SAN IGNACIO DE LOYOLA – ESCUELA ISIL

TÍTULO DE LA INVESTIGACIÓN

“La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú en 2025”

TRABAJO DE INVESTIGACIÓN PARA OPTAR EL TÍTULO PROFESIONAL DE

Bachiller en Dirección de Tecnologías de la información

PRESENTADO POR:

Nolasco Huamanchumo, Cesar Raul - Dirección de Tecnologías de la Información

ASESOR

Pelaez Valdivieso, Jose Víctor

LIMA, PERÚ

2025

ASESOR Y MIEMBROS DEL JURADO

ASESOR

Peláez Valdivieso, Jose Víctor

MIEMBROS DEL JURADO

Chávez Soriano, Silvia Nicolasa

Chupillón Barreto, Rosemary

Rojas Aguilar, Claudio Sergio

DECLARACIÓN JURADA DE ORIGINALIDAD

Yo, Cesar Raul Nolasco Huamanchumo, identificado con DNI N° 46174717 perteneciente al programa Dirección de Tecnologías de la Información, siendo mi asesor Sr. José Víctor Peláez Valdivieso, identificado con DNI N°: 18161446, y cuyo código ORCID es 0000-0002-2186-0398.

DECLARO BAJO JURAMENTO QUE:

a) Soy el autor del documento académico titulado “La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú en 2025”.

b) El trabajo de investigación es original y no ha sido difundido en ningún medio académico; por lo tanto, sus resultados son veraces y no es copia de ningún otro.

c) El asesor ha revisado minuciosamente el proyecto de investigación, incluyendo las citas a otros autores y las referencias bibliográficas. Este proceso se ha llevado a cabo cumpliendo con las pautas académicas y respetando las normas internacionales.

d) El trabajo de investigación cumplió con el análisis del sistema TURNITIN, el cual tiene el 10% de similitud.

e) Declaro conocer las consecuencias legales y/o administrativas que puedan derivar si se verifica la falsedad total o parcial de la presente declaración, de acuerdo con lo previsto en el artículo 411 del código penal y el numeral 34.3 del artículo 34 del Texto Único Ordenado de la Ley del Procedimiento Administrativo General, aprobado por Decreto Supremo 004-2019-JUS.

Fecha: 11/12/2025

Firma del autor

Huella

Firma del asesor

Huella

Índice temático

Índice de tablas

Índice de figuras

RESUMEN

La investigación tuvo como propósito comprender e interpretar los factores técnicos, temporales y humano–organizacionales que incidieron en la ausencia de pruebas unitarias en proyectos de desarrollo de software en Lima, Perú, en 2025, así como sus efectos percibidos en la calidad del producto y en la coordinación del trabajo entre roles. Se empleó un enfoque cualitativo, de tipo básico, con diseño no experimental y transeccional. La muestra estuvo conformada por tres informantes clave (un project manager y dos desarrolladores senior de organizaciones con características distintas), seleccionados mediante un muestreo no probabilístico de tipo intencional por criterios, priorizando su experiencia en desarrollo de software y su rol en la toma de decisiones técnicas. La técnica principal de recolección de datos fue la entrevista semiestructurada y el instrumento una guía de entrevista elaborada a partir de una matriz de categorización apriorística basada en las categorías “ausencia de pruebas unitarias” y “desarrollo de software”. Las entrevistas fueron grabadas, transcritas íntegramente y analizadas mediante codificación cualitativa, triangulación y apoyos como el diagrama de Ishikawa, el mapa de co-ocurrencias de códigos y el diagrama de Pareto. Los resultados mostraron que la ausencia de pruebas unitarias no respondió únicamente a limitaciones técnicas, sino a la combinación de presión por plazos, estimaciones ajustadas, cultura de priorizar entregas funcionales, falta de estándares y Definition of Done explícita, escasa formación específica en pruebas y decisiones organizacionales que relegan la calidad estructural. Se concluyó que esta ausencia se tradujo en detección tardía de defectos, retrabajo recurrente, acumulación de deuda técnica, sobrecarga del área de calidad y variabilidad en la calidad del software, lo que evidencia la necesidad de formalizar prácticas de pruebas unitarias en el proceso de desarrollo.

Palabras claves: Diseño de sistemas, Programación informática, Programa de ordenador, Perú.

ABSTRACT

This research aimed to understand and interpret the technical, temporal, and human–organizational factors that influenced the absence of unit tests in software development projects in Lima, Peru, in 2025, as well as their perceived effects on product quality and work coordination between roles. A qualitative approach was used, with a basic, non-experimental, cross-sectional design. The sample consisted of three key informants (one project manager and two senior developers from organizations with different characteristics), selected through non-probabilistic purposive sampling based on key informant criteria. The main data collection technique was the semi-structured interview, and the instrument was an interview guide developed from an a priori categorization matrix based on the categories “absence of unit tests” and “software development”. The interviews were audio-recorded, fully transcribed, and analyzed through qualitative coding, triangulation, and support tools such as an Ishikawa diagram, a code co-occurrence map, and a Pareto diagram. The results showed that the absence of unit tests did not stem solely from technical limitations, but from a combination of deadline pressure, tight estimates, a culture focused on delivering functional features, lack of standards and an explicit Definition of Done, limited training in testing, and organizational decisions that deprioritize structural quality. It was concluded that this absence led to late defect detection, recurrent rework, accumulation of technical debt, overload of the quality area, and variability in software quality, highlighting the need to formalize unit testing practices within the development process.

Keywords: System design, Computer programming, Computer software, Peru.

Introducción

El desarrollo de software se ha convertido en un componente crítico para la operación de organizaciones públicas y privadas. La calidad del software no solo implica que “no falle” frente al usuario, sino también que sea mantenible, confiable y capaz de evolucionar con bajo riesgo, tal como plantea el modelo de calidad ISO/IEC 25010 (ISO, 2023). En este marco, las pruebas unitarias cumplen un rol clave al permitir detectar defectos en etapas tempranas, sobre unidades pequeñas de código, antes de su integración a componentes más complejos (Laura Mamani, 2023; Rojas Illanes, 2023). No obstante, diversos estudios han mostrado que la presencia real de pruebas unitarias en proyectos de software es limitada y que muchos equipos continúan apoyándose principalmente en pruebas manuales tardías o validaciones funcionales (Díaz Figueredo et al., 2021; Trautsch et al., 2017; Wang et al., 2024).

En el contexto de Lima, esta brecha entre lo que recomiendan los marcos de calidad y lo que se practica en los equipos de desarrollo se traduce en fallas que llegan a ambientes de prueba o producción, ciclos de retrabajo, deuda técnica y mayor presión sobre las áreas de aseguramiento de la calidad y los usuarios finales. Antecedentes en organizaciones públicas y privadas muestran que cuando se formalizan los procesos de prueba y se integran actividades como pruebas de unidad, planes de prueba y automatización, se reduce la cantidad de no conformidades y se mejora la detección temprana de errores (Díaz Figueredo et al., 2021; Cabrera et al., 2024; Quispe Gutiérrez, 2023; Paz Díaz, 2023; Espinoza, 2025). La persistencia de equipos que trabajan sin pruebas unitarias plantea, por tanto, un problema relevante para la mejora de procesos de desarrollo de software en la ciudad.

En este contexto, la presente investigación tuvo como objetivo general comprender e interpretar los factores técnicos, temporales y humano–organizacionales que inciden en la ausencia de pruebas unitarias en proyectos de desarrollo de software en Lima, Perú, así como sus efectos percibidos en la calidad del producto y en la coordinación entre roles. Para ello se plantearon dos objetivos específicos: (a) comprender las razones, creencias y barreras que sostienen la no adopción de pruebas unitarias en los equipos de desarrollo, y (b) explorar

cómo la ausencia de estas pruebas incide en el flujo de desarrollo, el retrabajo, la deuda técnica y la coordinación entre desarrolladores, líderes técnicos y QA.

La investigación se desarrolló con un enfoque cualitativo, de tipo básico, diseño no experimental y transeccional. Se trabajó con una muestra intencional de tres informantes clave (un project manager y dos desarrolladores senior) pertenecientes a organizaciones de distinto tamaño y alcance. La técnica principal de recolección de datos fue la entrevista semiestructurada y el instrumento una guía de entrevista diseñada a partir de una matriz de categorización apriorística basada en dos grandes categorías: ausencia de pruebas unitarias y desarrollo de software. Las entrevistas fueron grabadas, transcritas íntegramente y analizadas mediante codificación cualitativa y triangulación con apoyos como el diagrama de Ishikawa, el mapa de co-ocurrencias de códigos y el diagrama de Pareto.

El documento se organiza de la siguiente manera: el Capítulo I presenta la información general del proyecto, el problema, los objetivos y las justificaciones. El Capítulo II desarrolla el marco referencial, que incluye antecedentes, bases teóricas y conceptos clave. El Capítulo III describe la metodología empleada. El Capítulo IV expone los resultados del análisis cualitativo y la discusión de los hallazgos frente a la literatura revisada. Finalmente, el Capítulo V presenta las conclusiones y recomendaciones, seguidas de las referencias bibliográficas y anexos.

I. Información general

1.1. Título del proyecto

La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú.

1.2. Área estratégica de desarrollo prioritario

Esta investigación pertenece al ámbito de Tecnologías de la Información (TI), calidad de software y testing con foco específico en pruebas unitarias. La pertinencia estratégica se sustenta en que, en el contexto limeño, la ausencia de pruebas unitarias se asocia con problemas de calidad y eficiencia en el desarrollo, generando errores no detectados, retrabajo y retrasos en la entrega de proyectos, tal como se enuncia en la formulación del problema del propio estudio.

1.3. Actividad económica en la que se aplicaría investigación

Esta investigación se enmarca en la línea de investigación institucional Mejora de Procesos, ya que se orienta al análisis y mejora del proceso de desarrollo de software en organizaciones de Lima, específicamente en la incorporación de pruebas unitarias como práctica sistemática de aseguramiento de calidad. El estudio busca aportar a la eficiencia y calidad de los productos de software, reduciendo retrabajo, errores en producción y deuda técnica, al optimizar la forma en que los equipos de desarrollo, líderes técnicos y áreas de QA gestionan su proceso de trabajo, tal como se expone en el planteamiento del problema y en los objetivos del proyecto.

1.4. Localización o alcance de la solución

La presente investigación se aplica al ecosistema de desarrollo y servicios de TI en Lima, Perú y tiene alcance organizacional y técnico sobre equipos que construyen y mantienen software. La motivación práctica es clara: la incorporación sistemática de pruebas unitarias disminuye la probabilidad de que errores triviales lleguen a producción y favorece la mantenibilidad, escalabilidad y rendimiento tras cambios en el código (refactorizaciones).

II. Descripción de la investigación

2.1. Planteamiento del problema

La industria de software enfrenta retos constantes para asegurar la calidad de los productos en un contexto de alta presión por entrega y complejidad creciente de los sistemas. Estudios recientes muestran que, pese al amplio cuerpo de conocimiento en pruebas de software, las actividades de gestión y automatización de pruebas siguen siendo de las más desafiantes para las organizaciones, y existe una brecha entre los problemas que reportan los practicantes y los temas priorizados por la investigación académica (Garousi et al., 2020). Al mismo tiempo, investigaciones sobre proyectos de código abierto han evidenciado que la adopción efectiva de pruebas unitarias dista de ser una práctica consolidada: en muchos proyectos se observan pocos casos de pruebas automatizadas o incluso su ausencia, aun cuando los proyectos que sí las utilizan tienden a mostrar mejores indicadores de calidad y mantenimiento (Wang et al., 2024). Estos hallazgos se contrastan con marcos de referencia como el modelo de calidad ISO/IEC 25010:2023, que resaltan atributos como confiabilidad, mantenibilidad y testabilidad como dimensiones críticas de la calidad del software (ISO, 2023), para cuya gestión las pruebas unitarias son una práctica clave.

En el ámbito latinoamericano y peruano, diversas investigaciones recientes se han orientado a fortalecer los procesos de prueba y el rol del aseguramiento de la calidad. Díaz Figueredo et al. (2021) proponen un proceso de pruebas de software integrado a un modelo de calidad institucional en Cuba, destacando la necesidad de definir actividades e indicadores específicos para incrementar la efectividad de las pruebas. De forma similar, Cabrera Vigil et al. (2024) evidencian que, en una empresa de desarrollo a medida en El Salvador, la falta de un plan de pruebas alineado a la norma ISO/IEC/IEEE 29119 se asociaba con detección tardía de errores y retrabajo, por lo que diseñan e implementan un plan basado en dicho estándar para mejorar la calidad y reducir incidencias en producción. En el contexto peruano, Quispe Gutiérrez (2023) aborda la automatización de pruebas funcionales y de software en el proceso de control de calidad del Ministerio de Educación, mientras que Paz Díaz (2023)

plantea el uso de automatización apoyada en ChatGPT para mejorar el proceso de calidad en empresas del sector financiero en Lima, y Espinoza (2025) analiza las competencias desarrolladas en procesos de optimización de pruebas en una empresa multinacional como Globant. Estos trabajos confirman la importancia estratégica de las pruebas de software y muestran esfuerzos por formalizar procesos y automatizar pruebas funcionales, pero no tienen como foco central la ausencia de pruebas unitarias en el trabajo cotidiano de los equipos de desarrollo.

En este contexto, estudios de caso como el de Rojas Illanes (2023), centrado en la refactorización e implementación de pruebas unitarias en un motor de transacciones digitales, muestran que la incorporación de pruebas unitarias puede integrarse exitosamente en proyectos específicos, pero aún como iniciativas puntuales y no como una práctica generalizada. La revisión de la literatura disponible y la experiencia profesional del investigador no permitieron identificar, para el caso de Lima, Perú, 2025, investigaciones cualitativas que describan desde la perspectiva de project managers y desarrolladores senior las razones por las cuales numerosos equipos de desarrollo no implementan pruebas unitarias de forma consistente, ni cómo esta ausencia se relaciona con la calidad del producto, la mantenibilidad del código y la aparición de defectos en etapas tardías. Este vacío configura el núcleo del problema abordado en la presente investigación: la ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú, 2025, y la necesidad de comprender, desde las experiencias y discursos de los profesionales del desarrollo, los factores técnicos, temporales y humano–organizacionales que explican dicha ausencia y sus efectos en la calidad del software y en la coordinación del trabajo entre roles.

2.1.1. Problemas de investigación

2.1.1.1. Problema general

¿Cómo se explican los factores técnicos, temporales y humano–organizacionales que sostienen la ausencia de pruebas unitarias en proyectos de desarrollo de software en Lima,

Perú, en 2025, y qué efectos perciben los actores en la calidad del producto y en la coordinación del trabajo entre roles?

2.1.1.2. Problemas específicos

PE1: ¿Cuáles son las razones, creencias y barreras de tipo técnico, cultural y de gestión que sostienen la no adopción de pruebas unitarias en los equipos de desarrollo de software en Lima, Perú, en 2025?

PE2: ¿Cómo incide la ausencia de pruebas unitarias en el flujo de desarrollo de software, en términos de calidad, retrabajo, estabilidad de los despliegues, mantenimiento y ciclo de vida, desde la perspectiva de los actores involucrados en proyectos de desarrollo en Lima, Perú, en 2025?

