UNIVERSIDAD CÉSAR VALLEJO

ESCUELA DE POSGRADO

PROGRAMA ACADÉMICO DE MAESTRÍA EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Aplicación ISO 25000 para el proceso de desarrollo de software en el Área de TI en una financiera, Lima 2023

TESIS PARA OBTENER EL GRADO ACADÉMICO DE:

Maestra en Ingeniería de Sistemas con Mención en Tecnologías de la Información

AUTORA:

Rios Jorge, Gina Rosario (orcid.org/0000-0001-7633-3571)

ASESOR:

Dr. Acuña Benites, Marlon Frank (orcid.org/0000-0001-5207-9353)

CO-ASESOR:

Dr. Flores Zafra, David (orcid.org/0000-0001-5846-325X)

LÍNEA DE INVESTIGACIÓN:

Auditoria de Sistemas y Seguridad de la Información

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo económico, empleo y emprendimiento

LIMA – PERÚ

2023

Dedicatoria

A mi madre por su amor, apoyo y comprensión, por sus consejos para lograr cada objetivo propuesto. Todos mis logros siempre serán suyos. A mi novio, por su apoyo y motivación para alcanzar cada una de mis metas profesionales. A cada persona cercana que estuvo apoyándome y dándome sus consejos para seguir adelante.

Agradecimiento

Gracias a los docentes de la universidad que nos acompañaron durante la maestría, por cada consejo y enseñanza. A mi asesor, que con sus sugerencias y correcciones logró que entregara una tesis de calidad.

Índice de contenidos

Índice de tablas

Pag.

Índice de figuras

Pag.

Resumen

La ISO 25000 tiene el objetivo principal es guiar el desarrollo de los productos de software mediante la especificación de requisitos y evaluación de características de calidad. El objetivo de la investigación es determinar de qué manera la aplicación del ISO 25000 ayudara a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023; se utilizó el tipo de Investigación aplicada, con enfoque cuantitativo y un diseño pre-experimental con 2 grupos, se tomó como población una cantidad de 50 registros. Como instrumento de recolección de datos se tuvo las fichas de registros, se realizó un análisis descriptivo por cada indicador. Se uso Kolgomorov-Smirnov para la prueba de normalidad y Wilcoxon para el análisis inferencial.

Con la aplicación del ISO 25000 se obtuvo como resultado una mejora significativa en el proceso de desarrollo de software, teniendo como puntos clave la mejora de los indicadores, productividad del desarrollador que incrementó un 50%, defectos en el software que disminuyo en un 10% y el rendimiento del software que aumentó un 46%.

Palabras clave: ISO 25000, sistemas, desarrollo, SCRUM

Abstract

ISO 25000 has the main objective of guiding the development of software products through the selection of requirements and evaluation of quality characteristics. The objective of the research is to determine how the application of ISO 25000 will help to improve the software development process in the IT area in a financial institution, Lima 2023; The type of Applied Research was used, with a quantitative approach and a pre-experimental design with 2 groups, one of 50 records was taken as a population. As a data collection instrument, the records were used, a descriptive analysis was carried out for each indicator. Kolgomorov-Smirnov was used for the normality test and Wilcoxon for the inferential analysis.

With the application of ISO 25000, a significant improvement in the software development process was obtained, having as key points the improvement of the indicators, Percentage of speed of software development that increased by 50%, Percentage of detection of defects that decreased by 10% and the Confidence Percentage increased by 46%.

Keywords: ISO 25000, systems, development, SCRUM

I. INTRODUCCIÓN

El software es una herramienta que permite optimizar el desarrollo de la empresa, con la meta de satisfacer las necesidades del usuario, por lo tanto, es necesario garantizar la calidad del producto, para esto es importante conocer al usuario y sus necesidades para contar con un proceso de calidad, y de esta manera se incrementará la integridad, se reducirá el mantenimiento, aumentará la satisfacción del usuario con respecto al software y se detectaran errores de forma temprana. Koh (2017) destaca que, el poder tener una garantía de la calidad del software es necesario aplicar procesos que permitan lograr un producto de calidad, tomando en cuenta el funcionamiento y necesidad que tiene el producto a implementar. La calidad de software no solo trata de cumplir con las necesidades del cliente, sino que también se logrará una mejora durante la etapa de desarrollo de un producto, lo cual permitirá que la organización crezca y alcanzar una posición el mercado.

Según Carrizo y Alfaro (2018) indicaron que, hoy en día a nivel internacional desarrollar y entregar un servicio de calidad que pueda satisfacer a las personas interesadas de una entidad es el objetivo de todas las empresas que desarrollan software para lograr tener una buena posición en el mercado. Pero ¿Como satisfacer las necesidades del cliente al momento de desarrollar un producto software?, pues esto se logra a través de la calidad. Es decir que la calidad lograra que el usuario se muestre a gusto y conforme con el servicio brindado, dicho de otro modo, se debe incluir y dar más tiempo al proceso de pruebas dentro de la etapa de desarrollo de software para mantener el producto funcionando de manera correcta y sin errores al momento de la entrega al usuario final.

En el plano nacional de acuerdo con el trabajo de investigación de Callapiña et al. (2019) concluyeron que, en el sector financiero, se vienen desarrollando proyectos de TI, donde la calidad es el principal factor en un proyecto software, por lo tanto, debe gestionarse de manera apropiada para contribuir a su éxito. Solo ciertas empresas priorizan la calidad de sus proyectos, lo cual llega a ser grave ya que si se manifiestan defectos pueden resultar tarde y esto impactaría al negocio de manera negativa. El poco conocimiento del aseguramiento de calidad en los

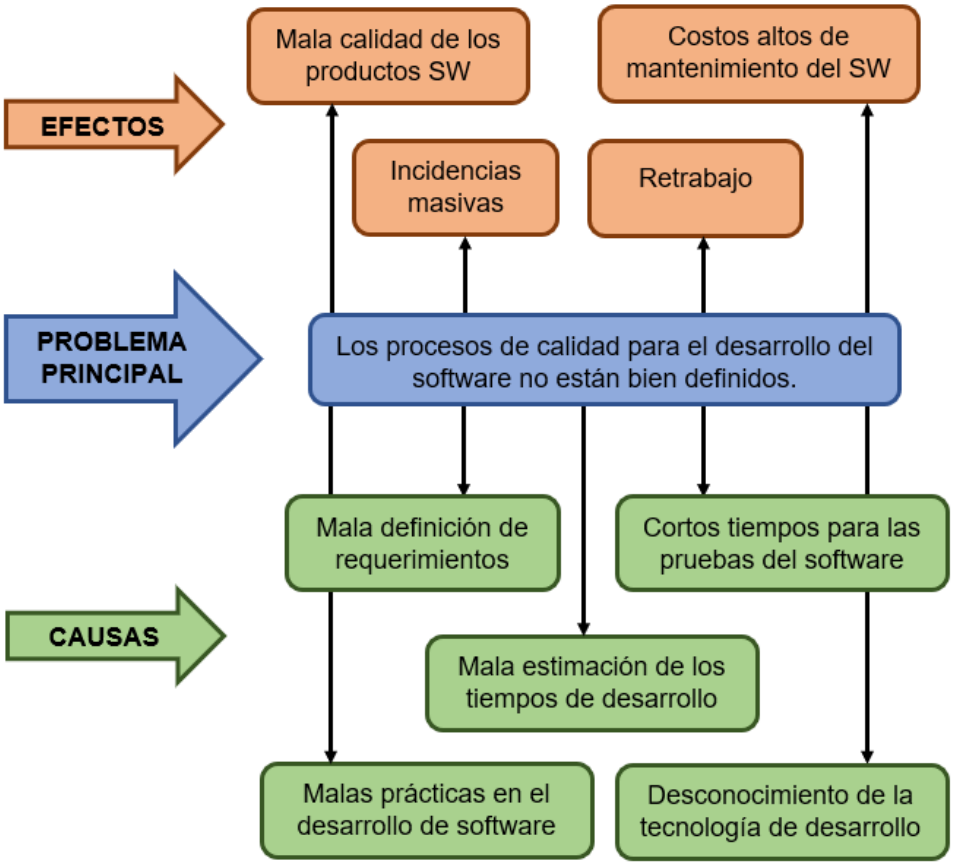
proyectos hizo que la gestión de calidad sea puesta asequible para muchos, pero en realidad es puesta en práctica por pocos. Un software, tiene etapas para su construcción y puesta en marcha, pero durante su ciclo de vida pueden surgir una variedad de problemas, esto se debe a que no se tiene en claro las funcionalidades y necesidades, o que también en el camino de desarrollo se viene presentando cambios frecuentes el cual puede ser confuso para las personas que se encargan del desarrollo debido a la falta de especificación de requisitos (Yusong, 2018).

En el mapa local Espejo et al. (2016) mencionaron que, las empresas invierten una gran suma de dinero para incorporar el área de TI e su organización, por lo tanto, estas deben prestar mucha atención al proceso de calidad, haciendo uso de los estándares de desarrollo de software. Para que una organización tenga éxito este debe tener su software con un correcto funcionamiento, es por esto que una de las principales preocupaciones de las industrias tecnológicas y financieras, es desarrollar programas y productos de calidad.

