Universidad Nacional Federico Villarreal

VRIN | VICERRECTORADO DE INVESTIGACIÓN

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

IMPLEMENTACIÓN DE UN MODELO DE DESARROLLO DE SOFTWARE PARA MEJORAR LA CALIDAD DEL PRODUCTO SAAS ERP

Línea de investigación:
Ingeniería de software, simulación y desarrollo de TICs

Tesis para optar el Título Profesional de Ingeniero de Sistemas

Autor

Cuadra Rivera, Carlos Joaquín

Asesor

Alfaro Bernedo, Juan Oswaldo

ORCID: 0000-0002-9803-5986

Jurado

Flores Vidal, Higinio Exequiel

Bazán Briceño, José Luis

Ccasani Allende, Julián

Lima - Perú

2024

TESIS_Carlos_Joaquin_Cuadra_Rivera_Version_08102024.docx

INFORME DE ORIGINALIDAD

30%	28%	11%	20%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorio.unfv.edu.pe Fuente de Internet	2%
2	hdl.handle.net Fuente de Internet	1%
3	docobook.com Fuente de Internet	1%
4	www.armadilloamarillo.com Fuente de Internet	1%
5	doi.org Fuente de Internet	1%
6	repositorio.ucv.edu.pe Fuente de Internet	1%
7	cynoteck.com Fuente de Internet	1%
8	repositorio.concytec.gob.pe Fuente de Internet	1%
9	Submitted to Pontificia Universidad Catolica del Ecuador - PUCE	1%

Universidad Nacional Federico Villarreal

VRIN | VICERRECTORADO DE INVESTIGACIÓN

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

IMPLEMENTACIÓN DE UN MODELO DE DESARROLLO DE SOFTWARE PARA MEJORAR LA CALIDAD DEL PRODUCTO SAAS ERP

Línea de investigación:
Ingeniería de Software, simulación y desarrollo de TIC's

Tesis para optar el Título Profesional de Ingeniero de Sistemas

Autor:
Cuadra Rivera, Carlos Joaquín

Asesor:
Alfaro Bernedo, Juan Oswaldo
ORCID: 0000-0002-9803-5986

Jurado:
Flores Vidal, Higinio Exequiel
Bazán Briceño, José Luis
Ccasani Allende, Julián

Lima - Perú
2024

INDICE

INDICE DE TABLAS

INDICES DE FIGURAS

RESUMEN

El propósito de esta tesis fue identificar herramientas y metodologías que puedan conformar una nueva estrategia para mejorar la calidad de un sistema ERP en la nube bajo la modalidad Software como Servicio (SaaS). Además, se analizó cómo estas herramientas interactúan entre sí para contribuir a dicha mejora. Para alcanzar este objetivo, se validaron hipótesis utilizando la herramienta estadística SPSS, versión 26. La metodología de investigación adoptó un enfoque aplicativo, no paramétrico y preexperimental, desarrollada en un periodo de 10 meses durante el año 2023. Se recopilaron datos a partir de una muestra de 265 programas del sistema SaaS ERP, registrando los efectos antes y después de la implementación de la nueva metodología. Esto permitió comparar ambas muestras y evaluar si existían diferencias significativas. Asimismo, se aplicó una encuesta de 16 preguntas en escala de Likert a 20 usuarios del sistema, con el fin de conocer sus percepciones sobre el impacto experimentado tras el cambio metodológico. Los resultados evidenciaron una relación significativa entre la nueva metodología y la mejora en la calidad del sistema ERP. Este hallazgo se sustentó con la prueba de Wilcoxon, cuyo coeficiente de correlación arrojó un valor sigma de 0,000**, muy por debajo del umbral de significancia teórica de 0,05. Esto llevó a la aceptación de la hipótesis alternativa, confirmando que la implementación de esta metodología tiene un impacto positivo en la calidad del sistema evaluado.

Palabras clave: SaaS ERP, calidad de software, mejora de procesos.

ABSTRACT

The purpose of this thesis was to identify tools and methodologies that could form a new strategy to improve the quality of a cloud-based ERP system under the Software as a Service (SaaS) model. Additionally, the study analyzed how these tools interact to contribute to such improvement. To achieve this objective, hypotheses were validated using the statistical software SPSS, version 26. The research methodology followed an applied, non-parametric, and pre-experimental approach, conducted over a period of 10 months during 2023. Data were collected from a sample of 265 SaaS ERP system programs, recording effects before and after the implementation of the new methodology. This allowed for comparison between both samples to determine whether significant differences existed. Furthermore, a 16-question Likert scale survey was administered to 20 system users, aiming to gather their perceptions regarding the impact experienced after the methodological change. The results showed a significant relationship between the new methodology and the improvement in ERP system quality. This finding was supported by the Wilcoxon test, whose correlation coefficient yielded a sigma value of 0.000**, well below the theoretical significance threshold of 0.05. This led to the acceptance of the alternative hypothesis, confirming that the implementation of this methodology has a positive impact on the quality of the evaluated system.

Keywords: SaaS ERP, software quality, process improvement.

I. INTRODUCCION

1.1 Descripción y formulación del Problema

Los sistemas de tipo "Enterprise resource planning" o Planificación de Recursos Empresariales" (ERP) son esenciales para la gestión empresarial, pero solían ser costosos y complejos de implementar. La adopción de soluciones de ERP basadas en la nube y el Software como Servicio (conocido como SaaS) está cambiando la forma en que las empresas los ven. Mover los ERP a la nube simplifica los requisitos tecnológicos y acelera el retorno de la inversión. Las soluciones en la nube están ofreciendo la misma funcionalidad que las locales y se integran fácilmente con otras tecnologías. La popularidad de los servicios en la nube está en aumento debido a su funcionalidad moderna y ahorro de costos. Para el éxito de un sistema SaaS ERP, es crucial aprovechar suites integradas y plataformas en la nube que ofrezcan alto rendimiento, escalabilidad y seguridad mejorada. (Oracle Net Suite, 2022)

El mercado de ERP se encuentra en una fase de expansión rápida, y se espera que el tamaño total del mercado supere los $49.5 mil millones para el año 2025.

Los servicios digitales han experimentado un aumento significativo durante la pandemia de Covid-19, con un enfoque en experiencias de pago sin contacto y compras en línea. Esto representa un desafío y una oportunidad para la transformación digital en muchos negocios. (Oracle NetSuite, 2022)

En una encuesta de IDC revela que, antes de la emergencia sanitaria, las empresas tenían entre dos y diez accesos remotos. Para el año 2023 el 56.5% de empresas elevó su demanda hacia el cloud computing, mientras que el 52.7% aumentó su inversión o planea hacerlo. (Diario El Peruano, 2023)

Para CEPAL" (La Comisión Económica para América Latina y el Caribe), el 42% de los adultos latinoamericanos hace un uso regular de los pagos digitales, y un 11% adoptó este canal de pago a raíz de la pandemia. El uso de monederos electrónicos (*e-wallets*) ha crecido con fuerza al

40% anual, y actualmente supone el 11% de los pagos", destaca el Informe Sociedad Digital en América Latina 2023. (Fundación Telefónica, 2023).

Se espera que la inversión en servicios de computación en la nube en el Perú alcance los $1.491 millones en 2026, según datos proporcionados por Global Data y compartidos por Internexa. Esto corresponde a una tasa de crecimiento anual del 18,4%.

A pesar de este crecimiento, la inversión en computación en la nube en el Perú sigue siendo relativamente baja en comparación con otros países de la región.

Sin embargo, la industria está creciendo significativamente debido a las ventajas de ahorro de costos que ofrece en comparación con la compra de infraestructura costosa.

El citado estudio también muestra que la participación de mercado del Perú en este campo será del 5,7% en 2022, con una inversión de 782 millones de dólares. En comparación, Brasil, Argentina, Chile y Colombia tienen cuotas de mercado del 56%, 13%, 12% y 11,52% respectivamente. (Noticias Ebiz, 2023).

1.1.1 Descripción del problema

Este cambio digital a la nube no es tarea fácil. Las empresas peruanas se están enfrentado a grandes reveses al respecto. Las organizaciones de hoy en día tienen trabajadores remotos, múltiples oficinas y numerosas herramientas y plataformas informáticas para trabajar. Esto requiere una planificación y ejecución eficaces en las aplicaciones por parte de las empresas de software que ofrecen sus servicios a través de la nube para garantizar el buen funcionamiento, la continuidad de las operaciones y un correcto funcionamiento de sus sistemas existentes en el futuro.

Mover las operaciones comerciales y financieras a la nube, aunque beneficioso tiene su propio conjunto de desafíos. Por ejemplo, la seguridad, en la que los expertos aseguran que es una de las principales preocupaciones de muchas empresas.

En lo que va del año 2023, las denuncias por estafas digitales y robos a través de aplicaciones se incrementaron en un 150%, según cifras de la Policía Nacional. Actualmente, se reportan más de cinco casos de cibercrimen por hora, una cifra que no refleja el alcance real de esta nueva amenaza. Frente al 2022, el riesgo de delito digital se incrementó en 17,000 veces gracias a la inclusión de inteligencia artificial en beneficio del crimen, de acuerdo con Kenneth Tovar, Country Manager de la empresa de ciberseguridad Palo Alto Networks. (Veliz, J, 2023).

La empresa Integrator SAC. Ofrece soluciones que optimizan los procesos operativos, facilitando una toma de decisiones más eficaz a través de su producto, Integrator ERP, un software como servicio en la nube (SaaS). La solución proporcionada por la empresa permite gestionar de manera modular e integrada procesos clave, como contabilidad, finanzas, planillas, ventas, logística, facturación y otros.

La adopción de muchas empresas de este producto de tipo SaaS, ha ocasionado un aumento del 25% en la base de clientes de la empresa en comparación con el período 2020-2023. Además, la demanda de nuevos desarrollos y adaptaciones financieras ha experimentado un crecimiento del 30% anual.

Este crecimiento, del uso del software, ha evidenciado deficiencias en el soporte y calidad del producto Integrador ERP. Solucionarlo plantea un desafío significativo para la empresa, ya que debe garantizar la continuidad del servicio mientras se enfrenta a varios problemas como, por ejemplo:

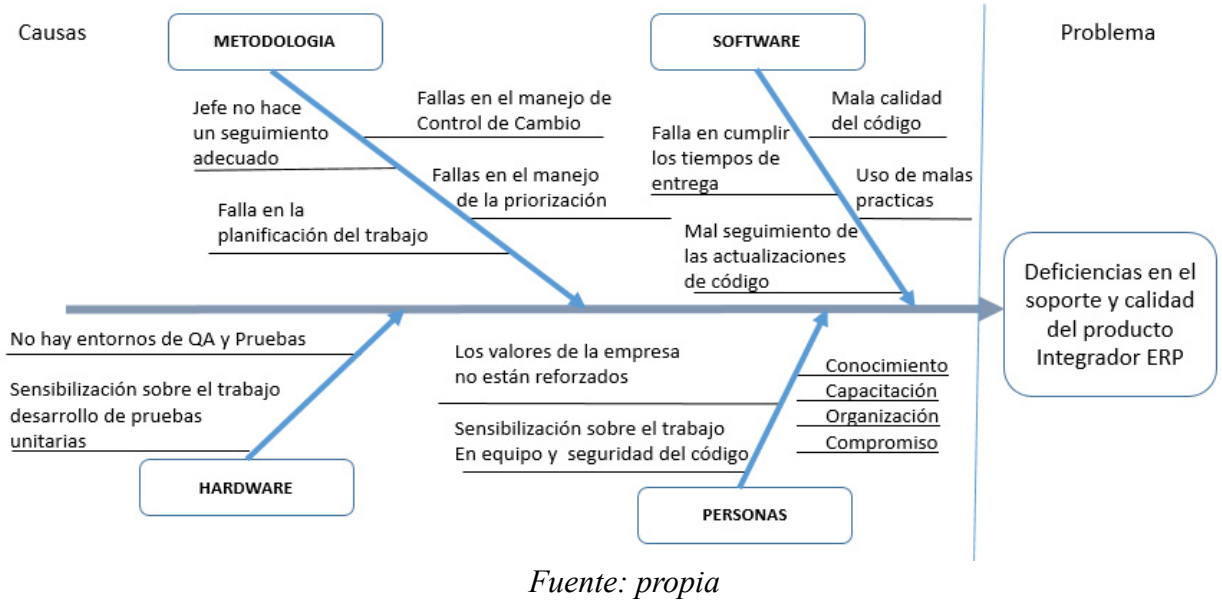
- El incremento de la cartera de clientes y la falta de demanda de cambios sobre el código y la falta de una metodología que sirva como marco a la hora de atender los desarrollos ha afectado la calidad del software, esto se ha visto reflejado en el alto porcentaje de código rechazada (40% del código no pasa las pruebas unitarias) y problemas con el manejo de fuentes (el 30% de los casos no se tiene la certeza que la versión del programa sea la versión vigente porque no se cuenta con un repositorio único de fuentes). Es un punto esencial para el éxito de la empresa, dado que quiere garantizar la satisfacción del cliente, la calidad del

código y mejorar la eficiencia operativa, como también mejorar los tiempos de entrega de los requerimientos y fortalecer la competitividad, la seguridad y la reputación de la empresa.

- Su equipo de software tiene muchos problemas con el manejo de cambios de los requerimientos, el manejo de las prioridades, un manejo adecuado del cambio y realización de pruebas dado que su modelo de desarrollo de software es en cascada.
- Seguridad del código: Se ha identificado un mal manejo de las mejores prácticas en lo que es codificación, dado que hacen uso de "código duro" y mala definición de variable (no manejan un estándar en un 80% de su código).
- No cuentan con un ambiente apropiado de QA para las pruebas unitarias y asegurar un correcto funcionamiento de las aplicaciones antes de ser desplegadas a los ambientes productivos.
- Personal Especializado: Contar con personal capacitado en la nube y en SaaS es crucial. La formación y la actualización constante del equipo son esenciales para mantenerse al día con las mejores prácticas. Actualmente no hay una evidencia de lo mismo.

Figura 1.

Diagrama de Ishikawa sobre la descripción del problema



Fuente: propia

1.1.2 Planteamiento del problema

En virtud de poder llegar a una solución de los problemas identificados, se plantea el uso una metodología para la construcción de un nuevo modelo de desarrollo de software que ayude a mejorar la calidad del software, lo cual involucra los siguientes cuestionamientos:

¿En función a qué criterios se podría seleccionar una metodología para garantizar una mejora calidad del software y una mejor rapidez para manejar el cambio?

¿Una vez implementada la metodología, se tendrá una mejora en el tiempo de la liberación de los requerimientos del software?

¿Una vez implementada la metodología se reducirá el porcentaje de código rechazada del software?

El cumplimiento de estos retos tiene el potencial de generar un impacto significativo en la empresa y en el bienestar de su cartera de cliente, especialmente en un campo tan competitivo como las soluciones SaaS en el Perú.

1.1.3 Formulación del problema

En virtud de los aspectos mencionados en la sección anterior, planteamos la siguiente formulación del problema orientado a revertir la problemática descrita.

Problema General

¿En qué medida la implementación de un modelo de desarrollo de software contribuye a mejorar la Calidad del Producto SaaS ERP?

Problemas Específicos

1.- ¿En qué medida, la implementación de un modelo de desarrollo de software contribuye a reducir los errores en el Producto del SaaS ERP?

2.- ¿En qué medida, la implementación de un modelo de desarrollo de software contribuye a reducir el tiempo de entrega del Producto SaaS ERP?

1.2 Antecedentes

1.2.1 Antecedentes Nacionales

Banda (2022) En la investigación titulada "*Automatización Pruebas De Calidad De Código Para Las Aplicaciones Web Con Herramientas De Código Abiertos*". Precisa que La Implementación del Proyecto de Automatización nos permite disminuir los tiempos de las fases de desarrollo del software, desde el levantamiento de información (Historias de Usuario), las pruebas de calidad y el posterior despliegue a producción. Validando de esa manera la hipótesis respecto a ella misma. Aportando valor a la empresa en agilidad de los procesos (p. 56)

Benites (2022) En la investigación titulada "*Aplicación de Aseguramiento de Calidad de Software y su influencia en proyectos de la empresa 3M*". Concluye y precisa que hay una diferencia significativa en los proyectos de la empresa 3M antes y después de la aplicación de Aseguramiento de Calidad de Software. Es decir, el área testing se encargó de encontrar las observaciones de las aplicaciones de 3M que fueron incluidas en los proyectos en estudio y se evitó que las fallas se presenten en el ambiente de producción al mitigarlos en el ambiente de pruebas. Por lo cual se concluye que la aplicación de Aseguramiento de Calidad de Software SI tiene efectos significativos sobre los proyectos de la empresa 3M (p. 28)

El estudio, titulado "Evaluación de la calidad de productos de software según la norma ISO/IEC 25000: un estudio de caso de un sistema de nómina en la provincia de Chiclayo", responde a la definición ISO de calidad como el nivel en el que un producto específico debe cumplir con las necesidades del usuario final.

La calidad tiene varias características como Relatividad, por su percepción está directamente relacionada con la perspectiva del observador dado el contexto en el que se evalúa. Multidimensional que incluye diferentes aspectos como funcionalidad, usabilidad, confiabilidad y aprendizaje. Restrictivo, ya que deben tener en cuenta limitaciones como el presupuesto asignado, los plazos, la capacidad de infraestructura, la experiencia disponible, etc.

Marín Chaman y Bautista Gutiérrez (2021) También consideran al intentar alcanzar un nivel suficiente de calidad de acuerdo con los estándares establecidos como también se puede evaluarse tanto desde el punto de vista cualitativo como cuantitativo. Estos conceptos e ideas evolucionaron paulatinamente y marcaron la línea de base que siguieron otras empresas y países en el campo de la calidad del software. (p. 16).

Saldaña (2020) En el trabajo de investigación titulada “Influencia Del Uso De La Norma ISO/IEC 25000 En El Nivel De Calidad Del Sistema Snyfuel De La Empresa Grifo 3B” precisa que El uso de la norma ISO/IEC 25000 influye de manera positiva en el nivel de calidad del sistema Snyfuel, ya que al considerar las características que indica del modelo de referencia que proporciona la norma se ha mejorado el nivel de calidad de dicho sistema, en los aspectos de calidad interna, externa y en uso. (p. 79)

Pinpingos (2018) En su estudio titulado "Implementación del control de calidad del software mediante técnicas ágiles centradas en pruebas en TCI Corporation".

Demuestra que los estándares de implementación y desarrollo en áreas como el control de calidad del software van de la mano. Si el proceso de desarrollo también es más estable, el proceso de prueba será más estable y eficiente. Se recomienda ampliar el alcance del control de calidad del software e involucrar al área de control de calidad. (p. 45)

1.2.2 Antecedentes Internacionales

Ribon Ramos y Morales Garzón (2021) En el estudio bajo el título “Diagnóstico Del Sistema De Gestión De La Calidad En El Procedimiento De Análisis, Diseño Y Desarrollo De Software Institucional Bajo Los Lineamientos De La Norma ISO 9001:2015 e ISO/IEC 25001:2014 – Córdova – Argentina”. Determinar el interés por la calidad del software crece en la medida que los usuarios son más exigentes y requieren de productos que cumplan con sus necesidades. Los estándares como son ISO 9001, ISO/IEC 25001 definen criterios que tienen como objetivo orientar la alta calidad y desarrollo del software de manera confiable. En relación con lo

anterior, se suman los desarrollos de las tecnologías digitales, que se convierten en uno de los componentes que promueve una sociedad basada en el conocimiento acompañada de las denominadas Tecnologías de la Información. (p. 7)

Jacobs (2020) En la Tesis doctoral bajo de título "Project Management Maturity: A Framework For Project Success In Sub-Saharan Higher Education Institutions Centres Of Excellence, 2020". Indica que La activación de procesos de cooperación y colaboración afecta el desempeño a través del aumento de la confianza entre los miembros del equipo del proyecto, basado en una comunicación de calidad. Confianza Se vuelve funcional a través de la previsibilidad y las expectativas del comportamiento del equipo. Miembros o la creencia en sus competencias. (p. 46)

Navarro (2020) En la investigación titulada "Modelo De Desarrollo E Implementación De Software Bajo El Marco De Buenas Prácticas Del Cmmi En El Área De Sistemas De Información De La Universidad Autónoma De Bucaramanga". El objetivo es demostrar que La adopción de las practicas propuestas por el CMMI, permitirá al departamento de tecnologías de información ofrecer software de calidad que soporte los procesos que apalancan la operación de la universidad; el modelo de desarrollo de software basado en SCRUM permitirá el involucramiento las partes interesadas en todo momento lo que conlleva a que se pueda reaccionar oportunamente las demandas del negocio de la entidad.

Radstaak (2019) En la Tesis de Maestria titulada "Developing DevOps Maturity Model Para la University of Twente". Afirma que la comunicación y coordinación de equipos se estandarizarán y se volverán más directas. La colaboración se mejorará aún más al compartir experiencias entre los equipos, lo que creará objetivos comunes que fomentarán una mayor colaboración. Los informes serán visibles para todos los equipos y se crearán estratégicamente. El historial de informes estará disponible para su comparación, y se podrá crear un panel de control para visualizar esto en todo el portafolio. Se realizará una prueba sistemática avanzada que se integrará en el proceso. La gestión se centralizará y se llevará a cabo de forma automática cada vez

que se realice un cambio en el código. Estos cambios se controlarán, lo que significará que se incorporarán en la rama principal con control de versiones, de modo que, en caso de que algo falle, será más fácil revertirlo. Los resultados se compararán con métricas de calidad para medir el rendimiento. La monitorización se realizará en contexto empresarial y de usuario final, utilizando los recursos de los entornos para garantizar la estabilidad. (p. 20)

Zavala (2019) En el estudio titulado "Mejoras con resultados comprobables y hablando cuantitativamente representan para el proyecto ahorros substanciales tanto en tiempo, esfuerzo y/o costo. Apoyando de esa manera al logro de los objetivos de negocio y estratégicos de la organización – para la Universidad Michoacana De San Nicolás De Hidalgo – México". Muestra que las organizaciones de tecnología de software en todos los niveles están cambiando cada vez más. Ya no se basan únicamente en proporcionar sistemas que realicen funciones básicas. Como siempre ha sido, es un problema para todas las organizaciones que carecen de procesos, lo que provoca aumentos presupuestarios, no puede entregar los proyectos a tiempo y obliga a los socios a trabajar demasiado y completar los proyectos en una semana. Si no conoce el alcance del proyecto y lo que se ha hecho, nunca sabrá cuándo estará terminado. (p. 9)

1.2.3 Artículos científicos publicados

Carrizo y Alfaro (2018) Indican la importancia al momento de atender la preocupación por la calidad en la industria del software que tiene como objetivo fundamental el desarrollo sistemático de productos y servicios de mejor calidad y el cumplimiento de las necesidades y expectativas de los clientes.

Aizprua et al. (2019) Indican que se debe tener una claridad en discernir entre la calidad del Producto de software y la calidad del Proceso de desarrollo de este. También en el mismo artículo incida que "No obstante, las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto".