2.2. Justificación

2.2.1. Justificación teórica

La presente investigación se sustenta en marcos conceptuales y empíricos que reconocen a las pruebas unitarias como una práctica clave para asegurar la calidad del software. Modelos de calidad como ISO/IEC 25010:2023 destacan atributos como confiabilidad, mantenibilidad y testabilidad como dimensiones críticas de la calidad interna y externa de los sistemas (ISO, 2023). En este marco, las pruebas unitarias y enfoques afines como el desarrollo guiado por pruebas (TDD) se conciben no solo como técnicas de verificación, sino también como mecanismos de diseño que contribuyen a mejorar la estructura interna del código y a reducir defectos desde etapas tempranas del ciclo de vida (Bissi et al., 2016). Estudios recientes en entornos reales de desarrollo refuerzan esta idea al mostrar que TDD y las pruebas unitarias tienden a mejorar la calidad interna y externa del software, aunque con un posible costo en productividad, y que la adopción efectiva de pruebas unitarias sigue siendo limitada en muchos proyectos, con coberturas reducidas o ausencia de pruebas en componentes críticos (Trautsch et al., 2017; Wang et al., 2024).

Al mismo tiempo, diversas investigaciones en la industria del software señalan que la automatización de pruebas, la gestión del proceso de testing y la alineación entre necesidades de negocio y prácticas de aseguramiento de la calidad constituyen focos recurrentes de dificultad para las organizaciones (Garousi et al., 2020). En paralelo, estudios sobre adopción de enfoques ágiles muestran que factores humanos y organizacionales —como la cultura de equipo, el soporte gerencial, el conocimiento disponible y la participación del cliente— influyen de manera decisiva en la adopción o resistencia a prácticas recomendadas, más allá de la mera disponibilidad de herramientas técnicas (Altuwaijri et al., 2022). Esta evidencia sugiere que la presencia o ausencia de prácticas como las pruebas unitarias no depende únicamente de aspectos técnicos, sino también de dinámicas de gestión, prioridades de negocio y capacidades organizacionales.

La necesidad de detectar defectos en etapas tempranas se vincula asimismo con la perspectiva clásica de costos de la calidad, que indica que el costo de corregir errores aumenta significativamente cuando estos se detectan en fases tardías del ciclo de desarrollo (Boehm & Basili, 2001). En este sentido, las pruebas unitarias sistemáticas se alinean con las recomendaciones de reducción de defectos y de prevención de retrabajo, al posibilitar la detección oportuna de fallas en unidades pequeñas y manejables del sistema, antes de que se propaguen a componentes de mayor complejidad. De esta manera, las pruebas unitarias se integran como un componente coherente con los modelos de calidad y con la visión de ingeniería de software orientada a minimizar riesgos operativos y la deuda técnica asociada a cambios y mantenimientos posteriores.

En el contexto latinoamericano y peruano, los antecedentes revisados se orientan principalmente a mejorar procesos de pruebas y de aseguramiento de la calidad —por ejemplo, mediante refactorización e implementación de pruebas unitarias en motores de transacciones (Rojas Illanes, 2023), diseño de procesos de pruebas alineados con modelos de calidad institucionales (Díaz Figueredo et al., 2021) o elaboración de planes de pruebas basados en normas internacionales (Cabrera Vigil et al., 2024)—, así como a la

automatización de pruebas funcionales y de regresión en organizaciones públicas y privadas (Quispe Gutiérrez, 2023; Paz Díaz, 2023; Espinoza, 2025). Estos trabajos confirman la relevancia de las pruebas y evidencian esfuerzos por institucionalizar prácticas de calidad, pero no abordan como problema central las razones por las cuales los equipos de desarrollo en Lima, Perú, no incorporan sistemáticamente pruebas unitarias en su práctica cotidiana. En ese marco, esta investigación se justifica teóricamente como un aporte exploratorio y cualitativo que traslada dichos marcos conceptuales al análisis detallado de la ausencia de pruebas unitarias en el desarrollo de software en Lima en 2025, aportando un diagnóstico cualitativo situado sobre cómo factores culturales, organizacionales, técnicos y formativos interactúan para limitar la adopción de esta práctica ampliamente recomendada.

2.2.2. Justificación metodológica

Se adoptó un enfoque cualitativo porque el interés central de la investigación fue comprender en profundidad las percepciones, prácticas y retos que enfrentan los desarrolladores y las organizaciones tecnológicas en Lima respecto a la ausencia de pruebas unitarias. Este tipo de enfoque resulta pertinente cuando se busca interpretar significados y dinámicas organizacionales más que medir relaciones causales entre variables. En estudios aplicados al contexto peruano, como los de Quispe Gutiérrez (2023) en el Ministerio de Educación y Paz Díaz (2023) en empresas del sector financiero, se ha observado que la formalización y automatización de las pruebas de software permiten mejorar la detección temprana de defectos y la eficiencia del proceso de control de calidad, pero también que su implementación depende de condiciones culturales, de gestión y de capacidades técnicas. En esa línea, el enfoque cualitativo de esta investigación permitió indagar, mediante entrevistas en profundidad, cómo los profesionales de desarrollo justificaban en la práctica la postergación o ausencia de pruebas unitarias y qué efectos percibían sobre la calidad del software y el trabajo en equipo, siguiendo ejemplos metodológicos que privilegian el análisis interpretativo de la experiencia de los actores (Sagástegui et al., 2022; Espinoza, 2025).

2.2.3. Justificación práctica

La investigación se justifica en el plano práctico porque la ausencia de pruebas unitarias en el desarrollo de software se traduce en fallas en producción, retrabajo continuo, incremento de costos operativos y afectación de la confianza de usuarios y clientes. En contextos reales, como el Ministerio de Educación del Perú, se ha evidenciado que la mejora y automatización de las pruebas funcionales permite reducir tiempos de ejecución, aumentar la detección temprana de defectos y ampliar la cobertura de regresión en sistemas críticos (Quispe Gutiérrez, 2023). De manera similar, en el sector financiero, la implementación de estrategias de automatización de pruebas apoyadas en herramientas de inteligencia artificial ha mostrado beneficios en la eficiencia del aseguramiento de calidad y en la estabilidad de los servicios de software (Paz Díaz, 2023). Estos antecedentes confirman que fortalecer el proceso de pruebas tiene efectos directos sobre la continuidad operativa y la calidad del servicio. En este contexto, comprender por qué las pruebas unitarias siguen siendo marginales en equipos de desarrollo en Lima y cómo esta ausencia incide en el flujo de trabajo y en la coordinación entre roles aporta insumos prácticos para diseñar acciones de mejora en procesos, cultura de calidad y formación de los profesionales de software.

2.3. Marco referencial

2.3.1. Antecedentes de investigación

Internacionales

Rojas Illanes (2023) en su estudio titulado *Refactorización e implementación de pruebas unitarias en el motor de Transacciones Digitales de Instance* tuvo por objetivo mejorar la calidad del software mediante la incorporación de pruebas unitarias en un sistema de transacciones digitales. La investigación desarrollada se basó en un enfoque experimental donde se aplicó la refactorización del código junto con la implementación de pruebas unitarias en el motor de transacciones digitales. Como resultado se evidenció una reducción significativa de errores y una mejora en la mantenibilidad del software, además de un aumento en la confiabilidad del sistema. Concluyendo que, la incorporación de pruebas unitarias como

parte integral del ciclo de desarrollo contribuye a optimizar la calidad y la estabilidad del software, además de facilitar futuras modificaciones, lo cual es esencial para sistemas críticos como los de transacciones digitales.

Díaz Figueredo, Lazo Alvarado y Tamayo Oro (2021), en su artículo *Proceso de pruebas de software para un modelo de calidad en Cuba*, tuvieron por objetivo elaborar un proceso base de pruebas de software para el Modelo de Calidad para el Desarrollo de Aplicaciones Informáticas (MCDAI), de manera que aumente la efectividad de las pruebas durante el desarrollo de software. Se trata de un artículo original de carácter propositivo, en el que se definen un conjunto de requisitos específicos que sintetizan buenas prácticas de pruebas de software en un proceso base diseñado para integrarse al MCDAI, describiendo actividades, roles y subprocesos como la ejecución de pruebas de unidad/componente, junto con un sistema de indicadores para evaluar procesos y productos. Entre sus principales hallazgos, los autores muestran, a partir de diagnósticos en organizaciones desarrolladoras de software en Cuba, que existe un uso insuficiente de procesos formales de prueba (incluida la ausencia de pruebas unitarias y de aceptación en un porcentaje importante de organizaciones) y que la aplicación del proceso base propuesto, validado mediante grupos focales, criterio de expertos y proyectos piloto, incrementa la efectividad de las pruebas en etapas tempranas del desarrollo. En sus conclusiones, plantean que formalizar un proceso de pruebas alineado al MCDAI permite mejorar la gestión de las actividades de verificación y validación, reducir no conformidades en el entorno operacional y contribuir a elevar la calidad del software en el contexto estudiado.

Cabrera, L. E., Landa, I. E. y Vindel, J. (2024), en su tesis de maestría *Estudio y análisis sobre la creación e implementación de un plan en el proceso de pruebas de software, basado en la Norma ISO/IEC/IEEE 29119 apartados: 1:2022, 2:2021; 3:2021; 4:2021, aplicado al área de control de calidad en la empresa System Out of the Box, El Salvador*, tuvo por objetivo diseñar y documentar un plan de pruebas de software alineado con la norma ISO/IEC/IEEE 29119 para el área de control de calidad de una empresa de desarrollo a medida. Se trata de

un trabajo de maestría de carácter aplicado y enfoque descriptivo, en el que primero se analiza la situación del proceso de pruebas de la empresa —donde las pruebas se ejecutan según el criterio y experiencia de cada analista, sin un estándar formal ni un proceso definido— y se identifican como problemas principales la inversión adicional de tiempo y recursos para corregir errores no detectados en etapas tempranas, la aparición de incidencias cuando las aplicaciones ya están en producción y la pérdida de credibilidad frente a los clientes. Entre sus hallazgos, los autores muestran que esta ausencia de lineamientos claros en el proceso de pruebas genera ineficiencias y riesgos en la calidad del producto, por lo que proponen un plan de pruebas basado en los apartados 1, 2, 3 y 4 de la norma ISO/IEC/IEEE 29119 para estandarizar actividades, definir responsabilidades y reducir la dependencia del criterio individual. En sus conclusiones, la tesis plantea que contar con un plan de pruebas alineado a la norma permite ordenar el trabajo del área de calidad, disponer de un proceso de pruebas estructurado y mejorar la gestión de la calidad del software al disminuir la improvisación y la variabilidad en la ejecución de las pruebas.

Sagástegui et al. (2022) en el artículo *Nuevo modelo basado en desarrollo ágil para mejorar la implementación de un sistema integrado* tuvieron como objetivo diseñar un modelo que optimizara la calidad y entrega de software mediante prácticas ágiles. La metodología implementada fue un estudio descriptivo basado en la aplicación del modelo en un caso real. Se reportó que la ausencia de pruebas unitarias adecuadas generaba errores recurrentes que demoraban los ciclos, mientras que su inclusión sistemática mejoró el desempeño del sistema. Concluyeron que incorporar pruebas unitarias dentro de los ciclos ágiles da soporte a una entrega continua de software robusto y confiable.

Nacionales

Mamani (2023), en su artículo Software Testing for Microservices publicado en la revista Innovation and Software, tuvo por objetivo describir los desafíos que plantea la arquitectura de microservicios para las actividades de prueba de software y resaltar la importancia de enfoques y herramientas adecuados para afrontarlos. El trabajo se presenta como un artículo

original de carácter descriptivo, en el que se analizan las características de los microservicios —capaces de agrupar cientos o miles de servicios que evolucionan continuamente— y las implicancias que esto tiene para el proceso de testing. Entre sus principales hallazgos, el autor destaca que la coexistencia de múltiples servicios hace que las pruebas sean un reto considerable, y que para superar este desafío la automatización de las pruebas, junto con el uso de herramientas de testing eficientes y eficaces, adquiere un papel clave en el aseguramiento de la calidad del sistema. En sus conclusiones, el artículo subraya que, en entornos distribuidos basados en microservicios, el mantenimiento de atributos como la seguridad y el rendimiento depende de contar con estrategias de prueba sistemáticas y fuertemente apoyadas en la automatización, lo que pone de relieve la necesidad de integrar de manera planificada las pruebas de software dentro del ciclo de desarrollo.

Quispe Gutierrez (2023), en su estudio Automatización de Pruebas Funcionales y Pruebas de Software en el Proceso de Control de Calidad del Ministerio de Educación del Perú, 2022, tuvo por objetivo precisar en qué medida la automatización de pruebas funcionales mejora las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú. La metodología empleada fue de tipo aplicada, con diseño preexperimental y enfoque cuantitativo, realizando mediciones antes (pretest, pruebas manuales) y después (postest, pruebas automatizadas) sobre una muestra de 26 analistas de calidad de la Unidad de Calidad y Seguridad de la Información del MINEDU, mediante la técnica de observación directa con fichas de recolección de datos. Los resultados evidenciaron una reducción del tiempo de ejecución de casos de prueba en 71,5 %, un incremento del 33 % en la detección temprana de defectos, una tasa de cobertura de pruebas de regresión de 96 %, una mejora del 43 % en el rendimiento del diseño de casos de prueba automatizados y una eficiencia del 97 % en la detección de defectos. La conclusión señala que la automatización de pruebas funcionales mediante scripts mejora significativamente la ejecución de las pruebas de software, incrementando la eficiencia, la productividad y la confianza, optimizando el diseño

de los casos de prueba y reduciendo los costos del proceso de control de calidad en el Ministerio de Educación del Perú.

Paz Díaz (2023) en su estudio *Implementar Automatización de Pruebas usando ChatGPT para la mejora del proceso de Calidad de Software en empresas del sector financiero en Lima-Perú 2023*, tuvo por objetivo evaluar el impacto de la inteligencia artificial en la automatización de pruebas. La metodología usada fue de tipo exploratorio, con un piloto de implementación de pruebas automatizadas asistidas por IA. Se observó una disminución del tiempo dedicado a la creación y ejecución de pruebas, así como una mejora en la detección de defectos. La conclusión señala que la combinación de pruebas unitarias con herramientas de IA permite optimizar el proceso de aseguramiento de calidad en entornos complejos como el financiero.