Las organizaciones que brindan o realizan servicios de desarrollo de software necesitan tener un proceso bien definido durante la etapa de planificación y construcción del proyecto, tal que el equipo de calidad pueda participar desde una etapa temprana, si bien es cierto en la actualidad muchas empresas usan metodologías agiles que hacen posible brindar pequeños entregables al usuario final, la financiera en estudio no es ajena a esta situación, ya que trabaja con metodologías agiles, en el cual el equipo de calidad se involucra desde una etapa de inicio donde se definen las historias de usuario que se verán durante cada Sprint, pero se tiene la necesidad de definir bien el proceso para que los analistas de calidad puedan participar desde la etapa, donde se realiza la toma de requerimientos para prevenir futuros defectos que se puedan presentar, o en todo caso hacer saber las validaciones que este realizara durante la etapa de pruebas. Se requiere evaluar la calidad de los sistemas usando la Norma ISO 25000 para determinar si el software es o no óptimo, tomando en consideración las necesidades del usuario final, buscando que el sistema sea eficiente, eficaz y agradable.

Figura 1

Árbol de problemas



En base a lo comentado se formula el problema general, ¿De qué manera la aplicación del ISO 25000 ayudara a mejorar el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023? Como problemas específicos se plantearon: PE1: ¿Cómo determinar la influencia del ISO 25000 sobre la productividad del desarrollador en el área TI de una financiera?; PE2: ¿Cómo determinar la influencia del ISO 25000 sobre los defectos del software en el área TI de una financiera?; PE3: ¿Cómo determinar la influencia del ISO 25000 sobre el rendimiento del software en el área TI de una financiera?

Como justificación teórica del presente trabajo se mejoró el conocimiento teórico, debido a que se desarrollara un método el cual permitirá evaluar la calidad de los proyectos software en cada una de sus fases, y esto no será solo cuando el producto se encuentre la etapa de pruebas, sino que también se implementará en

cada etapa de desarrollo del software llegando con éxito a la puesta en producción del producto. El método será sostenido por el ISO 25000 como estándar. Con la ayuda de la tecnología, las empresas pueden procesar mejor los datos y, por lo tanto, proporcionar un sobresaliente análisis (Mendes, 2016). Como justificación práctica es importante tener en cuenta a la calidad como un elemento importante el cual debería ser gestionado durante su ciclo de vida, es decir desde su etapa de inicio a su fin, por lo tanto, es esencial tener con un método que posibilite realizar una buena gestión. Permite utilizar todas las tecnologías de la información analizando e integrando los datos realizados por el sistema operativo (Mendes, 2016). Como justificación metodológica, la tesis permitirá tener diferentes perspectivas de la organización, que tendrán un objetivo en común, el cual será detectar posibles errores y soluciones tempranas, determinar la calidad de software con la ISO 25000, permitirá garantizar la confiabilidad de la evolución de calidad que se vienen manejando dentro de la empresa. A causa de los nuevos sistemas de información, las organizaciones son capaces de comprender las amenazas al crecimiento que enfrentan otras empresas para poder implementar estrategias para prevenirlas (Fred, 2014).

El objetivo general es determinar de qué manera la aplicación del ISO 25000 ayudara a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023. Como objetivos específicos se tienen, OE1: Determinar la influencia del ISO 25000 sobre la Productividad del Desarrollador en el área TI de una financiera; OE2: Determinar la influencia del ISO 25000 sobre los defectos del software en el área TI de una financiera; OE3: Determinar la influencia del ISO 25000 sobre el Rendimiento del software en el área TI de una financiera.

Como hipótesis general se planteó lo siguiente: La aplicación del ISO 25000 ayudo a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023. Las hipótesis específicas son los siguientes, HE1: La ISO 25000 influye en la productividad del desarrollador en el área TI de una financiera. HE2: La ISO 25000 influye en los defectos del software en el área TI de una financiera. HE3: La ISO 25000 influye en el rendimiento del software en el área TI de una financiera.

II. MARCO TEÓRICO

En una investigación relacionada con el aseguramiento de calidad aplicando un proceso de desarrollo de software, según Espejo et al. (2016), tuvieron como problemática la calidad que no se estaba aplicando durante la implementación del software. Por otro lado, el tipo de investigación fue cuantitativo, aplico una metodología orientada a conseguir productos de calidad, tuvo un estudio de un total de 22 proyectos de desarrollo, después de ellos se realizó la comparación de 2 grupos de proyectos integrados por 11, donde en el pre-test no se utilizó un modelo y en el pos-test llegaron a adaptar un patrón para lograr la calidad durante la etapa de desarrollo de software, acorde a esto el porcentaje de costo en la calidad del grupo del pre-test fue de 19.84%, a cambio del pos-test donde el porcentaje de costo de calidad fue un 7.41%, asimismo, en el grupo de pre-test la cantidad de usuarios satisfechos fue de un 51.29% y del pos-test fue de 73.71%. El investigador llego a la conclusión de que aplicando el modelo del CMMI logró reducir costos en torno a la calidad en un 12.43% y logro aumentar la satisfacción del cliente en un 22.42% con respecto al producto software así mismo los defectos fueron reducidos, indica también que es importante que dentro del equipo de trabajo estos participen activamente, compartan conocimientos y tengan una mejor comunicación para lograr los objetivos estratégicos por la empresa.

Carrasco (2022), desarrolló un proyecto donde su objetivo fue reducir los tiempos de entrega del producto software y mejorar la comunicación entre el equipo del trabajo; describe que la problemática fue que no se cumplían con las fechas planificadas además durante el desarrollo se iban agregando nuevos requerimientos, no había un compromiso de parte de los involucrados en el negocio y no había una participación activa. Así mismo, el tipo de investigación fue pre-experimental, teniendo 25 trabajadores que representaron a la población ya que eran los encargados de los proyectos de software, tuvo como resultados que al aplicar un prototipo de gestión de proyectos logro disminuir los tiempos de entrega en un 9%, alcanzo incrementar el grado de conocimiento en un 17%, la comunicación con el equipo aumento un 18%, llegando a la conclusión que al

implementar un prototipo de gestión de proyectos mejora significativamente en el desarrollo de productos software.

El investigación Llactahuaman (2018), desarrolló un proyecto para el desarrollo de software empleando metodologías agiles, tuvo como problemática de no contar con un proceso estructurado para el desarrollo de productos software, el tipo de estudio fue de tipo pre-experimental, como población tuvo un total de 12 proyectos de desarrollo, para recolectar los datos hizo uso de las fichas de registro, tuvo como resultado que las metodologías agiles mejoran el nivel de eficiencia en los proyectos de desarrollo, en el pre-test tuvo un resultado de 53.75%, mientras que en el pos-test alcanzo un 94.63%, incrementando considerablemente en un 40%88% en nivel de eficiencia en sus proyectos software, llego a la conclusión de que las metodologías agiles mejoran el desarrollo de los proyectos software dentro de la organización.

En un estudio acerca del desarrollo de software el autor Falen (2020), tuvo como problemática la carencia de un patrón de apoyo para la implementación de un software; por lo tanto, los productos que brindaba tenían baja calidad y no podían cumplir con las necesidades del usuario ni los requisitos detallados al inicio del proyecto, no cumplían con los tiempos de desarrollo por ende las fechas de entrega no eran las planificadas, al hacer uso de una metodología tradicional el usuario recién podía ver el software una vez realizada la entrega, generando molestias en el caso que algo no estaba de acuerdo al diseño .La investigación que se realizo fue de enfoque cualitativo, con un tipo de investigación aplicada, se tuvo como población a 3 personas que eran especialistas en TI, como resultado de la investigación se llegó a la conclusión de que al aplicar la metodología SCRUM para el desarrollo de software este facilita el adecuado seguimiento al proyecto así como su control y evaluación, debido a que SCRUM trabaja con entregables cortos que son definidos como Sprints, en el cual se asignan roles de acorde a las habilidades de cada miembro del equipo así como las revisiones periódicas ya que se maneja un tablero donde se especifican a detalle las actividades de cada miembro del equipo.

En un estudio sobre un modelo de mejora para la calidad de software, donde el autor Flores (2018) menciona que, el problema fue que la carencia de una buena planificación; por lo tanto, siempre se generaban cuellos de botella debido a una mala administración de requisitos funcionales y no funcionales. Para la investigación el diseño fue no experimental de tipo descriptivo, con una población de 14 personas, tomando como muestra un total de 5 personas. Gracias a la aplicación de CMMI se pudo desarrollar un buen nivel de madurez, aplicando practicas importantes en la organización, lo cual permite adaptarse al mercado actual manteniendo una mejora continua dentro de la empresa. Hoy en día se cuenta con un modelo de calidad que permite asegurar que los procesos de desarrollo alcancen la calidad del software.