Rodríguez Barajas (2017) Precisa que una solicitud de atención de ser considerado de alta calidad, debe cumplir con las siguientes características:

- Correcto: debe satisfacer las necesidades reales de los usuarios.
- Claro: Esto debe tener una explicación clara. Completo: debe incluir todas las definiciones de funcionalidad requeridas, interfaces (externas y gráficas), entradas, salidas, procesos alternativos y términos.
- Comprobable: Debe haber mecanismos para verificar que el sistema se ajusta a los requisitos.
- Coherencia: no debe haber conflicto entre los requisitos y el alcance.
- Prioridad determinable: Debería ser posible determinar su importancia en relación con las necesidades del usuario. Modificable: Las definiciones creadas deben ser modificables de manera simple, completa y consistente.
- Trazabilidad: Debería ser posible rastrear fácilmente el origen del requisito y los componentes del sistema que necesitan realizar la función. Utilizable: Debería ser útil para tareas de mantenimiento, es decir.
- El documento de requisitos debe ser práctico y funcional para el equipo de desarrollo durante todo el ciclo de vida del software.

Acosta et al. (2017) Indica que el interés por la calidad del software se está incrementando en la medida que los usuarios son más exigentes y requieren productos que cumplan con sus necesidades. Para su cumplimiento, existe estándares o metodologías (como normas ISO (International Organización for Standardization), CMMI, SPICE, SQuaRE que definen criterios de desarrollo, cuyo objetivo es producir software confiable de alta calidad.

López Mora et al. (2019) Concluye en su investigación "El uso de las metodologías agiles y su importancia para el desarrollo de software" que "No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser

adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc. Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Esto ha llevado hacia un interés creciente en las metodologías agile"

Gaete et al. (2021) Comenta que los enfoques ágiles se encuentran en sintonía con los principios y valores expresados en el manifiesto ágil. Dentro de este contexto, se destaca la importancia otorgada al valor del Trabajo en Progreso (WIP), la asignación de responsabilidades individuales durante el desarrollo y la actitud positiva hacia los cambios en los requisitos. Scrum, uno de los enfoques ágiles más ampliamente aceptados y utilizados en la actualidad, se define como un marco de trabajo que permite a las personas abordar problemas complejos adaptativos, al tiempo que facilita la entrega de productos con el máximo valor posible de manera productiva y creativa

Thongtanunam y Hassan (2020) Precisa la importancia de fortalecer los procesos de revisión de código suelen llevarse a cabo en un entorno transparente, donde diversas informaciones (como comentarios de revisión) son visibles, con el objetivo de mejorar la colaboración activa y oportuna. Sin embargo, esta información visible puede afectar las prácticas de revisión de código y su efectividad. Por lo tanto, en este artículo, investigamos primero la dinámica de la revisión, es decir, la práctica de proporcionar una decisión de evaluación por parte de un revisor

Mahbub et al. (2023) Enfatiza la relevancia de corregir los errores de software dado que cuestan miles de millones de dólares a la economía global cada año y consumen aproximadamente el 50% del tiempo de desarrollo. Una vez que se informa un error, el desarrollador asignado intenta identificar y comprender el código fuente responsable del error para luego corregirlo. En las últimas cinco décadas, ha habido una investigación significativa sobre la detección o corrección automática

de errores de software. Sin embargo, ha habido poca investigación sobre la explicación automática de errores a los desarrolladores, una tarea esencial pero altamente desafiante.

Esparza (2023) Precisa que, para garantizar un desarrollo de software seguro, es crucial adoptar un enfoque holístico que abarque todo el ciclo de vida del desarrollo. Esto implica la implementación de medidas de seguridad desde las fases iniciales de diseño y arquitectura, pasando por el desarrollo y las pruebas, hasta la etapa de implementación y el mantenimiento continuo. Para alcanzar este objetivo, se recurre a diversas técnicas, algoritmos y herramientas, como SonarQube, con el fin de identificar problemas comunes como vulnerabilidades de seguridad, errores de programación, código duplicado y malas prácticas de codificación, entre otros. Estas herramientas no solo ofrecen una visión detallada de la calidad del código fuente, sino que también permiten a los desarrolladores detectar y corregir problemas antes de que surjan en la producción del software. Los informes generados por estas herramientas proporcionan detalles exhaustivos sobre los problemas encontrados, incluyendo la ubicación exacta, recomendaciones y soluciones conocidas.

Lubomirsky et al. (2022) Indica que la deuda técnica se describe como las consecuencias de decisiones de desarrollo que priorizan la funcionalidad sobre consideraciones de diseño e implementación y parámetros de calidad. Inicialmente vinculada al código fuente, ha evolucionado para abarcar otros tipos de deuda, como la de arquitectura, requerimientos y documentación. Entre estos, destaca la deuda técnica de diseño de experiencia de usuario (UX debt), definida como un diseño incompleto que se pospone corregir. Aunque se reconoce a la UX debt como un tipo de deuda técnica, aún falta una caracterización que facilite su identificación y control.

1.3 Objetivos

A continuación, se expone los propósitos que pretende alcanzar en el desarrollo de la siguiente investigación.

1.3.1 Objetivo General

OG. Determinar el grado de influencia que ejerce la implementación de un modelo de desarrollo de software en la calidad del Producto del SaaS ERP de la empresa Integrator.

1.3.2 Objetivos Específicos

OE1.- Determinar el grado de influencia que ejerce la implementación de un modelo de desarrollo de software en la reducción de errores en el Producto del SaaS ERP de la empresa Integrator.

OE2.- Determinar el grado de influencia que ejerce la implementación de un modelo de desarrollo de software en la reducción del tiempo de entrega del Producto SaaS ERP de la empresa Integrator.

1.4 Justificación

1.4.1 Teórica

Mediante la aplicación de la nueva metodología se pretende mejorar el proceso de calidad de software SaaS ERP de la empresa Integrator SAC, con el objetivo de lograr una mejora significativa y medible dentro del proceso de acuerdo con las necesidades de la empresa.

Sobre dicha problemática se han formulado las posibles soluciones a través de las hipótesis; luego se ha establecido los propósitos que persigue el trabajo por intermedio de los objetivos. Todos estos elementos se han formado en base a las variables e indicadores de la investigación.

1.4.2 Metodológica

Contribuye a futuros trabajos de investigación en el contexto del modelo de optimización de procesos de calidad de software para reforzar la disponibilidad, escalabilidad y flexibilidad para los sistemas de información operativa y gerencial.

1.4.3 Practica

Al culminar el proyecto la empresa Integrator contara con una metodología que le permitirá mejorar la calidad de su producto SaaS ERP y además esta lograra reducir sus costos, perdidas e incrementar sus ganancias.

1.4.4 Limitaciones

Las fuentes de datos para esta investigación están conformadas por los programas fuentes de las aplicaciones del software Integrator ERP de los años 2019 – 2023 de la empresa Integrator SAC.

- Las muestras para este estudio fueron acotadas por el área de TI a un porcentaje no mayor del 30% de la totalidad de programas.
- Este trabajo de investigación se desarrollará para la Integrator SAC
- Este trabajo de investigación se desarrollará en la plataforma Cloud Computing de la empresa Azure y AWS.
- En este trabajo, no intenta optimizar las aplicaciones del sistema y/o el tiempo de ejecución del programa, sino que explora un conjunto de técnicas para aplicar las mejores técnicas recomendadas para el desarrollo de software.
- Para salva guardar la información utilizada en este trabajo se firmó un acuerdo de confidencialidad con la empresa investigada por el uso y manipuleo de los datos empleados
- Se han definido una población de 265 programas para su análisis.
- El alcance de este trabajo de investigación se ha limitado a la población general debido a restricciones logísticas y financieras, y se centrará en la asesoría tecnológica para el diseño de una metodología compuesta por la parte de mejoramiento de código y la parte de la organización del trabajo de programación para mejorar la calidad de código de la aplicación y mejorar la presencia de la empresa. Cabe destacar que no se llevará a cabo un despliegue

completo de la solución, ya que este estudio se ha desarrollado únicamente como una demostración inicial del concepto.

1.5 Hipótesis

A continuación, se expone las hipótesis que pretende ser demostrada en el desarrollo de la siguiente investigación.

1.5.1 Hipótesis General

HG. - La implementación de un modelo de desarrollo de software, contribuye en mejorar la Calidad del Producto SaaS ERP de la Empresa Integrator.

1.5.2 Hipótesis Específica

HE1.- La implementación de un modelo de desarrollo de software contribuye a reducir los errores en el Producto del SaaS ERP.

HE2.- La implementación de un modelo de desarrollo de software contribuye a reducir el tiempo de entrega del Producto SaaS ERP.

II. MARCO TEÓRICO

2.1 Bases Teóricas sobre el tema de Investigación

2.2 Modelo de Desarrollo de Software

2.2.1 Software

"Se define como una configuración empaquetada, componentes o un servicio basado en un producto intangible, con materiales auxiliares, que se libera y comercializa en un mercado específico" (Xu y Brinkkemper, 2005).

2.2.2 Modelo de desarrollo de Software

Modelos de desarrollo de software son una colección de técnicas y sistemas organizacionales para crear software de computadora. El objetivo de los diversos enfoques es estructurar equipos de trabajo para que puedan construir las funcionalidades del programa de la manera más eficiente posible. Modelos de desarrollo de software proporcionar un marco para controlar el desarrollo de los sistemas de información. Desde la planificación hasta el mantenimiento, un Ciclo de vida del desarrollo de programas (SDLC) describe todos los procesos en un proyecto de desarrollo de software. Estos marcos incluyen el desarrollo de programas, así como las herramientas necesarias para ayudar en el proceso de desarrollo. (Sharma, 2022)

2.2.3 Metodología de desarrollo de Software

Una metodología se refiere a un sistema coordinado de técnicas y enfoques que se utilizan de manera uniforme y flexible para manejar todas las etapas de un proyecto de desarrollo. Es esencialmente un procedimiento detallado y exhaustivo para la creación de software. (Maida y Pacienzia, 2015)

En el contexto de la ingeniería del software, una metodología se caracteriza por:

- Optimizar tanto el desarrollo del software como el resultado final del producto.

- Incorporar enfoques que guíen la planificación y desarrollo del software.
- Establecer pautas sobre qué acciones realizar, cómo llevarlas a cabo y cuándo hacerlo a lo largo de todo el ciclo de vida del proyecto, incluyendo el desarrollo y el mantenimiento.

2.2.4 Metodología Tradicional

Las metodologías tradicionales en desarrollo de software imponen una disciplina estricta al proceso, destacando la planificación detallada antes de iniciar el proceso de construcción del producto. Se enfocan en la revisión y control de las actividades mediante roles definidos, actividades detalladas, artefactos y herramientas específicas. Sin embargo, carecen de adaptabilidad a cambios imprevistos, lo que las hace menos adecuadas para entornos donde los requisitos son inciertos o pueden cambiar. Además, presentan desafíos como altos costos al implementar cambios y falta de flexibilidad en proyectos con entornos volátiles. (Board Infinity, 2020).

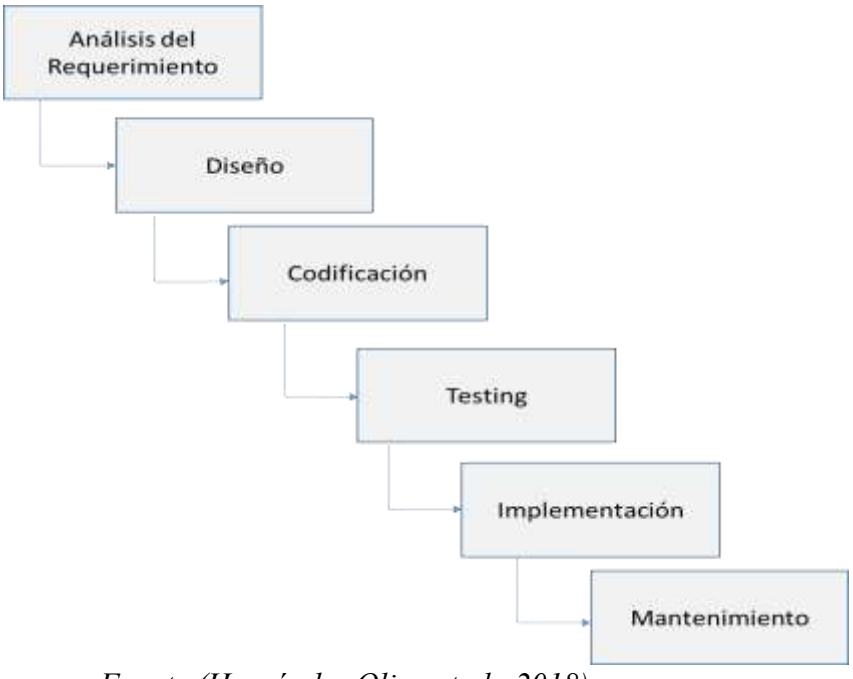
2.2.5 Metodología Tradicional - Modelo Waterfall (Cascada)

El modelo Cascada, es un enfoque tradicional del desarrollo de software y puede ilustrarse a través del modelo de cascada, que es el tiempo-probado y es fácil de entender. El modelo de cascada es un modelo estático y aborda la construcción de sistemas en una manera escalonada y horizontal, esperando concluir una tarea antes que del inicio de otra. (Adenowo y Adenowo, 2020)

El estilo cascado divide proyectos basados en actividades: requisito análisis, diseño, codificación y pruebas.

Figura 2.

Fases del Modelo Cascada



Fuente (Hernández Oliva et al., 2018)

También se identifica las actividades como: comunicación (que involucra inicio del proyecto y recopilación de requisitos), planificación (estimación, programación y seguimiento), modelado (análisis y diseño), construcción (codificación y pruebas), e implementación (entrega, soporte y retroalimentación).

El modelo en cascada suele tener objetivos distintos para cada etapa de la construcción. Una vez que una etapa está completamente desarrollada, el desarrollo continúa con la siguiente fase y no hay oportunidad de volver atrás y revisar etapa anterior como se muestra en la *figura. 2* arriba. El modelo así apoya un enfoque estructurado y basado en procesos

En la organización geeksforgeeks plataforma de desarrollo de software (geeksforgeeks.org, 2023) detalla las siguientes Fases:

Análisis de Requerimiento:

- En la etapa de análisis y especificación de requerimientos, el propósito es comprender las necesidades precisas del cliente y documentarlas de manera adecuada. Esta fase implica dos actividades distintas:

- Recopilación y análisis de requisitos: En primer lugar, se recopilan todos los requisitos relacionados con el software del cliente y luego se analizan. El objetivo de la parte de análisis es eliminar la incompletitud (un requisito incompleto es aquel en el que se han omitido algunas partes de los requisitos reales) y las inconsistencias (un requisito inconsistente es aquel en el que una parte del requisito contradice otra parte).
- Especificación de requisitos: Estos requisitos analizados se documentan en un documento de Especificación de Requisitos de Software (SRS, por sus siglas en inglés). El documento SRS sirve como un contrato entre el equipo de desarrollo y los clientes. Cualquier disputa futura entre los clientes y los desarrolladores puede resolverse examinando el documento SRS. (geeksforgeeks.org, 2023)

Diseño

- El alcance de esta etapa es convertir los pedidos adquiridos en el SRS a un formato que pueda ser codificado en un lenguaje de programación. Esto implica el diseño tanto a nivel alto como detallado, así como la arquitectura general del software. (geeksforgeeks.org, 2023)

Codificación

- En la etapa de programación, el diseño del software se convierte, utilizando cualquier lenguaje de programación adecuado en un producto usable. De esta manera, cada parte usado en el proceso de diseñado se codifica. El objetivo de la fase de pruebas unitarias es verificar si cada módulo está funcionando correctamente o no. (geeksforgeeks.org, 2023)

Integración y Prueba (Testing)

- La integración de distintos módulos se lleva a cabo poco después de que han sido codificados y sometidos a pruebas unitarias. La integración de diferentes módulos se realiza paso a paso a través de varios pasos.

- En cada etapa de integración, se agregan módulos previamente planificados al sistema semi-integrado y se prueba el sistema resultante.
- Finalmente, después de que todos los módulos se hayan integrado y probado con éxito, se obtiene un sistema completo y completamente funcional y se prueba el sistema.

Después de que el software ha sido entregado, el cliente realiza pruebas de aceptación para determinar si aceptar o rechazar el software entregado.

Implementación

- El Software pasa para usar. (geeksforgeeks.org, 2023)

Mantenimiento

El mantenimiento es el paso más importante en el ciclo de vida del software.

La carga de trabajo de mantenimiento representa el 60% de la carga de trabajo total de desarrollo de software. El mantenimiento se divide básicamente en tres tipos

- Mantenimiento Correctivo: Este tipo de mantenimiento se realiza para corregir errores que no fueron descubiertos durante la fase de desarrollo del producto.
- Mantenimiento Perfectivo: Se lleva a cabo para mejorar las funcionalidades del sistema según la solicitud del cliente.
- Mantenimiento Adaptativo: Por lo general, se requiere para adaptar el software y hacerlo funcionar en un nuevo entorno, como en una nueva plataforma de computación o con un nuevo sistema operativo. (geeksforgeeks.org, 2023)

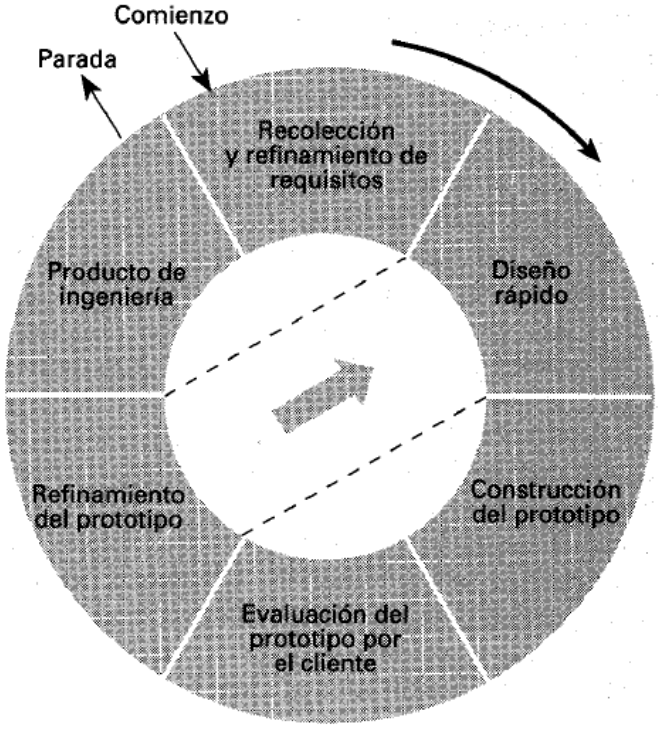
2.2.6 Metodología Tradicional - Modelo Prototipado

Este método implica crear una versión inicial de un sistema de información destinada a demostración o evaluación. La creación del desarrollo de este requisito implica la creación de una implementación parcial de un sistema con el objetivo de comprender mejor sus requisitos. Puede construirse y entregarse rápidamente a los usuarios, clientes o sus representantes para que lo prueben. Los comentarios de estos usuarios se registran en la especificación de requisitos para guiar

el desarrollo del producto actual. Estos se pueden utilizar durante o antes de la fase de definición de requisitos o incluso antes del desarrollo incremental. (Board Infinity, 2020).

Figura 3.

Modelo Prototyping



Fuente (Board Infinity, 2020)

2.2.7 Metodología Tradicional - Modelo Spiral

Esta metodología combina las bondades de los dos modelos de desarrollo previos (cascada y prototipos), añadiendo una actividad relacionada a la definición de análisis de riesgo. Se estructura en cuatro actividades:

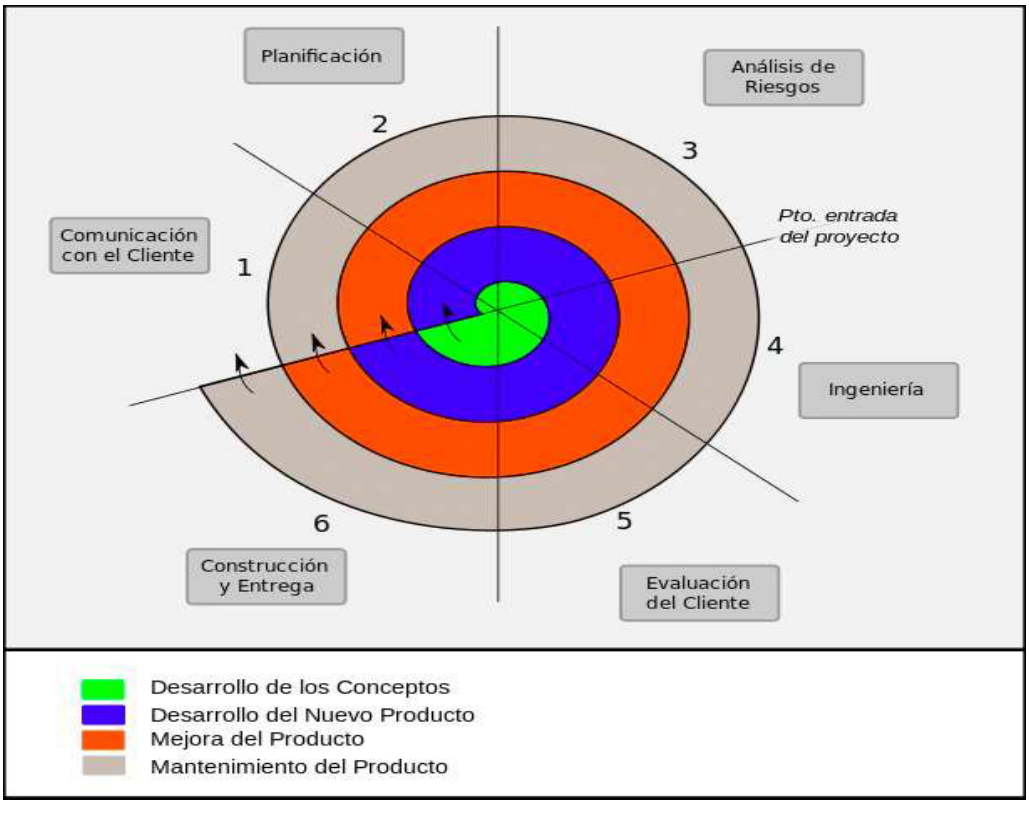
- Planificación: Recopilación de los requisitos iniciales o aquellos que se añadirán en la iteración actual.
- Análisis de riesgos: Evaluación de las opciones de desarrollo de software según los requisitos y decisión sobre si proceder.

- Desarrollo: Creación de modelos preliminares basados en los requisitos obtenidos durante la fase de planificación.
- Revisión del cliente: El cliente examina el modelo preliminar. Si está satisfecho, se concluye el proceso; de lo contrario, se añaden nuevos requisitos para la próxima iteración (Roche, 2017)

Esta metodología busca abordar riesgos potenciales desde el principio, permitiendo ajustes continuos en función de la retroalimentación del cliente y asegurando un desarrollo más eficiente y adaptativo.

Figura 4.

Modelo Espiral



Modelo en Espiral. (Roche, 2017)

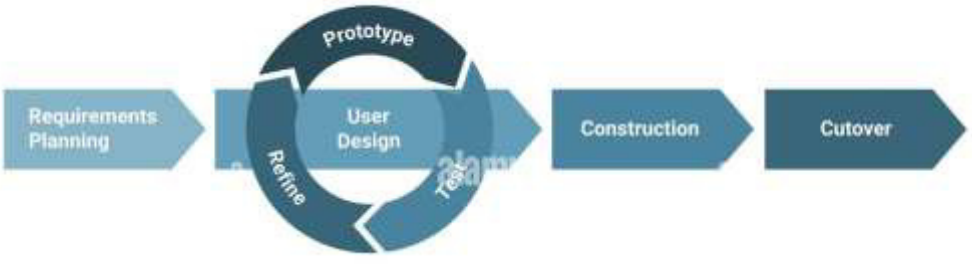
2.2.8 Metodología Tradicional - Modelo Desarrollo Rápido de Aplicaciones

Este modelo es muy común debido a la necesidad obtener un producto en corto plazo. El RAD es un flujo de construcción destinado a producir software de computadora de buena calidad. Incluye desarrollo interactivo, construcción de prototipos y utiliza utilidades CASE (Ingeniería de Software Asistida por Computadora). Tradicionalmente, el desarrollo rápido de aplicaciones se centra en su uso, la utilización y la velocidad de sus componentes al momento de usarlo.