Espinoza Calderón (2025), en su trabajo de suficiencia profesional *Competencias desarrolladas en el proceso de optimización de pruebas de software en la empresa Globant*, tuvo por objetivo describir las competencias y mejoras asociadas al proceso de optimización de pruebas de software en el área de Quality Engineering de Globant Perú, en el contexto de proyectos de transformación digital. Se trata de un trabajo de suficiencia profesional de licenciatura, de carácter descriptivo, basado en la sistematización de la experiencia profesional en dicha empresa, donde se presenta el modelo de actividades de testing a lo largo del ciclo de vida del desarrollo, los servicios especializados de pruebas (como Agile Testing, Test Automation, Load & Performance Testing y otros) y el rol del área de calidad en la reducción de riesgos comerciales de los clientes. Entre sus hallazgos, identifica una oportunidad de optimización en las pruebas recurrentes en entornos de desarrollo, orientada a reducir el sobretrabajo y evitar la re-ejecución innecesaria de pruebas, e incorpora prácticas como las pruebas continuas, la automatización de pruebas y la adopción de Integración Continua y Despliegue Continuo (CI/CD) coordinadas con los desarrolladores, con el fin de mantener altos estándares de calidad en desarrollo y producción. En sus conclusiones, plantea que la optimización del proceso de pruebas de software exige integrar estas prácticas técnicas a lo largo del ciclo de vida, de manera alineada con las necesidades del negocio,

reforzando el papel del área de Quality Engineering como responsable de garantizar productos de software de alta calidad.

2.3.2. Marco teórico

2.3.2.1. Ausencia de pruebas unitarias

a) Subcategoría: Errores / Bugs

Los defectos de software (bugs) se conciben como fallas o imperfecciones en los artefactos de software que pueden producir resultados inesperados o comportamientos incorrectos si no son detectados y corregidos oportunamente (Boehm & Basili, 2001). En ausencia de pruebas unitarias, los errores elementales que podrían identificarse en la capa de código se desplazan hacia fases posteriores —pruebas funcionales, aceptación de usuario o incluso producción—, lo que incrementa el costo de corrección, el retrabajo y la probabilidad de generar deuda técnica acumulada. Intervenciones que combinan refactorización e incorporación de pruebas unitarias evidencian reducciones significativas de fallos y mejoras en la mantenibilidad del código (Rojas Illanes, 2023), mientras que experiencias en organizaciones públicas peruanas muestran que fortalecer el pipeline de pruebas disminuye la recurrencia de incidencias en operación (Quispe Gutiérrez, 2023).

b) Subcategoría: Fecha límite (deadlines)

Las fechas límite (deadlines) son hitos temporales establecidos para la entrega de funcionalidades, versiones o proyectos completos. La literatura sobre gestión de proyectos de software señala que la presión por cumplir plazos rígidos suele concentrarse en la entrega de funcionalidad visible para el cliente, recortando actividades que no se perciben como “productivas” a corto plazo, entre ellas las pruebas unitarias (Trautsch et al., 2017; Wang et al., 2024). Esto conduce a reasignaciones de tiempo y recursos que disminuyen la cobertura de pruebas. Estudios aplicados reportan que, cuando se incorporan prácticas de automatización y pruebas en etapas tempranas, los plazos se gestionan de manera más realista, se reduce el retrabajo y se estabilizan los ciclos de entrega (Cabrera et al., 2024;

Quispe Gutiérrez, 2023). La presión de tiempo, por tanto, no solo comprime tareas, sino que también reconfigura las prioridades del equipo, situando las pruebas unitarias como un esfuerzo prescindible frente a la urgencia de cumplir el deadline.

c) Subcategoría: Trabajo en equipo

El trabajo en equipo es un componente central en el desarrollo de software, pues la calidad del producto depende de la coordinación entre desarrolladores, testers y otros roles del ciclo de vida. La colaboración efectiva se sustenta en la comunicación constante, la distribución clara de responsabilidades y la existencia de acuerdos compartidos sobre calidad, incluyendo quién escribe, ejecuta y mantiene las pruebas (Sagástegui et al., 2022). La literatura sobre Quality Engineering y pruebas de software resalta que la experiencia del equipo y las competencias desarrolladas en torno a testing y automatización son determinantes para sostener prácticas de calidad (Espinoza, 2025; Wang et al., 2024). En contextos donde no se institucionalizan las pruebas unitarias ni se establecen mecanismos como la revisión de código por pares, el manejo de errores queda supeditado a la iniciativa individual y al conocimiento tácito de algunos miembros, lo que aumenta la variabilidad y dificulta la consolidación de aprendizajes colectivos (Bacchelli & Bird, 2013). De este modo, la ausencia de pruebas unitarias no solo es un problema técnico, sino también una expresión de debilidades en la coordinación del trabajo y en la forma como los equipos distribuyen y supervisan las tareas relacionadas con la calidad.

2.3.2.2. Desarrollo de software

a) Subcategoría: Buenas prácticas

Las buenas prácticas en desarrollo de software incluyen la modularización del código, el diseño orientado a la prueba, el uso de patrones de diseño, la refactorización continua y la revisión de código por pares. Estas prácticas buscan facilitar el aislamiento de unidades, reducir el acoplamiento y mejorar la legibilidad, de modo que el software sea más fácil de probar y mantener (Bissi et al., 2016; Rojas Illanes, 2023). Investigaciones en arquitecturas

modernas muestran que equipos que combinan buenas prácticas de diseño con pruebas automatizadas tienen mayor capacidad para gestionar la complejidad, introducir cambios y responder a incidentes sin deteriorar la calidad (Laura Mamani, 2023; Wang et al., 2024). La ausencia de estas prácticas, por el contrario, incrementa la dependencia de conocimiento individual y dificulta la incorporación posterior de pruebas unitarias.

b) Subcategoría: Definición de Hecho (Definition of Done)

La Definición de Hecho (Definition of Done, DoD) se entiende como un conjunto explícito de criterios que debe cumplir un incremento de software para ser considerado completo. En metodologías ágiles, estos criterios suelen incluir actividades de desarrollo y verificación (por ejemplo, revisión de código, pruebas unitarias y pruebas de integración), así como artefactos asociados, tales como planes, procedimientos y registros (Cabrera et al., 2024). Cuando la DoD se formula y acuerda a nivel de equipo y organización, actúa como un mecanismo de alineamiento entre desarrollo y calidad, porque obliga a incorporar actividades de prueba dentro del flujo regular de trabajo.

c) Subcategoría: Programación guiada por pruebas (Test-Driven Development, TDD)

La programación guiada por pruebas (Test-Driven Development, TDD) es una práctica de desarrollo que avanza en ciclos breves: primero se escribe una prueba que falla (rojo), luego se implementa el código mínimo para que la prueba pase (verde) y, finalmente, se refactoriza el diseño conservando la cobertura de pruebas (Bissi et al., 2016). Esta dinámica convierte a las pruebas unitarias en un artefacto de diseño, no solo de verificación, ya que orientan la estructura del código y facilitan la detección temprana de defectos. Estudios empíricos reportan que equipos que adoptan TDD tienden a producir software con mayor mantenibilidad y menor incidencia de errores regresivos (Rojas Illanes, 2023; Wang et al., 2024). Sin embargo, la literatura también muestra que su adopción exige tiempo de aprendizaje, disciplina y apoyo organizacional, por lo que con frecuencia se percibe como una práctica difícil de incorporar en entornos donde los plazos son ajustados y las prioridades están

centradas en la entrega rápida de funcionalidad (Mamani, 2023; Espinoza, 2025). En este sentido, TDD representa un caso límite de integración entre desarrollo y pruebas unitarias, cuya ausencia en muchos equipos refleja la tensión entre productividad inmediata y calidad estructural del software.

2.3.3. Glosario de Términos

Pruebas unitarias (unit testing).

Pruebas automatizadas que verifican el comportamiento de la unidad más pequeña del software (función, clase o módulo) en aislamiento para detectar defectos de forma temprana y sostener la calidad del sistema, incluidas arquitecturas de microservicios donde sirven como primera barrera de control. (Laura Mamani, 2023; Wang et al., 2024).

Programación guiada por pruebas (Test-Driven Development, TDD).

Práctica de desarrollo que progresa en ciclos breves: primero se escribe una prueba automatizada que falla (rojo), luego se implementa el mínimo código para que pase (verde) y, finalmente, se refactoriza el diseño conservando el comportamiento; el ciclo se repite para construir el sistema de forma incremental. En TDD, las pruebas actúan como especificaciones ejecutables que guían el diseño hacia componentes más modulables y verificables, con efectos sobre calidad y productividad que varían según el contexto, tal como reporta la evidencia sintetizada en una revisión sistemática (Bissi, de Freitas, de Mello, & Goldman, 2016).

Refactorización.

Proceso de reestructurar el código sin cambiar su comportamiento observable para mejorar su diseño interno (legibilidad, acoplamiento, complejidad), habitualmente acompañado de pruebas unitarias para reducir regresiones. (Rojas Illanes, 2023).

Mantenibilidad

Característica de calidad del producto software que, según el modelo ISO/IEC 25010, engloba subcaracterísticas como modularidad, reusabilidad, analizabilidad, modificabilidad y probabilidad de ser probado; orienta especificación, medición y evaluación de la calidad. (ISO, 2023).

Defecto de software (bug)

Falla o imperfección en el artefacto software que puede producir resultados incorrectos o comportamiento no deseado; la reducción de defectos exige técnicas sistemáticas de ingeniería y pruebas a lo largo del proceso. (Boehm & Basili, 2001).

Revisión de código (code review).

Práctica de inspección por pares, hoy predominantemente “moderna”: informal, apoyada en herramientas (p. ej., revisiones en pull requests) y realizada de forma regular antes de integrar cambios. Aunque uno de sus móviles es encontrar defectos, en la práctica sus resultados más frecuentes incluyen transferencia de conocimiento entre desarrolladores, mayor consciencia de equipo y soluciones alternativas de mejor calidad; es decir, funciona tanto como control técnico como mecanismo de aprendizaje y coordinación del trabajo. (Bacchelli & Bird, 2013).

2.4. Resumen ejecutivo

El desarrollo de software se ha convertido en un recurso estratégico para organizaciones públicas y privadas en Lima. Sin embargo, muchos equipos trabajan sin incorporar pruebas unitarias de manera sistemática en su proceso de desarrollo. Esta ausencia se traduce en defectos que se detectan tarde, retrabajo constante, acumulación de deuda técnica, sobrecarga del área de calidad y mayor riesgo en los despliegues a producción. Aunque existen marcos de referencia y buenas prácticas que recomiendan el uso de pruebas unitarias, en la práctica estos lineamientos no siempre se reflejan en el día a día de los equipos de desarrollo. En este contexto, resulta necesario comprender por qué no se están

aplicando estas pruebas y qué efectos concretos tiene esta situación en la calidad del software y en la forma de trabajo.

La investigación se enmarca en la línea de Mejora de procesos y se orienta a analizar la ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú, en el año 2025. El propósito del estudio es comprender e interpretar los factores técnicos, temporales y humano–organizacionales que explican por qué los equipos no incorporan pruebas unitarias, así como los efectos que esta ausencia genera en la calidad del producto y en la coordinación entre roles. Más que medir cuántos equipos hacen o no pruebas unitarias, el foco está en entender cómo se toma la decisión de no implementarlas, qué argumentos se utilizan para justificar esa elección y cómo impacta en el flujo de trabajo y en los resultados de los proyectos.

Para ello se utilizó un enfoque cualitativo, de tipo básico, con diseño no experimental y transeccional. Se trabajó con tres informantes clave: un project manager y dos desarrolladores senior que participan directamente en proyectos de desarrollo de software en organizaciones con distinta dimensión y alcance. La información se obtuvo mediante entrevistas semiestructuradas basadas en una matriz de categorización apriorística organizada en dos grandes ejes: ausencia de pruebas unitarias y desarrollo de software. Las entrevistas fueron grabadas, transcritas y analizadas mediante codificación cualitativa y triangulación, apoyándose en herramientas como un diagrama de Ishikawa, un mapa de coocurrencias de códigos y un diagrama de Pareto para ordenar y priorizar los hallazgos.

Los resultados muestran que la ausencia de pruebas unitarias no se debe únicamente a una falta de conocimiento técnico, sino a una combinación de factores interrelacionados. Entre ellos destacan la presión por cumplir plazos ajustados, estimaciones centradas en la entrega de funcionalidad visible, una cultura que prioriza “sacar el sistema” antes que asegurar su calidad estructural, la ausencia de estándares y de una Definition of Done que exija evidencia de pruebas unitarias, y una formación limitada en pruebas dentro de la trayectoria de muchos desarrolladores. A esto se suman decisiones organizacionales que

relegan la calidad de código frente a otros objetivos, y una alta dependencia del área de calidad y de las pruebas funcionales para “descubrir” errores que pudieron haberse detectado en etapas más tempranas

Esta combinación de factores tiene efectos directos en el funcionamiento de los equipos y en los resultados de los proyectos. La falta de pruebas unitarias favorece la detección tardía de defectos, genera ciclos de retrabajo, incrementa la deuda técnica y dificulta el mantenimiento del software. Además, eleva la carga del área de calidad, aumenta la incertidumbre en cada despliegue y hace que la estabilidad del sistema dependa en gran medida de la experiencia individual y de conocimientos implícitos no documentados. Estos hallazgos evidencian que la ausencia de pruebas unitarias no es un problema aislado, sino un síntoma de cómo se están gestionando el tiempo, las prioridades y la calidad en los equipos de desarrollo.

A partir de este diagnóstico, la investigación plantea la necesidad de formalizar la presencia de pruebas unitarias dentro del proceso de desarrollo. Esto implica, entre otros aspectos, incorporar criterios explícitos relacionados con pruebas unitarias en la Definition of Done, ajustar la planificación para considerar el esfuerzo asociado a escribir y mantener pruebas, fortalecer la formación de los desarrolladores en testing y promover prácticas de revisión de código que incluyan la verificación de pruebas asociadas. Si bien el estudio no implementa directamente una solución en un equipo específico, ofrece un mapa claro de las causas y efectos de la ausencia de pruebas unitarias en contextos reales de Lima, proporcionando insumos para que las organizaciones diseñen e implementen planes de mejora de sus procesos de desarrollo y de su cultura de calidad.

2.5. Objetivos General y Específicos: Propósito del Proyecto

2.5.1. Objetivo general

Comprender e interpretar los factores técnicos, temporales y humano-organizacionales que explican y sostienen la ausencia de pruebas unitarias en proyectos de desarrollo de

software en Lima, Perú en 2025, así como sus efectos percibidos en la calidad del producto y en la coordinación del trabajo entre roles.

2.5.2. Objetivos específicos

OE1: Comprender las razones, creencias y barreras (técnicas, culturales y de gestión) que sostienen la no adopción de pruebas unitarias en equipos de desarrollo en Lima, explorando experiencias, decisiones cotidianas y trade-offs reportados por desarrolladores y líderes técnicos.

OE2: Explorar cómo la ausencia de pruebas unitarias incide en el flujo de desarrollo (calidad, retrabajo, estabilidad de despliegues, mantenimiento y ciclo de vida) desde la mirada de los actores, interpretando prácticas, rituales de entrega y mecanismos alternativos.

2.6. Limitaciones

La presente investigación presentó algunas limitaciones que es necesario reconocer para contextualizar el alcance de sus resultados. En primer lugar, se trabajó con una muestra reducida de tres participantes seleccionados mediante muestreo intencional, conformada por un project manager y dos desarrolladores senior de empresas con características organizacionales distintas. Esta decisión permitió obtener información rica y detallada desde actores con amplia experiencia en desarrollo de software; sin embargo, restringió la posibilidad de generalizar los hallazgos a todos los equipos de desarrollo de Lima, por lo que las conclusiones deben entenderse como válidas para contextos similares y no como estimaciones representativas a nivel estadístico.