En un entorno internacional el trabajo de investigación orientado a una herramienta que permita dar soporte al aseguramiento de calidad, el autor Ibarra (2018), cuya problemática fue el no contar con una herramienta que brinda buenas prácticas a los desarrolladores y testers; así mismo en el proceso de desarrollo de sus actividades para asegurar la calidad de su producto se hizo uso de la metodología cascada para que las personas del equipo puedan tener sus responsabilidades definidas e identificables. Para la recolectar los datos hizo uso del cuestionario ya que son completados de forma individual utilizando la escala de Likert a una población de usuarios. El autor concluye que, las herramientas de aseguramiento de calidad se enfocan en una manera de optimizar las pruebas del software; por otra parte realizo un análisis el cual indica que los errores más comunes encontrados son defectos menores, error en la documentación, mala integración del código o en la misma interfaz del usuario, en pocas palabras gracias al estudio se dieron cuenta que los defectos que se encontraron no son tomados en cuenta, por consiguiente a futuro se llegan a tener problemas críticos en el mismo software con un total de 47% de defectos en el cual deberían de enfocarse para aplicar un modelo de calidad.

En la investigación acerca de un modelo de calidad de software que será aplicado en un sistema, la autora Vaca (2017), la problemática fue el no contar con procesos que facultan el cumplimiento de la calidad dentro de su equipo

garantizado la eficiencia, seguridad, funcionalidad y eficiencia; tuvo como objetivo determinar la calidad del software apoyándose en un modelo de calidad, por consiguiente gracias a su estudio desarrolló un instrumento basado en la ISO 25000 el cual ayudo a detectar las debilidades y fortalezas de los módulos del sistema que eran probados. El tipo de investigación fue cuantitativo asimismo uso la investigación documental y descriptiva, con una población de 40 personas, como resultados logró un 6.54 sobre los 10 puntos de calidad de uso y un 6.50 de 10 sobre la calidad total, basándose en esto se pudo proponer mejoras basados en la experiencia para el proceso de pruebas, los cuales también fueron aplicados en los desarrollos a futuro, al implantar un modelo de calidad este permitió que los sistemas incrementen su calidad y logren la satisfacción del usuario con respecto al sistema.

Para usar un producto software es necesario que este tenga mantenibilidad por lo tanto Mendoza (2018), cuya problemática fue el no contar con procesos establecidos para que el software que manejan tenga mantenimiento a través del tiempo, lo cual genera demoras para la definición de nuevos requerimientos o requerimientos que necesitan ser cambiados. En el proyecto se aplicó la metodología de ciencia del diseño para poder implantar una propuesta, en su tesis indica que hizo un estudio con el objetivo de realizar una propuesta con estándares de calidad de software para el mantenimiento del producto, así pues se logró implementar un proceso bajo los estándares de calidad de software que permite aplicar el proceso de mantenimiento donde este no tenga impacto con futuros desarrollos que puedan perjudicar de alguna manera al software en uso, la propuesta tiene un 50% de usuarios que consideran sencilla la aplicación, sin embargo, el 50% restante indican que es complejo ya que necesitarían aprendizaje continuo y capacitaciones, por ello recomienda que una persona se responsabilice del cumplimiento de los procesos de calidad.

Un punto importante en considerar es la evaluación del software antes de aplicar el ISO 25000, Reyes (2021) comenta que, en su trabajo de investigación cuyo objetivo fue comparar las ventajas de un producto que no usa buenas prácticas contra un producto que requiere aplicar buenas prácticas o normas ISO,

para esto empleo las métricas definidas en la ISO 25000, para evaluar la funcionalidad y fiabilidad del producto software, resumiendo concluye que es importante hacer una evaluación de la calidad del software antes de implementar la ISO 25000, ya que de esta manera reducirá los gastos de dinero y tiempo. Debido al gran interés de contar con modelos de evaluación para la calidad de software. Ramos (2016) indica en su investigación que su problemática fue el no contar con un mecanismo que pueda dar soporte a sus procesos de desarrollo de software y que este pueda ser aplicado a todo tipo de producto. Al aplicar el ISO 25000 llega a la conclusión de que todo producto software debe lograr la eficiencia, la efectividad y satisfacción del usuario final, resalta que para lograr esto es importante establecer las características del software, y realizar cálculos para obtener las métricas que permitan la calidad del sistema.

En una investigación basada en la implementación de las métricas para la valorar el proceso de aseguramiento de calidad en los proyectos de software, Caiza (2019), tuvo como meta implementar métricas que puedan determinar el proceso de aseguramiento de calidad en el desarrollo de productos software, la metodología de investigación-Accion que sugiere etapas, como son las de planificar, diagnosticar, evaluar, tomar acciones y especificar aprendizaje, presentando a la metodología como un proceso reflexivo e interactivo, se tomó como población la cantidad de 3 proyectos a ser analizados, dando como resultado: para el caso 1 donde se tenían que implementar mejoras para la automatización de procesos se tuvo como resultados que el grado de calidad del proyecto software fue de 7.28 de 10 puntos, para el caso 2 donde se tenían que implementar mejoras en el proceso de desarrollo se tuvo un grado de satisfacción con respecto a la calidad del producto software un total de 8.71 de 10 puntos, por último, para el caso 3 donde se quería automatizar el tiempo de respuesta y mejorar el proceso de registro de solicitudes se tuvo como resultado que el grado de satisfacción del proceso de calidad fue de 6.26 de 10 puntos. Como conclusión el autor comenta que para implementar un proceso de evaluación de calidad es necesario realizar un estudio previo a proceso de pruebas que se estaba llevando a cabo y ya en base a esto se podrían aplicar procesos que permitan la implementación de estas métricas de calidad, como lo fue en este caso que se trabajó bajo la norma ISO 25023 a la par con las buenas

prácticas del fundamentos del ISTQB que es una certificación internacional de calificaciones de pruebas de software, gracias a esto se pudo implementar y mejorar el proceso de pruebas, se encontraron defectos tempranos que pudieron ser corregidos, y de esta forma brindaron un producto de calidad.

Por otro lado, el autor Luna (2017) nos dice que, ante la problemática de querer optimizar el tiempo, recursos y tener una garantía con respecto a la calidad del producto, se realizó un análisis con la finalidad de estructurar los procesos para el desarrollo del software en el área de TI. La metodología que se aplico fue mixta combinando estudios cualitativos y cuantitativos para el estudio tomaron como referencia 44 proyectos que habían finalizado y 47 proyectos que estaban siendo solicitados para poder aplicar un método de calidad durante el proceso de desarrollo, al momento de aplicar las buenas prácticas el resultado fue satisfactorio ya que se logró enriquecer el proceso de desarrollo de software disminuyendo un total del 20% de horas que se retrasan durante el proceso de desarrollo, eliminando un 99% de errores en el código en conjunto con la estructura de sus datos, el autor destaca que al aplicar un modelo de calidad no tendrá fin ya que es necesario continuar estableciendo procesos que permitan que la organización alcance la calidad de sus productos software brindados.

Cabe señalar que el autor Jimenez (2017), tuvo como objetivo elaborar un plan de pruebas que garantice la calidad de un producto software; gracias a su investigación sobre las diferentes metodologías para la calidad del software, llego a la conclusión de que es necesario aplicar el proceso de calidad desde la fase inicial del desarrollo del software, se deben realizar pruebas a nivel de código, pruebas de caja negra y caja blanca; así mismo dependerá de que tipo de software se está implementando, el tipo de desarrollo y contar con un procedimiento que permita identificar defectos tempranos para poder garantizar la calidad y mejora continua. Dado que el mundo de TI es cada vez más desafiante, las empresas deben tener contemplado la calidad para poder sobresalir con respecto a otras organizaciones. En relación a la calidad de un producto software Rashid y Wasif (2016) mencionan que, la calidad logra que un usuario se encuentre a gusto con el producto que se le brinda de manera que esto solo puede lograrse mediante la

aplicación de normas y procedimientos adecuados en el tiempo del proceso de construcción de software. Para que se pueda mejorar la calidad del software es vital adoptar una variedad de técnicas, de acuerdo con el problema que se presente, si se combinan diferentes técnicas se podrán alcanzar resultados excepcionales en un proyecto, en conclusión, al implementar nuevas técnicas en cada etapa del ciclo de vida de un software, ayudará a aumentar la ventaja y garantía de calidad.

En un estudio que evalúa la usabilidad de las aplicaciones móviles los autores Karima y Abran (2016) comentan que, el uso de los teléfonos ha incrementado considerablemente con un total del 81% con respecto al año 2011, por lo tanto, se plantea el objetivo de evaluar la usabilidad de las aplicaciones haciendo el uso de la ISO 25000, para este experimento participaron 32 personas con diferentes dispositivos móviles teniendo un conjunto de casos de pruebas como de Google Maps y Google Apps, llegando a la conclusión que la calidad de software es influenciada considerablemente dependiendo del sistema operativo, y que la capacidad de almacenamiento o la resolución de la pantalla juegan un papel importante al evaluar la usabilidad de las aplicaciones. Al momento de analizar los resultados resalta que los desarrolladores deberían de tener en cuenta el diseño de las aplicaciones para que estos puedan ser fáciles de usar por los usuarios. Según Ribeiro y Soares (2017) en su artículo donde desarrollan un método para determinar la calidad de un software hace uso de la ISO 25000, su objetivo fue indicar el grado de calidad de un producto software, para esto toma un requerimiento con un índice de calidad bajo para aplicar los estándares de las norma ISO 25000, teniendo como resultado mejoras al momento del desarrollo, además que gracias a este método los programadores pueden identificar problemas tempranos en la etapa de planificación, para de esta forma proponer soluciones eficientes.

