción depende de condiciones tecnológicas que no siempre están disponibles en todos los entornos.

Desde la perspectiva del aseguramiento de la calidad, la revisión permitió analizar la coherencia entre trazabilidad y evaluación empírica, aspecto señalado por Vega Zepeda (2010) y reforzado por Díaz y Peralta Luján (2022) como componente esencial de control. Esta premisa encuentra continuidad en la propuesta de Koneru (2021), quien integra herramientas de análisis de seguridad en pipelines de CI/CD, lo cual ejemplifica cómo los métodos contemporáneos buscan validar la calidad dentro del propio ciclo de desarrollo. Sin embargo, la revisión también identificó limitaciones en la validación práctica de ciertas propuestas, como ocurre con Gobov y Zuieva (2025), cuya aplicación directa es restringida. En línea con lo advertido por Esterkin y Pons (2017), la ausencia de pruebas empíricas amplias dificulta determinar la aplicabilidad sostenida de algunos modelos.

El análisis comparativo igualmente permitió examinar la integración entre evaluación técnica y seguridad, cuestión desarrollada por Mellado Fernández et al. (2010) como requisito para una visión integral de calidad. Este planteamiento se materializa en propuestas recientes como la de Czekster (2024), donde la evaluación continua de riesgos se incorpora al ecosistema DevSecOps. No obstante, el contraste con modelos integrativos como el de Andrade de Casañas

(2006) revela que la sofisticación técnica puede reducir la accesibilidad para contextos con menor infraestructura. En consecuencia, la revisión no solo identificó métodos avanzados, sino que también determinó limitaciones asociadas a su aplicabilidad, cumpliendo así con el propósito de analizar críticamente su viabilidad en escenarios diversos.

En relación con los atributos no funcionales, la revisión permitió identificar vacíos persistentes en su medición objetiva, aspecto señalado por Castaño (2021) como uno de los desafíos estructurales de la ingeniería de software. Esta dificultad se refleja en los hallazgos empíricos de Syed-Mohamad et al. (2025), quienes reconocen la escasez de métricas automatizadas para evaluar propiedades como mantenibilidad o escalabilidad sin intervención humana. Asimismo, Delgado Jojoa et al. (2024) introducen variables psicológicas del equipo de desarrollo, ampliando la perspectiva hacia dimensiones humanas. Esta integración coincide con lo planteado por Serna (2011), quien advierte que la calidad no puede separarse del factor humano, aunque su medición represente un reto metodológico constante.

En cuanto al análisis metodológico, la revisión sistemática permitió identificar una predominancia de marcos ágiles y prácticas de integración continua, tendencia que Chipe Ortiz (2024) vincula con procesos de transformación digital. Sin embargo, el examen crítico de estudios como los de Terrón-Macías et al. (2025) y Klotins et al. (2022) evidenció dificultades de integración en ciclos de desarrollo acelerados. Esta situación contrasta con lo expuesto en Agile software development: Methodologies and trends (2020), donde se plantea que la agilidad debe acompañarse de adaptabilidad en los modelos de calidad. La revisión, por tanto, no solo clasificó metodologías, sino que también determinó tensiones entre evaluación y rapidez operativa, elemento central para valorar su aplicabilidad actual.

Desde el plano metodológico, el contraste con los criterios de Hernández Sampieri et al. (2014) permitió evaluar el rigor de los estudios incluidos en la revisión. Investigaciones como las de Rahman et al. (2024) y Zhang y Zhang (2024) presentan experimentación controlada con datos cuantificables, lo cual respalda su validez empírica. En contraste, trabajos como los de Poth et al. (2020) y Hanief et al. (2025) se apoyan en revisiones o percepciones, lo que limita la comprobación directa en campo. Esta heterogeneidad confirma lo señalado por Mellado Fernández et al. (2010) acerca de la inconsistencia en la aplicación de métodos empíricos, evidenciando que el objetivo general fue alcanzado al determinar niveles diferenciados de validación y madurez metodológica.