En segundo lugar, la investigación se centró exclusivamente en profesionales vinculados directamente al desarrollo de software, sin incluir otros roles como especialistas de aseguramiento de la calidad, product owners o usuarios clave. Esta delimitación permitió profundizar en las percepciones y prácticas de quienes implementan o postergan las pruebas unitarias en el código, pero dejó fuera la mirada de otros actores que también participan en

la gestión de la calidad y que podrían aportar perspectivas complementarias sobre las consecuencias de la ausencia de pruebas unitarias en el ciclo de vida del software.

Una tercera limitación estuvo asociada al alcance geográfico y temporal del estudio. El trabajo se desarrolló con profesionales que operaban en Lima en el año 2025, en un contexto marcado por demandas específicas de clientes locales e internacionales y por determinadas condiciones de mercado y de madurez tecnológica. En consecuencia, los resultados describieron una realidad situada en términos de tiempo y lugar; cambios posteriores en las prácticas de desarrollo, en la adopción de metodologías ágiles o en la cultura de calidad de las organizaciones podrían modificar las dinámicas observadas.

Finalmente, el uso de entrevistas en profundidad como única técnica principal de recolección de datos implicó que la información se basara en los relatos y percepciones de los participantes, con los sesgos inherentes a la memoria y a la deseabilidad social. Para mitigar este riesgo, se grabaron y transcribieron íntegramente las entrevistas, se revisaron de manera reiterada las transcripciones y se aplicó triangulación entre categorías apriorísticas, códigos emergentes y diferentes productos de análisis cualitativo. No obstante, el estudio no incorporó otras fuentes de evidencia, como métricas de repositorios de código o documentación interna de procesos de prueba, lo que abre una línea de trabajo para futuras investigaciones que deseen contrastar las percepciones recogidas con indicadores cuantitativos de calidad y de uso efectivo de pruebas unitarias.

2.7. Metodología del Proyecto

2.7.1. Categorización Apriorística

Para orientar el proceso de recolección y análisis de los datos, se elaboró una matriz de categorización apriorística a partir del problema de investigación y de los objetivos planteados. A partir de esta estructura se diseñó la guía de entrevista semiestructurada, de modo que cada bloque de preguntas se correspondiera con una categoría, subcategoría y código específico, asegurando la pertinencia de la información frente al problema y los

objetivos del estudio. Posteriormente, la misma matriz sirvió como referente para la codificación de las transcripciones y para la organización de la triangulación de la información, garantizando coherencia entre los objetivos, las preguntas formuladas y la interpretación de los hallazgos.

Categoría 1: Ausencia de pruebas unitarias

Definición conceptual: La ausencia de pruebas unitarias se concibe como la falta de aplicación de mecanismos de verificación del código que garanticen la calidad y estabilidad del software en sus fases iniciales. Rojas Illanes (2023) sostiene que la implementación de pruebas unitarias dentro del ciclo de desarrollo permite detectar tempranamente errores y optimizar el rendimiento del sistema antes de su integración completa, lo cual evidencia que su omisión expone a los proyectos a mayores riesgos operativos, incremento de fallos en producción y mayores costos de mantenimiento. La carencia de estas prácticas refleja una deficiencia en la cultura de aseguramiento de la calidad, así como una limitada comprensión de su impacto en la eficiencia del desarrollo. En consecuencia, esta categoría aborda la influencia que tiene la no ejecución de pruebas unitarias en la calidad final del producto y en la gestión del tiempo dentro del proceso de desarrollo de software.

Categoría 2: Desarrollo de software

Definición conceptual: El desarrollo de software se define como el proceso estructurado mediante el cual se diseñan, construyen, prueban y mantienen aplicaciones o sistemas informáticos, integrando recursos técnicos, humanos y organizacionales para garantizar su correcto funcionamiento. Cabrera, Landa y Vindel (2024) señalan que el desarrollo de software exige la implementación de procesos sistemáticos de verificación basados en estándares internacionales, como la norma ISO/IEC/IEEE 29119, que buscan asegurar la consistencia y fiabilidad del producto final. Por su parte, Espinoza Calderón (2025) explica que este proceso implica también la optimización continua de las pruebas y la formación de competencias técnicas dentro de los equipos de trabajo, ya que la calidad del software

depende tanto de la metodología aplicada como del conocimiento y la colaboración de los desarrolladores (pp. 10–14). En este sentido, el desarrollo de software se entiende como un proceso integral donde la aplicación de buenas prácticas, el uso de pruebas unitarias y la gestión eficiente del ciclo de vida del producto constituyen elementos esenciales para garantizar resultados sostenibles y de alta calidad.

2.7.2. Enfoque de Investigación

En el presente estudio se consideró que la investigación cualitativa resultaba apropiada para explorar fenómenos vinculados con competencias organizacionales y procesos de trabajo dentro de los equipos de desarrollo, ya que permite interpretar las experiencias y significados que los actores atribuyen a sus prácticas cotidianas más allá de la medición estadística de variables. En este mismo sentido, Rojas Illanes (2023) evidenció que la falta de pruebas unitarias no se limita a un problema técnico aislado, sino que refleja carencias en la cultura de calidad y en la estructura organizacional de los equipos de desarrollo, lo cual exige un enfoque interpretativo capaz de capturar las percepciones y prácticas de los profesionales.

2.7.3. Tipo de investigación

El presente estudio corresponde a una investigación de tipo básico, dado que su propósito fundamental es generar conocimiento teórico y comprensivo sobre la ausencia de pruebas unitarias en el desarrollo de software, sin buscar una aplicación o intervención directa en el entorno de estudio. Este tipo de investigación se orienta a la comprensión y análisis de causas y significados, más que a la implementación de soluciones inmediatas, lo que la distingue de la investigación aplicada.

Rojas Illanes (2023) identifica que, en muchos entornos tecnológicos, la falta de pruebas unitarias no se debe a limitaciones técnicas, sino a la ausencia de una cultura de validación y documentación del código, lo que plantea un campo de estudio teórico para comprender cómo los desarrolladores construyen sus prácticas profesionales. Del mismo modo, Cabrera,

Landa y Vindel (2024) afirman que la consolidación de un proceso de pruebas basado en estándares internacionales, como la norma ISO/IEC/IEEE 29119, exige un entendimiento conceptual y metodológico sobre la naturaleza del aseguramiento de la calidad en los equipos de software. Esta perspectiva coincide con el objetivo de una investigación básica, orientada a profundizar en los fundamentos y principios que explican las prácticas de prueba dentro del ciclo de desarrollo.

2.7.4. Diseño de investigación

El diseño de esta investigación es no experimental y de tipo transeccional. Se denomina no experimental porque no se manipulan variables, sino que se observan y analizan los hechos tal como ocurren en su contexto natural. Según Hernández Sampieri (2021), en este tipo de diseño el investigador no interviene sobre la realidad, sino que interpreta los fenómenos a partir de la información proporcionada por los participantes.

El estudio es transeccional o transversal porque se desarrolla en un único momento temporal (Lima, año 2025), con el objetivo de describir la situación actual de la ausencia de pruebas unitarias en equipos de desarrollo de software. No se realiza seguimiento longitudinal, sino una recopilación de percepciones en un punto específico del tiempo.

2.7.5. Niveles de investigación

El nivel de la presente investigación es exploratorio, dado que su propósito es comprender en profundidad un fenómeno poco estudiado en el contexto nacional. Este nivel se caracteriza por su orientación hacia el descubrimiento, la comprensión y la generación de aproximaciones iniciales a un problema, sin pretender establecer relaciones causales ni generalizaciones estadísticas. En concordancia con el enfoque cualitativo, el estudio busca describir y analizar las experiencias, percepciones y prácticas de los desarrolladores y líderes técnicos frente a la implementación o falta de pruebas unitarias en los proyectos de software.

Cabrera, Landa y Vindel (2024) sostienen que la adopción de normas y procesos estandarizados de prueba, como la ISO/IEC/IEEE 29119, representa un desafío cultural y

organizacional que requiere ser comprendido más allá del plano técnico, reforzando así la pertinencia de un abordaje exploratorio. Espinoza Calderón (2025) indica que la mejora de los procesos de prueba depende del fortalecimiento de competencias organizacionales y de la reflexión colectiva sobre la práctica, lo cual se alinea con el propósito de este estudio de indagar en las percepciones y experiencias de los profesionales del sector.

2.7.6. Población

La población de la presente investigación estuvo conformada por profesionales del sector tecnológico que participan directamente en el desarrollo de software en Lima, Perú, tales como desarrolladores full-stack y tech leads, quienes poseen un rol activo en la codificación, integración, aseguramiento de calidad y toma de decisiones técnicas dentro de sus organizaciones. Este grupo representa el segmento donde se manifiesta con mayor claridad el fenómeno de la ausencia de pruebas unitarias en los proyectos de software, dado que son los principales responsables de aplicar o supervisar las prácticas de verificación durante el ciclo de desarrollo.

De acuerdo con Rojas Illanes (2023), la falta de implementación de pruebas unitarias en entornos de desarrollo genera vulnerabilidades y errores en producción, lo que afecta directamente la calidad y confiabilidad del producto final. Esta problemática, al ser recurrente en organizaciones tecnológicas, justifica la elección de una población conformada por profesionales que enfrentan cotidianamente dichos desafíos.

2.7.7. Muestreo y muestra

El presente estudio empleó un muestreo no probabilístico de tipo intencional por criterios, orientado a seleccionar informantes clave que pudieran aportar información relevante sobre la ausencia de pruebas unitarias en el desarrollo de software. Se definió como población objetivo a profesionales del sector tecnológico que participan directamente en el desarrollo de software en Lima, Perú; en particular desarrolladores full-stack y líderes técnicos, que cuentan con al menos cuatro años de experiencia en la actividad. Bajo este enfoque, el

propósito no fue alcanzar representatividad estadística, sino profundizar en la comprensión del fenómeno a partir de casos que cumplieran condiciones específicas de experiencia y responsabilidad dentro de los equipos de desarrollo.

La muestra final quedó conformada por tres participantes, considerados informantes clave: un project manager que dirige equipos de desarrollo en proyectos para empresas como Cálidda; un desarrollador senior asociado a un emprendimiento de soluciones de TI con clientes como el SENAMHI y la Federación Peruana de Fútbol; y un tech lead de una empresa transnacional con operaciones en Guatemala y sede en Perú. Esta composición permitió contrastar perspectivas provenientes de organizaciones de distinto tamaño y alcance, manteniendo como criterio común la experiencia directa en el ciclo de vida del software y el tiempo de experiencia de 7 años.

2.7.8. Técnicas e instrumentos de recolección de datos

2.7.8.1. Técnicas de recolección de Datos

La técnica de recolección de datos utilizada en la investigación fue la entrevista semiestructurada, elegida por su capacidad para obtener información detallada, interpretativa y contextual sobre las experiencias de los desarrolladores y líderes técnicos en relación con la ausencia de pruebas unitarias en el desarrollo de software en Lima. Este método se considera apropiado en estudios cualitativos de nivel exploratorio, donde el objetivo es comprender fenómenos desde las percepciones de los propios actores y no medir variables de manera cuantitativa.

En el ámbito de la ingeniería de software, diversos estudios han mostrado la utilidad de recabar directamente la experiencia de los profesionales para comprender cómo se llevan a cabo las prácticas de desarrollo y aseguramiento de la calidad en contextos reales. Por ejemplo, Bacchelli y Bird (2013) combinaron observaciones, entrevistas y encuestas para analizar las expectativas, resultados y desafíos de la revisión de código en equipos industriales, mientras que Garousi et al. (2020) recogieron la opinión de profesionales de

distintos países mediante una encuesta de opinión para caracterizar los principales desafíos de la industria en pruebas de software. De forma coherente con este tipo de investigaciones empíricas, en el presente estudio las entrevistas semiestructuradas permitieron profundizar en las motivaciones, creencias y prácticas que explican por qué las pruebas unitarias no se incorporan de manera sistemática en los equipos de desarrollo analizados.

2.7.8.2. Instrumentos de Recolección de Datos

Para la recolección de información se utilizó una guía de entrevista semiestructurada diseñada ad hoc, instrumento idóneo en investigaciones cualitativas de tipo básico y nivel exploratorio. La guía se elaboró a partir de los objetivos de la investigación, las categorías definidas en la matriz de categorización apriorística y el marco teórico sobre pruebas de software y desarrollo ágil, organizando las preguntas en torno a temas como el proceso de desarrollo, la gestión de la calidad, el uso (o ausencia) de pruebas unitarias, la coordinación entre roles y la percepción de riesgos y deuda técnica. Este diseño permitió captar las percepciones, experiencias y significados que los profesionales atribuían a su práctica de desarrollo y al uso o ausencia de pruebas unitarias, facilitando la obtención de datos interpretativos de alta riqueza contextual.

La decisión de construir un instrumento específico para este estudio es coherente con investigaciones empíricas en pruebas de software que definen instrumentos alineados a sus objetivos y contextos particulares, como los estudios que analizan la implementación de procesos de prueba y la optimización de prácticas de aseguramiento de calidad en organizaciones reales (Bacchelli & Bird, 2013; Garousi et al., 2020; Sagástegui et al., 2022). En esa misma línea, la guía de entrevista de esta investigación fue sometida a revisión y ajuste iterativo para asegurar la claridad de las preguntas y su capacidad para generar información relevante respecto al problema planteado.

2.7.9. Validez y confiabilidad

2.7.9.1. Validez del instrumento

La validez de contenido se estableció mediante juicio de expertos. Se consultó a un especialista en desarrollo de software con grado de magíster en Ingeniería Informática, quien evaluó la guía de entrevista según los criterios de claridad, objetividad, actualidad, organización, suficiencia, intencionalidad, consistencia, coherencia, metodología y pertinencia. La valoración se realizó con una escala ordinal de cinco niveles (deficiente, regular, buena, muy buena y excelente), asociada a rangos porcentuales de 0–20 %, 21–40 %, 41–60 %, 61–80 % y 81–100 %, respectivamente. Al codificar estos niveles de 1 a 5 puntos, la suma de puntajes obtenidos fue de 40 sobre un máximo de 50, lo que equivale a un 80 % de la puntuación máxima. De acuerdo con esta escala, la guía se ubica en un nivel de validez “Muy bueno”, por lo que se considera adecuada para su aplicación en la investigación. Adicionalmente, en la matriz de pertinencia todos los 19 reactivos fueron calificados como suficientes, concluyéndose que el instrumento podía ser aplicado tal como estaba elaborado (ver Anexo 4).

2.7.9.2. Confiabilidad de la investigación

En el enfoque cualitativo, la confiabilidad se vincula con la credibilidad y la consistencia del proceso más que con coeficientes estadísticos. En este estudio, todas las entrevistas se realizaron con la misma guía semiestructurada, se grabaron íntegramente y se transcribieron de forma literal. Las transcripciones fueron revisadas más de una vez y posteriormente codificadas siguiendo la matriz de categorización apriorística, incorporando códigos emergentes cuando fue necesario. El análisis se reforzó mediante triangulación entre los discursos de los tres informantes, las categorías teóricas y diferentes representaciones analíticas (matriz de triangulación, diagrama de Ishikawa, mapa de coocurrencias y diagrama de Pareto), lo que permite rastrear el recorrido desde los registros originales hasta los resultados y respalda la confiabilidad del estudio.