El autor Rahman (2020), investiga acerca de los estándares de calidad basados de las ISO/IEC 25000, para desarrollar un software de calidad. Su objetivo fue determinar la clave de la calidad basándose en la ISO 25000. Gracias a su investigación concluye que se logró identificar las características de calidad que sirven para evaluar su sistema E-Learning y de esta forma no tendrán problemas

con futuras actualizaciones que puedan perjudicar las pruebas de regresión de su software. Según el autor Perez (2016) comenta que, hoy en día el desarrollo de aplicaciones de negocio se ha convertido en el corazón de toda organización, por ello durante el desarrollo de software se están planificando las diferentes pruebas que se realizaran como las funcionales, no funcionales, de rendimiento, de seguridad, fiabilidad y usabilidad, es por esto que el objetivo de su investigación es desarrollar una herramienta que facilite el proceso de pruebas funcionales. Como conclusión se desarrolló la herramienta gexrenof el cual ayudo a la mejora de la gestión de proyectos a nivel gerencial, ya que se logró definir un procedimiento para las pruebas automatizadas las cuales se integraron con Jenkins, esto permitió agregar más casos de pruebas que ayudaban considerablemente con las pruebas de regresión.

El autor Ali et al. (2017) indican que, debido a la naturaleza del desarrollo del software desde el punto global, la calidad del software es un tema importante ya que una gran cantidad de estos tuvieron resultados insatisfactorios en la industria, según un estudio se revelo que solo un 6.2% de proyectos software fueron culminados dentro de un tiempo definido, y es por esto que consideran que la calidad del software es un desafío, de acuerdo a esto se proponen técnicas para renovar el procedimiento de desarrollo de software. La metodología de investigación empleada fue la revisión sistemática de la literatura que ayudo a establecer factores de éxito, prácticas para la implementación y las barreras existentes, hizo uso del cuestionario de encuesta para el estudio del caso, como resultados se tuvo que se lograron identificar 211 factores de éxito con un impacto positivo, 22 barreras negativas, como conclusión se tuvo que al implementar nuevas técnicas ayudan a mejorar y evaluar la efectividad de un producto software.

Azeem et al. (2017), en su artículo consideran que la calidad es el componente principal en el proceso de desarrollo de software dado que se relaciona directamente con la satisfacción del usuario y esto conlleva al, por tanto el objetivo es lograr que el producto software alcance las expectativas planteadas, culminen dentro del tiempo acordado y se relacionen con la satisfacción del usuario, para esto se propuso la implementación del modelo AZ como un nuevo ciclo de

desarrollo de software, que consta de 3 fases, siendo estas la participación del cliente, la fase del desarrollo y la fase del lanzamiento, teniendo una participación del cliente para minimizar riesgos ya que los requisitos tienden a cambiar, es por ello que se realizan pruebas unitarias durante la fase de diseño para minimizar el riesgo en conjunto con la pruebas de usabilidad; para comprobar la eficacia de la metodología se aplicó en 22 organizaciones demostrando resultados satisfactorios, teniendo como resultados que el modelo AZ logra que las organizaciones puedan brindar un producto de calidad dentro del tiempo acordado y a un presupuesto ya determinado.

Lindsjorn et al. (2016) explican que, en los equipos de desarrollo ágil donde los equipos son pequeños y autodirigidos es cuando se logra un mejor desempeño y por ende que el producto software sea de calidad, por ejemplo en su estudio se evaluó el desempeño del equipo, para esto se aplicó la encuesta a 477 personas que fueron encuestadas, como resultado se tuvo un efecto positivo en la calidad de trabajo que realizaban para lograr un producto de calidad, logrando una satisfacción con respecto al producto. Por otro lado, Moyo y Mnkandla (2020) afirman que, las metodologías de desarrollo de software son principalmente diseñadas para mejorar la calidad del software, por lo tanto, propone una metodología ágil SSDM que permitirá desarrollar un software seguro que promueva la calidad y la seguridad para lograr un buen desarrollo. Como resultado se tuvo que se logró crear productos seguros y de calidad, sin afectar la agilidad en el desarrollo, este puede ser utilizado tanto en desarrollos grandes o medianos.

En referencia a la calidad de software el autor Behutiye et al. (2020) destacan que, la calidad de software tiene un papel importante para alcanzar los propósitos trazadas por la empresa, es por ello que ante la problemática de que durante el desarrollo ágil de hoy en día muchas veces la definición de casos de prueba están mal definidos, debido a los cortos tiempos que se manejan, dado esto tienden a descuidar los requisitos de calidad, es por ello que se plantea a sintetizar una buena gestión de los requisitos de calidad con el desarrollo rápido del software, con un estudio sistemático, se llegó a la conclusión de que el manejo de los requisitos de calidad son importantes durante el desarrollo rápido del software, se identificaron

brechas en la investigación donde se concluye que son necesarias aplicar más herramientas y pautas para los cortos ciclos de iteración; del mismo modo es necesario definir más estrategias para lograr que durante el desarrollo ágil, la calidad de software no se quede con cortos tiempos y se pueda participar desde una etapa de inicio para alcanzar un software de calidad durante el tiempo establecido.

Según Arcos y Mauricio (2020), la problemática en la industria de TI es que al aplicar metodologías agiles no garantizan que un software tendrá éxito y lo demuestran con las altas tasas de proyectos software que están retrasados, no fueron terminados, fueron rechazados o abandonados, por lo tanto, si no aplicas la calidad desde un inicio están destinados a que el proyecto fracase, siendo la calidad el principio vital en todo proceso de desarrollo de software, se sugiere un modelo de calidad para el proceso de desarrollo de software el cual consiste en 4 categorías de prácticas agiles y 8 características de calidad que se basan en la ISO 25010. Se aplicó el modelo a una población de 146 personas que desarrollan software mediante las metodologías agiles, en conclusión, las buenas prácticas con una buena gestión de proyectos tienen una influencia positiva en la funcionalidad del producto y por ende una buena calidad, logrando que el software sea compatible, potable, seguro y usable.

El autor Hovorushchenko (2018) comenta que, la mayor parte de las actividades de las organizaciones están dedicadas a los sistemas de información, que son softwares; lo más importante para los usuarios y partes interesadas es el grado de calidad que un producto software tenga ante la necesidad del usuario final, mediante una evaluación cuantitativa acerca de la calidad de software cuando la documentación tiene requisitos incompletos estos no se llegan a implementar de manera exitosa, ya que la mayoría de errores que se tienden a presentan son los que surgen durante la definición de requisitos teniendo un margen del 10% al 23% de errores, por ende solucionar estos defectos pueden llegar a costar hasta 100 veces el costo del proyecto, debido a esto se planea desarrollar métodos para la determinación de calidad según la norma ISO 25010, como resultados se tuvo que se necesita aplicar un modelo ontológico donde se tenga un buen nivel de

conocimientos, para facilitar la capacidad de análisis y comprensión de la información, gracias a esto se pudo verificar la suficiencia de la información tomando en consideración los estándares del desarrollo del software para un dominio temático, el cual permite la evaluación desde los primeros pasos de la etapa de desarrollo de software.

Bautista (2018), desarrollo su investigación donde tuvo como objetivo determinar la calidad de software durante la etapa de desarrollo tomando como referencia la ISO 25000, como resultado logro establecer requerimientos de evaluación para evaluar el producto software mediante una selección de módulos que tenían cambios funcionales, tuvo como conclusión que, al momento de implementar un proceso de pruebas, el producto software tuvo buena calidad y logro satisfacer las necesidades del usuario. Para Garcia y Mazo (2017) quienes resaltan que, en la actualidad se vienen desarrollando una gran cantidad de proyectos software, y por ello la ejecución de pruebas se ha vuelto una problemática. En la investigación se aplicó la ISO 25000 como objetivo de validar los requerimientos funcionales, no funcionales, la seguridad, confiabilidad y usabilidad, como resultado del trabajo se implementó un procedimiento que ayudo con el proceso de desarrollo para alcanzar la calidad de un software de acuerdo a las pruebas funcionales y no funcionales.