Hoy en día este método se emplea comúnmente al referirse al desarrollo rápido de interfaces gráficas de usuario, como Glade, o entornos de desarrollo integrado completos, como Visual Studio, Lazarus, Gambas, Delphi, Foxpro, Anjuta, Game Maker, Velneo o Clarion. Además, en el ámbito de la autoría multimedia, software como Neosoft Neoboo y MediaChance Multimedia Builder proporcionan plataformas para el desarrollo rápido de aplicaciones, aunque dentro de ciertos límites. (Levin y Vector, 2020)

Figura 5.

Modelo RAD



Modelo RAD (Levin y Vector, 2020)

2.2.9 Metodologías Agiles

"Ágil" es una palabra de moda que probablemente caerá en desuso algún día, lo que hará que este libro parezca obsoleto. Está cargada con diferentes significados que se aplican en diferentes circunstancias. Una forma de definir el "desarrollo ágil" es observar los cuatro valores del Manifiesto Ágil. (Crispín, 2008)

El enfoque Agile se basa en cuatro valores fundamentales:

- Individuos e interacciones sobre procesos y herramientas**: Prioriza la comprensión del comportamiento del software en funcionamiento y el feedback temprano de los usuarios sobre la documentación exhaustiva de requisitos.
- Software funcionando sobre documentación exhaustiva**: Destaca la importancia de ver cómo se comportan las funcionalidades del software en funcionamiento y el feedback de los usuarios frente a la creación de una documentación detallada de requisitos.
- Colaboración con el cliente sobre negociación contractual**: Aboga por la colaboración continua con el cliente durante el desarrollo del producto en lugar de seguir rigurosamente un contrato preestablecido.
- Respuesta al cambio sobre seguir un plan**: Reconoce que, en entornos volátiles como el desarrollo de software, es más valiosa la capacidad de respuesta y adaptación a los cambios que seguir un plan preestablecido. (Sentrio, 2021).

Principios Agiles y la Agilidad

Según los principios ágiles enunciados en el Manifiesto Ágil, desarrolladores de software motivados y empoderados, confiando en la excelencia técnica y diseños simples, generan valor empresarial entregando software funcional a los usuarios en intervalos cortos y regulares. Estos principios han dado origen a diversas prácticas que se cree ofrecen mayor valor a los clientes. En el núcleo de estas prácticas está la idea de equipos auto organizado cuyos miembros no solo están localizados, sino que trabajan a un ritmo que sostiene su creatividad y productividad. Los principios fomentan prácticas que se adaptan a cambios en los requisitos en cualquier etapa del proceso de desarrollo. Además, los clientes (o sus representantes) participan activamente en el proceso de desarrollo, facilitando la retroalimentación y reflexión que pueden conducir a resultados más satisfactorios. Los principios no constituyen una definición formal de agilidad, sino más bien pautas para entregar software de alta calidad de manera ágil. (Torgeir Dingsøyr, 2012)

- Pruebas Agiles

Es posible que use el término "tester" para describir a una persona cuyas principales actividades se centran en las pruebas y el aseguramiento de la calidad. También verá que a menudo usamos la palabra "programmer" para describir a una persona cuyas actividades principales giran en torno a escribir código de producción. No pretendemos que estos términos suenen limitados o insignificantes. Los programadores hacen más que convertir una especificación en un programa. No los llamamos "desarrolladores", porque todos los involucrados en la entrega de software son desarrolladores. Los testers más allá de realizar "tareas de prueba". Cada miembro del equipo ágil está enfocado en entregar un producto de alta calidad que aporte valor empresarial. Los testers ágiles trabajan para garantizar que su equipo entregue la calidad que necesitan sus clientes. Utilizamos los términos "programmer" y "tester" por conveniencia. (Crispin, 2008)

¿Por qué probamos? La respuesta podría parecer obvia, pero, de hecho, es bastante compleja. Probamos por muchas razones: para encontrar errores, para asegurarnos de que el código sea confiable y a veces solo para ver si el código es utilizable. Realizamos diferentes tipos de pruebas para lograr diferentes objetivos. La calidad del producto de software tiene muchos componentes. (Crispin, 2008)

- Los cuatro cuadrantes del Testing Agile

El Cuadrante de Pruebas Ágiles es un concepto utilizado en el ámbito de las pruebas de software en el contexto de metodologías ágiles. Fue propuesto por *Brian Marick* para proporcionar una estructura que organice los distintos tipos de pruebas en proyectos ágiles.

Los cuadrantes a la izquierda incluyen pruebas que respaldan al equipo mientras desarrolla el producto. Este concepto de realizar pruebas para ayudar a los programadores es algo nuevo para muchos *testers* y es la mayor diferencia entre las pruebas en un proyecto tradicional y las pruebas en un proyecto ágil. Las pruebas realizadas en los Cuadrantes 1 y 2 son más ayudas para la

especificación de requisitos y el diseño que lo que normalmente pensamos como pruebas. (Lisa Crispin, 2008)

Figura 6.

Los cuatro cuadrantes del Testing Agile

Nota: Adaptado de (Crispín, 2008)

Sector 1

Este sector se refiere al método Desarrollo Guiado por Pruebas (o TDD), se incorporan pruebas de componentes y unidades de implementación. Al escribir pruebas primero, los desarrolladores pueden producir un mejor código. Las pruebas unitarias verifican que una pequeña parte del sistema, como un objeto o un método, funcione correctamente. Las pruebas de componentes evalúan el comportamiento de una parte más grande (como un conjunto de categorías). Todas estas pruebas están automatizadas y escritas en el mismo idioma que la aplicación asociada. Estas pruebas determinan lo que Kent Beck llama la "calidad intrínseca" del código, que no se discute con el cliente, sino que la determina el programador. (Rodríguez, 2020)

Sector 2

Incluye pruebas que ayudan a los equipos de desarrollo a elaborar software, aunque desde una perspectiva más general y empresarial. Nos permiten confirmar que la función funciona según lo esperado por el cliente o el experto empresarial que ayuda a redactar estas pruebas, y deben expresarlas en su propio idioma. Estamos hablando aquí de las "propiedades extrínsecas". Idealmente, la mayoría de estas pruebas (scripts con parámetros de ejemplo, simulacros, pruebas de validación de UI, API de terceros, etc.) están automatizadas y forman parte del proceso de construcción e integración continua, proporcionando comentarios frecuentes a los desarrolladores. Pero en este sector también incluimos el modelo inicial y el diseño, que guía al equipo en la interfaz de usuario y sus interacciones (esto será parte del trabajo de descubrimiento del equipo de producto). (Rodríguez, 2020).

Sector 3

Las pruebas de los sectores 1 y 2 ayudaron al equipo a desarrollar el producto. Sin embargo, incluso cuando se implementan pruebas orientadas al negocio, los requisitos del cliente pueden ignorarse o malinterpretarse. El software puede no ser lo que los clientes realmente quieren o no ser lo suficientemente competitivo. Aquí es donde se necesitan pruebas manuales: solo una persona, ya sea un cliente o alguien que se ponga en el lugar de otra persona usando su intuición y creatividad, puede juzgar si un producto es realmente lo que debería ser. Este tipo de pruebas suele permitirnos encontrar los errores más relevantes. Estamos hablando exactamente de pruebas exploratorias, donde los evaluadores diseñan y ejecutan pruebas simultáneamente. También existen pruebas de usabilidad en entrevistas, focus groups, etc. (Rodríguez, 2020).

Sector 4

Son pruebas diseñadas para cubrir requisitos no funcionales como seguridad, velocidad de carga, escalabilidad, etc. Si es posible, las automatizaremos en el sector 1 o 2 o usaremos herramientas para hacerlo. (Rodríguez, 2020).

2.2.10 Metodologías Agiles – Programación Extrema (XP)

La Programación Extrema (XP) podría marcar un hito en la ingeniería del software. Concebida a fines de los años noventa por Kent Beck, Ward Cunningham y Ron Jeffries, esta metodología ha evolucionado desde una idea para un solo proyecto hasta infiltrarse en numerosas fábricas de programas computacionales. Algunos la llaman un movimiento ideológico que desplaza el enfoque de los estudios preliminares al momento de desarrollar el software hacia los profesionales de las unidades operacionales del negocio, destacando la necesidad de desarrollar soluciones frente a los rigores de los contratos de desarrollo.

XP destaca entre los procesos ágiles de desarrollo de software y, al igual que ellos, se diferencia de las metodologías tradicionales al priorizar la adaptabilidad sobre la previsibilidad. Los defensores de XP consideran que los cambios de requisitos son naturales, inevitables e incluso deseables en el desarrollo de proyectos. Argumentan que la capacidad de adaptarse a cambios en cualquier fase del proyecto es una aproximación más realista y eficiente que intentar definir todos los requisitos al inicio y luego gestionar los cambios de manera reactiva. (Raeburn, 2020).

2.2.11 Metodologías Agiles – SCRUM

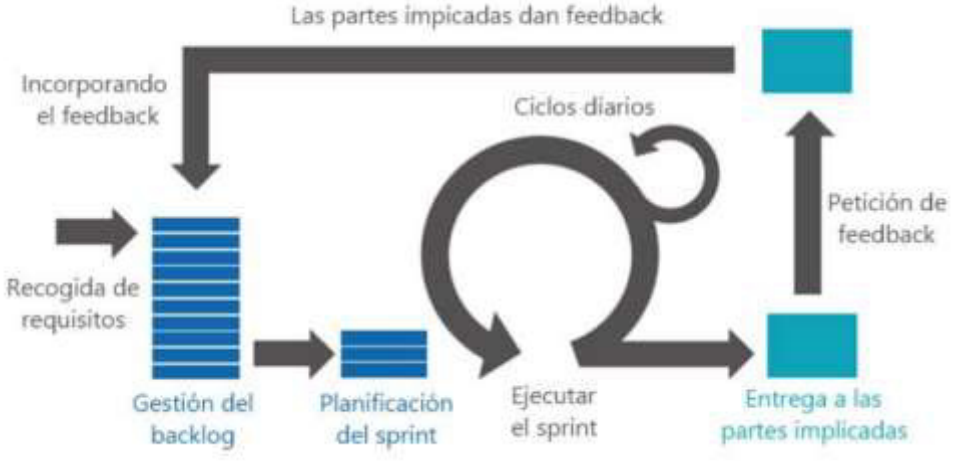
Esta disciplina tiene sus origines en los años 80’s, originada por la necesidad de procesos de reingeniería liderados por Goldratt, Takeuchi y Nonaka. Este enfoque se inspiró en los nuevos métodos de desarrollo utilizados en productos exitosos en Japón y los Estados Unidos. Los equipos que desarrollaban estos productos trabajaban con requisitos generales y novedosos, con la presión de llegar al mercado en un tiempo significativamente más corto que los productos anteriores. La forma de trabajo de estos equipos altamente productivos y multidisciplinarios se comparó con la

colaboración entre jugadores de rugby y su formación de scrum, de donde proviene el nombre. (Maida y Pacienzia, 2015).

Scrum es un proceso que aplica regularmente un conjunto de buenas prácticas para trabajar colaborativamente y obtener los mejores resultados posibles en un proyecto. Estas prácticas se respaldan mutuamente y se seleccionan a partir de un estudio sobre el trabajo de equipos altamente productivos. Scrum se basa en un controlado "caos" pero establece mecanismos para manejar la indeterminación, manipular lo impredecible y controlar la flexibilidad. En Scrum, se realizan entregas parciales y regulares del producto final, priorizadas según el beneficio que aportan al receptor del proyecto. Por lo tanto, Scrum es especialmente adecuado para proyectos en entornos complejos, donde se necesita obtener resultados rápidamente, los requisitos son cambiantes o poco definidos, y donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales. (Maida y Pacienzia, 2015).

Figura 7.

Metodología SCRUM



Fuente: (Maida y Pacienzia, 2015)

2.2.11 Proceso de Aseguramiento de Calidad (QA)

El proceso de QA (*Quality Assurance*) es el conjunto de tareas y/o actividades que asegura la calidad es un producto software o de un servicio. Como lo indica Garcia (2022), "Tiene el objetivo de cumplir las expectativas del cliente, mejorar la experiencia de usuario y facilitar un proyecto que funcione óptima y correctamente. El medio para hacerlo es detectar errores y corregirlos antes de las entregas. Por ello, es de vital importancia que el equipo de QA esté presente en todas las fases de desarrollo, en cada integración, nuevo módulo, o funcionalidad. Se suele utilizar bastante en la metodología ágil o scrum, corrigiendo los bugs poco a poco en cada sprint. De esta manera, se inicia de forma correcta el desarrollo, no se acumulan los fallos y es más sencillo continuar programando, mejorando el rendimiento. Por lo que va ganando popularidad en la actualidad."

Sanchís (2016) en su trabajo de fin de grado de ingeniería titulado "Definición e implantación de un proceso QA para desarrollo de software", define los siguientes conceptos propios dentro de un proceso de QA:

Fallo: comportamiento no deseado del producto, visible desde el punto de vista externo del producto.

Defecto: la imperfección que causa el fallo. Un defecto puede existir en el producto y no manifestarse aún como fallo.

Error: lo que causó la introducción del defecto en el producto. El error siempre es cometido directa o indirectamente por una acción (o no acción) humana.

- Proceso de QA Testing

Inesdi (2022), Preciso que el proceso de QA testing es un proceso por el cual se asegura la calidad en un proyecto de software, permitiendo evitar errores y bugs en el desarrollo de estos. Se puede considerar un compromiso con el cliente o usuarios para que los productos digitales que vaya a utilizar se correspondan con lo precisado. El objetivo de esta prueba es detectar los errores

en el menor tiempo posible y lo ideal es que se lleve a cabo a lo largo de todo el proceso de desarrollo.

- Diferencias entre QA y Testing

Mientras que el QA está dirigido a asegurar la calidad del producto en todas sus fases, el testing se enfoca exclusivamente a probar el artículo durante el desarrollo para encontrar fallos.

Para saber qué es QA en desarrollo software frente al testing, encontramos la diferencia de que por un lado el *Quality Assurance* trata de prevenir errores futuros, conociendo cada requerimiento y fase del proyecto. Y, por otro lado, el testing se centra en arreglar los bugs existentes y después reportarlos al equipo de desarrollo para que lo modifique.

El QA amplía las labores del tester, evoluciona este concepto yendo más allá y consiguiendo mejorar la productividad y rendimiento del proyecto. Una actividad preventiva orientada en todo el proceso para cumplir con los requisitos, frente una labor correctiva orientada al producto para encontrar bugs. Garcia (2022).

- Proceso QA Interno

Es el procedimiento enmarcado en el proceso de revisión de pares, como lo indica Scrum Manager BoK (2023), “La revisión por pares (*peer review*, en inglés) puede explicarse, en una primera aproximación, como la revisión del código realizada por otro miembro del equipo diferente al autor original del código, y cuyo objetivo principal es la búsqueda de defectos y la propuesta y evaluación de soluciones alternativas o de mejoras en el diseño y los algoritmos utilizados. Además, la revisión por pares sirve como facilitador para la difusión del conocimiento a través del equipo y, en su caso, de la organización.”

Los objetivos de esta técnica son:

1. Tener un primer filtro sobre el código que vaya a pasar a los procesos de certificación
2. Hacer hallazgos previos sobre la calidad y revisión de código

3. Hacer una revisión temprana para que el código cumpla con los RFP (*Request for propossals*) o las propuestas técnicas, en los siguientes puntos:

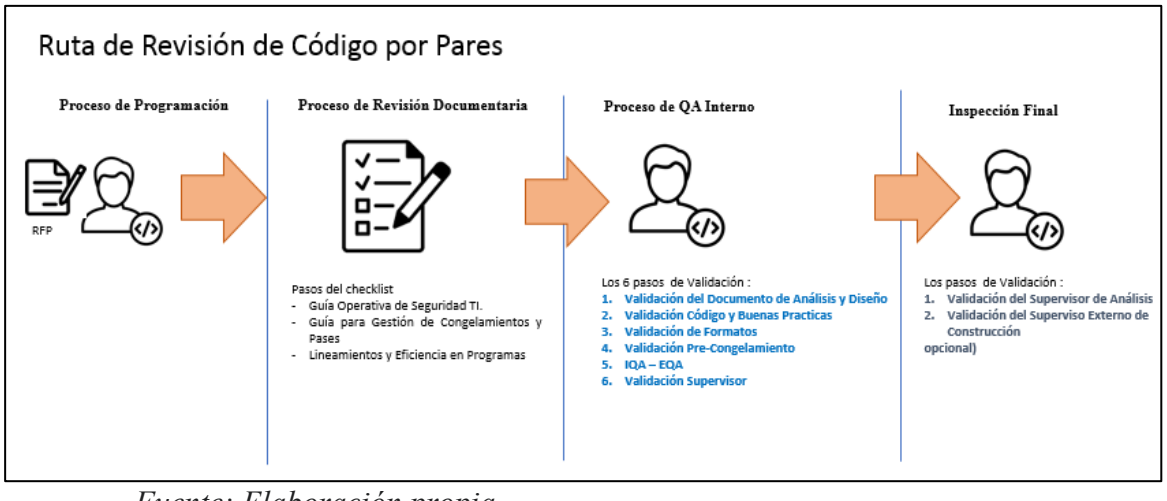
- Funcionabilidad (Lo que el producto puede hacer i)
- Fiabilidad (Que cumpla una determinada función bajo ciertas condiciones)
- Usabilidad (el software sea sencillo de usar, cumpliendo las expectativas del usuario)
- Eficiencia (alcázar el objetivo fijado)
- Mantenimiento (darle mantenimiento de una manera sencilla)
- Portabilidad (Compatible)

Proceso del proceso de QAinterno:

Al finalizar el proceso de desarrollo, se ejecuta un proceso de QAinterno, en la cual un par (del mismo grupo de desarrolladores u otra tribu Ágil), procede a revisar el código. Son 6 pasos los que usualmente se recomienda.

Figura 8.

Procedimiento de revisión de Pares como parte del QA Interno



Fuente: Elaboración propia

Para la ejecución de estos pasos, se deben de elaborar formatos alineados al alcance de cada una de estas actividades, asegurándose que esta actividad cumpla con lo especificado en estos documentos:

1. **Validación del Documento de Análisis y Diseño**: Proceso de revisión del diseño y estándares en los formatos
2. **Validación Código y Buenas Prácticas:** Proceso de revisión de código fuente, mejores prácticas de programación
3. **Validación de Formatos:** Proceso de revisión de documentación del requerimiento
4. **Validación Pre-Congelamiento:** Proceso de revisión del proceso PAR y herramientas de Versionamiento de código
5. **QAInterno Revisión** general del requerimiento realizada por Analista Sr. interno en el equipo y externo otro dominio
6. **Validación Supervisor:** Proceso de revisión supervisor para confirmar y autorizar el congelamiento

Adicionalmente puede existir una inspección final los cuales son 2 pasos:

1. **Validación del supervisor de analistas** Proceso de revisión de un supervisor externo:
    - Validación del Documento de Análisis y Diseño
    - Validación Supervisor.
    - QAInterno
2. **Validación del supervisor de construcción de software** Proceso de revisión de un supervisor externo:
    - Validación de Formatos
    - Validación Pre-congelamiento
    - EQA – QA Externo (opcional)

SonarQube

Es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente con el objetivo de obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.

2.3 Calidad de Software

2.3.1 Calidad del Software

La calidad de software como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo plenamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente", con base en los requisitos funcionales y no funcionales identificados en la etapa de análisis del sistema, insumo principal para implementar dichos requisitos con los atributos mínimos de calidad, fomentando la aplicación de procesos estandarizados y criterios necesarios en cada una de sus etapas, así se fomenta que el avance en el ciclo de vida del software minimice el riesgo de fracaso del proyecto. (Callejas-Cuervo et al., 2017)

.

2.3.2 Norma ISO / IEC 25000

La norma ISO/IEC 25000, también conocida como SQuaRE (Software Quality Requirements and Evaluation), se estableció con el propósito de establecer un marco unificado para la evaluación de la calidad de los productos de software. Este conjunto de estándares representa una evolución de normativas previas, especialmente la ISO/IEC 9126, que define modelos de calidad para productos de software, y la ISO/IEC 14598, centrada en el proceso de evaluación de dichos productos. La serie ISO/IEC 25000, ilustrada en la Figura 5, comprende cinco partes distintas. (ISO25000, 2024).

Figura 9.

Cinco divisiones del ISO/IEC 25000

Fuente: División para gestión de calidad (ISO25000, 2024)

2.3.3 ISO / IEC 25010:2011

El modelo de calidad juega un papel fundamental en el diseño de un sistema para evaluar la calidad de un producto. Este modelo sirve para identificar las cualidades esenciales que deben tenerse en cuenta al evaluar un producto de software específico. El modelo de calidad juega un papel fundamental en el diseño de un sistema para evaluar la calidad de un producto. Este modelo sirve para identificar las cualidades esenciales que deben tenerse en cuenta al evaluar un producto de software específico. (ISO25000, 2024)

La excelencia de un producto de software se puede entender como el nivel en que cumple con las expectativas de los usuarios y ofrece utilidad. Estas exigencias, como funcionalidad, desempeño, seguridad, mantenibilidad, entre otras, están incorporadas en el modelo de calidad. Este modelo descompone la calidad del producto en atributos y sub atributos. La definición de calidad del producto según ISO/IEC 25010 incluye ocho aspectos de calidad, tal como se presenta en la siguiente representación gráfica:

Figura 10.

Las 8 Características de la calidad de software

División para gestión de calidad (ISO25000, 2024.)

2.3.4 ISO / IEC 25040:2011

ISO/IEC 25040 define el proceso para llevar a cabo la evaluación del producto software. Dicho proceso de evaluación consta de un total de cinco actividades (ISO25000, 2024)

Figura 11.

Cuadro de Actividades ISO / IEC 25040

División para gestión de calidad (ISO25000, 2024)

En La familia de normas ISO/IEC 25000; El proceso de evaluación de la calidad del software consta de varias etapas (ISO25000, 2024):

- Actividad 1: Definir los requerimientos de la evaluación

  El primer paso en el proceso de evaluación es determinar los requisitos de evaluación.

  - Tarea 1.1: Definir el objetivo.

En esta tarea se registra el motivo de la evaluación de calidad del software por parte de la organización, que puede abarcar desde garantizar la calidad del producto y decidir su aceptación, hasta evaluar la viabilidad del proyecto en curso y comparar la calidad del producto con la de la competencia. (ISO25000, 2024)

  - Tarea 1.2: Determinar los estándares de la calidad del software.

En esta tarea se reconocen las partes involucradas en el software (desarrolladores, compradores potenciales, usuarios, proveedores, etc.) y se establecen los estándares de calidad del producto mediante un modelo específico de calidad. (ISO25000, 2024)

  - Tarea 1.3: Identificar las partes del producto a evaluar.

Es esencial identificar y registrar las partes del software que serán evaluadas. El alcance del producto a evaluar (ya sea especificaciones de requisitos, diagramas de diseño, documentación de pruebas, etc.) varía según la etapa del ciclo de vida en la que se realice la evaluación y su objetivo específico. (ISO25000, 2024)

- Tarea 1.4: Determinar el nivel de la evaluación.