2.7.10. Consideraciones éticas

La investigación respetó principios éticos básicos vinculados con el respeto a las personas, la confidencialidad de la información y el uso responsable de los datos. En primer lugar, la participación de los profesionales fue voluntaria y se obtuvo su consentimiento informado antes de iniciar cada entrevista, explicando el propósito del estudio, la forma de utilización de la información y la posibilidad de retirarse en cualquier momento sin consecuencias. En segundo lugar, las entrevistas fueron grabadas únicamente con autorización de los participantes y, posteriormente, se transcribieron y seudonimizaron, evitando registrar nombres reales de personas u organizaciones, de modo que no fuera posible identificar individualmente a los informantes. En tercer lugar, los registros audiovisuales y las transcripciones fueron almacenados en repositorios digitales de acceso restringido, destinados solo a fines académicos y de análisis, sin compartirlos con terceros ajenos al estudio. De este modo, se procuró salvaguardar la privacidad de los participantes y asegurar que los resultados presentados en el informe final reflejaran su experiencia sin exponer su identidad ni la de las instituciones a las que pertenecían.

III. Estimación del costo del proyecto

3.1. Estimación de los costos necesarios para la implementación

Tabla 1

Estimación de costos para el desarrollo de la investigación (prorrateo 1 mes)

Naturaleza del gasto	Descripción	Cantidad	Precio unitario (S/.)	Precio total (S/.)
Servicios	Servicio de internet	1	99.00	99.00
	Electricidad	1	99.00	99.00
	Plataforma de videollamadas (Zoom Basic: plan gratuito)	1	0.00	0.00

	Respaldo de videos en YouTube (privado)	1	0.00	0.00
	Transcripción automática de entrevistas (capa gratuita)	1	0.00	0.00
Bienes	— (no se requirieron adquisiciones para este estudio)	—	—	0.00
Personal	— (sin honorarios ni tercerización)	—	—	0.00
RESUMEN	**Servicios**			198.00
	Bienes			0.00
	Personal			0.00
	TOTAL			S/. 198.00

Nota. Elaboración propia.

Interpretación de resultados

Los costos del proyecto son bajos debido al diseño remoto: el 100% del gasto corresponde a servicios operativos prorrateados (internet y electricidad). Las plataformas utilizadas (Zoom en plan básico y YouTube en modo privado), así como la transcripción automática en su capa gratuita, no generaron egresos. Esta estructura de costos es consistente con un estudio cualitativo de entrevistas por videollamada realizado en menos de un mes. Si se extendiera el periodo o se incorporaran licencias de software de pago, los montos se ajustarían proporcionalmente.

IV. Resultado de investigación

4.1. Análisis de resultados

Tabla 2

Matriz de triangulación

Subcategoría	Pregunta (a priori)	Evidencia – Stefani	Evidencia – Cinthya	Evidencia – Félix	Convergencias y divergencias	Interpretación preliminar
Fecha límite / Deadline	¿Cómo determinan las fechas límite para los proyectos en los que trabajas?	"Se priorizan entregables,… hay reuniones con el equipo para definir lo que se va a usar y con el usuario… se revisan historias de usuario, el equipo estima el tiempo… trata de poner holgura"	"Yo misma estimo los tiempos, agrego algo de tiempo para pruebas de regresión y como colchón".	"El equipo no lo estima, lo estima el gerente del proyecto. Hay proyectos donde la fecha se puede mover por faltas del cliente y en otros la fecha no se mueve."	Convergencias: Se consulta con el cliente para recoger información y se agrega colchón de tiempo. Divergencia: Stefani estima el tiempo con el equipo. Cinthya estima los tiempos del equipo según su criterio, experiencia; Félix, no forma parte de la decisión.	Existen tres formas de estimar: en conjunto con el equipo, mediante tanteo individual o acatando decisiones externas, añadiendo siempre un margen por contingencias.
	¿Cuál es el nivel de cumplimiento	"A veces se demoran, … reprogramaciones	"A veces cumplimos… si lo tenemos antes lo	"En la mayoría de los casos no se llegan a	Convergencias: mayormente se cumple. Si no se	Es importante comunicar con el cliente para

de las fechas límite en tu equipo de trabajo?	de entregables siempre son coordinadas con el usuario… si me demoro …, lo puedo entregar en la siguiente presentación …”.	presentamos…por complejidad informo que tomará más tiempo.”	cumplir por la falta de tiempo por mala planificación que no depende del equipo.”	cumple se coordina el entregable. Divergencias: Stefani, si demoran lo planifican para siguientes entregables. Felix: No cumplen porque los tiempos se estiman mal.	replantear los entregables en caso de retraso. Una buena estimación de tiempo es importante para el cumplimiento.
¿Cuáles son las principales causas de retrasos en la entrega de proyectos?	“Al hacer pruebas de regresión surgen observaciones que se tienen que corregir.”	“El usuario no explicó bien por celos o por no ser idóneo y no pregunté.”	“Experiencia del programador, la claridad del requerimiento y cambios en los deadline.”	Convergencias: Felix y Cinthya coinciden en la mala información de parte del usuario Divergencias: Stefani reporta que bugs retrasan. Cinthya dice que los usuarios no informan bien los requerimientos y Felix, indica que la experiencia y cambios en el deadline retrasan	La adecuada recopilación de requerimientos, junto con mecanismos de prevención de errores, resulta clave para evitar retrasos en las entregas.
Si existe retrabajo, ¿incide en el	Reprogramar, agregar 2 o máximo 5 días a	Sí incide, tenemos que presentarlo sí o sí, se puede	Si no lo terminas, se empieza a arrastrar. O lo		

	retraso de la entrega?	pesar de la holgura. Se puede trabajar en paralelo con otros desarrollos.	coordinas una extensión, y si no se tiene que trabajar horas extras.	presentas con errores o queda como deuda técnica.		
Errores / Bugs	¿Cuál es el impacto de estos bugs en términos de retrabajo?	“En las pruebas internas… hay observaciones que corregir… (dejan) otra actividad para atender esa observación” (impacto en tiempo).	“Hicimos una reingeniería… para ser el producto escalable” “Toma tiempo… reorganizar lo que ya tienes… es difícil… (la planificación) se te va a mover”.	“Impacta mucho ya que debemos corregir y como estamos con el tiempo justo tenemos que trasnochar para llegar a tiempo”	Convergencia: bugs frecuentes generan retrabajo y demora. Divergencia: origen percibido (pruebas, deuda técnica histórica).	Los bugs y deuda técnica incrementan retrabajo y mueven cronogramas, reforzando el círculo de no probar a tiempo.
	¿Con qué frecuencia encuentran bugs en el código que desarrollan?	--	Bastante, por falta de conocimiento sobre el sistema por parte de los programadores, mueven algo y se rompen otras cosas.	Es muy frecuente, por los requerimientos mal explicados por parte del usuario o el gerente. A veces ofrecen funcionalidades que no desarrollamos.	Divergencias: Stefani no tiene conocimiento por no estar involucrada en la programación. Cinthya expresa que la falta de conocimiento de la lógica de programación causa bug y felix menciona mala comunicación	Es importante la comunicación entre todos los involucrados para que estén alineados y se logren los objetivos.

					entre usuarios y el gerente.	
	¿Qué estrategias emplean para minimizar la incidencia de errores en el desarrollo de software?	Pruebas de regresión, pruebas automatizadas, pruebas de rendimiento, pruebas en producción en conjunto con el usuario. Code review	Uso Try Catch para contener errores y manejar los mensajes de error. Todo se maneja por logs.	Trabajamos con QA y con un Excel donde se tienen las incidencias y observaciones. No tenemos una metodología para manejar los bugs.	Convergencias: No se manejan pruebas a nivel de programación, todo se hace luego del desarrollo. Divergencias: Diferentes formas de controlar errores.	No se dispone de una estrategia sólida de prevención de errores; la ausencia de estándares genera retrasos en las entregas.
	¿Cómo manejan los bugs encontrados en desarrollos anteriores?					
Trabajo en equipo	¿Qué tan estricta es la política de tu empresa respecto al seguimiento de buenas prácticas y	“Revisión de código… el líder técnico (verifica) buenas prácticas… pruebas de regresión, rendimiento y seguridad;	“No hay políticas, todo es conocimiento mío que lo transmito a los nuevos.”	“Según la documentación se les intruye a los nuevos”	Convergencia: se intenta asegurar calidad con documentación. Divergencia: formalización varía (documentada vs. informal/experiencial).	Cuando las prácticas están institucionalizadas, hay mayor control; donde son informales, depende del seniority y se diluye la calidad.

	patrones de diseño?	pruebas con usuario.”				
Buenas prácticas	¿Estás familiarizado con las buenas prácticas de programación y patrones de diseño?	“No opino porque no participo de la programación.”	“Una buena estructura de base de datos replicada en la programación manteniendo la nomenclatura, comentarios en el código.”	“Todo está documentado y estandarizado.”	Convergencias: Creen que buenas prácticas es seguir una estructura o patrones de nomenclatura. Divergencias: Cinthya sigue reglas no escritas mientras que Felix usa documentación.	Los entrevistados evidencian un desconocimiento de las buenas prácticas de programación, lo que dificulta la incorporación de mecanismos como las pruebas unitarias..
Definición de Hecho (DoD)	¿Cómo definirían el concepto de “definición de hecho” en su equipo de trabajo?	Lo que se pide en los documentos funcionales e historias de usuario.	Que el programa funcione como el usuario lo pedido	Lo que el cliente pide y coordina con gerencia.	Convergencias: Stefani y Cinthya coinciden en que la definición de hecho es que el entregable cumpla con lo solicitado y se verifique antes de darlo por terminado. Divergencias: Stefani la vincula a documentos e historias de usuario; Cinthya y Felix lo	La definición de hecho se entiende como cumplimiento práctico de requisitos, más que como un conjunto formal de criterios de calidad que incluya evidencias sistemáticas de pruebas unitarias.

					asocian más a lo que pide el jefe.	
	¿En tu trabajo usan el término “definición de hecho”?	“Se define de acuerdo con los documentos funcionales… historias de usuario… (se) verifica con el usuario.”	“Sí… con todo tipo de jefe: tú programas y agregas tareas dentro de la misma, pero no se maneja formalmente ese término.”	“Definición de hecho no… nos basamos en requerimientos y escenarios”	Divergencia clara: en un equipo sí existe DoD operativo; en otro no se usa el término ni práctica formal.	La presencia de DoD aporta claridad de “terminado”; su ausencia deja ambigüedad y brecha de criterios de calidad.
Programación guiada por pruebas (TDD)	¿Tienes experiencia con programación guiada por pruebas (TDD)?	No tiene experiencia TDD	Prefiere pruebas manuales “como QA”; razones: tiempo y no investigar TDD; ve las unitarias “útiles hasta un punto”	No tiene claro lo que significa	Convergencia: ninguno practica TDD; las unitarias se ven costosas bajo presión temporal.	Falta de conocimiento y tiempo inhibe TDD; se prioriza la validación manual de regresión/usuario.
	¿En qué medida se utiliza la programación guiada por pruebas en las empresas donde has trabajado?	No tengo experiencia con TDD.	No he usado TDD; sé que podría hacer pruebas, pero no lo aplico en la práctica	Usé pruebas automatizadas en un trabajo previo con .NET para validar HTML; actualmente no uso.	Convergencias: Actuales los tres entrevistados no aplica. Divergencias: Felix menciona una experiencia previa con pruebas	La programación guiada por pruebas no forma parte de las prácticas habituales de desarrollo; cuando aparece, lo hace como experiencia aislada y ligada a

				automatizadas en .NET.	proyectos específicos, no como estándar organizacional.
¿Cuáles son las principales razones por las que crees que no se adopta ampliamente la programación guiada por pruebas?	No se usa porque el equipo no tiene experiencia y agrega tiempo al desarrollo, necesitamos entregarlo en poco tiempo.	Nunca lo he usado porque no lo he investigado. Como siempre me piden el desarrollo para ya, prefería probarlo en caliente.	Se intentó usar pruebas, pero la práctica de pruebas unitarias internas no se mantuvo en el tiempo.	Convergencia: Coinciden en que no hay tiempo para implementar las pruebas por falta de tiempo y falta de conocimiento	Las principales razones para no utilizar pruebas en el desarrollo son la escasez de tiempo y la limitada formación en su uso.
¿Cuál es tu opinión sobre la necesidad de realizar pruebas unitarias?	Son muy importantes para entregar software de calidad.	Son importantes en el desarrollo. El equipo de QA lo prueba, pero de todas maneras el desarrollador debería hacer pruebas.	Son útiles hasta cierto punto, se complica cuando el sistema crece y ya no le veo sentido	Convergencia: Las pruebas unitarias son importantes. Divergencias: Felix indica que se complejiza en desarrollos grandes.	Son indudablemente relevantes; sin embargo, la carencia de conocimiento lleva a percibir las pruebas unitarias como difíciles de implementar.
¿Has observado que el desarrollo se alarga debido a la implementación	Sí, se alarga, por eso debemos prescindir de estas pruebas para que los	Sí, como mencioné antes, los usuarios lo piden para ya, eso hace	No se alarga mucho, porque antes se consideraba el tiempo, pero ahora nos	Convergencias: Se alarga el desarrollo y hay presión de tiempo.	Con una buena planificación y comunicación se puede incorporar el tiempo necesario para el

	de pruebas unitarias?	entregables no demoren tanto.	imposible que se pueda hacer.	imponen el tiempo y ya no se puede.	Divergencias: En algunos entornos organizados es posible implementar las pruebas	uso de las pruebas unitarias.
Definición de hecho / Definition of Done (DoD)	¿Qué mecanismos utilizan para asegurar que el producto final cumpla con los requisitos del cliente o DoD?	"Pruebas en conjunto con el usuario para validar"	"En la gran mayoría de equipos de trabajo no tengo QA; me ha tocado hacer pruebas yo sola para no quedarme corta de tiempo de desarrollo." / "Yo particularmente suelo probar el producto… y volver a probar."	"Se maneja un Excel de incidencias… se prioriza con el implementador (frente al cliente)"	Convergencia: hay mecanismos prácticos de validación y priorización. Divergencia: foco usuario final (Stefani) vs. implementador/cliente (Félix).	La validación existe, pero orientada a operación/cliente, no a calidad interna automatizada.
Buenas prácticas	¿Consideras que el código existente en tu empresa sigue estas buenas prácticas?	"Sí, aunque puede ser auditado por el área de TI de la empresa o por el cliente"	"Yo diría que sí, aunque puede tener errores, pero no es código spaguetti."	"Sí, aunque puedes encontrar código spaguetti pero está comentado para fácil ubicación."	Convergencia: Definición de buenas prácticas no es correcta y el código está muy acoplado. Divergencia: Stefani puede recibir auditorias mientras con Felix y Cinthya no y su código es	Falta de conocimiento de buenas prácticas de programación dificulta prevenir errores y sostener calidad.

	complejo de leer y refactorizar.