Los autores Xing y Meng (2019), realizaron una investigación donde su objetivo fue evaluar los modelos de calidad en la actualidad en el cual tuvo una población de 716 artículos donde tomo como muestra 31, tuvo como resultado el descubrimiento de oportunidades de mejora las cuales son: definir criterios para que el modelo de calidad se pueda adaptar, investigar a profundidad las fortalezas y debilidades de los nuevos métodos, realizar un estudio con casos reales, llegaron a la conclusión de que en la actualidad se pueden adaptar más contextos de mejoras de calidad al desarrollar un producto software. En una investigación realizada por Kyoo (2016) desarrolló una nueva perspectiva basándose en la innovación del desarrollo de software, teniendo como objetivo la mejora del ciclo de vida del desarrollo de software, su investigación fue aplicada y explicativa tomando como población a 10 empresas que se dedican a TI; tuvo como resultado el conocer

cuál es el modelo para adaptar un proyecto de desarrollo de forma efectiva. El investigador concluye que, al aplicar un modelo de innovación le permite generar nuevas ideas que permiten que se termine un producto software de forma satisfactoria y de calidad.

Hynninen et al. (2018) hicieron un estudio donde el objetivo fue implementar un marco de trabajo que ayuda a calcular las métricas del tiempo de ejecución de un software y que también ayuda a medir la calidad de uso, para esto hizo uso de la norma ISO 25000, como resultado logro implementar un conjunto de métricas que fueron adaptadas durante la etapa de mantenimiento del software, concluyendo que al aplicar métricas de calidad el producto software puede ser modificado y analizado mediante nuevos requisitos. Del mismo modo Vargas (2018) en su investigación acerca de una metodología de calidad de software tuvo como objetivo desarrollar una metodología que permita evaluar las herramientas de los sistemas de información. El autor llego a la conclusión de que, si aplica las metodologías de calidad durante el ciclo de vida de desarrollo de software se puede entregar un producto de calidad.

Como teorías generales el modelo de calidad de la serie ISO/IEC 25000 según Suárez y Garzás (2014) nos indican que, la norma ISO 25000 nace ante la necesidad de determinar la calidad del software en base a modelos y procesos, asimismo, esta norma se conoce como SQuaRE (Software Product Quality Requiremens and Evaluation). La ISO/IEC 25000 se compone por distintas partes en las que destacamos.

La ISO/IEC 25010 determina las propiedades de la calidad del software, las cuales son 8 y se pueden apreciar también en la figura 2: Seguridad, usabilidad, Funcionalidad, fiabilidad, portabilidad, rendimiento, mantenibilidad y compatibilidad.

Figura 2

Modelo de calidad del producto software basado en la ISO/IEC 25010

Nota: ISO/IEC 25010 (2011).

En la ISO/IEC 25040 se definen las 5 actividades para que la calidad de software pueda ser evaluada: (a) establecer los requisitos que se consideraran para la evaluación del producto; (b) especificar la evaluación estableciendo los criterios de medición y de las métricas: (c) diseñar la evaluación elaborando un plan de actividades que se deben realizar para evaluar el producto; (d) ejecutar la evaluación efectuando las funciones de evaluación y medición del producto software; (c) concluir la evaluación realizando el informe del producto evaluado.

La ISO/IEC 25020, donde se definen las métricas de calidad de software, pero aún está pendiente debido a que no se tienen los indicadores que determinan la calidad del software de forma estandarizada.

Según Crespo (2018), la norma ISO 25000 también conocida como SQuaRE se especializa en la evaluación de productos software, se denomina SQuaRE a los sistemas y requisitos de calidad de software y evaluación, el objetivo principal de la ISO 25000 es brindar un panorama en general de los contenidos del SQuaRE, además de centrarse en 2 principales procesos las cuales son la especificación de requisitos de calidad de software y la evaluación de la calidad del software, las cuales son soportadas en el proceso de medición de calidad de software. Se divide en 5 áreas las cuales son: (a) ISO 2500n: gestión de calidad; (b) ISO 2501n: modelo

de calidad: compuesto entre otros por fiabilidad, seguridad, mantenibilidad y usabilidad; (c) ISO 2502n: medición de calidad; (d) ISO 2503n: requisitos de calidad; (c) ISO 2504n: evaluación de calidad.

Como teorías específicas Méndez (2019) indica que, la norma ISO 25010 surge con la necesidad que un producto pueda ser eficaz, de confianza, eficiente y con garantía de seguridad, para satisfacer las necesidades del usuario. Para esta norma se tiene 8 características que son calidad de uso, mantenibilidad, eficiencia, funcionalidad, usabilidad, fiabilidad, portabilidad y seguridad, esto ayudara a garantizar la calidad del software demostrando que el producto es de total confianza.

La primera dimensión es la usabilidad, donde la ISO 25010. (s/f), lo describe como el grado en el que el software cumple los objetivos definidos con eficiencia, eficacia y satisfacción, como complemento indica que este debe tener: (a) reconocimiento de la idoneidad donde se mide el punto considerado por el usuario cuando el software satisface sus necesidades a partir de un impacto inicial al conocer el software; (b) capacidad de aprendizaje, donde un grupo de usuarios específicos se plantean objetivos para aprender a usar el software con eficiencia, ausencia de riesgos, eficacia y con la satisfacción que cumpla su uso especificado; (c) operabilidad donde el software sea fácil de controlar y operar; (d) protección contra errores de usuario para evitar que el usuario pueda cometer errores al usar el software; (e) estética de la interfaz de usuario donde la pantalla principal debe ser agradable ante los ojos del usuario haciendo un buen uso del color y el diseño gráfico; (f) accesible es decir que el software pueda ser fácil de usar en personas de todas las edades y características físicas. Para Ormeño (2019) la funcionalidad es cuando se valida que el sistema cumpla con los requisitos especificados haciendo solo las cosas necesarias para completar las tareas, esto se puede comprobar realizando pruebas del sistema, unitarias y de regresión.

La segunda dimensión es la seguridad, la ISO 25010. (s/f), describe que el software debe resguardar los datos e información almacenados de cada usuario, esta dimensión debe tener: (a) confidencialidad para garantizar que solo los

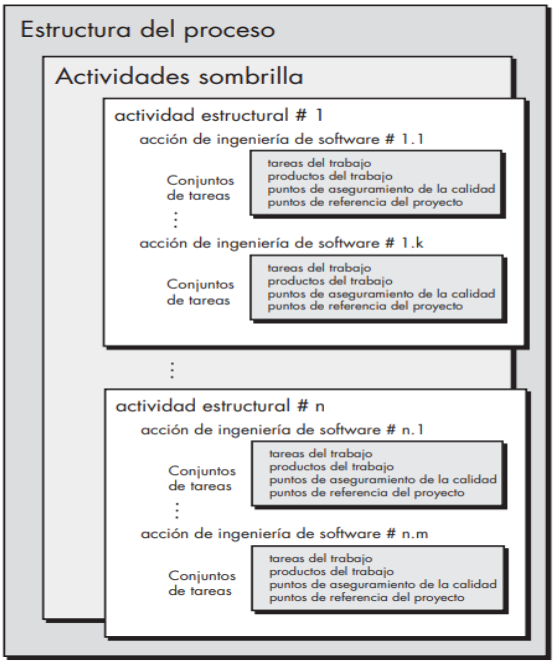
usuarios que están autorizados puedan acceder a datos sensibles; (b) integridad que evitara el acceso no autorizado; (c) no repudio donde se puedan tener evidencias que las acciones se hayan dado, para que en el futuro estas no puedan ser repudiadas; (d) responsabilidad para que las acciones solo puedan ser rastreadas por las misma entidad; (e) autenticidad para poder probar que comprobar que un tema o recurso corresponde al reclamo. De acuerdo con Ormeño (2019) la seguridad es sin duda unos de los puntos más importantes ya que hoy en día es importante proteger la información de cada usuario.

La tercera dimensión es la mantenibilidad, la ISO 25010. (s/f) describe que, debe de tener el grado de eficiencia y eficacia para que se pueda modificar un sistema o producto software, ya sean por mejoras, correcciones, adaptación a un cambio de entorno o un cambio en los requisitos funcionales, esto significa que el software tendrá una actualización, en sus características específicas se tiene: (a) modularidad que es el modo en que el cambio que se realice tenga un impacto mínimo en otros componentes; (b) reutilización que permitirá que un activo pueda ser utilizado en la construcción de otro activo; (c) analizable siendo la capacidad de evaluar el impacto que se tendrá en una de las partes del software; (d) modificabilidad siendo el grado en que se pueda modificar un componente sin tener defectos que puedan perjudicar al software existente; (e) portabilidad siendo el grado de eficacia y eficiencia en la que se pueda migrar a un nuevo sistema operativo; (f) adaptabilidad donde el producto pueda adaptarse satisfactoriamente a un hardware, software o sistema operativo; (g) instalable donde este producto pueda ser instalado o desinstalado según se especifique; (h) reemplazable donde se permitirá reemplazar por otro producto que realice las mismas funcionalidades que este.