Una comparación adicional con la propuesta de Camacho Quintero (2023) permitió analizar la transversalidad de los métodos identificados, especialmente en contextos educativos. Mientras modelos industriales como los de Najmi y El-Dosuky (2025) priorizan aseguramiento técnico en entornos productivos, enfoques pedagógicos responden a dinámicas distintas. Este contraste reafirma lo planteado por Carvallo et al. (2008) respecto a la necesidad de modularidad en los modelos de evaluación. La revisión sistemática evidenció que muchos métodos contemporáneos presentan especialización contextual, lo cual limita su transferencia automática a otros dominios, cumpliendo así con el objetivo de identificar limitaciones que inciden en su aplicabilidad.

Por último, el marco de mejora continua descrito por Soto Durán y Reyes Gamboa (2007) sirvió como referente para evaluar si los métodos actuales integran retroalimentación permanente. Aunque Olivero et al. (2023) destacan el valor de los mapeos sistemáticos para apoyar decisiones en el ciclo de vida, la revisión mostró que la implementación de retroalimentación en tiempo real aún no es generalizada. Propuestas como las de Koneru (2021)

y Czekster (2024) apuntan en esa dirección, aunque su adopción es desigual. En consecuencia, el análisis confirma que el objetivo general fue cumplido, ya que la revisión permitió identificar métodos, analizar su validez empírica y determinar limitaciones estructurales que condicionan su aplicabilidad en contextos actuales de desarrollo.

Conclusiones

El objetivo general planteó realizar una revisión sistemática de la literatura científica para identificar, clasificar y analizar los métodos de evaluación de software, con el fin de determinar su validez empírica y las limitaciones que inciden en su aplicabilidad en los contextos actuales de desarrollo. Este propósito fue alcanzado mediante la selección, sistematización y análisis comparativo de dieciséis estudios publicados entre 2020 y 2025, lo que permitió construir una estructura clasificatoria coherente y evaluar el nivel de respaldo empírico de cada enfoque. La investigación no se limitó a describir propuestas, sino que examinó críticamente su grado de validación, su contexto de implementación y las restricciones que afectan su adopción. De esta manera, la monografía logró integrar identificación, análisis empírico y evaluación crítica en un marco interpretativo alineado con el objetivo central.

El primer objetivo específico consistió en identificar los métodos de evaluación de software descritos en fuentes científicas especializadas y clasificarlos según sus enfoques, características y contextos de aplicación. Este objetivo se cumplió mediante la elaboración de una matriz de clasificación que permitió agrupar los métodos en categorías como DevOps y DevSecOps, métricas automatizadas, metodologías ágiles y tradicionales, evaluación de requerimientos y gestión de deuda técnica. La sistematización evidenció que la evaluación de software no responde a un modelo único, sino a enfoques diferenciados que se articulan con el entorno de desarrollo. Asimismo, la clasificación permitió reconocer patrones comunes, como la tendencia hacia la automatización y la integración continua, estableciendo una base estructurada para el análisis posterior de efectividad y limitaciones.

El segundo objetivo específico buscó analizar la evidencia empírica asociada a la efectividad y validez de los métodos identificados. Este propósito se alcanzó mediante la

diferenciación de niveles de validación, distinguiendo estudios con experimentación controlada, investigaciones aplicadas en contextos reales, revisiones sistemáticas y propuestas de carácter conceptual. El análisis permitió determinar que algunos enfoques presentan respaldo empírico robusto basado en datos cuantificables, mientras otros aún se encuentran en fases exploratorias con validación limitada. Esta evaluación comparativa evidenció la heterogeneidad metodológica del campo y permitió establecer el grado de madurez de cada propuesta. En consecuencia, la investigación no solo identificó métodos, sino que examinó críticamente la solidez de la evidencia que respalda su efectividad.

El tercer objetivo específico se orientó a analizar críticamente las fortalezas, limitaciones y vacíos de los métodos de evaluación de software, con el fin de identificar tendencias y proponer mejoras. Este objetivo se cumplió al identificar fortalezas como la integración de métricas en pipelines CI/CD, la automatización de procesos y la detección temprana de defectos, así como limitaciones relacionadas con la escasa validación en entornos industriales amplios, la dificultad de integración en ciclos ágiles acelerados y la debilidad en la medición de atributos no funcionales. Además, se evidenció un vacío persistente en la estandarización de métricas y en la transversalidad de los modelos hacia contextos educativos o públicos. El análisis permitió, por tanto, delinear líneas de mejora orientadas a fortalecer la validación empírica, la adaptabilidad contextual y la medición integral de calidad.

