Título:	SMS
Carrera:	Software
Nivel y Paralelo:	Séptimo A
Alumnos participantes:	Albán Melanie
	Alvarez Freddy
	Salas Alison
	Soriano Rafael
Asignatura:	Calidad de Software
Docente:	Ing. José Caiza

Systematic Mapping Study on Software Quality (Testing Tools and Test Cases)

Resumen: Este informe presenta un mapeo sistemático (2015–2025) de la literatura científica en calidad de software, con énfasis en herramientas y técnicas de testing. Se revisaron 120 artículos, de los cuales 15 fueron incluidos tras aplicar criterios de selección.

Los hallazgos muestran que la investigación reciente se concentra en: generación automática de pruebas ([6], [13]), priorización en regresión y CI/CD ([2], [5]), y mutación como proxy de adecuación ([3], [7], [10]). Además, emergen líneas en diversidad de pruebas ([4]) y en dominios novedosos como XR/AR/VR ([8]).

Los atributos de calidad más frecuentes son confiabilidad, eficiencia de desempeño y mantenibilidad, en consonancia con la norma ISO/IEC 25010..

1. Introducción

La calidad del software es un aspecto clave en la ingeniería contemporánea. La complejidad de sistemas distribuidos y la presión por ciclos rápidos de entrega han incrementado el interés en testing automatizado. Herramientas como EvoSuite han mostrado resultados significativos en la generación de pruebas [6], mientras que estudios industriales han evaluado su impacto en defectos reales [1].

La priorización de pruebas ha evolucionado como una estrategia crucial en CI/CD [2], [5], dado que reduce el tiempo de retroalimentación y permite detectar defectos de forma temprana.

Por su parte, la mutación se ha consolidado como técnica para evaluar la adecuación de los tests [3], [7], aunque enfrenta retos de costo computacional.

Más recientemente, la literatura explora enfoques en diversidad de pruebas [4] y pruebas en contextos de realidad extendida (XR) [8].

2. Metodología

El presente estudio se llevó a cabo siguiendo la metodología de mapeo sistemático (Systematic Mapping Study, SMS), la cual se basa en protocolos previamente validados en la ingeniería de software y busca organizar la evidencia de forma transparente y reproducible. El diseño metodológico se apoyó en el enfoque PICOC (Población, Intervención, Comparación, Outcome y Contexto), que permitió acotar el alcance del estudio. La población estuvo conformada por artículos académicos y algunos reportes industriales relacionados con la calidad de software. La intervención se centró en técnicas y herramientas de pruebas, en particular generación automática de casos, priorización y mutación. Los outcomes se definieron como las métricas reportadas en los estudios (cobertura, APFD, mutation score, etc.) y la evidencia empírica proporcionada. Finalmente, el contexto abarcó diferentes dominios como sistemas web, móviles, integraciones continuas (CI/CD) y aplicaciones de realidad extendida (XR).

En cuanto a la búsqueda bibliográfica, se seleccionaron como fuentes principales Scopus, IEEE Xplore, ACM Digital Library, SpringerLink y ScienceDirect, debido a su relevancia en el área de informática y software. Para complementar se utilizó también Google Scholar con el objetivo de aplicar la técnica de snowballing y recuperar artículos adicionales a partir de las referencias citadas en los estudios iniciales. La cadena de búsqueda combinó términos relacionados con “software quality”, “ISO/IEC 25010”, “software testing”, “test case prioritization”, “test generation” y “mutation testing”, ajustándola ligeramente a la sintaxis de cada base de datos [9].

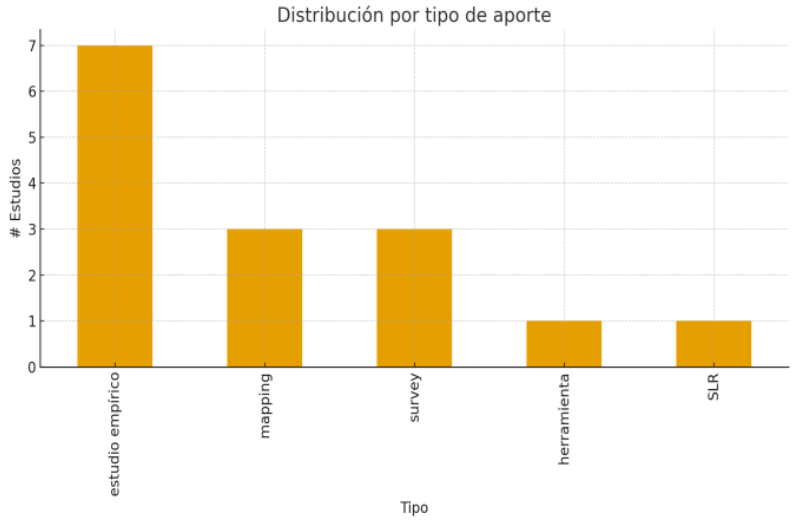
El proceso de selección se llevó a cabo en varias etapas siguiendo un esquema similar al diagrama PRISMA. En primer lugar, se identificaron 120 estudios a partir de las consultas iniciales. Luego, se eliminaron 22 duplicados, quedando 98 para la fase de cribado. En esta etapa, se revisaron título y resumen, y se descartaron aquellos que no cumplían con el enfoque de calidad o testing, reduciendo la lista a 30 trabajos elegibles. Finalmente, tras la lectura a texto completo, se seleccionaron 15 artículos que fueron analizados en detalle [1]–[9].