Nota: Elaboración propia

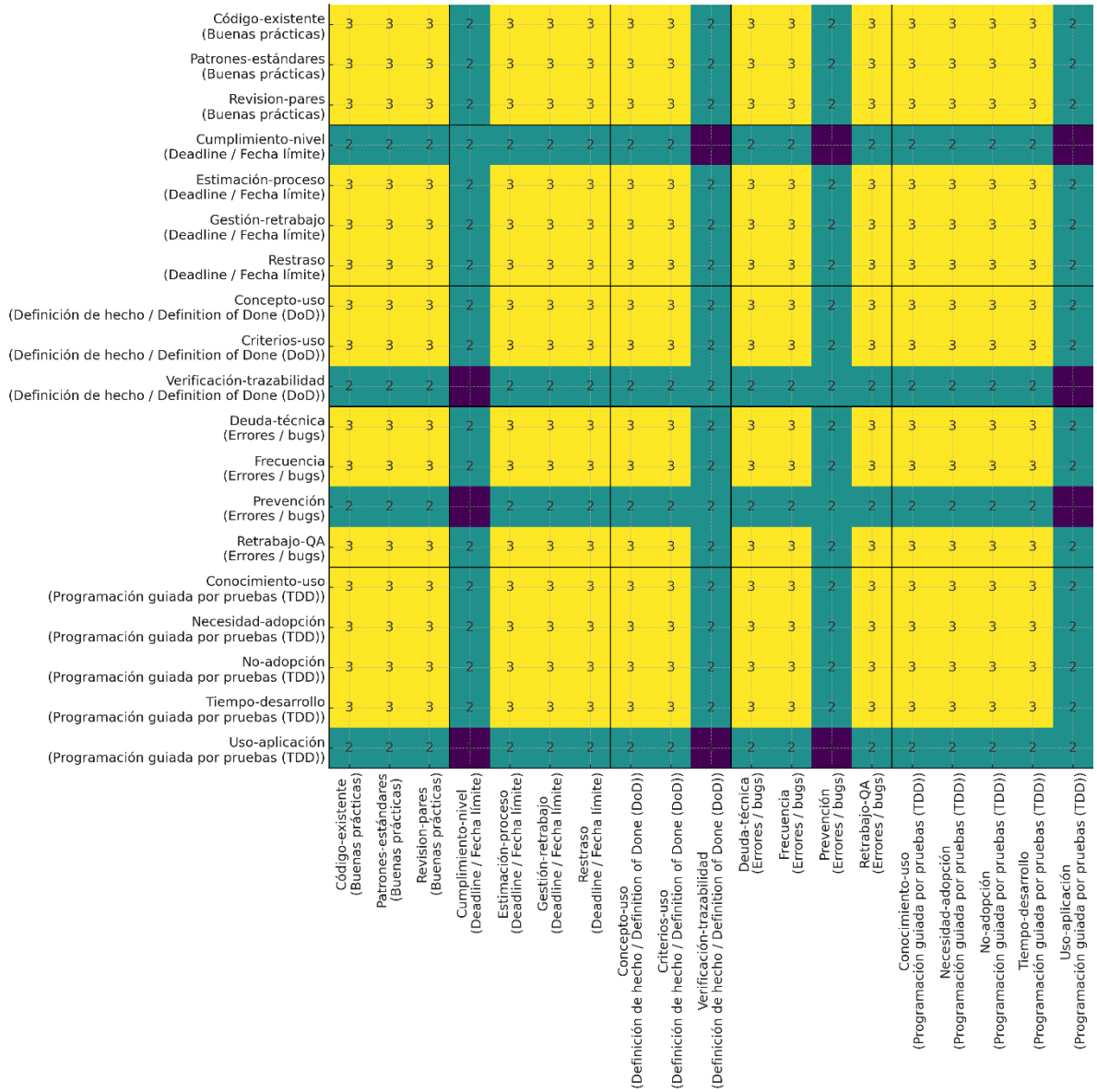
Figura 1

Diagrama causa–efecto (Ishikawa)

Nota. Elaboración propia

Figura 2

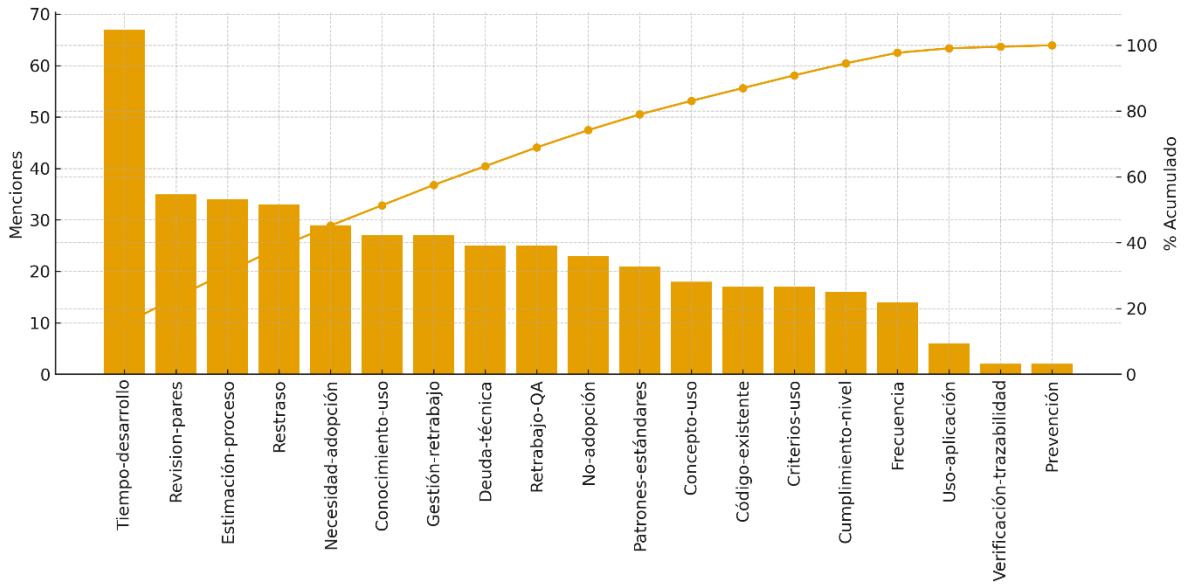
Mapa de co-ocurrencias de códigos (entrevistas)



Nota. Elaboración propia.

Figura 3

Diagrama de pareto



Nota. Elaboración propia

A partir de la matriz de triangulación (Tabla 2) y de las representaciones gráficas construidas a partir de los códigos (Figura 1: diagrama causa–efecto, Figura 2: mapa de coocurrencias y Figura 3: diagrama de Pareto), se identifican varios patrones comunes en las tres entrevistas. En primer lugar, la calidad se describe como gestionada de manera principalmente reactiva: el trabajo se organiza alrededor de la atención de incidencias reportadas por QA, por el usuario o registradas en un Excel de bugs, más que en torno a la prevención sistemática de errores. Las estrategias mencionadas para reducir la incidencia de errores —como pruebas de regresión, algunas pruebas automatizadas, uso de bloques try–catch, manejo de mensajes de error y pruebas en conjunto con el usuario— se aplican después del desarrollo y no como pruebas unitarias a nivel de programación. Según la interpretación preliminar de la matriz, la ausencia de una estrategia sólida de prevención hace que los problemas reaparezcan y consuman horas en correcciones repetidas.

Un segundo eje que atraviesa los relatos es la heterogeneidad en los criterios de cierre del trabajo. La Figura 1 sintetiza que “terminado” no significa lo mismo para todos: en un caso se define a partir de documentos funcionales e historias de usuario revisadas con el cliente; en otro se agregan tareas y ajustes según cada jefe, sin manejar formalmente el término Definition of Done; y en otro se trabaja únicamente con requerimientos y escenarios. Algo

similar ocurre con las buenas prácticas y la revisión de código: en un equipo el líder técnico revisa el código y contrasta con documentación y estándares, mientras que en otros predomina el conocimiento personal del desarrollador y reglas no escritas que se transmiten a los nuevos integrantes. Esta combinación de cierres poco claros y prácticas de revisión informales favorece que se escapen defectos que luego aparecen como retrabajo y alimentan la deuda técnica.

Un tercer patrón se relaciona con la tensión constante entre estimaciones y plazos. Según la matriz, las fechas se determinan combinando reuniones con el cliente, estimaciones realizadas por el propio equipo o decisiones del gerente del proyecto, añadiendo cierto margen por contingencias. No obstante, los tres entrevistados coinciden en que con frecuencia los tiempos se ajustan demasiado y que, cuando surgen imprevistos o errores no previstos, la fecha se mantiene y se reprograman entregables o se recortan actividades internas. El mapa de coocurrencias (Figura 2) muestra una fuerte asociación entre los códigos vinculados a plazos, estimación, bugs, retrabajo y deuda técnica, mientras que el diagrama de Pareto (Figura 3) concentra precisamente esos factores como los más frecuentes en los discursos. En la práctica, cuando el tiempo escasea, las pruebas unitarias quedan relegadas frente a la presión por cumplir la fecha de entrega.

Finalmente, en patrones, estándares y documentación, así como en trabajo en equipo y gestión de incidencias, la situación es desigual. La matriz de triangulación recoge casos donde existen documentos y estándares para orientar el desarrollo y para instruir a los nuevos miembros, junto con otros donde todo descansa en la experiencia individual y en la transmisión informal del conocimiento. En los mecanismos de validación predominan las pruebas con el usuario y la priorización de incidencias en coordinación con QA o con el implementador frente al cliente, más que esquemas formales de aseguramiento de la calidad interna automatizada. Leída en conjunto, la evidencia de la Tabla 2 y de las figuras sugiere encadenamientos consistentes: estimaciones ajustadas, plazos rígidos, criterios de cierre ambiguos, revisión de código poco formal e implementación irregular de estándares

configuran un entorno en el que las pruebas unitarias se postergan o se omiten, se normaliza “apagar incendios” mediante retrabajo y se incrementa la deuda técnica.

4.2. Discusión de los resultados

Objetivo específico 1: Comprender las razones, creencias y barreras que llevan a los equipos de desarrollo a no incorporar pruebas unitarias en su práctica cotidiana.

En relación con el objetivo específico 1, los resultados mostraron que la ausencia de pruebas unitarias forma parte de una manera de trabajar ya asumida como “normal” por los equipos. La matriz de triangulación y las figuras de resultados evidencian dinámicas centradas en cumplir fechas y entregar funcionalidades, donde la validación se concentra en QA y en el usuario final, y el retrabajo y las correcciones tardías se integran al flujo habitual de trabajo. En ese marco, las pruebas unitarias se perciben como una actividad que “quita tiempo” a las tareas visibles para el negocio y por ello se postergan o se omiten cuando los plazos son ajustados. También se identificaron barreras organizacionales y culturales: inexistencia de una Definition of Done que exija explícitamente pruebas unitarias, estándares de desarrollo poco compartidos, documentación desigual y brechas de formación específica en pruebas.

Este resultado coincide con los antecedentes revisados donde se reporta que las pruebas unitarias suelen ser las menos aplicadas o incluso inexistentes en varios entornos de desarrollo, trasladando la responsabilidad de detectar errores a usuarios funcionales o áreas de QA. En estudios de proyectos empresariales se ha descrito que la presión por cumplir plazos y la prioridad otorgada a la funcionalidad observable lleva a reducir o eliminar actividades de aseguramiento de calidad en etapas tempranas, reforzando esquemas de validación tardía similares a los observados en esta investigación. A la vez, contrasta con experiencias documentadas en organizaciones que sí han incorporado pruebas unitarias y automatización de pruebas como parte de su proceso estándar, donde se reportan mejoras en mantenibilidad, escalabilidad y estabilidad del software. Frente a estos casos, tus

hallazgos muestran un nivel de madurez menor: los profesionales conocen el concepto de pruebas unitarias, pero no lo han interiorizado como práctica no negociable.

Desde el marco teórico, los resultados se entienden a la luz de modelos de calidad como ISO/IEC 25010:2023, que resaltan mantenibilidad y testabilidad como dimensiones centrales de la calidad interna (ISO, 2023), y de los aportes que muestran que las pruebas unitarias y enfoques como TDD mejoran la estructura del código y reducen defectos tempranos (Bissi et al., 2016; Garousi et al., 2020). Estudios empíricos han reportado, sin embargo, baja adopción o coberturas reducidas de pruebas unitarias por decisiones de gestión, presión de tiempo y priorización de funcionalidades visibles (Trautsch et al., 2017; Wang et al., 2024; Altuwaijri et al., 2022), lo que coincide con el peso que, en esta investigación, tienen los plazos ajustados, la cultura de “cumplir fechas”, la ausencia de Definition of Done explícita y la falta de estándares y formación específica en pruebas como barreras para incorporar unit tests en la práctica cotidiana (Boehm & Basili, 2001).

En síntesis, para el objetivo específico 1, la discusión muestra que la decisión de no realizar pruebas unitarias no se explica solo por “falta de tiempo”, sino por la combinación de presión por plazos, cultura de priorizar entregas funcionales, ausencia de una Definition of Done que exija evidencia de pruebas unitarias, estándares poco compartidos y brechas de formación en pruebas. Esto genera un contexto donde la renuncia a las pruebas unitarias se normaliza y deja de percibirse como deuda explícita. La principal contribución de este estudio es reconstruir esas racionalizaciones cotidianas en equipos que desarrollan software en Lima en 2025, ofreciendo una explicación situada de por qué las pruebas unitarias siguen siendo marginales a pesar de su reconocimiento discursivo.

Objetivo específico 2: Explorar cómo la ausencia de pruebas unitarias incide en el flujo de desarrollo, la calidad del producto y la coordinación entre roles.

Respecto al objetivo específico 2, los resultados evidenciaron que la ausencia de pruebas unitarias impacta directamente en el flujo de desarrollo, la calidad del producto y la

coordinación entre roles. La matriz de triangulación y las figuras de resultados (especialmente el diagrama de Ishikawa y el Pareto) muestran que los códigos de retrabajo, errores y bugs, deuda técnica, deadline y estimación concentran buena parte de las menciones. Los participantes describieron ciclos de trabajo marcados por la reapertura de tareas, la corrección de incidencias de versiones anteriores y la necesidad de alternar entre nuevas funcionalidades y errores pendientes. QA y el usuario final se convierten en filtros tardíos de calidad y ciertos módulos, por su fragilidad y deuda técnica acumulada, se vuelven difíciles de modificar. Además, la implementación desigual de estándares, documentación y mecanismos de trabajo en equipo refuerza estas dificultades y genera fricciones entre desarrollo, QA y otros roles.