Recomendaciones

1. Se requiere un aumento en estudios que prueben los modelos propuestos en entornos reales, especialmente en organizaciones de distinto tamaño y sector. Esto permitirá mejorar la generalización y transferibilidad de los resultados.
2. Es necesario que los modelos de evaluación se diseñen con flexibilidad para integrarse a diferentes marcos de desarrollo, incluyendo Agile, DevOps, e incluso procesos educativos y gubernamentales.
3. Se recomienda el desarrollo de herramientas automatizadas que midan de forma más precisa y objetiva aspectos como mantenibilidad, escalabilidad y usabilidad, sin depender exclusivamente de juicios expertos o encuestas.
4. Se sugiere explorar más profundamente el impacto de las competencias, actitudes y organización de los equipos de desarrollo sobre la calidad del software, considerando variables psicológicas, motivacionales y culturales.
5. Los modelos propuestos deben buscar compatibilidad con normativas existentes como ISO/IEC 25000, de manera que su adopción sea más sencilla en entornos normativos exigentes.
6. Es clave que las investigaciones incluyan datos abiertos, herramientas reproducibles y documentación clara para facilitar su replicación, adaptación y evolución en nuevas investigaciones.

Referencias Bibliográficas

Agile software development: Methodologies and trends. (2020). *International Journal of Interactive Mobile Technologies (iJIM), 14*(25), 121–139. https://www.su.edu.om/sujsb/index.php/ejournal/article/download/3/11/120

Andrade de Casañas, M. (2006). Modelo de evaluación para determinar la calidad de software de apoyo a la enseñanza de la física, basado en la concepción integrativo-adaptativa. *Revista Mexicana de Física, 52*, 26–27. http://www.redalyc.org/articulo.oa?id=57019196006

Briano, J. (Comp.). (2025). *Ingeniería de software*. Universidad de Buenos Aires. https://bibliotecadigital.econ.uba.ar/download/libros/Briano_compilacion_apuntes.pdf

Callejas-Cuervo, M., & Alarcón-Aldana, A. C. (2017). Modelos de calidad del software: Un estado del arte. *Entramado, 13*(1), 15–32. https://doi.org/10.18041/entramado.2017v13n1.25125

Camacho Quintero, C. L. (2023). Metodología para la evaluación del software educativo: Una visión desde la formación a la praxis. *Revista Scientific, 8*(27), 62–80. https://doi.org/10.29394/Scientific.issn.2542-2987.2023.8.27.3.62-80

Campoverde-Molina, M., Álvarez, A., Bravo-Torres, J., & Vinueza, S. (2021). Systematic review of literature on software architecture of educational websites. *IET Software, 15*(6), 453–470. https://digital-library.theiet.org/doi/full/10.1049/sfw2.12024

Cárdenas Castellanos, M. J. (2017). *Evaluación de metodologías de desarrollo de software utilizadas para valorar el impacto de un proyecto de software durante un semestre de clases* [Tesis de maestría]. Universidad Autónoma de Bucaramanga. https://repository.unab.edu.co/handle/20.500.12749/3412

Carvallo, J. P., Franch, X., & Quer, C. (2008). *Calidad de componentes software* [Informe técnico]. Universidad Politécnica de Cataluña. https://upcommons.upc.edu/entities/publication/28ac9c87-5d67-4264-82a0-35597a9b7d71

Castaño, J. F. (2021). Métricas en la evaluación de la calidad del software: Una revisión conceptual. *Computer and Electronic Sciences: Theory and Applications, 2*(2), 21–26. https://doi.org/10.17981/cesta.02.02.2021.03

Czekster, R. M. (2024). Continuous risk assessment in secure DevOps. *arXiv*. https://doi.org/10.48550/arXiv.2409.03405