Como teoría general el autor Pressman (2010), define proceso de desarrollo como un grupo de actividades que se ejecutaran para obtener un entregable, esto permite que un equipo de trabajo elija un conjunto de tareas y acciones a realizar para el desarrollo el software, entregándolo de manera oportuna y con calidad para poder satisfacer las exigencias del usuario, esto incluye un compuesto de 5 actividades de un proceso en general: (a) la comunicación para entender los

objetivos y definir los requerimientos y funciones del software; (b) la planeación donde se definen las especificaciones técnicas que se realizaran, probables riesgos, recursos a utilizar y a definir el programa de actividades; (c) modelado con la finalidad de entender los requerimientos, es necesario realizar un bosquejo para entender el panorama total del problema un la solución; (d) la construcción en esta actividad se realizara la generación del código y las pruebas para descartar los errores, y si existen solucionarlo antes de su puesta en producción; (e) el despliegue aquí ya se hace la entrega del software funcionando para que el usuario pueda usarlo y evaluarlo de acuerdo a sus necesidades. Para cada caso, el proceso de desarrollo es distinto, pero siempre se realizan las mismas actividades ya mencionadas. En la figura 3, se muestra como son las actividades donde se definen las tareas que se deben realizar, el producto software que se desarrollara, los puntos de aseguramiento de calidad y la evaluación del avance. Para los autores Owens y Khazanchi (2009) se debe evaluar el aseguramiento de calidad durante toda la etapa de desarrollo de software para un programa, con la finalidad de reducir el riesgo, cumplir con los requisitos funcionales y no funcionales, para incrementar la calidad del software.

Figura 3
Procesos del software



Nota: Pressman (2010).

Como teorías especificas tenemos a Navarro et al. (2013) quienes comentan que, lo que buscan las metodologías agiles de desarrollo de software es reducir costes y tiempo, en cambio las metodologías tradicionales buscan disciplinar el proceso de desarrollo para que pueda ser más eficiente y predecible, pero se tiene un problema con las metodologías tradicionales ya que se debe seguir un proceso estático y esto muchas veces retrasa el proceso de desarrollo, a comparación de las metodologías agiles que pueden adaptarse ante cualquier situación, orientándose a las personas y no a los procesos. Tavares et al. (2017) comentan que, una de las metodologías más comunes es Scrum, ya que brinda un conjunto de buenas prácticas que ayuda a agregarle valor a los flujos de trabajo, reduciendo el tiempo en el desarrollo de software, además de tener transparencia dentro de la organización ya que cada Sprint es monitoreado mediante indicadores de cumplimiento.

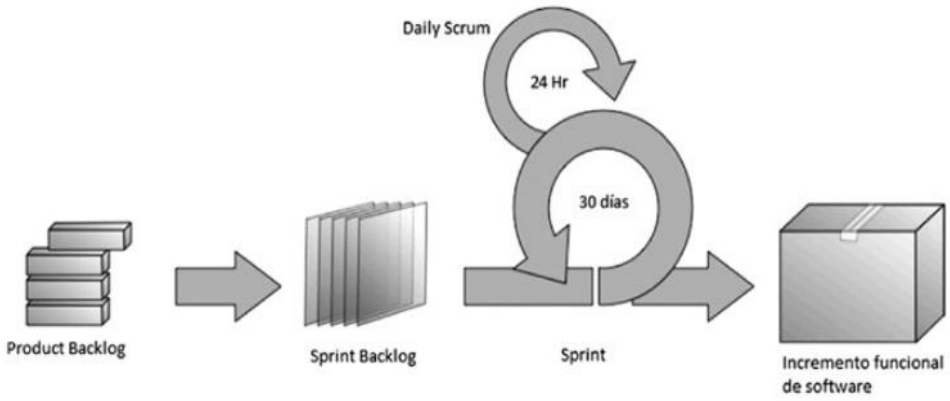
Dima y Maassen (2018) comentan que, la metodología cascada es uno de los métodos tradiciones, en pocas palabras un método para el desarrollo de software el cual se impulsa en una planificación incluyendo todas las etapas del ciclo de vida de modelo incremental, se utiliza comúnmente en grandes proyectos, pero su principal inconveniente es que resulta complicado implementar nuevos cambios durante su etapa de desarrollo. Según Galvan et al. (2021) comentan que, como una metodología ágil se tiene a Scrum que es un proceso de desarrollo de software donde los miembros del equipo participan de forma colaborativa, se consideran a 3 principales actores, el scrum master, el scrum team y el product owner, scrum tiene buenas prácticas agiles que buscan aprovechar al máximo los recursos para evitar tareas que no tengan valor; el tiempo de entrega de los sprint suele ser de 2 a 4 semanas. Del mismo modo, Poscoba (2020) resalta que, para todo proyecto ágil de desarrollo este se basa en las personas que trabajan para brindar un buen producto software y de esta forma lograr satisfacer las exigencias del cliente, de la misma manera estos pueden lidiar con rápidos cambios dentro del proyecto a solicitud del usuario. Para Saeedi y Visvizi (2021) sugieren que, las buenas prácticas de desarrollo deben ser aplicados tanto en metodologías tradicionales como en metodologías agiles, aunque resalta que más de un 70% de

las organizaciones se adaptan a las metodologías agiles, para la entrega de sus productos software.

Para Navarro et al. (2013) indican que, una de las metodologías agiles más representativas es scrum, siendo este un marco de trabajo que permite colaborar eficazmente entre los equipos de trabajo, por ejemplo definen roles para lograr un correcto funcionamiento, con scrum se puede tener una mejor visibilidad del proceso de trabajo ya que permite adaptarse minimizando los impactos; el marco de trabajo SCRUM trabaja con interacciones donde se van agregando funcionalidades de acuerdo a los requerimientos los cuales son llamados sprints, y cada sprint es un proyecto independiente, cabe resaltar que un sprint puede durar entre 15 a 30 días, dentro de esto se manejan 4 ceremonias importantes: (a) sprint planning donde se crean las tareas del trabajo que se realizara durante el Sprint y también se realiza la estimación de esfuerzo que requerirá cada tarea; (b) daily scrum donde el equipo le dedica 15 minutos diarios para indicar lo que hizo, lo que hará y que obstáculos tiene para que estos puedan ser resueltos; (c) sprint review esta ceremonia se da al final de cada sprint, aquí el PO revisa lo que se logró durante el sprint y discute con el equipo los problemas que tuvieron durante el desarrollo y la forma en la que fueron resueltos; (d) la retrospectiva donde analiza la comunicación con el equipo, qué hicieron bien, qué hicieron mal y qué podrían mejorar en los siguientes sprints. En la figura 4, se contemplan las fases de un sprint.

Figura 4

Metodología Scrum, fases de un sprint

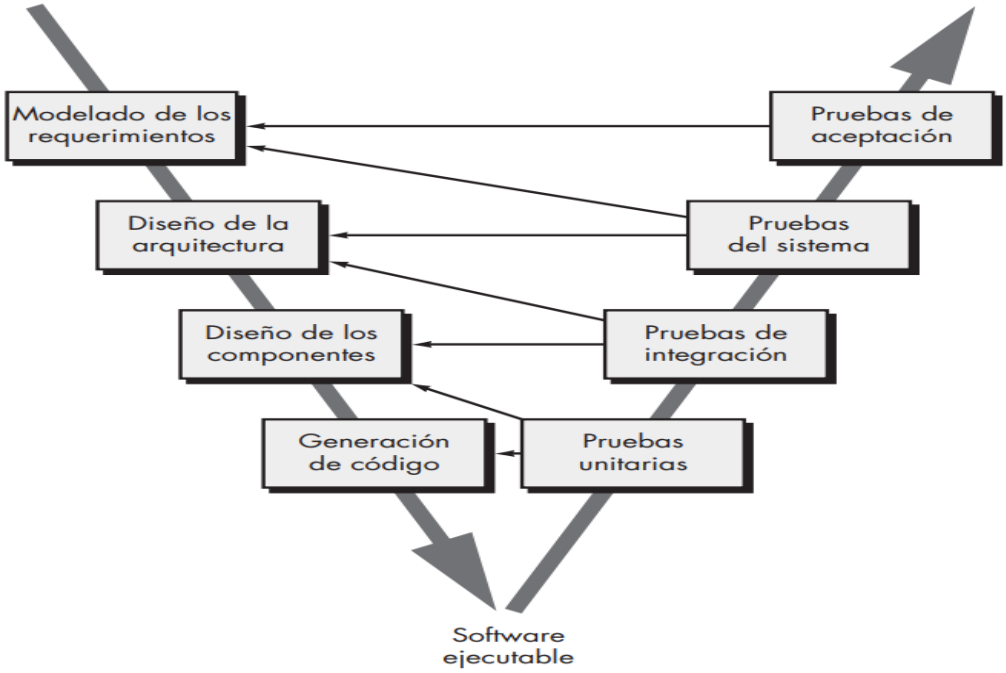


Nota: Navarro et al. (2013).