Es crucial reconocer y registrar las componentes del producto de software que serán objeto de evaluación. El tipo de producto a evaluar, ya sea especificaciones de requisitos, diagramas de diseño, documentación de pruebas, entre otros, varía según la fase del ciclo de vida en la que se lleve a cabo la evaluación y su propósito específico. (ISO25000, 2024)

- Actividad 2: Determinar la evaluación

En esta tarea se detallan los conjuntos de evaluación, los cuales incluyen métricas, herramientas y técnicas de medición, junto con los criterios de decisión que se utilizarán durante el proceso de evaluación.

- Tarea 2.1: Identificar sobre los módulos a evaluar.

Durante esta etapa, el evaluador elige meticulosamente las métricas de calidad, técnicas y herramientas (los módulos de evaluación) que aborden de manera integral todos los requisitos de la evaluación. Estas métricas deben ser capaces de facilitar comparaciones confiables basadas en sus valores, permitiendo así la toma de decisiones fundamentada. La Norma ISO/IEC 25020 puede ser una referencia útil en este proceso. (ISO25000, 2024)

- Tarea 2.2: Definir los criterios para las métricas.

Es esencial establecer criterios de decisión para las métricas seleccionadas. Estos criterios son valores numéricos que se pueden asociar con los requisitos de calidad y luego con los criterios de evaluación para determinar la calidad del producto. Dichos umbrales pueden basarse en benchmarks, límites de control estadísticos, datos históricos, requisitos del cliente, entre otros factores. (ISO25000, 2024)

- Tarea 2.3: Determinar las cualidades de las características evaluadas.

Es necesario definir criterios para diferentes características, que se evalúan en forma de sub-características e indicadores de calidad. Estos estándares permiten una evaluación global de la calidad de los productos de software en un nivel más alto de abstracción. (ISO25000, 2024)

- Actividad 3: Configuración de la evaluación

  En esta tarea se crea el plan con los trabajos de evaluación que se deben realizar.

  - Tarea 3.1: Organizar las tareas de la evaluación.

La planificación de las actividades de evaluación debe tener en cuenta la disponibilidad de recursos humanos y materiales que puedan ser necesarios. A la hora de planificar se deben tener en cuenta el presupuesto, los métodos de evaluación y las normas adaptadas, las herramientas de evaluación, etc. El plan de evaluación será revisado y actualizado con la información adicional necesaria durante el proceso de evaluación. (ISO25000, 2024)

- Actividad 4: Hacer la evaluación

En este ejercicio se realizan pasos de revisión para obtener indicadores de calidad mediante la aplicación de estándares de evaluación.

  - Tarea 4.1: Tomar medidas sobre el producto

Es necesario llevar a cabo la evaluación del producto de software y sus elementos para adquirir los datos correspondientes a las métricas previamente seleccionadas y especificadas en el plan de evaluación. Todos los resultados obtenidos deben ser registrados de manera adecuada. (ISO25000, 2024)

  - Tarea 4.2: Poner los criterios en las métricas.

Se aplican los estándares de evaluación a los indicadores elegidos utilizando los valores derivados de la medición del producto. (ISO25000, 2024)

  - Tarea 4.3: Aplicar estándares de evaluación a las cualidades.

En esta etapa final, se aplican los estándares de evaluación a nivel de características y sub-características de calidad. Esto resulta en una evaluación que determina en qué medida el producto de software cumple con los requisitos de calidad establecidos. (ISO25000, 2024)

- Actividad 5: Fin de la evaluación

En esta etapa se finaliza la evaluación de la calidad del producto de software mediante la elaboración del informe de resultados, que será entregado al cliente. Además, se lleva a cabo una revisión conjunta con el cliente para analizar los resultados obtenidos.

- Tarea 5.1: Repasar los hallazgos con el cliente

Durante esta actividad, el evaluador y, en caso de estar presente, el cliente de la evaluación, llevan a cabo una revisión conjunta de los hallazgos obtenidos. El propósito es lograr una interpretación más precisa de la evaluación y una detección más efectiva de posibles errores. (ISO25000, 2024)

- Tarea 5.2: Desarrollar el documento de la evaluación.

Una vez examinados los resultados, se prepara el informe de evaluación que incluye los requisitos de la evaluación, los hallazgos, las limitaciones y restricciones identificadas, el equipo evaluador involucrado, entre otros aspectos pertinentes. (ISO25000, 2024)

- Tarea 5.3: Evaluar la calidad de la evaluación y recabar retroalimentación.

El evaluador analizará los resultados de la evaluación, así como la efectividad del proceso, los indicadores y las métricas empleadas. La retroalimentación obtenida de esta revisión se utilizará para mejorar tanto el proceso de evaluación de la organización como las técnicas de evaluación utilizadas. (ISO25000, 2024)

- Tarea 5.4: Administrar la información recopilada durante la evaluación.

Una vez concluida la evaluación, el evaluador debe manejar los datos y los elementos de la evaluación de acuerdo con lo acordado con el cliente (si este es una tercera parte). Esto puede implicar devolver, archivar o eliminar los datos y objetos según corresponda. (ISO25000, 2024)

2.3.4 Six Sigma

Six Sigma, ideado por Bill Smith de Motorola en los años ochenta, es un enfoque para reducir drásticamente los defectos en los productos. General Electric revitalizó este concepto a

finales del siglo XX, aplicándolo en toda su organización para mejorar tanto la fabricación como los servicios, obteniendo resultados impresionantes. (Conexión Esan, 2016)

Six Sigma, también conocido como DMAIC (Definir, Medir, Analizar, Mejorar y Controlar), representa un enfoque de gestión de calidad que se emplea en cada proceso. Estas cinco fases son esenciales para llevar a cabo la mejora continua en la calidad y eficiencia de los procesos organizacionales. (Sejzer, 2016)

Figura 12.

Las Etapas del Six Sigma

Fases de Six Sigma (Sejzer, 2016)

Etapas del Six Sigma:

Definición: se establecen los procesos que serán evaluados por la dirección de la empresa, así como el equipo de trabajo y los objetivos de mejora.

Medición: se evalúa el estado actual del problema o defecto en el proceso, clasificando y midiendo cada parte del mismo para identificar las variables relacionadas.

Análisis: se interpretan los resultados de la medición, comparando la situación actual con el historial del proceso para identificar los orígenes del problema.

Mejora: se implementan las acciones necesarias para mejorar el proceso.

III. MÉTODO

3.1 Tipos de investigación

3.1.1 Tipos de investigación

Para Arias Gonzáles y Covinos Gallardo (2021).” La investigación empleada se clasifica como Investigación Aplicada, dado que se aprovechan los conocimientos previos junto con la información obtenida de diversas fuentes bibliográficas relacionadas con las metodologías de calidad de software.” (p.68). En consecuencia, se utilizará el tipo de investigación Aplicativa.

3.1.2 Nivel de Investigación

Para Arias Gonzáles y Covinos Gallardo (2021). La investigación se enmarca en un enfoque explicativo. “Este alcance tiene la característica de establecer causa – efecto entre sus variables, son más profundas y estructuradas a diferente de los alcances previos. Existen las variables independientes (causas) y las variables dependientes (efectos) y las hipótesis se pueden plantear de forma que se establezca causalidad.”

3.1.3 Diseños de Investigación

Según Arias Gonzáles y Covinos Gallardo (2021). Se llama diseño pre experimental “por qué no cumple con los parámetros del experimento, por tal razón está fuera del campo de dicho diseño, al trabajar con un solo grupo de estudio este experimento carece de validez interna y externa en sus resultados; asimismo, la desventaja de este tipo de diseño es que el investigador no puede saber con total certeza los efectos que se han producido por causa de la variable independiente sobre la variable dependiente. El pre experimento tiene las siguientes características:

- Son grupos o sujetos que ya están conformados previamente.
- Solo existe un grupo llamado “grupo experimental”.
- Se puede aplicar un pre test y pos test.
- Se realizan las mediciones en no más de dos tiempos diferentes.

- En consecuencia, el nivel de la investigación es pre experimental.

3.2 Ámbito temporal y espacial

3.2.1 Ámbito temporal

Esta investigación tiene un horizonte temporal próximo, programado para el período 2023-2024, pero se prevé que la solución propuesta continúe siendo utilizada a largo plazo.

3.2.2 Ámbito espacial

El espacio de este proyecto corresponde al distrito de La Molina, Departamento de Lima, Perú (2023), con un tiempo aproximando de trabajo de 10 meses.

3.3 Variables

En el presente trabajo de investigación se ha identificado las siguientes variables:

3.3.1 Variable independiente

- Modelo de Desarrollo de software

3.3.2 Variables dependientes

- Calidad del Producto SaaS ERP

3.3.3 Operacionalización de variables

Tabla 1

Operacionalización de la Variable Independiente: Modelo de Desarrollo de Software

Variable	Detalle	Definición conceptual	Definición Operacional
Variable Independiente	El modelo de desarrollo de Software	Modelos de desarrollo de software son una colección de técnicas y sistemas organizacionales para crear software de computadora. El objetivo de los diversos enfoques es estructurar equipos de trabajo para que puedan construir las funcionalidades del programa de la manera más eficiente posible. Modelos de desarrollo de software proporcionar un marco para controlar el desarrollo de los sistemas de información. Desde la planificación hasta el mantenimiento, un Ciclo de vida del desarrollo de programas (SDLC) describe todos los procesos en un proyecto de desarrollo de software. Estos marcos incluyen el desarrollo de programas, así como las herramientas necesarias para ayudar en el proceso de desarrollo (Sharma, 2022)	La variable se evaluará mediante la información de los Project, herramientas de progreso de actividad y tiempo de desarrollo durante el proceso

Variable	Detalle	Indicador	Índice
Variable Independiente	El modelo de desarrollo de Software	Tasa de Errores	(1) Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100 (2) Eliminación de Defectos (EED) = E / (E + D)
		Liberación del Producto	(3) Tiempo de Entrega = Tiempo pre producción + Tiempo fabricación + Tiempo de Envío

Fuente: Elaboración propia

Donde el Indicador Tasa de Errores de Variable Independiente:

1. Disponibilidad: es la probabilidad de que un sistema, equipo o componente realice la función prevista cuando sea requerido. Se expresa en porcentaje y tiene en cuenta tanto la confiabilidad como la mantenibilidad del sistema Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100. (Fracttal, 2023).

2. Eliminación de Defectos: La eficacia de la eliminación de defectos (EED), es una métrica que permite medir la habilidad de filtrar las actividades de la garantía de calidad y de control, ya que es aplicable a todas las actividades del marco de trabajo del proceso. Donde E= es el número de errores encontrados antes de la entrega del software al usuario final y D= es el número de defectos encontrados después de la entrega Formula: EED = E / (E + D). (Wikia Calidadsoftware, 2023).

Donde el Indicador Liberación del Producto de Variable Independiente:

1. Tiempo de Entrega: Es el tiempo de ciclo se define como el tiempo que transcurre desde que se empieza a trabajar en una historia de usuario que está lista para ser implementada, hasta que está lista para ser entregada (liberada)el tiempo de ciclo se define como el tiempo que transcurre desde que se empieza a trabajar en una historia de usuario que está lista para ser implementada, hasta que está lista para ser entregada (liberada) Formula : Tiempo de Entrega = Tiempo pre producción + Tiempo fabricación + Tiempo de Envío (Nimble, 2023).

Tabla 2

Operacionalización de la Variable Dependiente: Calidad del Producto SaaS ERP

Variable	Detalle	Definición conceptual	Definición Operacional
Variable Dependiente	Calidad del producto SaaS ERP	Es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo plenamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente", con base en los requisitos funcionales y no funcionales identificados en la etapa de análisis del sistema, insumo principal para implementar dichos requisitos con los atributos mínimos de calidad, fomentando la aplicación de procesos estandarizados y criterios necesarios en cada una de sus etapas, así se fomenta que el avance en el ciclo de vida del software minimice el riesgo de fracaso del proyecto (Callejas-Cuervo et al., 2017)	La variable se evaluará mediante el análisis de una ficha de resultados y del llenado de un formulario para evaluar la calidad del software

Variable	Detalle	Indicador	Índice
Variable Dependiente	Calidad del producto SaaS ERP	Calidad de Software	(1) Precisión = (Número de pases correctas / Total de pases realizadas) * 100 (2) Índice de Productividad = Numero de Errores / Líneas de Código generadas / 100 (3) Eficiencia en el uso del tiempo = (Hora Trabajas al Mes / Horas Trabajadas Real) * 100 (4) Hitos alcanzados = Pases Realizados *100 / Pases Correctos (5) Índice de BUG = Número de líneas físicas/Número de líneas observadas (6) Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100

Fuente: Elaboración propia

Donde el Indicador Calidad del Software de la Variable Dependiente:

1. Calidad del trabajo = Esta métrica pretende medir la Precisión a evaluar el porcentaje de trabajo sin errores:

Formula: Precisión = (Número de pases correctas / Total de pases realizadas) * 100

2. Índice de Productividad: Esta Métrica mide la cantidad de trabajo realizado en relación con la tasa de defectos. (Saavedra Martínez et al., 2019).

Formula: Productividad = Numero de Errores / Líneas de Código generadas / 100

3. Eficiencia en el uso del tiempo: Esta métrica va a Evaluar la distribución del tiempo entre tareas importantes y otras actividades.

Formula: Eficiencia en el uso del tiempo = (Hora Trabajas al Mes / Horas Trabajadas Real) * 100.

4. Hitos alcanzados: Mide el cumplimiento de los objetivos de evitar que un Requerimiento pase a un Segundo ciclo de pruebas.

Formula: Hitos alcanzados = (Pases Realizados * 100 / Pases Correctos)

5. Índice de BUG en el Código: Mide el % de observaciones por la detección y corrección de defectos y vulnerabilidades para la mejora continua de la calidad del código. (Larios Calleja, 2021)

Formula: Índice de BUG = Número de líneas físicas/Número de líneas observadas

6. Disponibilidad: es la probabilidad de que un sistema, equipo o componente realice la función prevista cuando sea requerido. Se expresa en porcentaje y tiene en cuenta tanto la confiabilidad como la mantenibilidad del sistema.

Formula: Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100. (Fracttal, 2023).

3.4 Población y muestra

3.4.1 Población

La población de la investigación es de por 850 programas de Software SaaS ERP Integrator dados a disponibilidad por el área de desarrollo de software de la empresa Integrator SAC.

3.4.2 Muestra

La muestra de estudio corresponde a las denominadas muestras probabilísticas especificadas proporcionalmente para las empresas Integrator SAC.

La muestra fue de tipo aleatoria-sistemática y su tamaño será calculado usando la siguiente fórmula, de la figura 9 y figura 10, de población finita con proporciones:

Datos:

- Población (N): 850 programas fuentes del Software SaaS ERP.
- Z: 1.96 Coeficiente de confiabilidad para el 95% de nivel de confianza.
- P y q: 0.5 Probabilidad de la población de estar o no incluidas en la muestra.
- E: 5% Representa el error estándar de la estimación

Figura 13.

Formula: de Muestra Población Finita

$$n = \frac{(p.q)\ Z^2 . N}{(E^2)\ (N-1)\ + (p.q)\ Z^2}$$

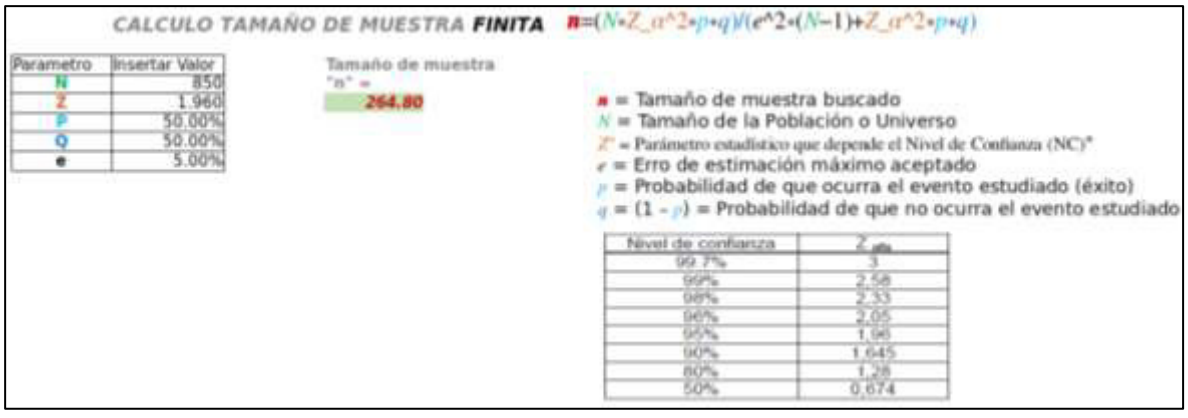
Fuente: Elaboración propia

El tamaño de la muestra es de 265

Sustituyendo y Calculando:

Figura 14.

Desarrollo del Cálculo tamaña de Muestra Finita



Fuente: Elaboración propia

3.5 Instrumentos

3.5.1 Técnicas e Instrumentos de Recolección de datos

Según Hernández Sampieri et al. (1991). Las técnicas de recolectar los datos implican tres actividades estrechamente vinculadas entre sí:

- Seleccionar un instrumento de medición de los disponibles en el estudio del comportamiento o desarrollar uno (el instrumento de recolección de los datos). Este instrumento debe ser válido y confiable, de lo contrario no podemos basamos en sus resultados.
- Aplicar ese instrumento de medición. Es decir, obtener las observaciones y mediciones de las variables que son de interés para nuestro estudio (medir variables).
- Preparar las mediciones obtenidas para que puedan analizarse correctamente (a esta actividad se le denomina codificación de los datos).

Los instrumentos de recolección de los datos deben reunir dos requisitos esenciales: confiabilidad y validez. La confiabilidad de un instrumento de medición se refiere al grado en que su aplicación repetida al mismo sujeto u objeto produce iguales resultados. (Hernández Sampieri, 1991).

- La confiabilidad de un instrumento de medición se refiere al grado en que su aplicación repetida al mismo sujeto u objeto produce iguales resultados.
- La validez, en términos generales, se refiere al grado en que un instrumento realmente mide la variable que pretende medir.
- Las principales técnicas e instrumentos a utilizar en la investigación son:

Tabla 3

Técnicas e instrumentos utilizados

Técnica	Instrumento de Recolección
Encuesta	Google Forms / Cuestionarios /Entrevista
Análisis Documental	SonarQube

Fuente: Elaboración propia

También se incluirá:

- ISO 25000
- Acta de requerimientos del sistema SaaS Integrator ERP
- MS Project
- SPSS (versión 26).

La investigación incluirá una entrevista con el gerente de los proyectos y a los equipos de desarrollo tanto del lado de la empresa como el personal de QA de los clientes, esto con el objetivo de evaluar el estado actual de la calidad de los programas, las principales falencias y el alcance que se desea obtener con la aplicación del nuevo modelo propuesto la implementación del sistema SaaS ERP.

Además, se utilizarán cuestionarios, que podrán ser completados tanto en línea como fuera de línea, aprovechando herramientas como Google Forms. El cuestionario virtual constará de 16

preguntas con escala de Likert, diseñadas según las variables definidas en la investigación y que fueron contestadas por 20 personas de los proyectos del sistema SaaS ERP (ver tabla 4).

Las preguntas cerradas, como lo indica la tabla 5, se emplearán para obtener una cobertura exhaustiva del tema, y se validarán posteriormente. La escala de Likert se presenta detalladamente en la Matriz de Consistencia lógica. La elaboración de las preguntas está en concordancia con las dimensiones e indicadores de cada una de las variables (Ver Anexos), En concordancia con la Matriz de Consistencia.

Tabla 4

Lista de entrevistados y su distribución

Cargo	Empresa	Número de Personas Entrevistadas
Product Leader	Empresa Consultora	1
Equipo de Desarrollo	Empresa Consultora	3
Personal de QA	Empresa Consultora	2
PMO	Cliente A	1
Personal de QA Funcional	Cliente A	2
Usuario Final	Cliente A	3
PMO	Cliente B	1
Personal de QA Funcional	Cliente B	2
Usuario Final	Cliente B	1
PMO	Cliente C	1
Personal de QA Funcional	Cliente C	1
Usuario Final	Cliente C	2
Total		20

Fuente: Elaboración propia

Tabla 5

La escala está definida de la siguiente manera

Codificación en los Criterio de Evaluacion/Escala de Valoracion				
Totalmente desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
1	**2**	**3**	**4**	**5**

Fuente: Elaboración propia

3.5.2 Validación y Confiabilidad de instrumentos

Validez

Según Espinosa-Solís et al. (2021) define "Juicio Experto" a la opinión de personas con trayectoria en el tema, reconocidas por otros. Por lo cual la presente investigación, el instrumento de validación a usar, son el juicio Experto.

El instrumento ha sido validado por tres expertos doctores, todos con especialidad en Desarrollo de Sistemas y un experto en estadística.

Tabla 6

Relación de Juicio Experto

#	Nombre y Apellido	Grado Académico	Tipo de Experto	Determinación
1	IVAN CARLO PETRILK AZABACHE	Dr. Ing. De Sistemas	Especialista	Aplicable
2	LEZAMA GONZALES PEDRO MARTIN	Dr. Ing. de Sistemas	Especialista	Aplicable
3	GUILLERMO PASTOR MORALES	Mg. Ing. De Sistemas	Especialista	Aplicable

Fuente: Elaboración propia

Confiabilidad

Según Peraza et al. (2017), hay que afirmar que la confiabilidad se refiere al grado en el que se obtienen resultados sólidos y consistentes, es decir, la medida en que se obtienen resultados similares al utilizar métodos y medidas de coherencia. En esta investigación, se empleó el coeficiente *c* como instrumento de medición. Este coeficiente requiere una sola administración del instrumento y produce valores que varían entre cero y uno. Su ventaja radica en que no es necesario dividir los ítems del instrumento en dos partes, ya que se aplica la medición y se calcula el coeficiente directamente.

Por lo general, el alfa es calculada mediante el uso de software estadístico, como SPSS, R-Studio, cuya fórmula (figura 15) es la siguiente:

Figura 15.

Formula del Alfa de Cronbach

$$\alpha = \frac{N * \bar{c}}{\bar{v} + (N - 1) * \bar{c}}$$

Fuente: Elaboración propia

Donde:

- N = el número de elementos.
- $\bar{c}$ = covarianza promedio entre pares de ítems.
- $\bar{v}$ = varianza promedio.

Los valores del Alfa de Cronbach van de 0 a 1. (Ver figura 15), Mayor fiabilidad cuando se acerca a 1, menor fiabilidad o consistencia cuando se acerca a cero. Cuando tiende a cero indica que no hay ninguna correlación entre los elementos. Son totalmente independientes; Es decir conocer el valor de una respuesta a una pregunta no proporciona información sobre las respuestas a las otras preguntas. Cuando tiende a 1 indica que están perfectamente correlacionados. Conocer

el valor de una respuesta proporciona información completa sobre los demás elementos. (gplresearch.com, 2023).

Figura 16.

Rangos del Alfa de Cronbach

Alfa de Cronbach	Consistencia Interna
α ≥ 0,9	Excelente
0,8 ≤ α < 0,9	Buena
0,7 ≤ α < 0,8	Aceptable
0,6 ≤ α < 0,7	Cuestionable
0,5 ≤ α < 0,6	Pobre
α < 0,5	Inaceptable

Rangos del Alfa de Cronbach (gplresearch.com, 2023).