Chipe Ortiz, W. B. (2024). *Metodologías y prácticas de desarrollo de software en el contexto de la transformación digital* [Trabajo de titulación, Universidad Politécnica Salesiana]. UPS. https://dspace.ups.edu.ec/bitstream/123456789/28125/1/UPS-GT005506.pdf

Delgado Jojoa, J. D., Revelo Sánchez, O., & Vallejo Chamorro, S. (2024). *Efectos de la meticulosidad como dimensión de la personalidad en los atributos de mantenibilidad y funcionalidad del software en equipos de desarrollo* [Tesis de maestría, Universidad de Nariño]. SIRED. https://sired.udenar.edu.co/15839/

Díaz, Y. M. S., & Peralta Luján, J. L. (2022). Modelo de medición y evaluación de calidad del software basado en la norma ISO/IEC 25000 para medir la usabilidad en productos de software académicos universitarios. *TecnoHumanismo, 2*(4), 181–204. https://doi.org/10.53673/th.v2i4.125

Domínguez-Acosta, M. F., & Garcia-Mireles, G. A. (2021). *Identificación de actividades relacionadas con la calidad de software en entornos de DevOps*. Universidad de Sonora.

https://investigadores.unison.mx/es/publications/identificaci%C3%B3n-de-actividades-relacionadas-con-la-calidad-de-soft/

Esterkin, V., & Pons, C. (2017). Evaluación de calidad en el desarrollo de software dirigido por modelos. *Ingeniare. Revista Chilena de Ingeniería*, *25*(3), 449–463. http://www.redalyc.org/articulo.oa?id=77252700008

Femmer, H., Méndez Fernández, D., Wagner, S., & Eder, S. (2017). Rapid quality assurance with requirements smells. *Journal of Systems and Software, 131*, 295–313. https://doi.org/10.1016/j.jss.2016.02.047

Fox, J. (2023). The evolution of software development from SDLC to SSDLC. *Cobalt Blog*. https://www.cobalt.io/blog/evolution-of-software-development-from-sdlc-to-ssdlc

Giraldo-Hurtado, J. S. (2023). *Prueba de concepto basada en herramientas Cloud Native para el desarrollo de software* [Trabajo de fin de máster, Universidad Internacional de La Rioja]. https://reunir.unir.net/bitstream/handle/123456789/15912/Giraldo-Hurtado%2C%20Johan%20Sebastian.pdf

Gobov, D., & Zuieva, O. (2025). Software quality attributes in requirements engineering. *International Journal of Information Technology and Computer Science, 17*(4), 38–48. https://doi.org/10.5815/ijitcs.2025.04.04

Haindl, P. (2021). *Measuring and evaluating the value of software features in DevOps* [Doctoral dissertation, Johannes Kepler University Linz]. epub.jku.at. https://epub.jku.at/download/pdf/6620538.pdf

Hanief, S., Sutikno, T., & Riadi, I. (2025). Systematic literature review on software quality using agile approach. *International Journal of Informatics and Computation, 7*(2), Article 205. https://doi.org/10.35842/ijicom.v7i2.205

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. del P. (2014). *Metodología de la investigación* (6.ª ed.). McGraw-Hill. https://apiperiodico.jalisco.gob.mx/api/sites/periodicooficial.jalisco.gob.mx/files/metodologia_de_la_investigacion_-_roberto_hernandez_sampieri.pdf

Hiberus Tecnología. (2025, abril 18). *Estándares de calidad del software*. https://www.hiberus.com/crecemos-contigo/los-estandares-de-calidad-del-software-mas-importantes/

Jafari, S. M., & Motlagh, S. M. (2021). A systematic literature review of the technical criteria for software development methodologies. *CSIT Proceedings*. https://csit.am/2021/proceedings/ITA/ITA_p2.pdf

Jústiz-Núñez, D., Gómez-Suárez, D., Delgado-Dapena, M. D., & Gómez, M. (2014). Proceso de pruebas para productos de software en un laboratorio de calidad. *Ingeniería Industrial, 35*(2), 131–145. http://scielo.sld.cu/pdf/ii/v35n2/ii03214.pdf

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering* (EBSE Technical Report). Keele University and Durham University. https://www.researchgate.net/publication/302924724_Guidelines_for_performing_Systematic_Literature_Reviews_in_Software_Engineering