Por otro lado tenemos a Pressman (2010), quien comenta acerca de otra metodología de desarrollo de software que es el modelo cascada, siendo este el ciclo de vida clásico, donde se describe una forma secuencial de desarrollo de software, empezando por la comunicación (se toman los requerimientos funcionales); planeación (se estima el tiempo de desarrollo); modelado(se realiza el análisis y diseño); construcción (etapa de desarrollo del código y pruebas del software); despliegue(entrega del producto final; Por otra parte se tiene al Modelo V, donde se da la relación entre las acciones que realizan durante cada etapa con el aseguramiento de la calidad, en la figura 5, se puede ver el modelo v donde por cada etapa de desarrollo se ejecutan una serie de pruebas para asegurar la calidad del software. Cuando se habla del modelo cascada nos referimos a uno de los paradigmas más antiguos de la ingeniería del software, pero también se tienen ciertos problemas con la mencionada metodología como por ejemplo, cuando se ingresa algún cambio imprevisto este puede generar confusión en el desarrollo; a menudo cuando el cliente describe sus necesidades no abarca todo en general y a medida que avanza quiere incluir nuevos cambios; otro de los inconvenientes es que el usuario no puede probar el software hasta que este se encuentre terminado, entonces si al final se encuentra un error esto traería consecuencias desastrosas.

El proceso de desarrollo de software es vital en la estructuración del desarrollo, ya que se tienen actividades secuenciales que son cumplidas para alcanzar su objetivo que es la entrega del producto software. En la actualidad se tiene una variedad de marcos de trabajo que van evolucionando a través del tiempo siendo accesible para todas las organizaciones, cada metodología se puede usar acorde a cómo se acomode al tipo de proyecto que se desarrolle, esto ya dependerá del equipo de trabajo.

Figura 5
Modelo en V



Nota: Pressman (2010).

Para Black (2020), para el proceso de desarrollo de software es vital apoyarse de las métricas ya que ayudan a demostrar de qué forma se desempeña el equipo y el producto software, así que, para tener una mejor medición del desempeño, las organizaciones trabajan bajo KPIs que permiten conocer el grado de productividad de los miembros del equipo de desarrollo, el rendimiento del software, la calidad del software y el agrado del usuario final.

Como primera dimensión se considera a la productividad del desarrollador donde Black (2020) comenta que, las métricas de productividad son una forma de medir la eficiencia del equipo de trabajo, gracias a esto los ejecutivos pueden cumplir de manera más eficiente con los proyectos de software, pueden conocer cuál es el grado de productividad que tiene un desarrollador y el tiempo que le toma desarrollar el entregable. Como característica se tiene a la velocidad ágil, donde se puede calcular la velocidad del desarrollador por medio de los puntos de historia que realizó durante el sprint.

Para la segunda dimensión se consideran los defectos, Black (2020) resalta que, gracias a esta métrica se tiene un nivel de comprensión del porqué fallan las aplicaciones para que en el futuro se puedan construir de mejor manera, como característica de evaluación se tiene al porcentaje de detección de defectos, para esto se evalúan los defectos que se encontraron durante la etapa de pruebas, contra un total de defectos que encuentra el usuario final al hacer uso de la aplicación.

Como tercera dimensión se tiene al rendimiento del software donde Black (2020) indica que, se refiere al grado en el que el software se comporta, es decir cómo se está desempeñando la aplicación que se desarrolló, de manera que cumpla con las especificaciones técnicas como el tiempo de funcionalidad del sistema para validar si cumple con las horas esperadas. Po otro lado Garcia (2022) comenta que, para calcular la efectividad se debe considerar el total de horas disponibles del sistema menos el total de horas de parada, debido a que el software se encontraba en mantenimiento no programado, dividiendo este sobre el total de horas disponibles.

III. METODOLOGÍA

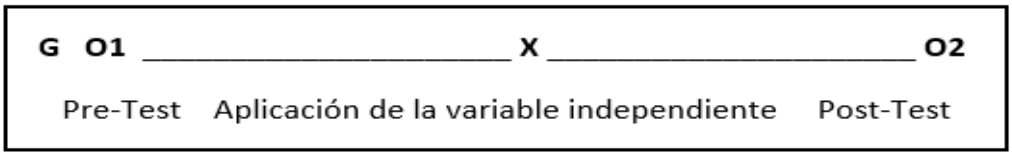
3.1. Tipo y diseño de investigación

Tipo de investigación: La presente investigación fue de tipo aplicada, ya que permitirá resolver el problema planteado en la investigación, consolidando el conocimiento para poder aplicarlo en el proceso del desarrollo de software. Según Martinez y Gonzales (2014), el tipo de investigación aplicada es todo el entendimiento que se obtenga durante la investigación y esto servirá para futuros problemas que se presenten.

Diseño de investigación: El diseño de investigación fue pre-experimental, donde Hernandez et al. (2014) mencionan que; se trabaja con 2 grupos en los cuales al grupo 1 no se aplica ninguna alteración siendo el pre-test, y el grupo 2 donde se estima después de que se haya aplicado la propuesta siento este el pos-test. Cuando se tiene el pre-test se tendrá como resultado O1 y con el pos-test será el O2.

Para el presente trabajo de investigación se usó del pre-test O1 y del pos-test O2, una vez que se haya aplicado la variable independiente el ISO 25000 sobre la variable dependiente tendremos como resultado el pos-test, y gracias a esto obtendremos los resultados mediante la comparación del pre-test.

Figura 6
Diseño pre-experimental



Enfoque: Cuantitativo

Se trabajó con el método cuantitativo ya que la investigación es de tipo pre-experimental y se podrá comprobar la hipótesis gracias al análisis estadístico. Según Hernandez et al. (2014) los estudios cuantitativos son analizados por medio

de las hipótesis y las bases teóricas, para que se acoplen los resultados en base a los conocimientos previos.

Nivel: El nivel fue explicativo debido a que se busca explicar la influencia de la norma ISO 25000 durante el desarrollo de software, el método elegido fue hipotético deductivo donde se determinara si la hipótesis establecida es verdadera o falsa.

3.2. Variables y operacionalización

Variable Independiente: ISO 25000

Definición conceptual: La norma ISO 25000 se especializa en la evaluación de productos Software, el objetivo principal de la ISO 25000 es brindar un panorama en general de los contenidos del SQuaRE, además de centrarse en 2 principales procesos las cuales son la especificación de requisitos de calidad de software y la evaluación de la calidad del software, las cuales son soportadas en el proceso de medición de calidad de software (Crespo, 2018).

Definición operacional: La norma ISO surge con la necesidad que un producto pueda ser eficaz, de confianza, eficiente y con garantía de seguridad, para que pueda satisfacer las exigencias del cliente. Para esta norma se tiene 8 características que son calidad de uso, mantenibilidad, eficiencia, funcionalidad, usabilidad, fiabilidad, portabilidad y seguridad, esto ayudara a garantizar la calidad del software demostrando que el producto es de total confianza.

Variable dependiente: Proceso de Desarrollo de software

Definición conceptual: El proceso de desarrollo de software incluye un conjunto de 5 actividades de un proceso en general: comunicación para entender los objetivos y definir los requerimientos y funciones del software; durante la planeación se definen las especificaciones técnicas que se realizara, probables riesgos, recursos a utilizar, y a definir el programa de actividades; modelado con la finalidad de entender los requerimientos en necesario realizar un bosquejo para entender el

panorama total del problema un la solución; la construcción en esta actividad se realizará la generación del código y las pruebas para descartar los errores, y si existen solucionarlo antes de su puesta en producción; despliegue aquí ya se hace la entrega del software funcionando para que el usuario pueda usarlo y evaluarlo de acuerdo a sus necesidades (Pressman, 2010).

Definición operacional: Para el proceso de desarrollo de software es vital apoyarse de las métricas ya que ayudan a demostrar de qué forma se desempeña el equipo y el producto software, así que, para tener una mejor medición del desempeño, las organizaciones trabajan bajo KPIs que permiten conocer el rendimiento de los miembros del equipo de desarrollo, el rendimiento del software, la calidad del software y la satisfacción del usuario final (Black, 2020).

3.3. Población, muestra y muestreo

Población: Guffante et al. (2016) indican que, al hablar de población se está haciendo referencia al total de los usuarios que interfieren o participan durante la investigación. Según Hernandez et al. (2014), la población debe definirse de acuerdo con las propiedades de la investigación, ya que esta será estudiada para poder ser analizada.

Por lo tanto, para la población se consideraron 50 proyectos de la financiera de una duración de 4 a 6 sprints (1 a 2 meses) considerando desde la etapa de planificación hasta la conformidad del usuario final.

Muestra: Para Muñoz (2015), la muestra es parte de la población, la cual es elegida para conseguir información acerca de las variables de estudio. Para Hernandez et al. (2014), la muestra llega a ser una parte o un sub-grupo de la población, para los tipos de muestra se tiene al muestreo probabilístico donde la parte seleccionada de la población tiene el mismo nivel de probabilidad para ser elegidos y el muestreo no probabilístico donde la extracción de elementos depende específicamente de la peculiaridad que tiene la investigación.

Dado que en la investigación la población tiene una cantidad establecida de proyectos de desarrollo es accesible, por lo tanto, se tomó los 50 proyectos de desarrollo de software como la población total.

3.4. Técnicas e instrumentos de recolección de datos

Técnica: Para Fidias (2014), cuando se aplica la técnica esta permite recolectar datos que podrán ser analizados de acuerdo a las necesidades del investigador.

Gracias a la técnica se puede estructurar los datos que se van obteniendo, en síntesis, la técnica es una necesidad para la investigación ya que de esa forma de obtiene la información que permitirá alcanzar el objetivo propuesto.