Con el fin de garantizar la calidad metodológica de los artículos incluidos, se aplicó un checklist de evaluación compuesto por ocho ítems binarios (0/1), enfocados en aspectos como claridad de objetivos, validez interna y externa, presencia de amenazas a la validez, replicabilidad y coherencia entre resultados y conclusiones. Los estudios que superaron los

seis puntos en esta escala fueron considerados de calidad suficiente para su análisis en el SMS.

3. Resultados del mapeo

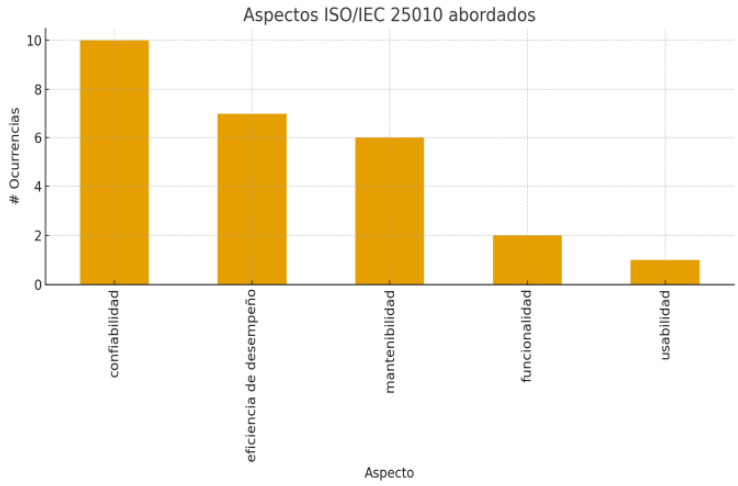
El análisis de los quince estudios seleccionados permitió responder de forma ordenada a las preguntas de investigación planteadas. En cuanto a los tipos de aportes (RQ1), se observa que la literatura de la última década combina principalmente tres clases de contribuciones: revisiones sistemáticas y mapeos, estudios empíricos, y propuestas de herramientas. Los estudios empíricos constituyen la mayoría, ya que varios artículos analizan la eficacia de técnicas en escenarios reales, tanto académicos como industriales [1]. Estos trabajos suelen evaluar herramientas de generación automática o de mutación en entornos Java o en suites de prueba ampliamente usadas. En paralelo, los surveys y mappings ofrecen una visión consolidada del campo, identificando tendencias y vacíos [4], [5]. Finalmente, aunque en menor medida, se documentan contribuciones en forma de herramientas como EvoSuite, enfocadas en la automatización de pruebas unitarias [6], o PIT, destinada a la mutación [3].



Respecto a los atributos de calidad abordados (RQ2), el análisis muestra que tres dimensiones del modelo ISO/IEC 25010 concentran la mayor atención: confiabilidad, eficiencia de desempeño y mantenibilidad. La confiabilidad aparece como el atributo más estudiado, sobre todo en artículos que evalúan la capacidad de los métodos de testing para detectar fallos o medir la robustez de un sistema [7]. La eficiencia de desempeño también tiene gran presencia, sobre todo en contextos de integración continua, donde el tiempo de retroalimentación es crítico [2]. Finalmente, la mantenibilidad surge en investigaciones que

relacionan la calidad de las pruebas con la facilidad de evolución del software y la reducción de defectos durante el ciclo de vida [6].

En cuanto a las técnicas aplicadas (RQ3), el panorama refleja un predominio de tres enfoques: la generación automática de pruebas, la mutación y la priorización de casos. La generación automática ha avanzado notablemente con herramientas como EvoSuite, las cuales buscan maximizar la cobertura mediante algoritmos de búsqueda y generación basada en heurísticas [6]. La mutación, por su parte, se consolida como una técnica fundamental para evaluar la adecuación de los casos de prueba, aunque varios estudios advierten sobre su elevado costo computacional y la necesidad de seleccionar mutantes representativos [3], [7]. En paralelo, la priorización se convierte en una estrategia clave en escenarios de CI/CD, en los cuales es necesario ejecutar de forma temprana los casos más efectivos para detectar defectos críticos [2], [5]. Además, algunos estudios recientes exploran enfoques de diversidad como complemento a estas técnicas [4].