Klotins, E., Unterkalmsteiner, M., & Gorschek, T. (2022). Towards cost–benefit evaluation for continuous software engineering activities. *Empirical Software Engineering*, *27*(157). https://doi.org/10.1007/s10664-022-10127-4

Koneru, N. M. K. (2021). Integrating security into CI/CD pipelines: A DevSecOps approach with SAST, DAST, and SCA tools. *International Journal of Science and Research Archive, 3*(1), 250–265. https://doi.org/10.30574/ijsra.2021.3.1.0080

Mellado Fernández, D., Rodríguez, M., Verdugo, J., & Piattini, M., & Fernández-Medina, E. (2010). *Evaluación de la calidad y seguridad en productos software* [Documento técnico]. Ministerio de Administración Electrónica, España. https://administracionelectronica.gob.es/pae_Home/dam/jcr:4bda2362-75fb-4668-9437-0afd55aad0e5/28inciativas-legales.pdf

Mera Paz, J. A. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería Solidaria, 12*(20), 163–176. https://doi.org/10.16925/in.v12i20.1482

Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios. (2020). Universidad del Valle. https://bibliotecadigital.univalle.edu.co/bitstream/10893/19567/1/9702-V473.pdf

Najmi, A., & El-Dosuky, M. (2025, June 27). Enhancing software quality through early defect identification. *SSRN*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5325973

Olivero, N. P., George, H. Á., & García-Mireles, G. A. (2023). Impact of DevOps practices on software product quality: Preliminary findings from a systematic mapping. *Proceedings of the 2023 12th International Conference on Software Process Improvement (CIMPS).* https://doi.org/10.1109/CIMPS61233.2023.10528820

Olsson, T., Sentilles, S., & Papatheocharous, E. (2022). A systematic literature review of empirical research on quality requirements. *Requirements Engineering, 27*, 249–271. https://doi.org/10.1007/s00766-021-00361-5

OpenWebinars. (2025, enero 29). *Gestión de la calidad en el desarrollo de software*. https://openwebinars.net/blog/gestion-de-la-calidad-en-el-desarrollo-de-software/

Pacienzia, J. (2015). *Metodologías de desarrollo de software: Evaluación de calidad y aseguramiento en proyectos de software* [Tesis de licenciatura]. Universidad Católica Argentina. https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Poth, A., Kottke, M., & Riel, A. (2020). Evaluation of agile team work quality. En R. Hoda (Ed.), *Agile processes in software engineering and extreme programming – Workshops* (pp. 101–110). Springer. https://doi.org/10.1007/978-3-030-58858-8_11

Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach (9th ed.)*. McGraw-Hill. https://www.researchgate.net/publication/365946272_Software_Engineering_A_Practitioner's_Approach_9_th_Edition

QuestionPro. (2023). *Evaluación de software: Qué es y cómo realizarlo*. https://www.questionpro.com/blog/es/evaluacion-de-software/

Rahman, A., Cysneiros, L. M., & Berry, D. M. (2024, April). An empirical study of the impact of Waterfall and Agile methods on numbers of requirements-related defects. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing* (pp. 1143–1152). ACM. https://doi.org/10.1145/3605098.3635901

Rodríguez Sinisterra, M. G., Quimis Peñafiel, N. J., Gonzalez Baque, Y. P., & Merino Marcillo, J. M. (2021). Calidad del software aplicado a la evaluación de la seguridad en sistemas de información. *UNESUM Ciencias, 5*(4), Artículo 411. https://doi.org/10.47230/unesum-ciencias.v5.n4.2021.411

Serna, E. (2011). Prueba del software: Más que una fase en el ciclo de vida. *Ingeniería y Desarrollo, 29*(3), 120–137. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-49932011000300006

Software development methodologies and practices in startups. (2018). *Master's Thesis*, University of Oulu. https://oulurepo.oulu.fi/bitstream/handle/10024/11894/nbnfioulu-201804041421.pdf