Confiabilidad Pre-Test

Para el instrumento que midió la variable dependiente antes de la aplicación del nuevo modelo de desarrollo de software y puede observar que el alfa de Cronbach fue de 0.876, que denota que es instrumento confiable.

Tabla 7

Resultado de Fiabilidad Pre-Test

Estadísticas de fiabilidad	
Alfa de Cronbach	N de elementos
,876	8

Fuente: Elaboración propia. Se muestra la confiabilidad del Pre-Test

Confiabilidad Pos-Test

Para el instrumento que midió la variable dependiente después de la aplicación del nuevo modelo de desarrollo de software y puede observar que el alfa de Cronbach fue de 0.934, que denota un instrumento confiable, motivo por el cual se aprueba el instrumento.

Tabla 8

Resultado de Fiabilidad Pos-Test

Estadísticas de fiabilidad	
Alfa de Cronbach	N de elementos
,934	8

Fuente: elaboración propia. Se muestra la confiabilidad del Pos-Test

3.6 Procedimientos

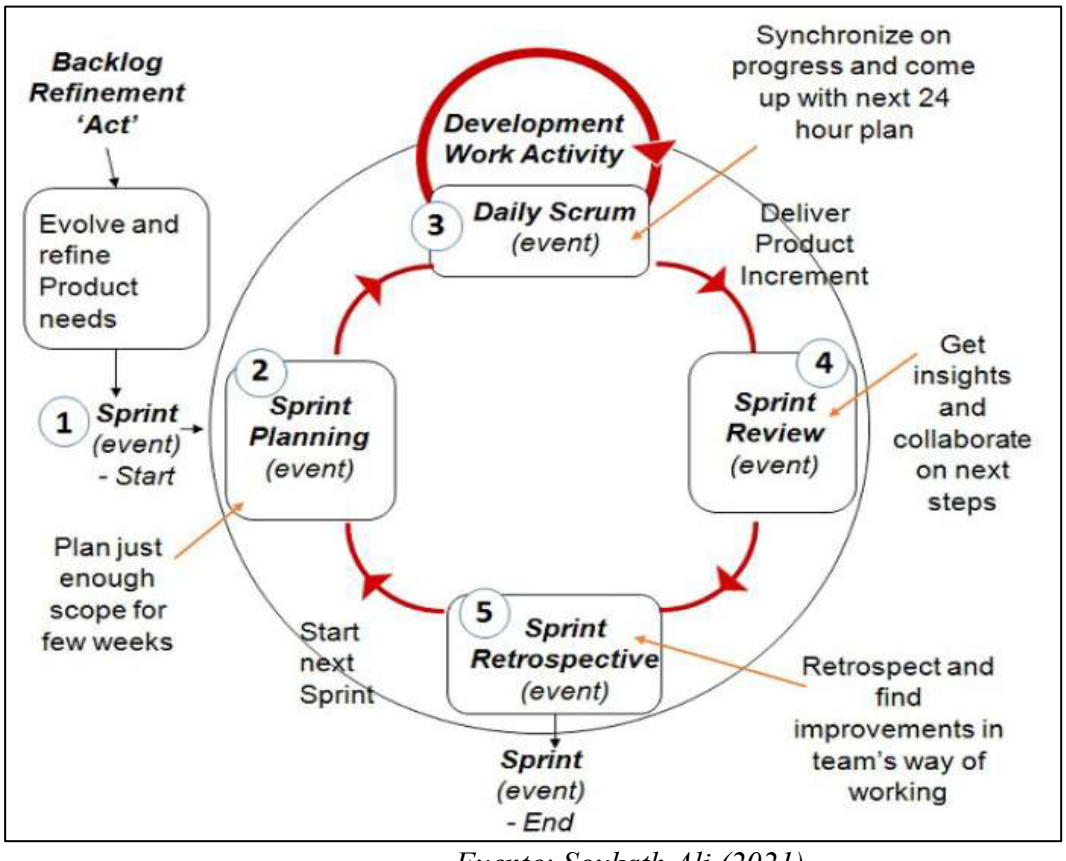
El proceso constara de las siguientes etapas:

- La primera parte será tomar muestras de programas observados en el periodo 2022 (Enero – Octubre), antes de la aplicación del nuevo modelo (pre test) de desarrollo de software.
- Implementar el nuevo modelo de desarrollo de software (variable independiente), que constara de las siguientes actividades:
- Crear actividades de evangelización del modelo SCRUM para remplazar el modelo Cascada.
- Organizar los procesos de organización, desarrollo, priorización de software, bajo un modelo SCRUM.
- Organizar el Backlog con las muestras
- Desarrollar las ceremonias propias del SCRUM (figura 17)
- Implementar el modelo de QA interno dentro de los procesos de revisión de código

- Usar la herramienta SONARQUBE para la identificación de malas prácticas en el código de los programas
- Luego tomara una segunda muestra luego de la aplicación del nuevo modelo de desarrollo (post test).
- Esos datos se llenarán en fichas de observación para su análisis y procesamiento.
- Utilizando la base de datos estadístico SPSS se procederá al análisis estadístico para obtener los siguientes resultados: Se procederá a describir los datos de cada variable a estudiar calculando el grado de significancia, la confiabilidad, la normalidad.
- Luego se procederá a medir la correlación entre dos variables (pre – test y post test), para lo cual se utilizará la correlación de T Wilcoxon para determinar si existe influencia significativa de las dimensiones con las variables.

Figura 17.

Eventos / Ceremonias Scrum



Fuente: Soukath Ali (2021)

3.7 Análisis de datos

Se analizó cual el nivel de significancia y la normalidad de los datos; para lo cual se utilizó el SPSS versión 26.0. Para calcular la Tabla de Pruebas de Normalidad se considera el número de muestras (NM). Si NM > 50 sujetos se usan Kolmogorov –Smirnov, caso contrario se deber de usar Shapiro – Wilk. En este caso la muestra es de 20 muestras por lo tanto se utilizará Shapiro – Wilk.

Posteriormente se pasó a verificar si los datos son normales, para lo cual se usó el Nivel de significancia (Sig). En caso el Sig >0,05 entonces quiere decir que los datos son normales, por lo que se puede aplicar Pruebas Paramétricas como R de Pearson. En caso contrario, los datos no son normales entonces puede aplicar Pruebas No Paramétricas como por ejemplo de T de Wilcoxon, Chi Cuadrado, Kendall o Rho de Spearman.

Tabla 9

Pruebas de Normalidad

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			**Shapiro-Wilk**		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Pre Test	,302	20	,000	,798	20	,001
Pos Test	,192	20	,053	,905	20	,051

a. Corrección de significación de Lilliefors

Fuente: Elaboración propia, Resultados de las pruebas de normalidad

En cuanto a la Prueba de Normalidad. Tomando los valores de la siguiente tabla, se observa un *pvalor* (Sig) **menor** que 0.05.

Para la primara muestra (Pre Test) se observa un *pvalor* (Sig) menor (0.001), lo que indica que sus datos **no son normales**.

Para la segunda muestra (Pos Test) se observa un *pvalor* (Sig) mayor (0.051) lo que indica que sus datos **son normales.**

Por lo tanto, al ser este una investigación pre-experimental, se usará la prueba de T de Wilcoxon de tipo no paramétricas.

3.8 Consideraciones Éticas

En el marco de este estudio de investigación, se recopiló la información de manera imparcial y objetiva, asegurando que no se alterara ni manipulara, con el objetivo de garantizar la fiabilidad de los resultados obtenidos. Se consideraron aspectos éticos, como la confidencialidad de los sujetos investigados y el respeto a la voluntad de los estudiantes que decidieron participar o no. Para asegurar una estructura sólida en el trabajo de investigación, se implementaron las pautas establecidas en las Normas APA 7ª Edición.

También dentro de las consideraciones Éticas se está considerando la firma de un acuerdo de confidencialidad y un acuerdo de tratamiento de datos personales bajo la Ley Peruana 29733, Ley de protección de datos personales

IV. RESULTADOS

4.1 Análisis e Interpretación de Resultados

Los resultados han sido obtenidos usando el software estadístico SPSS versión 26, dando como resultado de la prueba de Wilcoxon, siguiendo los objetivos de la investigación.

Objetivo general:

Determinar la influencia, a través de la comprobación de las hipótesis, del nuevo modelo de software para mejorar la calidad del software SaaS ERP tanto en la disminución de los tiempos en el análisis y desarrollo como en la reducción de defectos. (Ver presentación de resultado para más detalle).

Esto se evidencio al identificar el alto grado de correlación que existe entre las variables independientes y dependiente, lo que significa que mientras más se implementa el nuevo modelo de desarrollo de software, mejor será la calidad del software SaaS ERP.

Presentación de resultado para más detalle en la sesión de prueba de hipótesis sección 4.2).

A continuación, se presentan resultados obtenidos a lo largo de la investigación relacionada con la implementación del nuevo modelo de software para el ERP SaaS ERP. Los resultados se organizan de acuerdo con los objetivos específicos de la investigación, con el propósito de proporcionar una visión completa y detallada del rendimiento y la eficacia de la aplicación

4.2 Prueba de Hipótesis

4.2.1 Prueba de Hipótesis General

Hipótesis Nula

La implementación de un modelo de desarrollo de software no contribuye en mejorar la Calidad del Producto SaaS ERP de la Empresa Integrator en Lima – Perú año 2023

Hipótesis Alternativa

La implementación de un modelo de desarrollo de software contribuye en mejorar la Calidad del Producto SaaS ERP de la Empresa Integrator en Lima – Perú año 2023

Tabla 10

Prueba de rangos con signo de Wilcoxon para la Hipótesis General

Rangos

		N	Rango promedio	Suma de rangos
% Mejora en la Calidad Pos-Test - % Mejora en la Calidad Pre-Test	Rangos negativos	10[a]	6,80	68,00
	Rangos positivos	253[b]	136,95	34648,00
	Empates	1[c]		
	Total	264		

a. % Mejora en la Calidad Pos-Test < % Mejora en la Calidad Pre-Test

b. % Mejora en la Calidad Pos-Test > % Mejora en la Calidad Pre-Test

c. % Mejora en la Calidad Pos-Test = % Mejora en la Calidad Pre-Test

Estadístico de prueba[a]

	% Mejora en la Calidad Pos-Test - % Mejora en la Calidad Pre-Test
Z	-14,003[b]
Sig. asintótica (bilateral)	,000

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos negativos.

Fuente: Elaboración propia con SPSS.

Se puede observar en la estadística de la prueba de Wilcoxon el grado de significancia nos da un valor de (0.000) menor a 0.05, por lo cual se rechaza la hipótesis Nula y se acepta la hipótesis Alternativa. Lo que significa, que la implementación del nuevo modelo de desarrollo de software contribuirá en mejorar la Calidad del Producto SaaS ERP.

4.2.2 Prueba de Hipótesis Específica I

Hipótesis Nula

La implementación de un modelo de desarrollo de software no contribuye a reducir los errores en el Producto del SaaS ERP.

Hipótesis Alternativa

La implementación de un modelo de desarrollo de software contribuye a reducir los errores en el Producto del SaaS ERP.

Tabla 11

Prueba de rangos con signo de Wilcoxon para la hipótesis especifica I

Rangos

		N	Rango promedio	Suma de rangos
Tasa de Defectos Antes de la aplicación del modelo - Tasa de Defectos Antes de la aplicación del modelo	Rangos negativos	73[a]	45,42	3316,00
	Rangos positivos	17[b]	45,82	779,00
	Empates	174[c]		
	Total	264		

a. Tasa de Defectos Antes de la aplicación del modelo < Tasa de Defectos Antes de la aplicación del modelo

b. Tasa de Defectos Antes de la aplicación del modelo > Tasa de Defectos Antes de la aplicación del modelo

c. Tasa de Defectos Antes de la aplicación del modelo = Tasa de Defectos Antes de la aplicación del modelo

Estadístico de prueba[a]

	Tasa de Defectos Antes de la aplicación del modelo - Tasa de Defectos Antes de la aplicación del modelo
Z	-5,104[b]
Sig. asintótica (bilateral)	,000

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos positivos.

Fuente: Elaboración propia con SPSS.

Se puede observar en la estadística de la prueba de Wilcoxon el grado de significancia nos da un valor de (0.000) menor a 0.05, por lo cual se rechaza la hipótesis Especifica I Nula y se acepta

la hipótesis Alternativa. Lo que significa, que la implementación del nuevo modelo de desarrollo de software contribuirá a reducir los errores en el Producto del SaaS ERP.

4.2.3 Prueba de Hipótesis Específica II

Hipótesis Nula

La implementación de un modelo de desarrollo de software no contribuye a reducir el tiempo de entrega del Producto SaaS ERP.

Hipótesis Alternativa

La implementación de un modelo de desarrollo de software contribuye a reducir el tiempo de entrega del Producto SaaS ERP.

Tabla 12

Prueba de rangos con signo de Wilcoxon para la hipótesis especifica II

Rangos

		N	Rango promedio	Suma de rangos
Total de Horas al Mes después de aplicación del Modelo - Total de Horas al Mes antes de aplicación del Modelo	Rangos negativos	8[a]	5,50	55,00
	Rangos positivos	0[b]	0,00	0,00
	Empates	0[c]		
	Total	10		

a. Total de Horas al Mes después de aplicación del Modelo < Total de Horas al Mes antes de aplicación del Modelo

b. Total de Horas al Mes después de aplicación del Modelo > Total de Horas al Mes antes de aplicación del Modelo

c. Total de Horas al Mes después de aplicación del Modelo = Total de Horas al Mes antes de aplicación del Modelo

Estadístico de prueba[a]

	Total, de Horas al Mes después de aplicación del Modelo - Total de Horas al Mes antes de aplicación del Modelo
Z	-2,807[b]
Sig. asintótica (bilateral)	,001

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos positivos.

Fuente: Elaboración propia con SPSS.

Se puede observar en la estadística de la prueba de Wilcoxon el grado de significancia nos da un valor de (0.001) menor a 0.05, por lo cual se rechaza la hipótesis Especifica II Nula y se acepta la hipótesis Alternativa. Lo que significa, que la implementación del nuevo modelo de desarrollo de software contribuye en reducir el tiempo de entrega del Producto.

4.3 Presentación de Resultados

4.3.1 Análisis de Confiabilidad

Tabla 13

Análisis de Confiabilidad PRE TEST de instrumento

Resumen de procesamiento de casos

		N	%
Casos	Válido	20	7,5
	Excluido[a]	245	92,5
	Total	265	100,0

a. La eliminación por lista se basa en todas las variables del procedimiento.

Estadístico de fiabilidad

Alfa de Cronbach	N de elementos
,876	8

Fuente: Elaboración propia con SPSS.

Tabla 14

Análisis de Confiabilidad POS TEST de instrumento

Resumen de procesamiento de casos

		N	%
Casos	Válido	20	7,5
	Excluido[a]	245	92,5
	Total	265	100,0

a. La eliminación por lista se basa en todas las variables del procedimiento.

Estadísticas de fiabilidad

Alfa de Cronbach	N de elementos
,934	8

Fuente: Elaboración propia con SPSS.

4.3.2 Análisis de Normalidad

Tabla 15

Análisis de Normalidad para las muestras PRE y POS Test

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
Pre Test	20	7,5%	245	92,5%	265	100,0%
Pos Test	20	7,5%	245	92,5%	265	100,0%

Descriptivos

			Estadístico	Error estándar
Pre Test	Media		31,80	,687
	95% de intervalo de confianza para la media	Límite inferior	30,36	
		Límite superior	33,24	
	Media recortada al 5%		31,89	

	Mediana		33,00	
	Varianza		9,432	
	Desviación estándar		3,071	
	Mínimo		27	
	Máximo		35	
	Rango		8	
	Rango intercuartil		6	
	Asimetría		-,639	,512
	Curtosis		-1,431	,992
Pos Test	Media		35,35	,737
	95% de intervalo de confianza para la media	Límite inferior	33,81	
		Límite superior	36,89	
	Media recortada al 5%		35,39	
	Mediana		36,50	
	Varianza		10,871	
	Desviación estándar		3,297	
	Mínimo		30	
	Máximo		40	
	Rango		10	
	Rango intercuartil		6	
	Asimetría		-,422	,512
	Curtosis		-1,097	,992

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Pre Test	,302	20	,000	,798	20	,001
Pos Test	,192	20	,053	,905	20	,051

a. Corrección de significación de Lilliefors

Fuente: Elaboración propia con SPSS.

Figura 18.

Gráfico de Tallo y Hojas Muestra Pre Test

Pre Test

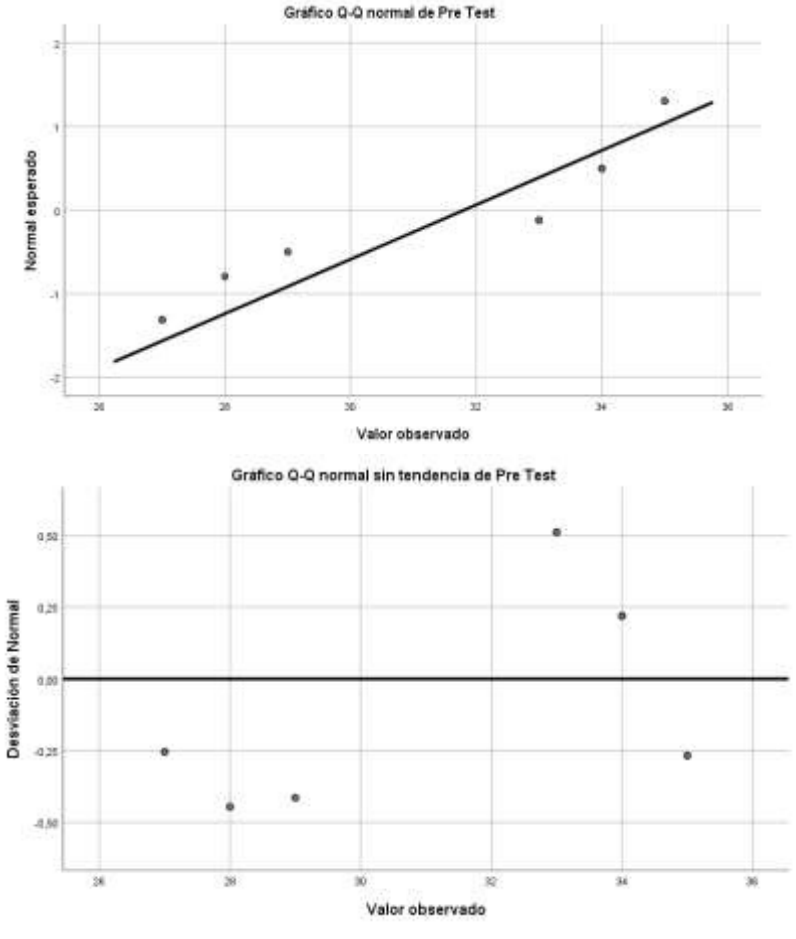
```
Pre Test Gráfico de tallo y hojas
```

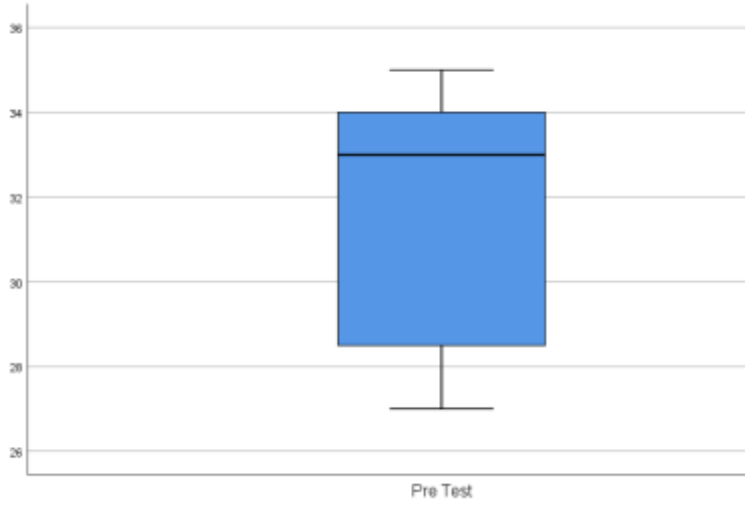
```
Frecuencia    Stem &  Hoja

     3,00       27 .  000
      ,00       27 .
     2,00       28 .  00
      ,00       28 .
     2,00       29 .  00
      ,00       29 .
      ,00       30 .
      ,00       30 .
      ,00       31 .
      ,00       31 .
      ,00       32 .
      ,00       32 .
     4,00       33 .  0000
      ,00       33 .
     6,00       34 .  000000
      ,00       34 .
     3,00       35 .  000

Ancho del tallo:         1
Cada hoja:        1 caso(s)
```

Fuente: Elaboración propia con SPSS.

Figura 19.

Gráficos de Normalidad PRE Test





Fuente: Elaboración propia con SPSS.

Figura 20.

Gráfico de Tallo y Hojas Muestra POS Test

Pos Test

Pos Test Gráfico de tallo y hojas

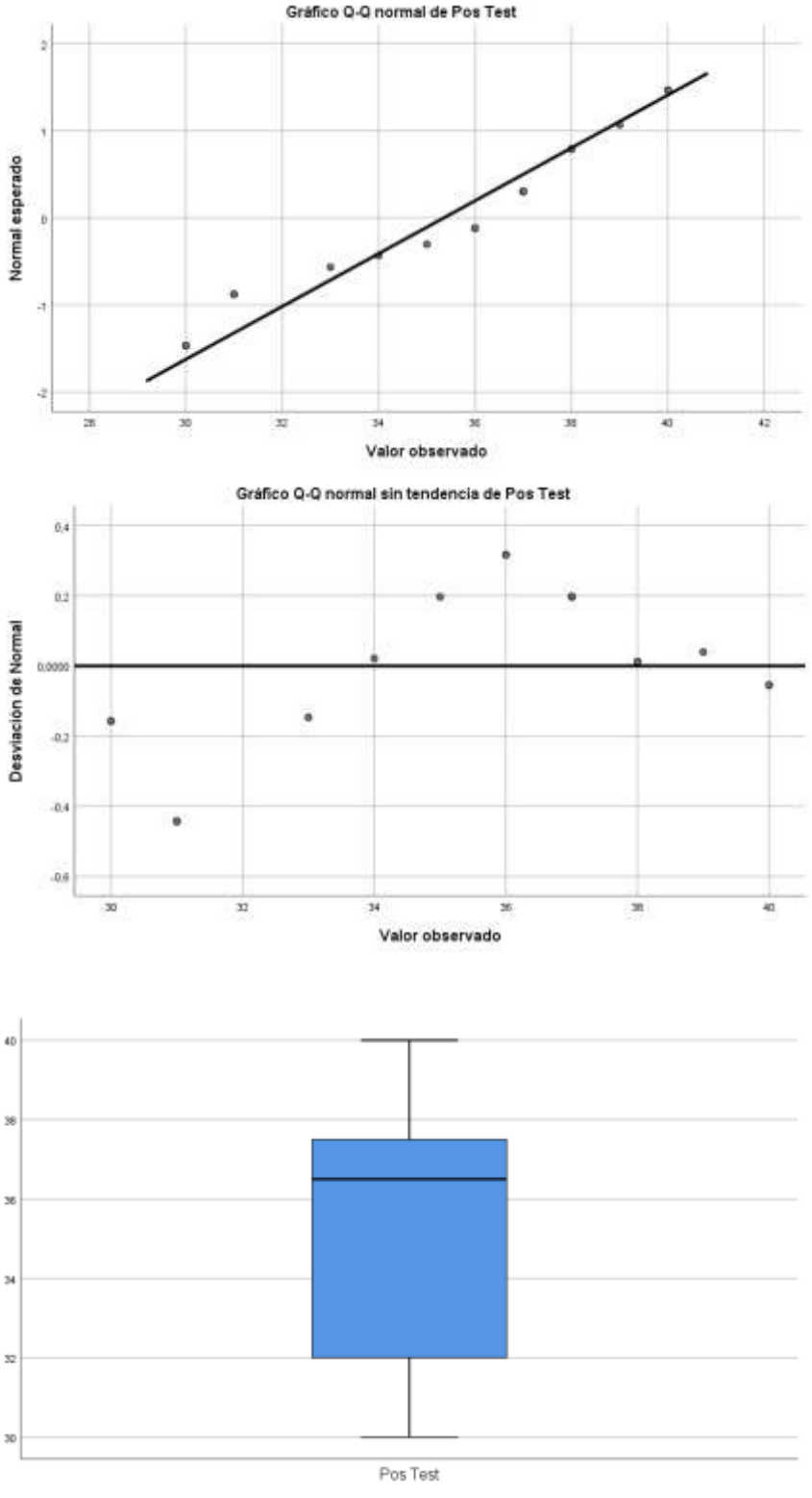
Frecuencia	Stem & Hoja
2,00	30 . 00
3,00	31 . 000
,00	32 .
1,00	33 . 0
1,00	34 . 0
1,00	35 . 0
2,00	36 . 00
5,00	37 . 00000
2,00	38 . 00
1,00	39 . 0
2,00	40 . 00

Ancho del tallo: 1
Cada hoja: 1 caso(s)

Fuente: Elaboración propia con SPSS.