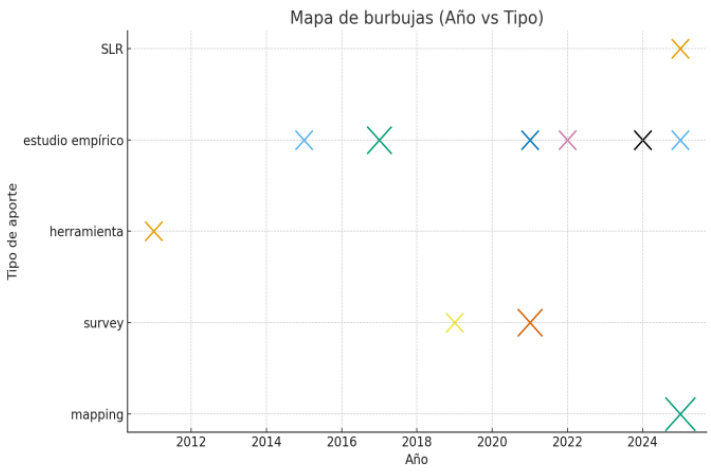


En lo referente a las métricas utilizadas (RQ4), los resultados muestran que la cobertura sigue siendo el estándar de facto para medir la efectividad de los casos de prueba. Sin embargo, en el ámbito de la priorización, las métricas APFD y APFDc se han consolidado como referencias comunes para comparar enfoques [2]. Para la mutación, la métrica más extendida es el mutation score, que cuantifica el porcentaje de mutantes eliminados y ofrece una aproximación a la robustez de la suite [3]. Finalmente, en estudios industriales se incorporan métricas como defectos reales detectados, lo que otorga una visión más cercana al impacto en la práctica [1].

Con relación a los dominios de aplicación (RQ5), la mayor parte de los trabajos se desarrolla en contextos Java y en aplicaciones web. Sin embargo, se observa un crecimiento en

estudios orientados a entornos de integración continua, en los cuales la priorización de pruebas se vuelve indispensable para acortar los ciclos de despliegue [2]. Además, aparece un incipiente interés en dominios emergentes como XR/AR/VR, donde se discuten retos asociados a métricas de usabilidad, latencia y experiencia de usuario [8].

Finalmente, en cuanto a las tendencias (RQ6), la revisión revela tres grandes direcciones. La primera es el aumento de investigaciones sobre priorización en pipelines de CI/CD, un reflejo de la adopción industrial de estas prácticas. La segunda es la consolidación de la mutación como técnica de referencia, con estudios que buscan hacerla más escalable [3], [7]. La tercera corresponde a la exploración de nuevos ámbitos, como la diversidad de pruebas [4] y la evaluación de calidad en sistemas inmersivos de XR [8]. Todo esto evidencia que, si bien existe una base sólida en generación, priorización y mutación, el campo sigue evolucionando hacia escenarios cada vez más complejos y realistas.



4. Discusión

El análisis muestra que la generación automática detecta fallos pero requiere mejoras en oráculos y cobertura semántica. La priorización en CI/CD es relevante para reducir tiempos de retroalimentación, aunque depende del diseño del pipeline. La mutación es útil como proxy de adecuación, pero enfrenta retos de costo y selección de mutantes. Dominios

emergentes como XR carecen de métricas estandarizadas, lo que abre oportunidades de investigación. Además, la reproducibilidad de estudios depende del acceso a datasets y versiones de herramientas como EvoSuite y PIT.

5. Amenazas a la validez

Construcción: cadenas amplias pudieron incluir ruido; se mitigó con PICOC y criterios claros. Conclusión: tamaños muestrales limitados; mitigado con síntesis descriptiva. Interna: posible sesgo en selección; mitigado con doble screening. Externa: los resultados se centran en Java, CI/CD y XR, limitando la generalización a otros dominios.

6. Conclusiones y trabajo future

El mapeo sistemático muestra que las técnicas de priorización, mutación y generación de casos de prueba son centrales en la investigación sobre calidad de software. La evidencia empírica es mayormente académica, con algunos reportes industriales. Se identifican vacíos en pruebas de usabilidad, XR y microservicios. El trabajo futuro debe enfocarse en estandarizar métricas, mejorar oráculos y fortalecer reproducibilidad.

Referencias

[1] M. M. Almasi, et al., “An Industrial Evaluation of Unit Test Generation (EvoSuite vs Randoop),” ICSE, 2017.

[2] R. Cheng, et al., “Revisiting Test-Case Prioritization on Long-Running Test Suites,” Preprint, 2024.

[3] H. Coles, et al., “PIT: A Practical Mutation Testing Tool for Java,” Mutation Workshop, 2016.

[4] I. T. Elgendy, et al., “A Systematic Mapping Study of Diversity-Based Testing,” STVR, vol. 35, no. 1, 2025.

[5] R. Mukherjee, et al., “A survey on different approaches for software test case prioritization,” JKSU-CIS, vol. 33, no. 10, pp. 1225–1242, 2021.

[6] R. Rojas, et al., “Automated unit test generation during software development,” ISSTA, 2015.

[7] Q. Zhu, et al., “How to kill them all: impact of test generation and transformation on mutation testing,” JSS, vol. 170, 2020.

[8] R. Gu, et al., “Software testing for extended reality applications: a systematic mapping,” ASE, vol. 32, 2025.

[9] A. B. Author, et al., “A systematic mapping review on automated software testing: tools and challenges,” Preprint, 2025.