Soto Durán, D. E., & Reyes Gamboa, A. X. (2007). Introducción a los modelos de calidad del software basados en procesos. *Cuaderno Activa, 1*(1), 25–36. https://ojs.tdea.edu.co/index.php/cuadernoactiva/article/view/71

Syed-Mohamad, S. M., Ngah, A., Ali, A.-F. M., & Keikhosrokiani, P. (2025). Measuring software maintainability: An exploration of metrics and continuous practices. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *63*(2), 181–195. https://doi.org/10.37934/araset.63.2.181195

Tello-Leal, E., & Marín, C. (2012). Revisión de los sistemas de control de versiones utilizando técnicas de grafos. *Ingeniería USBMed, 3*(1), 73–84. https://revistas.usb.edu.co/index.php/IngUSBmed/article/download/267/181/837

Terrón-Macías, V., Mejia, J., & Terrón-Hernández, M. (2025). A systematic literature review on technical debt in software development: Types, tools, and concerns. *International Journal of Combinatorial Optimization Problems and Informatics, 16*(4). https://doi.org/10.61467/2007.1558.2025.v16i4.1150

UNADM. (2025). *Desarrollo de software. Unidad 1: Arquitectura de software*. Universidad Abierta y a Distancia de México.

https://dmd.unadmexico.mx/contenidos/DCEIT/BLOQUE2/DS/04/DDRS/U1/descargables/DDRS_U1_Contenido.pdf

Valencia-Arias, A., Benjumea-Arias, M., & Castaño, J. (2024). Tendencias investigativas en el uso de Cloud Computing. *Revista ESPACIOS*, *45*(6). https://www.redalyc.org/journal/1942/194277606012/html/

Vega Zepeda, V. R. (2010). *Metodología para el aseguramiento de la calidad en la adquisición del software (proceso y producto) y servicios correlacionados (MACAD-PP)* [Tesis doctoral]. Universidad Politécnica de Madrid. https://oa.upm.es/14955/

Velasco, J., Moreno, M., & Pérez, F. (2025). Herramientas de inteligencia artificial usadas en el desarrollo de software: Una revisión sistemática de la literatura. *Revista Científica Multidisciplinar G-nerando*, *3*(2), 1–25. https://revista.gnerando.org/revista/index.php/RCMG/article/download/586/616/2649

Zhang, J. Y., & Zhang, Y. (2024). Quantitative DevSecOps metrics for cloud-based web microservices. *IEEE Access*, *12*, 1–1. https://doi.org/10.1109/ACCESS.2024.3486314

Anexos

Anexo 1. Clasificación de Fuentes según Enfoque Metodológico y Tipo de Validación

N°	Fuente	Enfoque metodológico	Tipo de validación
1	Zhang & Zhang (2024)	DevSecOps + métricas	Experimento controlado
2	Haindl (2021)	Valor de funcionalidades	Estudio de caso
3	Najmi & El-Dosuky (2025)	Identificación temprana de defectos	Simulación
4	Hanief et al. (2025)	Calidad en metodologías ágiles	Revisión sistemática
5	Olsson et al. (2022)	Requerimientos de calidad	Metaanálisis
6	Gobov & Zuieva (2025)	Atributos de calidad en ingeniería de req.	Revisión teórica
7	Olivero et al. (2023)	Prácticas DevOps	Mapeo sistemático
8	Terrón-Macías et al. (2025)	Deuda técnica	Revisión sistemática
9	Syed-Mohamad et al. (2025)	Mantenibilidad	Estudio exploratorio
10	Rahman et al. (2024)	Comparación metodologías	Empírico con datos
11	Domínguez-Acosta & Garcia-Mireles (2021)	Actividades DevOps	Estudio exploratorio
12	Delgado Jojoa et al. (2024)	Psicología y calidad	Tesis con casos
13	Poth et al. (2020)	Trabajo en equipo ágil	Experimentos
14	Koneru (2021)	Integración seguridad CI/CD	Estudio de herramientas
15	Klotins et al. (2022)	Costo-beneficio en ingeniería continua	Estudio empírico
16	Czekster (2024)	Evaluación de riesgo en DevSecOps	Modelo conceptual aplicado

Nota. Elaboración propia basado en cada referente mencionado en la tabla.