Figura 21.

Gráficos de Normalidad POS Test



Fuente: Elaboración propia con SPSS.

V. DISCUSIÓN DE RESULTADOS

Respecto al análisis de datos los resultados de la investigación fueron los siguientes:

Tras realizar el análisis de los datos y obtener los resultados, aceptando la hipótesis general que establece la relación positiva e influyente entre las variables tomadas antes de la aplicación del nuevo modelo de desarrollo de software y después de la aplicación del nuevo.

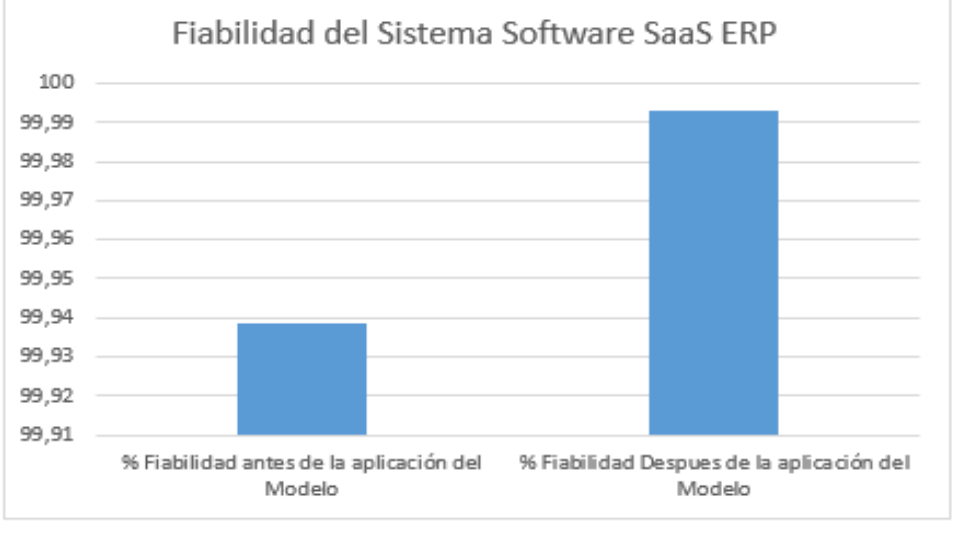
Dicho resultado tiene relación con lo que manifestó González (2008) en donde evidencia en los resultados en su trabajo de maestría , que luego de haber realizado su estudio sobre el efecto de las metodologías agiles sobre el desarrollo de software evidencia que la evolución de producto software utilizando una metodología SCRUM ,que En términos generales podemos concluir que el equipo ha sido ágil en la evolución del producto pues un 97% de las necesidades del cliente han sido cubiertas tras el mismo, en un espacio muy reducido de tiempo.

Al revisar el desarrollo de la hipótesis general podemos observar la influencia de la nueva metodología sobre el producto, software en la fiabilidad y la precisión de los programas.

Figura 22.

Impacto (%) sobre la fiabilidad de los programas

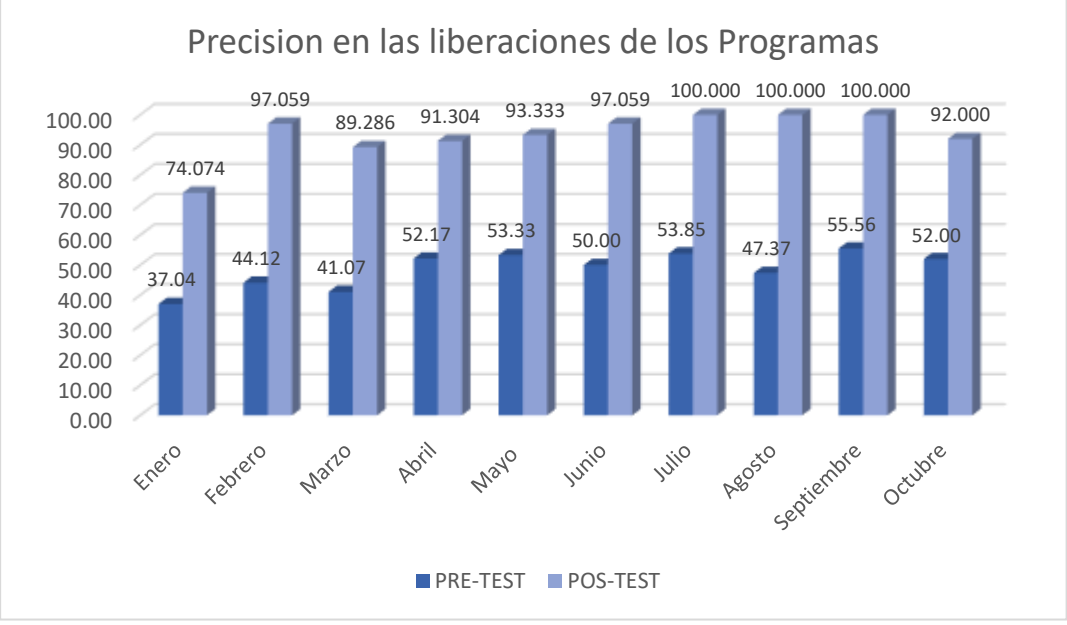
% Fiabilidad antes de la	% Fiabilidad Despues de la
99,93860549	99,99297652



Fuente: Elaboración propia tomando como datos los resultados obtenidos

Figura 23.

Impacto (%) sobre la Precisión de los programas



Fuente: Elaboración propia tomando como datos los resultados obtenidos

Por otro lado, el manejo de los errores del software no solo afecta, por un lado, la calidad del producto, sino también el tiempo de disponibilidad del servicio y la seguridad del producto y por otro lado al cliente, ya que su servicio podría estar comprometido sus operaciones administrativo financiero, así como la integridad de su información y su reputación.

Al implementarse el proceso de QAInterno ha permitido mejorar el control de calidad del código desarrollando seguimientos en las revisiones de Pares, revisiones independientes y auto revisiones.

Para Zazworka et al. (2011) “La deuda técnica es una metáfora que describe situaciones en las que los desarrolladores aceptan sacrificios en una dimensión del desarrollo (por ejemplo, la calidad del software) para optimizar otra dimensión (por ejemplo, implementar las funciones necesarias antes de una fecha límite). Se han desarrollado enfoques, como la detección de código erróneo (smell code), para identificar tipos particulares de deuda, es decir, deuda de diseño. Lo que aún no se ha entendido es el impacto que tiene la deuda de diseño en la calidad de un producto de software. Responder a esta pregunta es importante para comprender cómo la creciente deuda afecta

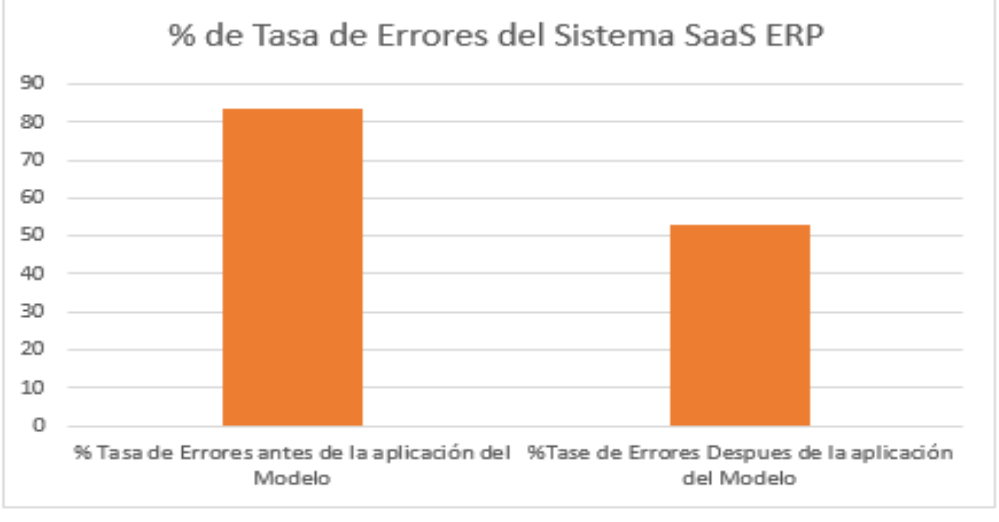
a un producto de software y cómo ralentiza el desarrollo, p. aunque se introducen modificaciones como la corrección de errores. “

Al revisar el desarrollo de la hipótesis especifica I (capitulo 4, anexo 2.2), podemos observar la influencia de la nueva metodología sobre el la reducción los errores en el Producto del SaaS ERP.

Figura 24.

Impacto (%) en la reducción de errores en los programas

% Tasa de Errores antes de la aplicación del Modelo	%Tase de Errores Despues de la aplicación del Modelo
83,71	53,03

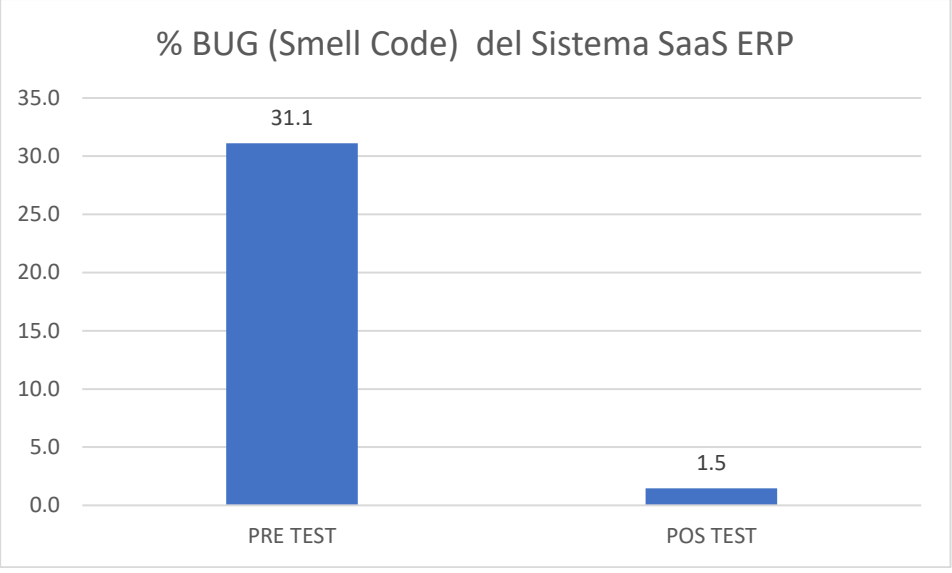


Fuente: Elaboración propia

Efecto en la aplicación de herramientas SONARQUBE (figura 21) dentro de los procesos de QA Interno (Anexo 10.5) para la mejora en la calidad del código son el objetivo de incrementar la eficiencia y calidad del código.

Figura 25.

Impacto (%) en la reducción BUG en los programas



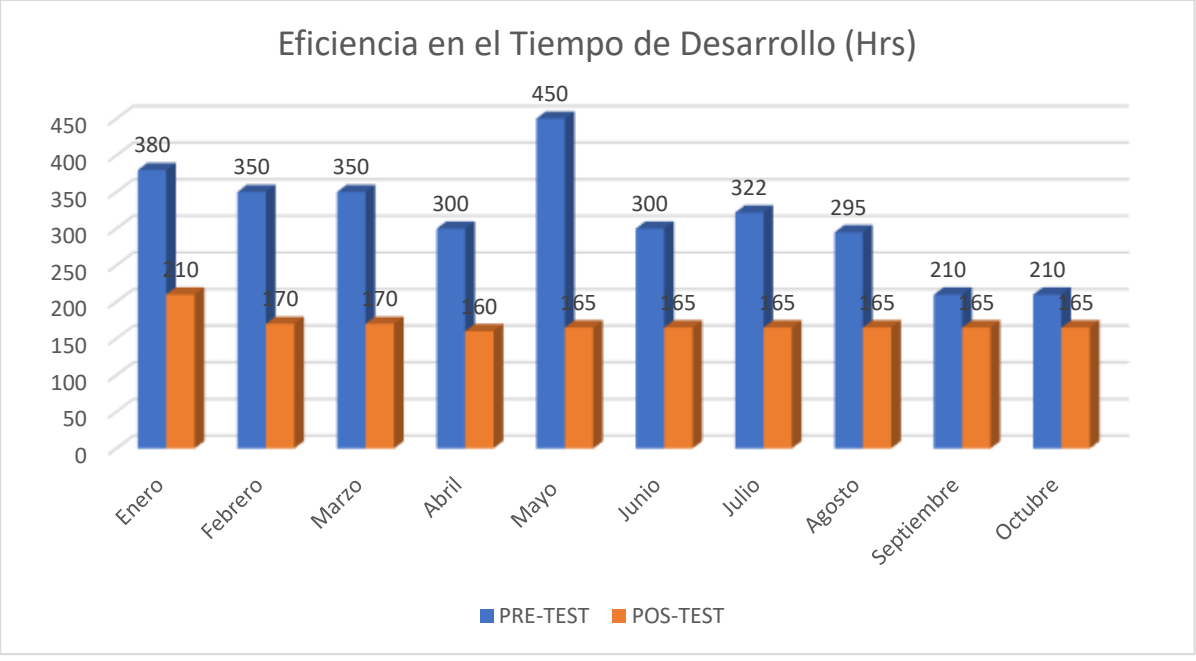
Fuente: Elaboración propia

Como parte de la aplicación del nuevo modelo de desarrollo, la organización del planeamiento, seguimiento, construcción, pruebas de calidad se vieron mejoradas. La implementación de la nueva organización del desarrollo, bajo el marco SCRUM, permitió cumplir con el control de estos procesos, todas con un sentido. También se evidencio el efecto de la implementación de las ceremonias dentro del desarrollo de los Sprint

Al revisar el desarrollo de la hipótesis Especifica II (capitulo 4, anexo 2.3), podemos observar la influencia de la nueva metodología sobre el producto, software en los tiempos de entrega de los programas y el efecto en la aplicación de las ceremonias y los Sprint (Anexo 10.5) con el objetivo de una mejora organización en el desarrollo y en el tiempo de entrega del código con valor al cliente.

Figura 26.

Eficiencia en el Tiempo de Desarrollo expresado en Horas



Fuente: Elaboración propia tomando como datos los resultados obtenidos

Basados en estos resultados podemos afirmar que las hipótesis descritas dentro de este proceso de investigación se relacionan eficazmente con el incremento de la calidad del Software SaaS ERP incrementando la eficacia del producto, el tiempo de entrega, la calidad en el código y reducciones los defectos de los programas.

VI. CONCLUSIONS

- La implementación de un modelo de desarrollo de software ayudó a mejorar la Calidad del Producto SaaS ERP respaldada por un valor de significancia es de 0,000, lo cual es significativamente inferior al umbral teórico de 0,05.
- La implementación de un modelo de desarrollo de software ayudó a reducir los errores en el Producto del SaaS ERP respaldada por un valor de significancia es de 0,000, lo cual es significativamente inferior al umbral teórico de 0,05.
- La implementación de un modelo de desarrollo de software ayudó a contribuir a la reducir el tiempo de entrega del Producto SaaS ERP, respaldada por un valor de significancia es de 0,001, lo cual es significativamente inferior al umbral teórico de 0,05.

VII. RECOMENDACIONES

- En vista a los resultados obtenidos, se recomienda utilizar el nuevo modelo de desarrollo de software para mejorar la calidad del sistema SaaS ERP, para mejorar la calidad del producto, mejorar los tiempos de los recursos y la reducción de errores en el código.
- Se debe de tener en cuenta que el equipo de desarrolladores deberá de continuar con la evangelización de las herramientas SCRUM y de sus artefactos con el objetivo de tener una mejora continua en sus procesos de desarrollo.
- Se debe considerar la posibilidad de integrar nuevas herramientas que ayude a mejorar los tiempos en las liberaciones de código, automatizando los procesos de liberación de productos lo cual ayudada a incrementar los procesos de monitoreo permitiendo crear un sistema de información unificado y eficiente.

VIII. REFERENCIAS

Acosta, N., Espinel, L. A., y García, J. (julio de 2017). *Estándares para la calidad de software*. Tecnología Investigación y Academia, 5(1), 75-84. Recuperado de URL: https://revistas.udistrital.edu.co/index.php/tia/article/view/8388/pdf

Adenowo, A., y Adenowo, B. (2020). *Software Engineering Methodologies: A Review of the Waterfall Model and Object Oriented Approach*. International Journal of Scientific & Engineering Research. 4. 427-434. Recuperado de URL: https://www.researchgate.net/publication/344194737_Software_Engineering_Methodologies_A_Review_of_the_Waterfall_Model_and_Object-_Oriented_Approach

Aizprua, S., Ortega Amable, y Von Chong, L. (2019). *Calidad del Software una Perspectiva Continua*. Revista científica CENTROS, 8(2), 120-134. Recuperado de URL: https://revistas.up.ac.pa/index.php/centros/article/view/741/632

Arias Gonzáles, J., y Covinos Gallardo, M. (2021). *Diseño y metodología de la investigación* (1 ed.). Arequipa, Peru: Enfoques Consulting. Recuperado de URL: https://gc.scalahed.com/recursos/files/r161r/w26022w/Arias_S2.pdf

Banda, R. (2022). *Automatización Pruebas de Calidad de Codigo dara das Aplicaciones Web don Herramientas de Código Abierto*. [Tesis de pregrado, Universidad San Ignacion de Loyola]. Repositorio Institucional USIL. https://repositorio.usil.edu.pe/server/api/core/bitstreams/71014249-7c5a-4971-b2e0-f9435209da2d/content

Benites C. (2022). *Aplicación de Aseguramiento de Calidad de Software y su influencia en proyectos de la empresa 3M,2022.* [Tesis de pregrado, Universidad Cientifica del Sur].

Repositorio Institucional CIENTÍFICA https://repositorio.cientifica.edu.pe/bitstream/handle/20.500.12805/2490

Board Infinity. (2020). *Una guía rápida para el modelo de prototipo en ingeniería de software*. Board Infinity. Recuperado de URL: https://www.boardinfinity.com/blog/a-quick-guide-to-prototype-model-in-software-engineering/

Callejas-Cuervo, M., Alarcón Adana, A. C., y Álvarez-Carreño, A. M. (2017). *Modelos de calidad del software, un estado del arte*. Revista entramado, 13(1), 236-250. Recuperado de URL: https://doi.org/10.18041/entramado.2017v13n1.25125

Carrizo, D., y Alfaro, A. (2018). *Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico*. Ingeniare. Revista chilena de ingeniería, 26(1), 12-20; 163-176. Recuperado de URL: https://www.scielo.cl/pdf/ingeniare/v26n1/0718-3305-ingeniare-26-01-00114.pdf

Conexión Esan. (2016). *La metodología Six Sigma*. Esan.edu.pe. Recuperado de URL: https://www.esan.edu.pe/conexion-esan/la-metodologia-six-sigma

Crispin, L. y Gregory, J. (2008). *Agile Testing: A Practical Guide for Testers and Agile Teams.* Addison-Wesley Professional. Agile Testing. Recuperada de URL: https://compress-pdf-free.obar.info/

Diario El Peruano. (2023). *El uso del cloud se mantiene en crecimiento en el Perú*. Recuperado de URL: https://elperuano.pe/noticia/104199--el-uso-del-cloud-se-mantiene-en-crecimiento-en-el-peru.

Esparza, R. (2023). *Análisis y corrección de vulnerabilidades de un producto software con SonarQube.* [Tesis de pregrado, Universitat Politècnica de València]. Repositorio Institucional UPV. https://riunet.upv.es/entities/publication/309e432c-0216-4bf2-acfd-513df962e626

Espinosa-Solís, J., Pizarro, N., Parra-Acosta, H., González-Carrillo, E., y Talavera-Sánchez, O. (2021). *Validación de un instrumento que mide el perfil actitudinal de los docentes y el desarrollo de competencias universitarias y transversales.* RIDE Revista Iberoamericana Para La Investigación Y El Desarrollo Educativo, 12(23). Recuperado de URL: *https://doi.org/10.23913/ride.v12i23.1003*.

Fracttal. (2023). Disponibilidad y Confiabilidad en el Mantenimiento: fractal.com. Recuperado de URL: https://www.fracttal.com/es/mantenipedia/que-es-la-confiabilidad-en-el-mantenimiento-y-como-calcularla

Fundacion Telefonica. (2023). *Informe Sociedad Digital en América Latina: Aumento del 1% en el índice de digitalización generaría un crecimiento de 0.3% en el PIB*. Telefonica. Recuperado de URL: https://telefonica.com.pe/informe-sociedad-digital-en-america-latina-aumento-del-1-en-el-indice-de-digitalizacion-generaria-un-crecimiento-de-0-3-en-el-pib/.