Estos hallazgos coinciden con los antecedentes que muestran que, cuando se fortalecen los procesos de prueba y se introducen prácticas de automatización y validación temprana, se reduce el retrabajo, se mejora la detección de defectos y se gana previsibilidad en el flujo de desarrollo. Estudios sobre automatización de pruebas en instituciones públicas y organizaciones del sector privado han reportado mejoras en tiempos de ejecución de pruebas, cobertura de regresión y estabilidad de los despliegues cuando se integran pruebas al proceso de desarrollo. En este contexto, los resultados de la investigación muestran el escenario contrario: al no contar con pruebas unitarias, los equipos dependen de pruebas manuales y de la detección de errores en etapas tardías, lo que fragmenta el flujo y convierte el retrabajo en un componente estructural del proceso. De forma complementaria, los antecedentes que describen la implementación de planes de prueba alineados con normas internacionales también señalan que la formalización de actividades, roles y responsabilidades reduce la improvisación; tus resultados indican que, donde esa formalización es parcial o débil, la ausencia de pruebas unitarias se vuelve uno de los síntomas más visibles.

En relación con el impacto de la ausencia de pruebas unitarias en el flujo de desarrollo y la calidad del producto, el marco ISO/IEC 25010:2023 advierte que sin mecanismos que

aseguren mantenibilidad, confiabilidad y testabilidad, aumenta la probabilidad de defectos recurrentes y dificultad para adaptar el software (ISO, 2023), lo que se alinea con los patrones de bugs, retrabajo y deuda técnica observados. La literatura sobre TDD y pruebas unitarias muestra que su uso disciplinado tiende a estabilizar el comportamiento del sistema, mientras que su ausencia desplaza la detección de errores a niveles superiores de prueba o producción, con mayor costo y riesgo (Bissi et al., 2016; Trautsch et al., 2017; Wang et al., 2024; Boehm & Basili, 2001). En contraste, estudios peruanos que fortalecen los pipelines de prueba con planes formales, automatización y normas como ISO/IEC/IEEE 29119 reportan mejoras en tiempos de prueba, detección temprana de errores y estabilidad de despliegues (Díaz Figueredo et al., 2021; Quispe Gutiérrez, 2023; Paz Díaz, 2023; Cabrera et al., 2024; Rojas Illanes, 2023); frente a ello, los hallazgos de esta investigación representan el escenario inverso, donde la falta de pruebas unitarias contribuye a un flujo reactivo centrado en correcciones, sobrecarga al equipo de calidad y tensión en la coordinación entre roles.

V. Conclusiones y recomendaciones

5.1. Conclusiones

5.1.1. Conclusiones respecto al objetivo general

En relación con el objetivo general, se concluyó que la investigación permitió comprender e interpretar de manera integrada cómo la combinación de factores técnicos, temporales y humano–organizacionales sostenía la ausencia de pruebas unitarias en el desarrollo de software en Lima en 2025 y cómo ello repercutía en la calidad del producto y en la coordinación entre roles. Los hallazgos mostraron que la práctica de pruebas unitarias ocupaba un lugar marginal porque convivía con estimaciones optimistas y plazos rígidos, una cultura de entrega orientada a “sacar” funcionalidades aun a costa de sacrificar verificación temprana, criterios de cierre poco claros y una aplicación desigual de estándares, guías y documentación. Esta configuración empujaba a reemplazar las pruebas unitarias por validaciones manuales tardías, normalizaba el retrabajo y favorecía la acumulación de deuda

técnica, afectando tanto la estabilidad del software como la carga operativa de desarrolladores, líderes técnicos y QA.

A partir de la triangulación de entrevistas, la matriz de categorización, el diagrama de Ishikawa, el mapa de co-ocurrencias y el diagrama de Pareto, se logró una comprensión global de cómo estos factores se encadenaban en la práctica cotidiana: la presión por cumplir fechas llevaba a recortar actividades de aseguramiento como las pruebas unitarias; ello incrementaba la probabilidad de errores y retrabajo; el retrabajo consumía capacidad del equipo y reforzaba la cultura de “apagar incendios”; y, en este contexto, la coordinación entre roles se organizaba alrededor de correcciones urgentes más que de prevención sistemática. De este modo, el objetivo general se cumplió al ofrecer una lectura articulada y situada de por qué, aun contando con marcos conceptuales y evidencia que respaldaban la importancia de las pruebas unitarias, estas seguían siendo postergadas en el ecosistema limeño de desarrollo de software, y cómo esa postergación impactaba en la calidad y en la forma en que los equipos se coordinaban para entregar valor.

5.1.2. Conclusiones respecto a objetivos específicos

Se concluyó que la ausencia de pruebas unitarias en los equipos de desarrollo analizados se explicó por la combinación de presión por cumplir plazos, estimaciones optimistas y una cultura orientada a la entrega rápida de funcionalidades, aun a costa de sacrificar la verificación temprana. Las entrevistas mostraron que las pruebas unitarias ocupaban un lugar marginal frente a otras actividades visibles para el negocio y eran las primeras en recortarse cuando el tiempo resultaba insuficiente. Esta lógica favorecía que los defectos se detectaran recién en etapas tardías del ciclo de vida —en QA, con el usuario final o incluso en producción—, lo que coincidía con lo planteado por la literatura sobre el incremento de costos cuando los errores se descubren en fases avanzadas del desarrollo (Boehm & Basili, 2001; Rojas Illanes, 2023).

Además, el estudio reveló barreras organizacionales y culturales que reforzaban esa ausencia de pruebas unitarias, como la falta de una Definition of Done que exigiera explícitamente evidencias de pruebas, la inexistencia de estándares compartidos de calidad y la delegación del aseguramiento casi exclusivamente al área de QA. Se observó también una brecha en la formación de los desarrolladores respecto al diseño, automatización y mantenimiento de pruebas unitarias, lo que contribuía a percibirlas como una práctica costosa y difícil de sostener. Estos hallazgos dialogaron con antecedentes que enfatizaron la importancia de procesos de prueba definidos, automatización y uso de herramientas adecuadas en contextos complejos como microservicios o empresas con planes de pruebas basados en normas internacionales (Laura Mamani, 2023; Cabrera et al., 2024), pero mostraron que, en el contexto estudiado, tales enfoques aún no se habían traducido en una cultura de pruebas unitarias sistemática.

También se concluyó que la ausencia de pruebas unitarias impactaba directamente en el flujo de desarrollo, en la calidad del producto y en la coordinación entre roles. La información recogida evidenció ciclos recurrentes de retrabajo, reapertura de historias y corrección de incidencias que interrumpían la planificación, generaban sobrecarga en el equipo y reforzaban una dinámica reactiva orientada a “apagar incendios”. Se observó que módulos considerados críticos o “frágiles” resultaban difíciles de modificar por la falta de una red de pruebas unitarias que respaldara los cambios, lo que incrementaba la deuda técnica y limitaba la capacidad de evolución de los sistemas. Estos resultados contrastaron con estudios que mostraron mejoras en tiempos, cobertura y detección temprana de defectos cuando se formalizaban y automatizaban las pruebas en organizaciones públicas y privadas, resaltando que el fortalecimiento del proceso de pruebas contribuía a estabilizar el flujo de trabajo y a reducir problemas en producción.

El diagrama de Pareto concentró en los primeros lugares los códigos asociados a retrabajo y carga de QA, errores y bugs, deuda técnica, deadlines y estimaciones, patrones/estándares, documentación/know-how y Definition of Done, lo que permitió

visualizar cómo la ausencia de pruebas unitarias se articulaba con otros factores estructurales del proceso de desarrollo. Se evidenció que la falta de pruebas unitarias no constituía un problema aislado, sino un síntoma de una brecha más amplia en la formalización del proceso y en la cultura de calidad: donde existían criterios claros de cierre, estándares compartidos y documentación viva, la introducción de prácticas de prueba resultaba más factible; allí donde estos elementos estaban ausentes, se normalizaba el trabajo reactivo y se consolidaba un estilo de gestión centrado en la corrección tardía de errores, convirtiendo dichos factores en componentes estructurales del día a día de los equipos.

5.2. Recomendaciones

Formalizar la Definición de Hecho incorporando evidencias de pruebas unitarias

A partir de la convergencia observada en torno a los criterios de cierre difusos, se recomienda que los equipos y organizaciones revisen y formalicen su Definición de Hecho, incorporando explícitamente evidencias de calidad interna antes de considerar una tarea como “lista”. En línea con la literatura que destaca el rol de las pruebas unitarias como primera barrera de control (Rojas Illanes, 2023; Laura Mamani, 2023), esta Definición de Hecho debería incluir, al menos, la ejecución de pruebas unitarias relevantes, la revisión de código por pares y la actualización de documentación asociada. Con ello se busca que las pruebas unitarias dejen de ser una actividad “deseable si hay tiempo” y pasen a ser un requisito operativo para cerrar trabajo dentro del sprint o iteración.

Ajustar prácticas de estimación y gestión de plazos para no recortar las pruebas

Dado que el análisis muestra que el binomio estimación–plazo empuja a posponer o eliminar pruebas unitarias, se recomienda revisar las prácticas de estimación de esfuerzo y de compromiso de fechas, de modo que el tiempo de diseño y ejecución de pruebas unitarias quede explícitamente contemplado. Los hallazgos de tu investigación y los antecedentes revisados evidencian que omitir esta reserva de tiempo traslada el costo a etapas posteriores en forma de retrabajo y deuda técnica (Boehm & Basili, 2001; Cabrera et al., 2024). Incorporar

explícitamente las actividades de prueba en la estimación y en la planificación no solo alinea las prácticas con lo recomendado por la literatura, sino que también reduce la percepción de que las pruebas unitarias son un “lujo” que puede sacrificarse frente a la presión temporal.

Fortalecer estándares, documentación y transferencia de conocimiento sobre pruebas unitarias

La evidencia de heterogeneidad en patrones/estándares y de brechas en documentación/know-how sugiere la necesidad de fortalecer los marcos de referencia internos. Se recomienda avanzar hacia la adopción y adaptación de estándares de prueba como ISO/IEC/IEEE 29119, que en los estudios revisados se asocian con una mayor formalización del ciclo de pruebas, mejor trazabilidad y menor variabilidad en la calidad de las entregas (Cabrera et al., 2024). Complementariamente, se propone fomentar prácticas de documentación viva (guías actualizadas, ejemplos de casos de prueba, plantillas) y espacios regulares de transferencia de conocimiento (revisiones de código, sesiones de aprendizaje entre pares), en línea con lo que la literatura reporta sobre el rol del Code review moderno como mecanismo de control técnico y aprendizaje colectivo (Bacchelli & Bird, 2013). Todo ello apunta a que la práctica de pruebas unitarias deje de depender de la experiencia de unos pocos y se convierta en un patrimonio compartido del equipo.

Desarrollar competencias en pruebas unitarias y automatización apoyadas en experiencias locales

Finalmente, considerando los antecedentes que muestran la relevancia de competencias técnicas y blandas para sostener procesos de prueba robustos (Espinoza Calderón, 2025; Paz Díaz, 2023), se recomienda diseñar planes de formación focalizados en pruebas unitarias y automatización que tomen como insumo tanto la literatura como las situaciones descritas por los participantes de este estudio. Dichos planes deberían abordar desde el diseño de pruebas unitarias efectivas y su integración en pipelines de integración continua, hasta habilidades de comunicación y coordinación necesarias para negociar plazos y criterios de

calidad dentro de los equipos. De este modo, las organizaciones pueden transformar la ausencia de pruebas unitarias, identificada en tu investigación, en una oportunidad para desarrollar capacidades que mejoren la calidad, la estabilidad de despliegues y la sostenibilidad del desarrollo en Lima, Perú.

VI. Referencias bibliográficas

Altuwaijri, F. S., Qureshi, M. R. J., Hussain, M., & Ajmal, M. M. (2022). Factors affecting agile adoption: An industry research study. *Journal of Systems and Software*, *192*, 111414. https://doi.org/10.1016/j.jss.2022.111414

Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. *Proceedings of the 2013 International Conference on Software Engineering, 712–721*. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ICSE202013-codereview.pdf

Bissi, W., de Freitas, R., de Mello, R. M., & Goldman, A. (2016). The effects of test-driven development on internal quality, external quality and productivity: A systematic review. *Information and Software Technology*, *74*, 45–54. https://doi.org/10.1016/j.infsof.2016.01.004

Boehm, B., & Basili, V. (2001). Software defect reduction top 10 list. *Computer*, *34*(1), 135–137. https://doi.org/10.1109/2.962984

Cabrera, L., Landa, I. y Vindel, J. (2024). Estudio y análisis sobre la creación e implementación de un plan en el proceso de pruebas de software, basado en la Norma ISO/IEC/IEEE 29119 apartados: 1:2022, 2:2021; 3:2021; 4:2021, aplicado al área de control de calidad en la empresa system out of the box, el salvador. [Tesis de maestría no publicada]. Universidad Don Bosco. http://hdl.handle.net/11715/2707

Espinoza, P. (2025). Competencias desarrolladas en el proceso de optimización de pruebas de software en la empresa Globant [Trabajo de suficiencia profesional de licenciatura, Universidad Continental]. Repositorio Institucional Continental. https://repositorio.continental.edu.pe/handle/20.500.12394/17277

Díaz Figueredo, L., Lazo Alvarado, Y., & Tamayo Oro, L. (2021). Proceso de pruebas de software para un modelo de calidad en Cuba. *I+D Tecnológico*, *17*(1), 23-35. https://doi.org/10.33412/idt.v17.1.2914

Garousi, V., Felderer, M., & Hacaloğlu, T. (2020). Exploring the industry's challenges in software testing: An empirical study. *Journal of Software: Evolution and Process*, *32*(11), e2251. https://doi.org/10.1002/smr.2251

ISO. (2023). *ISO/IEC 25010:2023 — Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. International Organization for Standardization. https://www.iso.org/standard/78176.html

Laura Mamani, C. A. (2023). Software testing for microservices. *Innovation and Software*, *4*(1), 151–160. https://doi.org/10.48168/innosoft.s11.a86

Paz Díaz, J. E. (2023). Implementar automatización de pruebas usando ChatGPT para la mejora del proceso de calidad de software en empresas del sector financiero en Lima, Perú 2023. https://hdl.handle.net/20.500.12867/8761

Quispe Gutiérrez, J. K. (2023). Automatización de pruebas funcionales y pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú, 2022. https://hdl.handle.net/20.500.12892/763

Rojas Illanes, S. (2023). *Refactorización e implementación de pruebas unitarias en el motor de Transacciones Digitales de Instance*. Repositorio Académico de la Universidad de Chile. https://repositorio.uchile.cl/handle/2250/198851

Sagástegui, L. A. C., Samaniego, H. H., Cuadros, D. L., & Raymundo, A. F. N. (2022). Nuevo modelo basado en desarrollo ágil para mejorar la implementación de un sistema integrado. *Ñawparisun – Revista de Investigación Científica de Ingenierías*, *4*(1). https://doi.org/10.47190/nric.v4i1.8

Trautsch, F., et al. (2017). Are there any unit tests? An empirical study on unit testing in open-source Java projects. University of Göttingen (Tech. Report).

Wang, H., et al. (2024). Beyond accuracy: An empirical study on unit testing in open-source projects. *ACM Transactions on Software Engineering and Methodology*. https://dl.acm.org/doi/10.1145/3638245

Anexo 1

Informe Turnitin

CESAR RAUL NOLASCO HUAMANCHUMO

La ausencia de pruebas unitarias en el desarrollo de software en Lima%2C Per%C3%BA en 2025.docx

Instituto San Ignacio de Loyola - ISIL

Detalles del documento

Identificador de la entrega
trn:oid:::30163:539244614

Fecha de entrega
10 dic 2025, 7:40 p.m. GMT-5

Fecha de descarga
17 dic 2025, 7:21 p.m. GMT-5

Nombre del archivo
La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú en 2025.docx

Tamaño del archivo
5.7 MB

76 páginas

17.461 palabras

100.321 caracteres

10% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Coincidencias menores (menos de 10 palabras)

Fuentes principales

9% Fuentes de Internet
1% Publicaciones
6% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.° de alertas de integridad para revisión

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo

Anexo 2

Registro de Impacto y Resultados

Tipo de documento: Trabajo de Investigación

Título del Trabajo de Investigación o Tesis:

"La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú en 2025"

Integrante:

1. Nolasco Huamanchumo, Cesar Raul

Asesor: Peláez Valdivieso, José Víctor

Impacto de la investigación

El impacto de una investigación se refiere a los efectos, tanto esperados como inesperados, que esta puede generar, abarcando aspectos económicos, políticos, culturales, ambientales, tecnológicos, sociales, entre otros.

Tecnológico: La identificación de factores que explican la ausencia de pruebas unitarias permitirá a las organizaciones tecnológicas limeñas comprender mejor las barreras que enfrentan y diseñar estrategias para superarlas, lo que contribuirá a la mejora de la calidad del software y a la reducción de defectos en producción.

Económico: La implementación de pruebas unitarias sistemáticas reducirá el retrabajo recurrente y la deuda técnica acumulada, lo que generará ahorros significativos en costos operativos al disminuir la necesidad de correcciones en etapas tardías del desarrollo.

Organizacional: Los hallazgos servirán como base para que las empresas revisen y fortalezcan sus procesos de desarrollo, incorporando prácticas de calidad fundamentales como la definición explícita de "hecho" (Definition of Done) que incluya evidencia de pruebas unitarias, mejorando así la coordinación entre roles y reduciendo la sobrecarga que recae actualmente en el área de calidad.

Resultado del proceso de investigación

Los resultados de un proyecto de investigación son los descubrimientos o conclusiones alcanzadas después de realizar el estudio. Estos reflejan los datos obtenidos durante el proceso investigativo y responden a las preguntas o hipótesis formuladas al comienzo del proyecto. Los resultados son fundamentales para evaluar, interpretar y comprender los efectos o la validez de lo investigado.

Los resultados mostraron que la ausencia de pruebas unitarias responde a factores interconectados: presión por plazos ajustados, estimaciones centradas únicamente en funcionalidad visible, cultura que prioriza "sacar el sistema" sobre la calidad estructural, y la falta de una Definition of Done explícita que exija evidencia de pruebas. Los equipos determinan fechas límite mediante estimaciones en conjunto, por decisión individual o acatando decisiones externas, siempre añadiendo márgenes insuficientes para pruebas unitarias.

Esta ausencia genera efectos concretos: detección tardía de defectos en QA o producción, ciclos recurrentes de retrabajo, acumulación progresiva de deuda técnica y mayor incertidumbre en despliegues. Los equipos dependen de mecanismos reactivos como pruebas manuales, logs de errores y Excel de incidencias, sobrecargando al área de calidad y postergando correcciones críticas.

La coordinación entre roles se fragmenta: desarrollo, QA y usuarios finales trabajan en ciclos desalineados, con QA funcionando como red de seguridad en lugar de estrategia preventiva. Los criterios de "terminado" varían según la organización (documentos funcionales, historias de usuario o simplemente lo que pide el cliente), sin incluir explícitamente pruebas unitarias como requisito no negociable.

Aunque los desarrolladores reconocen la importancia de las pruebas unitarias, estas no se interiorizan como práctica obligatoria debido a la presión temporal constante, la formación limitada en testing y decisiones organizacionales que relegan la calidad de código. La investigación concluye que esta ausencia no es un problema técnico aislado, sino un síntoma de cómo se gestionan el tiempo, las prioridades y la calidad en los equipos de desarrollo limeños.

Anexo 3

Tabla 3

Matriz de categorización apriorística

Categoría	Ámbito temático	Problema de investigación	Objetivos	Subcategorías	Códigos	Preguntas de investigación
Ausencia de pruebas unitarias	Los equipos de desarrollo de software de Lima omiten el desarrollo de pruebas unitarias por diversos motivos.	¿Cómo se manifiesta la ausencia de pruebas unitarias en las prácticas de desarrollo y calidad y qué implicancias perciben los actores en la confiabilidad y mantenibilidad del software?	General: Comprender e interpretar los factores técnicos, temporales y humano-organizacionales que inciden en la ausencia de pruebas unitarias en proyectos de desarrollo de software en Lima, Perú, así como sus efectos percibidos en la calidad del producto y en la coordinación del trabajo entre roles.	Errores / bugs	Frecuencia	¿Con qué frecuencia encuentran bugs en el código que desarrollan?
					Retrabajo-QA	¿Cómo manejan los bugs encontrados en desarrollos anteriores?
					Deuda-técnica	¿Cuál es el impacto de estos bugs en términos de retrabajo?
					Prevención	¿Qué estrategias emplean para minimizar la incidencia de errores en el desarrollo de software?
				Deadline / Fecha límite	Estimación-proceso	¿Cómo determinan las fechas límite para los proyectos en los que trabajas?
					Cumplimiento-nivel	¿Cuál es el nivel de cumplimiento de las fechas límite en tu equipo de trabajo?
					Restraso	¿Cuáles son las principales causas de retrasos en la entrega de proyectos?
					Gestión-retrabajo	Si existe retrabajo, ¿incide en el retraso de la entrega?
				Trabajo en equipo	Revisión-pares	¿Qué tan estricta es la política de tu empresa respecto al seguimiento de buenas prácticas y patrones de diseño?

Desarrollo de software	EL desarrollo de software tiene muchas herramientas para generar un producto de calidad, pero depende de la cultura, el conocimiento y la experiencia.	¿Cómo se relaciona la ausencia de pruebas unitarias con elementos clave del proceso de desarrollo y qué efectos percibidos tiene sobre calidad, coordinación y entregas?	OE1: Comprender las razones, creencias y barreras (técnicas, culturales y de gestión) que sostienen la no adopción de pruebas unitarias en equipos de desarrollo en Lima, explorando experiencias, decisiones cotidianas y trade-offs reportados por desarrolladores, líderes técnicos y QA.	Buenas prácticas	Patrones-estándares	¿Estás familiarizado con las buenas prácticas de programación y patrones de diseño?
					Código-existente	¿Consideras que el código existente en tu empresa sigue estas buenas prácticas?
					Concepto-uso	¿En tu trabajo usan el término “definición de hecho”? ¿Cómo lo definirían?
				Definición de hecho / Definition of Done (DoD)	Criterios-uso	¿Cómo definirían el concepto de “definición de hecho” en su equipo de trabajo?
					Verificación-trazabilidad	¿Qué mecanismos utilizan para asegurar que el producto final cumpla con los requisitos del cliente o la definición de hecho?
					Conocimiento-uso	¿Tienes experiencia con la programación guiada por pruebas (TDD)?
				Programación guiada por pruebas (TDD)	Uso-aplicación	¿En qué medida se utiliza la programación guiada por pruebas en las empresas donde has trabajado?
					No-adopción	¿Cuáles son las principales razones por las que crees que no se adopta ampliamente la programación guiada por pruebas?
					Necesidad-adopción	¿Cuál es tu opinión sobre la necesidad de realizar pruebas unitarias?
					Tiempo-desarrollo	¿Has observado que el desarrollo se alarga debido a la implementación de pruebas unitarias?

Anexo 4

Validación

INFORME DE JUICIO DE EXPERTOS DEL INSTRUMENTO DE INVESTIGACIÓN

I. DATOS GENERALES:

1.1. Apellidos y Nombres del experto: Felix Martin Adrian Caceres Padilla

1.2. Cargo e institución del experto: Coordinador de Desarrollo en TBS del Perú
Magister en Ingeniería Informática

1.3. Nombre del instrumento: Guía de entrevista

1.4. Autor del instrumento: Cesar Raul Nolasco Huamanchumo

1.5. Título de la investigación: La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú en 2025

II. ASPECTOS DE VALIDACIÓN:

CRITERIOS	INDICADORES	Deficiente 00-20%	Regular 21-40%	Buena 41-60%	Muy buena 61-80%	Excelente 81-100%
1. CLARIDAD	Está formulado con lenguaje apropiado y específico.				X	
2. OBJETIVIDAD	Está expresado en conductas observables.				X	
3. ACTUALIDAD	Adecuado al avance de la ciencia y tecnología.				X	
4. ORGANIZACIÓN	Existe organización lógica.				X	
5. SUFICIENCIA	Comprende los aspectos en cantidad y calidad.				X	
6. INTENCIONALIDAD	Adecuado para valorar aspectos de las estrategias.				X	
7. CONSISTENCIA	Basados en aspectos teóricos-científicos.				X	
8. COHERENCIA	Entre los índices, indicadores y dimensiones.				X	
9. METODOLOGÍA	La estrategia responde al				X	

	propósito del diagnóstico.					
10. PERTINENCIA	El instrumento es funcional para el propósito de la investigación.				x	
PROMEDIO DE VALIDACIÓN						x

PERTINENCIA DE LOS ÍTEMS O REACTIVOS DEL INSTRUMENTO

INSTRUMENTO	SUFICIENTE	MEDIANAMENTE SUFICIENTE	INSUFICIENTE
Ítem 1	x		
Ítem 2	x		
Ítem 3	x		
Ítem 4	x		
Ítem 5	x		
Ítem 6	x		
Ítem 7	x		
Ítem 8	x		
Ítem 9	x		
Ítem 10	x		
Ítem 11	x		
Ítem 12	x		
Ítem 13	x		
Ítem 14	x		
Ítem 15	x		
Ítem 16	x		
Ítem 17	x		
Ítem 18	x		
Ítem 19	x		

III. PROMEDIO DE VALORACIÓN: 80 %.

IV. OPINIÓN DE APLICABILIDAD:

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

( ) El instrumento debe ser mejorado antes de ser aplicado.

Lugar y fecha: Lima, 01/11/2025

Firma del experto: _____

DNI N°: 46127857

Anexo 5

Guía de entrevista a profundidad

Estudio: La ausencia de pruebas unitarias en el desarrollo de software en Lima, Perú

Versión: 1.0

Investigador/a: Cesar Nolasco

1. Propósito y alcance

Explorar cómo se manifiesta la ausencia de pruebas unitarias en proyectos de desarrollo de software en Lima, Perú, y cómo esta se relaciona con la calidad del producto y la organización del trabajo, tomando como referencia las categorías, subcategorías y códigos definidos en la matriz de categorización apriorística.

2. Tipo de entrevista y justificación

Entrevista semiestructurada: se emplea un conjunto de preguntas base asociadas a las categorías y subcategorías de la matriz, con libertad para repreguntar, sondar y ajustar el orden según el flujo natural de la conversación, preservando comparabilidad entre casos.

3. Perfil del participante

Profesionales involucrados en el desarrollo de software en Lima, Perú, tales como:

- Desarrolladores/as (backend, frontend, full stack).
- Líderes técnicos o arquitectos de software.
- Project managers con responsabilidad directa sobre proyectos de desarrollo.

Con experiencia directa en proyectos de software recientes (últimos años) y participación en decisiones o prácticas relacionadas con calidad y pruebas.

4. Logística y consideraciones éticas

Duración estimada: 45–60 minutos.

Confidencialidad: Seudonimización/anonimización de nombres de personas y empresas.

Consentimiento: Solicitar consentimiento informado y permiso para grabación de audio.

Almacenamiento: Resguardo seguro de registros y transcripciones, solo para fines académicos.

5. Guion operativo

Apertura (script sugerido):

"Gracias por participar. El objetivo es comprender prácticas y decisiones en torno a las pruebas unitarias y al proceso de desarrollo. Tu participación es voluntaria; puedes omitir cualquier pregunta. Con tu permiso, grabaremos para asegurar precisión. La información será tratada de forma confidencial."

Calentamiento: rol actual, años de experiencia, tipo de proyectos y tecnologías principales.

Cierre: Agradecimiento, verificación de contacto para aclaraciones y consulta final: "¿Hay algo relevante que no pregunté?"

6. Sondas neutrales

• ¿Podrías dar un ejemplo concreto?

• ¿Qué hicieron exactamente / quiénes participaron / con qué criterios?

• ¿Qué pasó después / cómo lo midieron / qué evidencia usaron?

• ¿Qué obstáculos enfrentaron y cómo los resolvieron?

• Si pudieras cambiar algo hoy, ¿qué cambiarías y por qué?

7. Cuerpo de la guía

Categoría 1: Ausencia de pruebas unitarias

Subcategoría: Errores / bugs — Códigos: Frecuencia / Deuda–técnica / Prevención

1) ¿Con qué frecuencia encuentran bugs en el código que desarrollan?

2) ¿Cómo manejan los bugs encontrados en desarrollos anteriores?

3) ¿Cuál es el impacto de estos bugs en términos de retrabajo?

4) ¿Qué estrategias emplean para minimizar la incidencia de errores en el desarrollo de software?

Subcategoría: Deadline / Fecha límite — Códigos: Estimación–proceso / Cumplimiento–nivel / Retraso / Gestión–retrabajo

5) ¿Cómo determinan las fechas límite para los proyectos en los que trabajas?

6) ¿Cuál es el nivel de cumplimiento de las fechas límite en tu equipo de trabajo?

7) ¿Cuáles son las principales causas de retrasos en la entrega de proyectos?

8) Si existe retrabajo, ¿incide en el retraso de la entrega?

Categoría 2: Desarrollo de software

Subcategoría: Buenas prácticas — Códigos: Patrones–estándares / Revisión–pares / Código–existente

9) ¿Estás familiarizado con las buenas prácticas de programación y patrones de diseño?

10) ¿Qué tan estricta es la política de tu empresa respecto al seguimiento de buenas prácticas y patrones de diseño?

11) ¿Consideras que el código existente en tu empresa sigue estas buenas prácticas?

Subcategoría: Definición de hecho / Definition of Done (DoD) — Códigos: Concepto–uso / Criterios–uso / Verificación–trazabilidad

12) ¿En tu trabajo usan el término “definición de hecho” (Definition of Done)?

13) ¿Cómo definirían el concepto de “definición de hecho” en su equipo de trabajo?

14) ¿Qué mecanismos utilizan para asegurar que el producto final cumpla con los requisitos del cliente o definición de hecho”?

Subcategoría: Programación guiada por pruebas (TDD) — Códigos: Conocimiento–uso / Uso–aplicación / No–adopción / Necesidad–adopción / Tiempo–desarrollo

15) ¿Tienes experiencia con TDD (programación guiada por pruebas)?

16) ¿En qué medida se utiliza la programación guiada por pruebas en las empresas donde has trabajado?

17) Si no utilizan TDD, ¿por qué razones?

18) ¿Perciben que existe una necesidad de adoptar TDD?

19) ¿Cómo perciben el impacto de TDD en el tiempo de desarrollo?