Gaete, J., Villarroel, R., Cornide-Reyes, H., Figueroa, I., y Muñoz, R. (2021). *Enfoque de aplicación ágil con Scrum, Lean y Kanban*. Revista chilena de ingeniería, 29(1), 141-157. Recuperado de URL: https://www.scielo.cl/pdf/ingeniare/v29n1/0718-3305-ingeniare-29-01-141.pdf

Garcia, S. (2022). *¿Qué es QA en desarrollo de software? La importancia del testing y QA*. Armadillo amarill, Recuperado de URL: https://www.armadilloamarillo.com/blog/que-es-qa-en-desarrollo-de-software-la-importancia-del-testing-y-qa/

GeeksforGeeks. (2023). *Modelo de cascad*a - Ingeniería de software, GeeksforGeeks. Recuperado de URL: Recuperado de: https://www.geeksforgeeks.org/waterfall-model/

Gplresearch. (2023). Coeficiente Alfa de Cronbach, *gplresearch.com*. Recuperado de URL: https://gplresearch.com/coeficiente-alfa-de-cronbach/

Hernández Oliva, A. M., Bron Fonseca, B., y Matamoros Benitez, L. C. (2018). *Modelos de desarrollo de software. ¿Cuál elegir ?* Revista Serie Científica de la Universidad de las Ciencias Informáticas, Recuperada de URL: https://dialnet.unirioja.es/descarga/articulo/8589950.pdf

Hernández Sampieri, C., Fernández Collado, C., y Baptista Lucio, P. (1991). *Metodología de la Investigación.* Editorial Mcgraw - Hill Interamericana de México. Recuperada de URL: https://www.uv.mx/personal/cbustamante/files/2011/06/metodologia-de-la-investigaci%C3%83%C2%B3n_sampieri.pdf

Inesdi. (2022). *¿Qué es un QA testing y cómo se hace?* Inesdi.com . Recuperado de URL: https://www.inesdi.com/blog/que-es-un-qa-testing/

ISO25000. (2024). *La familia de normas ISO/IEC 25000*. Iso 25000. Recuperado de URL: https://iso25000.com/index.php/normas-iso-25000

Jacobs, A. (2020). *Project Management Maturity: A Framework for Success in Sub-Saharan. Doctoral dissertation.* The International School of Management (ISM), Dortmund, Alemania. Recuperada de URL: https://ism.edu/images/ismdocs/dissertations/jacobs-dba-dissertation-2020.pdf

Larios, J. (2021). *Cómo montar un SonarQube en Cloud que nos sirva de Spike*. enmilocalfunciona.com. Recuperado de URL: https://www.enmilocalfunciona.io/como-montar-un-sonarqube-en-cloud-que-nos-sirva-de-spike-parte-1/?ref=enmilocalfunciona.io

Levin, K., y Vector, A. (27 de diciembre de 2020). *Metodología de desarrollo rápido de aplicaciones (RAD)*. alamy.com. Recuperado de URL: https://www.alamy.es/desarrollo-rapido-de-aplicaciones-metodologia-de-software-rad-esquema-detallado-de-vectores-de-proceso-marco-planificacion-de-requisitos-diseno-de-usuario-prototipo-prueba-image396522778.html

López Mora, D., Villamar Coloma, M., Bravo Pino, Á., y Lozano Rodríguez, E. (Abril de 2019). *El uso de las metodologías agiles y su importancia para el desarrollo de software. Revista Killkana Técnica. Vol. 3, No. 1, pp. 25-30*. Recuperada de URL: https://doi.org/https://doi.org/10.26871/killkana_tecnica.v3i1.473

Lubomirsky, A., Gardey, J., y Garrido, A. (Octubre de 2022). *Análisis de deuda técnica de UX en repositorios de GitHub*. Libro de actas - XXVIII Congreso Argentino de Ciencias de la Computación, 437-446. Recuperada de URL: http://sedici.unlp.edu.ar/handle/10915/149102

Mahbub, P., Rahman, M., Shuvo, O., y Gopal, A. (23 de agosto de 2023). *Explicación de errores: Aprovechar las estructuras de código para explicar errores de software mediante traducción automática neuronal.* Conferencia Internacional sobre Ingeniería de Software (ICSE) IEEE/ACM (v1), págs. 640-652. Recuperada de URL: https://arxiv.org/abs/2308.12267

Maida, E. G., y Pacienzia, J. (2015). *Metodologías de desarrollo de software*. [Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería. Universidad Católica Argentina]. Repositorion Institucional UCA. https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Marín Chaman, E. R., y Bautista Gutiérrez, J. J. (2021). Evaluación de la Calidad de Producto de Software Bajo Normas Iso/Iec 25000: Caso de Estudio Sistema de Planillas de la Municipalidad Provincial de Chiclayo. [Tesis de Título Profesional de Ingenieria. Universidad del Sr de Sipan], Repositorion Institucional USS https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/8086/Marin%20Chaman%2

0Edson%20%26%20Bautista%20Guti%c3%a9rrez%20Juan_.pdf?sequence=6&isAllowed=y

Navarro, F. (2020). Modelo de Desarrollo E Implementación De Software Bajo El Marco De Buenas Prácticas del Cmmi en el área de Sistemas de Información de la Universidad Autónoma de Bucaramanga. [Tesis de Maestria. Universidad Autónoma de Bucaramanga]. Repositorion Institucional UNAB https://repository.unab.edu.co/bitstream/20.500.12749/11966/1/2020_Tesis_Hermes_Fabian_Navarro.pdf

Nimble. (2023). *Métricas de tiempo de entrega y tiempo de ciclo.* nimblework. Recuperado de URL: https://www.nimblework.com/es/agile/metricas-de-tiempo-de-entrega-y-de-./

Noticias Ebiz. (2023). *Perú llegará a los US$ 1491 millones de inversión en cloud*. Noticias Ebiz. Recuperado de URL: https://noticias.ebiz.pe/: https://noticias.ebiz.pe/peru-llegara-a-los-us-1491-millones-de-inversion-en-cloud/

Oracle NetSuite. (2022). *tendencias del mercado, datos y análisis* Oracle NetSuite. Recuperado de URL: https://www.netsuite.com/portal/resource/articles/erp/erp-statistics.shtml.

Peraza, S., Uzcátegui, B., Guerrero, D., Medina, D., Ramírez, A. y Lezama, L. (2017). *Diseño, Confiabilidad, validez y normas de la escala de resiliencia para estudiantes universitarios.* Revista de Pedagogía, Vol. 38, N° 103, 2017, pp. 158 -176 Recuperado de URL: https://www.redalyc.org/pdf/659/65954978008.pdf

Pinpingos, A. (2018). *Implementación del área de control de calidad de software con una metodología enfocada en pruebas ágiles para la empresa TCI.* [Tesis de pregrado. Universidad Nacional Mayor de San Marcos], Repositorio institucional Cybertesis UNMSM.https://www.studocu.com/co/document/universidad-de-antioquia/ingenieria-de-sistemas/pinpingos-sa-sdadasdsadasdasd/29895619

Radstaak, J. (2019). Developing DevOps Maturity Model Para la University of Twente. [Tesis de maestría. Universidad de Twente], Recuperado de URL: https://essay.utwente.nl/77808/1/Thesis%20Jeroen%20Radstaak.pdf

Raeburn, A. (2020). La programación extrema (XP) produce resultados, pero ¿es la metodología adecuada para ti? asana.com. Recuperado de URL: https://asana.com/es/resources/extreme-programming-xp

Ribon Ramos, J., y Morales Garzón, L. (2021). *Diagnóstico del Sistema de Gestión de la Calidad en el Procedimiento de Análisis, Diseño y Desarrollo de Software Institucional bajo los lineamientos de la Norma ISO 9001:2015 EISO/IEC 25001:2014.* [*Proyecto de Grado.* Universidad de Córdoba]. Recuperado de URL: https://repositorio.unicordoba.edu.co/server/api/core/bitstreams/dfc005bb-b7e9-4037-a547-ff00f2abde12/content

Roche, J. (20 de diciembre de 2017). *Las 5 ceremonias Scrum: claves para la gestión de procesos*. *deloitte*.com Recuperado de URL: https://www.deloitte.com/es/es/services/consulting/blogs/todo-tecnologia/ceremonias-scrum.html

Rodríguez, C. (2017). *Impacto de los requerimientos en la calidad de software*. Revista de la Universidad Distrital Francisco José de Caldas, 5(2), 161-173. Recuperado de URL: https://revistas.udistrital.edu.co/index.php/tia/article/view/7607/pdf

Rodríguez, P. (2008). *Estudio de la Aplicación de Metodologías Ágiles para la Evolución de Productos Software*. [Tesis de Maestria, Universidad Politécnica de Madrid]. Repositorion Institucional UPM https://oa.upm.es/1939/1/Tesis_Master_Pilar_Rodriguez_Gonzalez.pdf

Rodríguez, S. (5 de setiembre 2020). *Agile testing: cuadrantes a considerar*. scrumio.com Recuperado de URL: https://www.scrumio.com/blog/agile-testing/.

Saavedra Martínez, J., Ibarguëngoitia González, M., y Fuentes Pineda, G. (2019). *Estimación del esfuerzo de proyectos de software con algoritmos de aprendizaje de máquinas.* ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica, vol. 8, núm. 1, pp. 1-22, 2019. Recuperado de URL: URL:https://www.redalyc.org/journal/5122/512259512010/html/

Saldaña, A. (2020). *Influencia del Uso de la Norma Iso/Iec 25000 en el nivel de Calidad del Sistema Snyfuel de la Empresa Grifo 3b*. [Tesis de pregrado. Universidad Privada del Norte]. Recuperado de URL: https://core.ac.uk/download/pdf/328899237.pdf

Sanchis, F. (2016). *Definición e implantación de un proceso QA para desarrollo de software*. [Trabajo Fin de Grado. Universitat Politècnica de València]. Recuperado de URL: https://riunet.upv.es/handle/10251/71381

Scrum Manager BoK. (2023). *Revisión por pares.* scrummanager.com. Recuperado de URL: https://www.scrummanager.com/bok/index.php/Revisi%C3%B3n_por_pares

Sejzer, R. (7 de junio de 2016). *DMAIC: Las 5 fases del proceso de implementación de Six. ctcalidad.blogspot.com*. Recuperado de URL: https://ctcalidad.blogspot.com/: http://ctcalidad.blogspot.com/2016/06/dmaic-las-5-fases-del-proceso-de.html

Sentrio. (13 de octubre 2021). *Metodologías Agile: los 4 valores y 12 principios del 'Manifiesto. Ágil'*. sentrio.io. Recuperado de URL: https://sentrio.io/blog/valores-principios-agile-manifiesto-agil/

Sharma, P. (9 de mayo de 2022). *Los 9 mejores modelos de desarrollo de software: fases y aplicaciones*. cynoteck.com. Recuperado de URL: https://cynoteck.com/es/blog-post/top-software-development-models-to-choose-from/

Soukath, M. (2021). *Scrum Narrative and PSM™ Exam Guide: All-in-one Guide for Professional Scrum Master™ (PSM™ 1) Certificate Assessment Preparation. goodreads.com.* Recuperado de URL: https://www.goodreads.com/book/show/54811919-scrum-narrative-and-psm-exam-guide

Thongtanunam, P., y Hassan, A. (26 de marzo de 2020). Dinámica de revisión y su impacto en la calidad del software. *IEEE Transactions on Software Engineering - Universidad de Melbourne*. Recuperado de URL: https://doi.org/DOI: 10.1109/TSE.2020.2964660

Torgeir Dingsøyr, S. N. (2012). *A decade of agile methodologies: Towards explaining agile software development.* The Journal of Systems and Software. Recuperado de URL:https://doi.org/https://doi.org/10.1016/j.jss.2012.02.033

Veliz, J. (2 de Octubre del 2023). *Ciberdelincuencia e Inteligencia Artificial ¿Puede el Perú defenderse ante estas amenazas digitales?* rpp.pe. Recuperado de URL: https://rpp.pe/tecnologia/mas-tecnologia/ciberdelincuencia-e-inteligencia-artificial-en-peru-2023-noticia-1508418?ref=rpp.

Wikia Calidadsoftware. (2023). *Métricas para la calidad del software.* Wikia Calidadsoftware calidadsoftwareunibague.fandom.com. Recuperado de URL: https://calidadsoftwareunibague.fandom.com/es/wiki/M%C3%A9tricas_para_la_calidad_del_software#:~:text=La%20eficacia%20de%20la%20eliminaci%C3%B3n,marco%20de%20trabajo%20del%20proceso.

Xu, L., y Brinkkemper, S. (2005). *Conceptos de software de producto: allanando el camino para una investigación urgentemente necesaria.* Institute of Information and Computing Sciences. Recuperado de URL: https://www.researchgate.net/publication/220921113_Concepts_of_Product_Software_Paving_the_Road_for_Urgently_Needed_Research

Zavala,C. (2019). *Aplicación ge Modelos ge Calidad y Madurez (Cmmi) Nivel 5 en Proyectos de desarrollo para su Implementacion E Integración en Valores Corporativos Softtek, logrando con ello la certificación por parte del Instituto de Ingenieria de Software (Sei)*. [Tesis De Maestria. Universidad Michoacana de San Nicolas de Hidalgo]. Recuperado de URL: http://bibliotecavirtual.dgb.umich.mx:8083/xmlui/handle/dgb_umich/6172

Zazworka, N., Shaw, M., Shull, F., y Seaman, C. (2011). *Investigating the impact of design debt on software quality.* Association for Computing Machinery, NY - EECC. Recuperado de URL: https://doi.org/https://doi.org/10.1145/1985362.1985366

IX. ANEXOS

Anexo A Matriz de Consistencia

Título: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

Problema	Objetivos	Hipótesis	Variables	Indicador	Índices	Metodología	Técnicas	Instrumento
Problema Principal	**Objetivo General**	**Hipótesis General**	**Variables Independiente**		(o) Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100			
PP: - ¿En qué medida la implementación de un modelo de desarrollo de software contribuye a mejorar la Calidad del Producto SaaS ERP de la Empresa Integrator	OP: Determinar el grado de influencia de un modelo de desarrollo de software en la Calidad del Producto SaaS ERP de la Empresa Integrator	HP: La implementación de un modelo de desarrollo de software, contribuye en mejorar la Calidad del Producto SaaS ERP de la Empresa Integrator	VI (X): Modelo de Desarrollo de software	Tasa de Errores	(o) Eliminación de Defectos (EED) = E / (E + D) (o) Indice_II_Ciclo = #Req al Mes/#Req II Ciclo * 100	Investigación de tipo "Aplicada", lo cual permite verificar y responder las Interrogantes planteadas en la presente investigación		
				Liberación del Producto	(o) Tiempo de Entrega = Tiempo Pre producion + Tiempo fabricación + Tiempo de Envió	Diseño No Experimental: No se va a manipular activamente las variables o realizar experimentos controlados, la investigación se basa en el uso de datos de muestra recopilados en un lapso determinado		- Cuestionarios - SonarQube - Ms Project - Herramientas de proceso de actividades - Acta de requerimientos del sistema SaaS Integrator ERP - Herramienta Visual Studio - SPSS (versión 26).
Problema Secundario	**Objetivo Secundario**	**Hipótesis Específica**	**Variables Dependientes**				Encuesta Análisis Documental	
P1: ¿En qué medida, la implementación de un modelo de desarrollo de software contribuye a reducir los errores en el Producto del SaaS ERP?	O1: - Determinar el grado de influencia que ejerce la implementación de un modelo de desarrollo de software en la reducción de errores en el Producto del SaaS ERP.	H1: - La implementación de un modelo de desarrollo de software contribuye a reducir los errores en el Producto del SaaS ERP.	VD(Y): Calidad de SaaS ERP	Calidad de Software	(o) Precisión = (Número de pases correctas / Total de pases realizadas) * 100 (o) Índice de Productividad = Numero de Errores / Líneas de Código generadas / 100 (o) Eficiencia en el uso del tiempo = (Resultados Logrados / Horas Trabajadas) * 100 (o) Índice de BUG = Número de líneas físicas/Número de líneas observadas (o) Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100			
P2: ¿En qué medida, la implementación de un modelo de desarrollo de software contribuye a reducirá el tiempo de entrega del Producto SaaS ERP?	O2: - Determinar el grado de influencia que ejerce la implementación de un modelo de desarrollo de software en la reducción del tiempo de entrega del Producto SaaS ERP	H2 - La implementación de un modelo de desarrollo de software contribuye a reducir el tiempo de entrega del Producto SaaS ERP.						

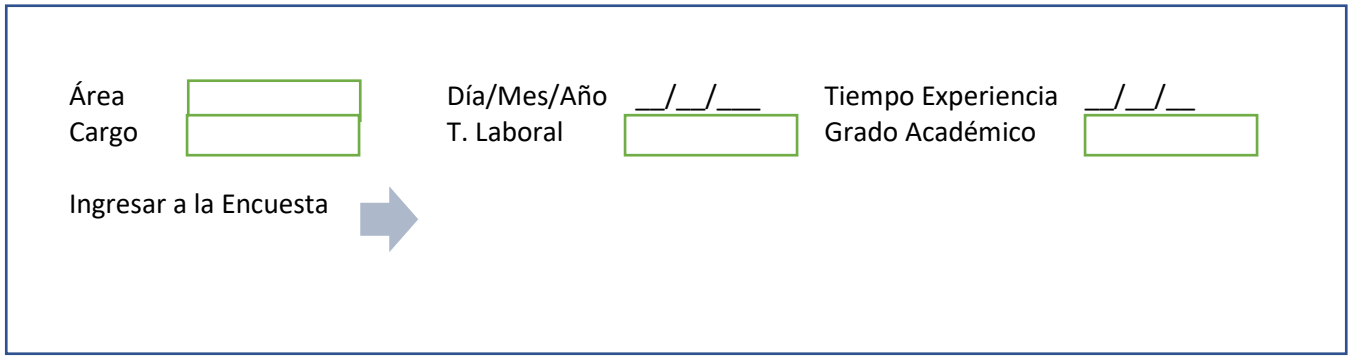
Anexo B Matriz de Operacionalización de Variables

Título: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios. Lima, Perú – 2023

Variable	Definición conceptual	Definición Operacional	Indicador	Índice
VI: El modelo de desarrollo de Software	Modelos de desarrollo de software son una colección de técnicas y sistemas organizacionales para crear software de computadora. El objetivo de los diversos enfoques es estructurar equipos de trabajo para que puedan construir las funcionalidades del programa de la manera más eficiente posible. Modelos de desarrollo de software proporcionar un marco para controlar el desarrollo de los sistemas de información. Desde la planificación hasta el mantenimiento, un Ciclo de vida del desarrollo de programas (SDLC) describe todos los procesos en un proyecto de desarrollo de software. Estos marcos incluyen el desarrollo de programas, así como las herramientas necesarias para ayudar en el proceso de desarrollo (1)	La variable se evaluará mediante la información de los Project, herramientas de progreso de actividades y tiempo de desarrollo durante el proceso de aplicación del modelo de desarrollo de software	Tasa de Errores	(o) Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100 (o) Eliminación de Defectos (EED) = E / (E + D) (o) Indice II Ciclo = #Req_al_Mes/#Req_II_Ciclo * 100
			Liberación del Producto	(o) Tiempo de Entrega = Tiempo Pre producion + Tiempo fabricación + Tiempo de Envió
VD: Calidad del producto software (SaaS ERP)	Es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo plenamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente", con base en los requisitos funcionales y no funcionales identificados en la etapa de análisis del sistema, insumo principal para implementar dichos requisitos con los atributos mínimos de calidad, fomentando la aplicación de procesos estandarizados y criterios necesarios en cada una de sus etapas, así se fomenta que el avance en el ciclo de vida del software minimice el riesgo de fracaso del proyecto (2)	La variable se evaluará mediante el análisis de una ficha de resultados y del llenado de un formulario para evaluar la calidad del software	Calidad de Software	(o) Precisión = (Número de pases correctas / Total de pases realizadas) * 100 (o) Índice de Productividad = Numero de Errores / Líneas de Código generadas / 100 (o) Eficiencia en el uso del tiempo = (Resultados Logrados / Horas Trabajadas) * 100 (o) Índice de BUG = Número de líneas físicas/Número de líneas observadas (o) Disponibilidad = (Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas x 100

Anexo C Instrumento de recolección de datos

Formulario con los datos generales (Google forms)



Instrucciones:

Las siguientes preguntas tienen que ver con varios aspectos del trabajo de control de calidad del Software. Señale con una **X** dentro del recuadro correspondiente a la pregunta, de acuerdo con el cuadro de codificación. Por favor, conteste con su opinión sincera, es su opinión la que cuenta y por favor asegúrese de que no deja ninguna pregunta en blanco.

Codificación en los Criterio de Evaluacion/Escala de Valoracion

Totalmente desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
1	2	3	4	5

Título de la Investigación: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios

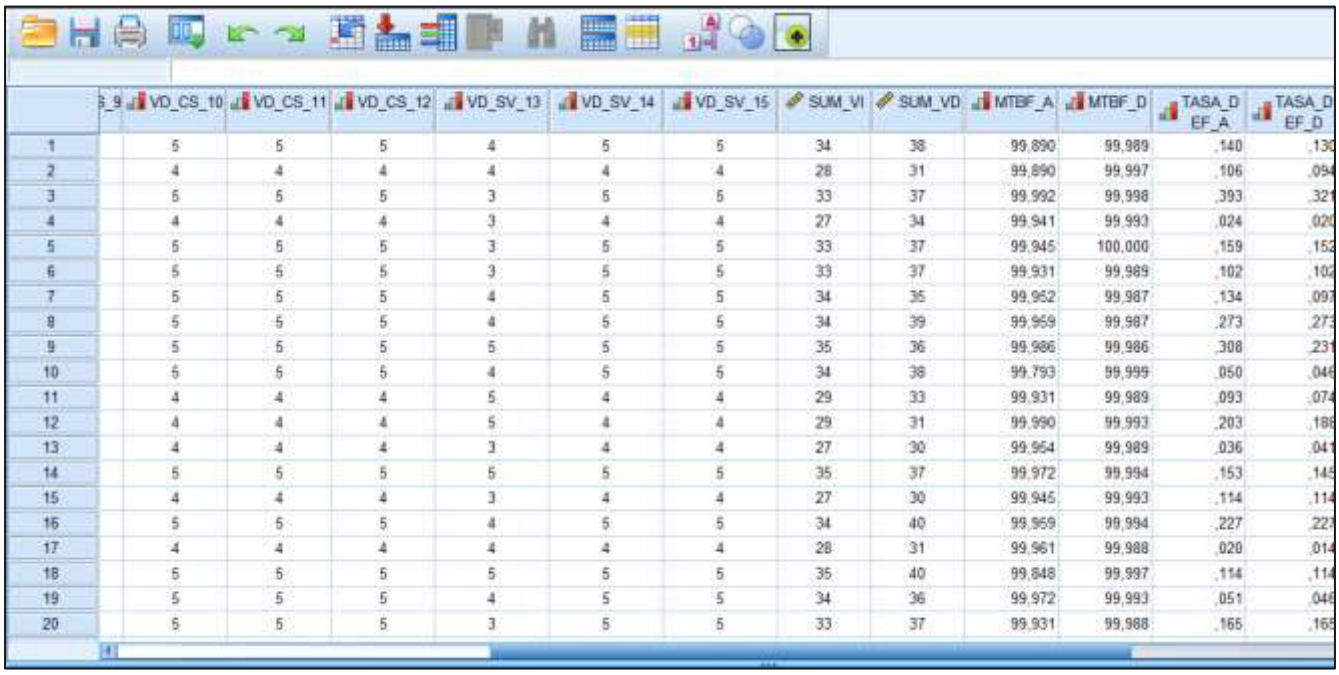
PRE- TEST

Variable	Dimensiones	Indicador	# Item	Pregunta	(1)	(2)	(3)	(4)	(5)
Variable Dependiente	Calidad del producto SaaS ERP	Calidad de Software	1	¿Considera que es necesario identificar los errores oportunamente para mejorar la calidad del codigo del programa ?					
			2	¿Considera que es necesario fortalezer los procesos de versionamiento para mejorar la disponibilidad del programa u artefacto ?					
			3	¿Cree que se deba priorizar el desarrollo de un "codigo limpio" en la elaboracion de un programa o artefacto ¿					
			4	¿Cree que este deban mejorar el proceso de desarrollo y QA para mejorar la disponibilidad del servicio SaaS ERP					
			5	¿ Cree que es importante cumplir con los tiempos de entrega usando las mejores prácticas para la liberacion de artefactos u programas?					
			6	¿Piensas que se debe incrementar las revisiones para garantizasr el tiempo de liberacion de los artefactos u programas ?					
			7	Es necesario realizar un análisis previo para estiamar mejor los esfuerzos a la hora de dar un tiempo de desarrollo de los artefactos u programas?.					
			8	¿Cree usted que una mejora en el modelo de desarrollo implementando controles y buenas practicas , podria mejorar la calidad del servico del aplicativo?					

POST - TEST

Variable	Dimensiones	Indicador	# Item	Pregunta	(1)	(2)	(3)	(4)	(5)
Variable Dependiente	Calidad del producto SaaS ERP	Calidad de Software	1	¿Considera que es ha mejorado el control de versiones antes de librar un artefacto u programa ?					
			2	¿Ve de utildiad el uso de una bitacora de cambios de los programas del sistema (backlog) ?					
			3	¿Considera ha mejorado el tiempo de estimacion un cambio del codigo del programa ?					
			4	¿Piensas que se ha mejorada la calidad del codigo con las las revisiones para la deteccion de codigo obsoleto/duplicidad /BUG con el objetivo de incementar la seguridad de programas ?					
			5	Concidera que se esta llevando un mejor control sobre la prioridad del cambio de los programas?					
			6	Cómo evaluaría la eficacia del nuevo modelo para incremntar la calidad del softwarepara cumplir lo ofrecido en términos de cumplimiento de sus expectativas					
			7	En qué medida considera que el nuevo modelo demuestra un cumplimiento genuino sobre las espectativas de mejoras					
			8	¿Esta deacuerdo que se a incrementado la disponibilidad y eficiencia del sistema con la aplicación del nuevo modelo?					

Anexo D Base de Datos SPSS



	5_9	VD_CS_10	VD_CS_11	VD_CS_12	VD_SV_13	VD_SV_14	VD_SV_15	SUM_VI	SUM_VD	MTBF_A	MTBF_D	TASA_DEF_A	TASA_DEF_D
1		5	5	5	4	5	5	34	38	99,890	99,989	,140	,130
2		4	4	4	4	4	4	28	31	99,890	99,997	,106	,094
3		5	5	5	3	5	5	33	37	99,992	99,998	,393	,321
4		4	4	4	3	4	4	27	34	99,941	99,993	,024	,020
5		5	5	5	3	5	5	33	37	99,945	100,000	,159	,152
6		5	5	5	3	5	5	33	37	99,931	99,989	,102	,102
7		5	5	5	4	5	5	34	35	99,952	99,987	,134	,097
8		5	5	5	4	5	5	34	39	99,959	99,987	,273	,273
9		5	5	5	5	5	5	35	36	99,986	99,986	,308	,231
10		5	5	5	4	5	5	34	38	99,793	99,999	,050	,046
11		4	4	4	5	4	4	29	33	99,931	99,989	,093	,074
12		4	4	4	5	4	4	29	31	99,990	99,993	,203	,186
13		4	4	4	3	4	4	27	30	99,954	99,989	,036	,041
14		5	5	5	5	5	5	35	37	99,972	99,994	,153	,145
15		4	4	4	3	4	4	27	30	99,945	99,993	,114	,114
16		5	5	5	4	5	5	34	40	99,959	99,994	,227	,227
17		4	4	4	4	4	4	28	31	99,961	99,988	,020	,014
18		5	5	5	5	5	5	35	40	99,848	99,997	,114	,114
19		5	5	5	4	5	5	34	36	99,972	99,993	,051	,046
20		5	5	5	3	5	5	33	37	99,931	99,988	,165	,165

Anexo E Hojas de trabajo para las Dimensiones

Título: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

Dimension	**Tasa de Errores**
Indicador	**Fiabilidad**

VI : Modelo de Desarrollo de Software

Fecha : 01/01/2022 - 30/10/2022

Tasa de Errores (Fallas)

Dimension	Tasa de Errores
Indicador	Fiabilidad

Incidencias

Numero de Fallas (F)	221	
Numero de Muestras	264	
Total	83,71212	%

Numero de Muestras	264	
Universo	850	
Tiempo Total Disponible (MI)	434880	100 %

MTBF — El Tiempo Medio Entre Fallos — (Numero de tiempo Disponible - Tiempo No Operativo /Numero de Paradas)

A mas Alto , Mas Fiable

Ref: https://blog.infraspeak.com/es/que-es-mtbf/

									Implementacion del Nuevo Modelo			
Programas	# de Cambios	N/M	Presento Errores ?	# Ciclos	# Minutos No Operativo	Horas	MTBF (MI)	%	# Minutos No OPerativo	# de Cambio	NuevoMTBF	%
Programa1	2	M	S	2	960	16	434400	99,88962	47	1	434833	99,98919
Programa2	2	M	S	2	960	16	434400	99,88962	11	1	434869	99,99747
Programa3	5	M	S	2	180	3	434844	99,99172	7	1	434873	99,99839
Programa4	4	M	S	2	1020	17	434625	99,94136	32	1	434848	99,99264
Programa5	4	M	S	2	960	16	434640	99,94481	2	1	434878	99,99954
Programa6	2	M	S	2	600	10	434580	99,93102	48	1	434832	99,98896
Programa7	4	M	S	2	840	14	434670	99,95171	58	1	434822	99,98666
Programa8	1	N	N	1	180	3	434700	99,95861	58	1	434822	99,98666
Programa9	3	M	S	2	180	3	434820	99,9862	60	1	434820	99,9862
Programa10	1	N	N	1	900	15	433980	99,79305	6	1	434874	99,99862

Título de la investigación: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

VI : Modelo de Desarrollo de Software

Fecha :		01/01/2022 - 30/10/2022	
Dimension	**Tasa de Errores**		
Indicador	**Disponibilidad**		
Numero de Muestras		264	
Universo		850	
Disponibilidad		**(Horas totales de funcionamiento planeadas - Horas en paradas) / Horas totales de funcionamiento planeadas**	
Tiempo Total Disponible (MI)		**434880**	100 %

Ref : https://www.fracttal.com/es/mantenipedia/que-es-la-confiabilidad-en-el-mantenimiento-y-como-calcularla

Programas	# Minutos No Operativo	Disponibilidad	%
Programa1	960	434879,9978	99,99999949
Programa2	960	434879,9978	99,99999949
Programa3	180	434879,9996	99,9999999
Programa4	1020	434879,9977	99,99999946
Programa5	960	434879,9978	99,99999949
Programa6	600	434879,9986	99,99999968
Programa7	840	434879,9981	99,99999956
Programa8	180	434879,9996	99,9999999
Programa9	180	434879,9996	99,9999999

E	Número de errores encontrados antes de la entrega del softwa (Trabajo de QA)
D	Número de defectos encontrados después de la entrega (Bug en Produccion)
Formula	EED = E / (E + D)

Dimension	Tasa de Errores
Indicador	Eliminacion de Defectos

Ref: https://calidadsoftwareunibague.fandom.com/es/wiki/M%C3%A9tricas_para_la_calidad_del_software#:~:text=La%20eficacia%20de%20la%20Eliminaci%C3%B3n,marco%20de%20trabajo%20del%20proceso.

								Despues de la implementacion del Nuevo Modelo						
Programas	# de Cambios	N/M	Presento Errores ?	# Ciclos	E	D	EED	# de Cambios	N/M	Presento Errores ?	# Ciclos	E	D	EED
Programa1	2	M	S	2	12	2	0,857143	2	M	N	1	12	1	0,923077
Programa2	2	M	S	2	16	3	0,842105	2	M	N	1	16	1	0,941176
Programa3	5	M	S	2	17	5	0,772727	5	M	N	1	17	1	0,944444
Programa4	4	M	S	2	5	1	0,833333	4	M	N	1	4	1	0,8
Programa5	4	M	S	2	19	4	0,826087	4	M	N	1	19	3	0,863636
Programa6	2	M	S	2	12	1	0,923077	2	M	N	1	12	1	0,923077
Programa7	4	M	S	2	16	9	0,64	4	M	N	1	16	2	0,888889
Programa8	1	N	N	1	12	0	1	1	N	N	1	12	0	1
Programa9	3	M	S	2	3	5	0,375	3	M	N	1	2	4	0,333333
Programa10	1	N	N	1	13	0	1	1	N	N	1	12	0	1
Programa11	1	N	N	1	5	0	1	1	N	N	1	4	0	1
Programa12	4	M	S	2	10	3	0,769231	4	M	N	1	10	2	0,833333
Programa13	3	M	S	2	6	1	0,857143	3	M	N	1	7	1	0,875

Título de la investigación: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

Dimension	**Tiempo de Entrega**
Indicador	**Tiempo de Liberacion**

TES	Tiempo Estimado (Horas)
TP	Tiempo de Analisis (Pre-Produccion) - Horas
TF	Tiempo de Fabricacion - Horas
TES	Tiempo de Envio - Horas
Formula	TL = TP + TF+ TE

Nota : Otra MUESTRA diferente a la Dimension de Tasa de Errores

Dimension	**Tiempo de Entrega**
Indicador	**Tiempo de Liberacion**

Ref: https://www.nimblework.com/es/agile/metricas-de-tiempo-de-entrega-y-de-ciclo/

Programas	TES	TP	TF	TE	TL
Programa1	9	2	13	1	16
Programa2	9	2	13	1	16
Programa3	1	1	1	1	3
Programa4	3	1	15	1	17
Programa5	8	1	14	1	16
Programa6	7	2	7	1	10
Programa7	9	2	11	1	14
Programa8	3	1	1	1	3

Nuevo Modelo usando Modelo Agiles

Programas	TES	TP	TF	TE	TL
Programa1	9	2	6	1	9
Programa2	9	2	6	1	9
Programa3	1	0	0	1	1
Programa4	3	1	1	1	3
Programa5	8	1	6	1	8
Programa6	7	2	4	1	7
Programa7	9	2	6	1	9
Programa8	3	1	1	1	3

Título de la investigación: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

VD : Calidad del producto Software

Fecha : 01/01/2022 - 30/10/2022

Dimension: Calidad de Software

Indicador: Precision — (Numero de Pases Correctos / Total Pases) * 100

Numero de Muestras: 264

Universo: 850

Mes	# de Pases correctos	# de Pases Realizados	Total (%)
Enero	10	27	37,04
Febrero	15	34	44,12
Marzo	23	56	41,07
Abril	12	23	52,17
Mayo	8	15	53,33
Junio	17	34	50,00
Julio	7	13	53,85
Agost	9	19	47,37
Septiembre	10	18	55,56
Octubre	13	25	52,00

Indice Prodructividad	Lineas de Codigo generados / Horas Trabajadas
Tasa de Defectos	Numero de Errores / Lineas de Codigo generadas

Dimension: Calidad de Software

Indicador: Productividad

(*) Datos de TAG Eliminacio de Defectos

Ref: https://www.redalyc.org/journal/5122/512259512010/html/

Programas	# de Lineas de Codigo Afectado	Horas Trabajadas	LP	E (*)	D	Tasa de Defectos	Impacto de la aplicación del nuevo modelo: # de Lineas de Codigo	Horas Trabajadas	LP	E (*)	D	Tasa de Defectos
Programa1	100	16	6,3	12	2	0,14000	100	9	11,1	12	1	0,13000
Programa2	180	16	11,3	16	3	0,10556	180	9	20,0	16	1	0,09444
Programa3	56	3	18,7	17	5	0,39286	56	1	56,0	17	1	0,32143
Programa4	252	17	14,8	5	1	0,02381	252	3	84,0	4	1	0,01984
Programa5	145	16	9,1	19	4	0,15862	145	8	18,1	19	3	0,15172
Programa6	127	10	12,7	12	1	0,10236	127	7	18,1	12	1	0,10236
Programa7	186	14	13,3	16	9	0,13441	186	9	20,7	16	2	0,09677
Programa8	44	3	14,7	12	0	0,27273	44	3	14,7	12	0	0,27273
Programa9	26	3	8,7	3	5	0,30769	26	3	8,7	2	4	0,23077
Programa10	259	15	17,3	13	0	0,05019	259	4	64,8	12	0	0,04633

Título de la investigación: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

VD : Calidad del producto Software

Fecha : 01/01/2022 - 30/10/2022

Dimension	**Calidad de Software**
Indicador	**Eficiencia en el uso del tiempo**

(Hora Trabajas al Mes / Horas Trabajadas Real)

Numero de Muestras 264

Universo 850

Horas Trabajadas (Mes) **188**

Mes	Horas Dedicadas al Desarrollo	%	Aplicación Nuevo Modelo: Horas Dedicadas al Desarrollo	Aplicación Nuevo Modelo: %
Enero	200	106,38	190	101,06
Febrero	205	109,04	200	106,38
Marzo	220	117,02	190	101,06
Abril	300	159,57	200	106,38
Mayo	450	239,36	202	107,45
Junio	188	100,00	190	101,06
Julio	222	118,09	210	111,70
Agost	195	103,72	188	100,00
Septiembre	188	100,00	188	100,00
Octubre	210	111,70	188	100,00
Total	2378			

VD : Calidad del producto Software

Fecha : 01/01/2022 - 30/10/2022

Dimension	**Calidad de Software**
Indicador	**Hitos Alcanzados**

(Pases Realizados *100 / Pases Correctos)

Numero de Muestras 264

Universo 850

Horas Trabajadas (Mes) **188**

Mes	# de Pases correctos	# de Pases Realizados	Diferencia	# Pases a un Segundo Ciclo	Diferencia (%)
Enero	10	27	17,00	62,96	-37,037
Febrero	15	34	19,00	55,88	-44,118
Marzo	23	56	33,00	58,93	-41,071
Abril	12	23	11,00	47,83	-52,174
Mayo	8	15	7,00	46,67	-53,333
Junio	17	34	17,00	50,00	-50,000
Julio	7	13	6,00	46,15	-53,846
Agost	9	19	10,00	52,63	-47,368
Septiembre	10	18	8,00	44,44	-55,556
Octubre	13	25	12,00	48,00	-52,000

Título de la investigación: Implementación de un modelo de desarrollo de software para mejorar la calidad del producto SaaS ERP de una empresa de servicios.

Dimension	**Calidad de Software**
Indicador	**Codigo BUG**

		ASIS del la calidad del Codigo						Efecto Aplicando el nuevo modelo de Desarrollo de Software					
		ISSUE				% de Codigo Duplicado	% de Severidad	ISSUE				% de Codigo Duplicado	% de Severidad
Programas	# de Lineas del Programa	Vulnerability	Bug	Security Hotspot	Code Smell			Vulnerability	Bug	Security Hotspot	Code Smell		
Programa1	693	226	69	21	52	31	57,6	0	0	0	0	31	4,5
Programa2	942	251	70	37	63	22	47,0	0	0	0	0	22	2,3
Programa3	1971	176	64	38	46	18	17,4	0	0	0	0	18	0,9
Programa4	983	247	67	59	52	31	46,4	0	0	0	0	31	3,2
Programa5	1628	259	72	71	76	22	30,7	0	0	0	0	22	1,4
Programa6	1994	89	60	46	59	20	13,7	0	0	0	0	20	1,0
Programa7	943	137	75	30	62	19	34,3	0	0	0	0	19	2,0
Programa8	1548	147	74	39	75	19	22,9	0	0	0	0	19	1,2
Programa9	1393	161	64	27	64	21	24,2	0	0	0	0	21	1,5
Programa10	1582	151	71	61	75	10	23,3	0	0	0	0	10	0,6
Programa11	916	155	67	32	58	8	34,9	0	0	0	0	8	0,9
Programa12	1539	176	71	72	51	16	25,1	0	0	0	0	16	1,0
Programa13	827	264	67	24	65	17	52,8	0	0	0	0	17	2,1
Programa14	714	107	63	70	45	20	42,7	0	0	0	0	20	2,8
Programa15	1233	113	64	22	52	21	22,1	0	0	0	0	21	1,7

Anexo F Hojas de Firmadas de Juicio Experto

UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
FACULTAD DE INGENIERIAINDUSTRIAL Y DE SISTEMAS
ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

FICHA DE VALIDACIÓN DEL INSTRUMENTO DE INVESTIGACIÓN
JUICIO DE EXPERTOS

I. **DATOS GENERALES**

1.1. Apellidos y Nombres: MORALES ROMERO, GUILLERMO PASTOR
1.2. Grado académico: Dr. Ciencias de la Educación – Maestro en Ingeniería de Sistemas
1.3. Cargo e Institución donde labora: Universidad Nacional de Educación Enrique Guzmán y Valle- Director de la Escuela de Matemática e Informática
1.4. Nombre del instrumento motivo de evaluación: CUESTIONARIO
1.5. Autor(A) de Instrumento: CUADRA RIVERA CARLOS JOAQUIN
1.6. Criterios de aplicabilidad:
a. De 01 a 09: (No válido, reformular)
b. De 10 a 12: (No válido, modificar)
c. De 13 a 15: (Válido, mejorar)
d. De 16 a 17: (Válido, precisar)
e. De 19 a 20: (Válido aplicar)

II. **ASPECTOS DE VALIDACIÓN**

INDICADORES DE EVALUCION DEL INSTRUMENTO	CRITERIOS CUALITATIVOS CUANTITATIVOS	Deficiente (01-09)	Regular (10-12)	Bueno (13-15)	Muy Bueno (16-18)	Excelente (19-20)
		1	2	3	4	5
1. CLARIDAD	Esta formulado con lenguaje comprensible.				X	
2. OBJETIVIDAD	Esta adecuado a las leyes y principios científicos.				X	
3. ACTUALIDAD	Esta adecuado a los objetivos y las necesidades reales de la investigación.				X	
4. ORGANIZACIÓN	Existe una organización lógica.				X	
5. SUFICIENCIA	Toma en cuenta los aspectos metodológicos esenciales				X	
6. INTENCIONALIDAD	Esta adecuado para valorar las variables de la Hipótesis.				X	
7. CONSISTENCIA	Se respalda en fundamentos técnicos y/o científicos.				X	
8. COHERENCIA	Existe coherencia entre los problemas objetivos, hipótesis, variables e indicadores.				X	
9. METODOLOGÍA	La estrategia responde una metodología y diseño aplicados para lograr probar las hipótesis.				X	
10. PERTINENCIA	El instrumento muestra la relación entre los componentes de la investigación y su adecuación al Método Científico.				X	

VALORACIÓN CUANTITATIVA (TOTAL X 0.4): **16**
VALORACIÓN CUALITATIVA: **VÁLIDO**
OPINIÓN DE APLICABILIDAD: **APLICAR**

Lima, 21 de noviembre del 2023

DNI No 10124478 Telf.: 939319870

FIRMA DEL EXPERTO INFORMANTE

UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS
ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

FICHA DE VALIDACIÓN DEL INSTRUMENTO DE INVESTIGACIÓN
JUICIO DE EXPERTOS

I. **DATOS GENERALES**

I.1 Apellidos y Nombres: LEZAMA GONZALES PEDRO MARTIN
I.2 Grado académico: DOCTOR EN INGENIERIA DE SISTEMAS
I.3 Cargo e Institución donde labora: UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
I.4 Nombre del instrumento motivo de evaluación: CUESTIONARIO
I.5 Autor(A) de Instrumento: CUADRA RIVERA CARLOS JOAQUIN
I.6 Criterios de aplicabilidad:

a. De 01 a 09: (No valido, reformular)
b. De 10 a 12: (No valido, modificar)
c. De 13 a 15: (Valido, mejorar)
d. De 16 a 17: (Valido, precisar)
e. De 19 a 20: (Valido aplicar)

II. **ASPECTOS DE VALIDACIÓN**

INDICADORES DE EVALUACION DEL INSTRUMENTO	CRITERIOS CUALITATIVOS CUANTITATIVOS	Deficiente (01-09)	Regular (10-12)	Bueno (13-15)	Muy Bueno (16-18)	Excelente (19-20)
		1	2	3	4	5
1. CLARIDAD	Esta formulado con lenguaje comprensible.					X
2. OBJETIVIDAD	Esta adecuado a las leyes y principios cientificos.					X
3. ACTUALIDAD	Esta adecuado a los objetivos y las necesidades reales de la investigación.					X
4. ORGANIZACIÓN	Existe una organización logica.					X
5. SUFICIENCIA	Toma en cuenta los aspectos metodologicos esenciales					X
6. INTENCIONALIDAD	Esta adecuado para valorar las variables de la Hipotesis.					X
7. CONSISTENCIA	Se respalda en fundamentos teoricos y/o cientificos.					X
8. COHERENCIA	Existe coherencia entre los problemas objetivos, hipotesis variables e indicadores.					X
9. METODOLOGÍA	La estrategia responde una metodología y diseño aplicados para lograr probar las hipotesis.					X
10. PERTINENCIA	El instrumento muestra la relación entre los componentes de la investigación y su adecuación al Método Científico.					X

VALORACIÓN CUANTITATIVA (TOTAL X 0.4): **20**.

VALORACIÓN CUALITATIVA: **VÁLIDO**

OPINIÓN DE APLICABILIDAD: **APLICAR**

Lima, 21 de noviembre del 2023

DNI No 09656793 Telf.: 945473135

FIRMA DEL EXPERTO INFORMANTE

UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS
ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

FICHA DE VALIDACIÓN DEL INSTRUMENTO DE INVESTIGACIÓN
JUICIO DE EXPERTOS

I. DATOS GENERALES

I.1. Apellidos y Nombres: IVAN CARLO PETRUK AZABACHE
I.2. Grado academico: DOCTOR EN INGENIERIA DE SISTEMAS
I.3. Cargo e institución donde labora: UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
I.4. Nombre del instrumento motivo de evaluación: CUESTIONARIO
I.5. Autor(A) de Instrumento: CUADRA RIVERA CARLOS JOAQUIN
I.6. Criterios de aplicabilidad:

a. De 01 a 09: (No valido, reformular)
b. De 10 a 12: (No valido, modificar)
c. De 13 a 15: (Válido, mejorar)
d. De 16 a 17: (Válido, precisar)
e. De 19 a 20: (Válido aplicar)

II. ASPECTOS DE VALIDACIÓN

INDICADORES DE EVALUACION DEL INSTRUMENTO	CRITERIOS CUALITATIVOS CUANTITATIVOS	Deficiente (01-09)	Regular (10-12)	Bueno (13-15)	Muy Bueno (16-18)	Excelente (19-20)
		1	2	3	4	5
1. CLARIDAD	Esta formulado con lenguaje comprensible.					X
2. OBJETIVIDAD	Esta adecuado a las leyes y principios cientificos.					X
3. ACTUALIDAD	Esta adecuado a los objetivos y las resultados reales de la investigación.					X
4. ORGANIZACIÓN	Existe una organización lógica.					X
5. SUFICIENCIA	Toma en cuenta los aspectos metodológicos esenciales					X
6. INTENCIONALIDAD	Esta adecuado para valorar las variables de la Hipótesis					X
7. CONSISTENCIA	Se respalda en fundamentos técnicos y/o cientificos.					X
8. COHERENCIA	Existe coherencia entre los problemas objetivos, hipótesis variables e indicadores.					X
9. METODOLOGIA	La estrategia responde una metodología y diseño aplicados para lograr probar las hipótesis.					X
10. PERTINENCIA	El instrumento muestra la relación entre los componentes de la investigación y su adecuación al Método Científico.					X

VALORACIÓN CUANTITATIVA(TOTAL X 0.4): **20**
VALORACIÓN CUALITATIVA: **VÁLIDO**
OPINIÓN DE APLICABILIDAD: **APLICAR**

Lima, 21 de noviembre del 2023

DNI No: 10140461 Telf.: 942198541

FIRMA DEL EXPERTO INFORMANTE