Fichaje: Para Martinez y Gonzales (2014), el fichaje permite recabar información para que esta pueda ser utilizada durante la investigación, se realiza el fichaje en base a la orientación de la investigación.

El fichaje ayuda a estudiar datos sin tener la necesidad de ser modificado para poder recolectar información basándose en la explotación de la información obtenida.

Instrumento: Según Fidias (2014), se usa un instrumento cuando se trabaja con recursos digitales o físicos para poder dar por escrita la información, por otro lado, Arias (2016) considera que, las fichas de registro con instrumentos que ayudan a recolectar los datos utilizados para almacenar, archivar y anotar la información.

El objeto que se empleó para esta exploración es la ficha de registro y la técnica es el fichaje. Se ha desarrollado una ficha por cada indicador.

3.5. Procedimientos

Para el trabajo de investigación se consideraron los próximos puntos, en primer lugar se definió la problemática de la investigación, después se continuo con la búsqueda de la información para los antecedentes, por medio de artículos que

ayudaron a describir las problemáticas y soluciones en distintos proyectos de investigación, por otro lado se buscó información sobre las variables de la investigación las cuales ayudaron en la identificación de los problemas, objetivo se hipótesis de la investigación, después de ello se continuo con la población y las técnicas de la investigación.

Luego se continuo con la elaboración de los instrumentos que ayudaran en la recolección de datos, las cuales fueron las fichas de registro, de acuerdo con los indicadores que fueron seleccionados en torno a las dimensiones, para recolectar los datos se solicitó apoyo de la financiera con la que se trabaja, así mismo, se obtuvo la información del estado actual de los proyectos en el pre-test, y luego de aplicar la ISO 25000 se recolectaron los datos para el pos-test, para luego continuar con el estudio de los datos.

Durante el análisis de datos se trabajó con el aplicativo IBM SPSS Statistics 25.0; una vez que se obtuvieron los datos se interpretaron las hipótesis formuladas en la investigación.

3.6. Método de análisis de datos

Se trabajo con un enfoque cuantitativo debido a que la investigación es pre-experimental de manera que se pueda comprobar la hipótesis por medio de estudios estadísticos. Según Hernández et al. (2014), los estudios que son cuantitativos pueden ser analizados por medio de la hipótesis y las bases teóricas que brindan enseñanza para poder acoplar los conocimientos previos.

3.7. Aspectos éticos

La investigación en estudio muestra datos que son de autoría propia, por medio de la recopilación y análisis de datos que fueron recolectados, habiendo desarrollado nuevos conocimientos que ayudaran en futuras investigaciones.

Para las referencias bibliográficas, citas, encabezados, fueron hechos en base a las normas American Psychological Association (APA) donde se sigue un

conjunto de normas y estructuras que ayudan a la presentación del trabajo de investigación en los proyectos de grado, del mismo modo durante el trabajo de investigación se trabajó con la herramienta del turnitin que permite identificar casos de plagio, de manera que permite asegurar la integridad académica al momento de realizar el proyecto de investigación.

Para la evaluación de originalidad y como guía para la elaboración del proyecto se basó en la resolución del Vicerrectorado de Investigación N°110-2022-VI-UCV. Se siguieron lineamientos referidos por la Universidad Cesar Vallejo dispuesto en la Resolución del Consejo Universitario N°0200-2018/UCV.

IV. RESULTADOS

Análisis descriptivo

Estadístico descriptivo para el indicador 1 productividad del desarrollador

En la tabla 1, se observa el resultado del indicador productividad del desarrollador, donde la media obtenida en el pos-test después de la aplicación de la ISO 25000 fue de 94% y la media del pre-test antes de la aplicación de la ISO 25000 fue del 44%, por lo tanto, se tuvo una mejora en el porcentaje de productividad del desarrollador con una variación del 50% en su valor medio, en otras palabras al implementar la ISO 25000 se cumple uno de los objetivos propuestos en el sprint, el cual es terminar las historias de usuario planificadas en el Sprint en curso.

Tabla 1

Resultado descriptivo estadístico del indicador productividad del desarrollador

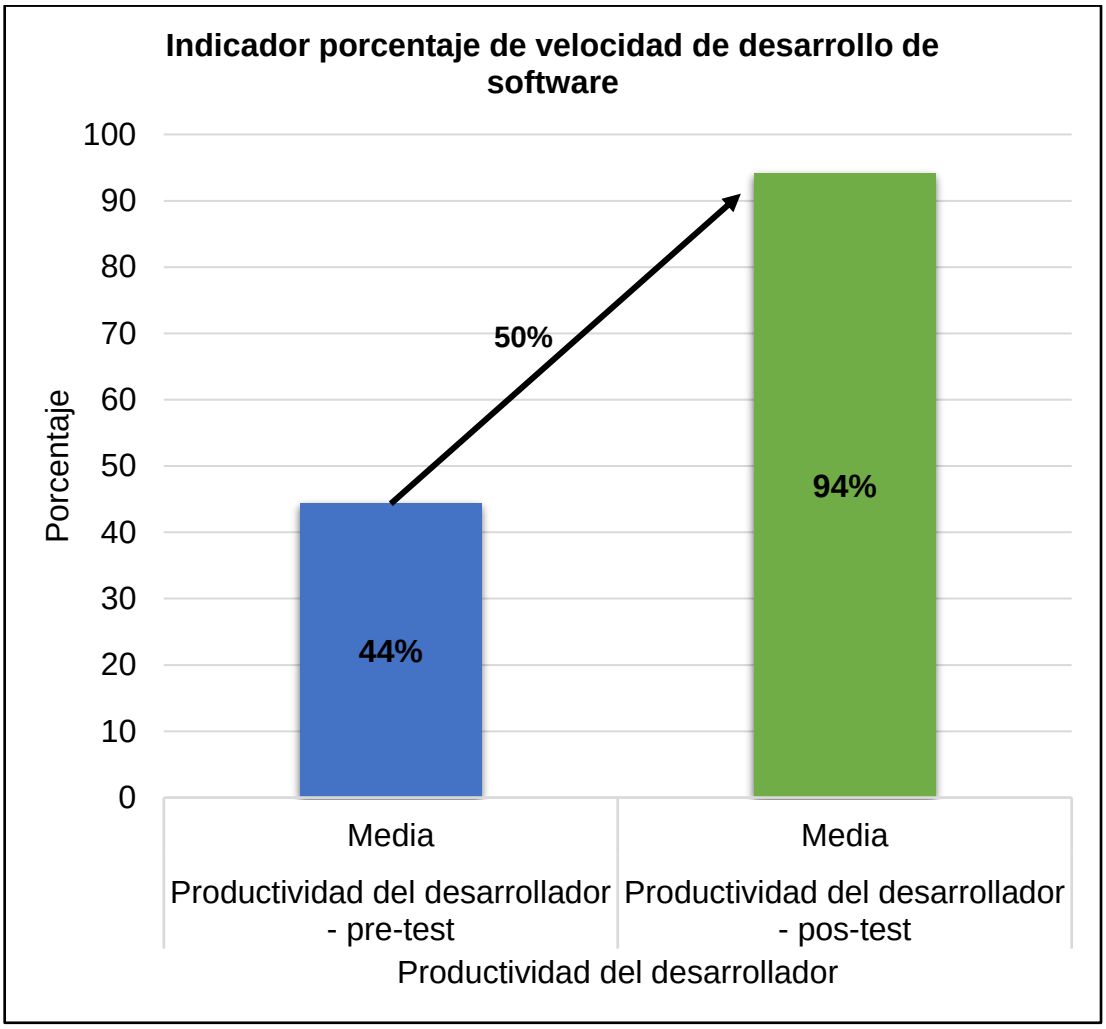
	N	Mínimo	Máximo	Media	Desv. Desviación
Productividad del desarrollador - pre-test	50	0	100	44.42	32.138
Productividad del desarrollador - pos-test	50	50	100	94.18	12.678
N válido (por lista)	50				

Nota. Base de datos procesado en SPSS Statistics versión 25

En la figura 7, se contempla de forma gráfica el comportamiento del indicador productividad del desarrollador en un antes y después de haber aplicado la ISO 25000, se aprecia que existe un incremente de velocidad durante el desarrollo de software.

Figura 7

Medias comparativas del indicador productividad del desarrollador



Estadístico descriptivo para el indicador 2 defectos del software

En la tabla 2, se analizó el resultado del indicador defectos del software, donde la media obtenida en un pos-test después de la aplicación del ISO 2500 fue de 1% y la media del pre-test antes de la aplicación del ISO 25000 fue del 10%, por lo tanto, se tuvo una disminución considerable en el indicador defectos del software con una variación del 9% en su valor medio, dicho de otro modo al aplicar la ISO 25000 se pudo realizar un mejor análisis de los casos de prueba que permiten detectar defectos desde una etapa temprana, par que al momento de entregar el producto al usuario final no se tengan defectos que impidan su uso y su estabilidad operativa.

Tabla 2

Resultado descriptivo estadístico del indicador defectos del software

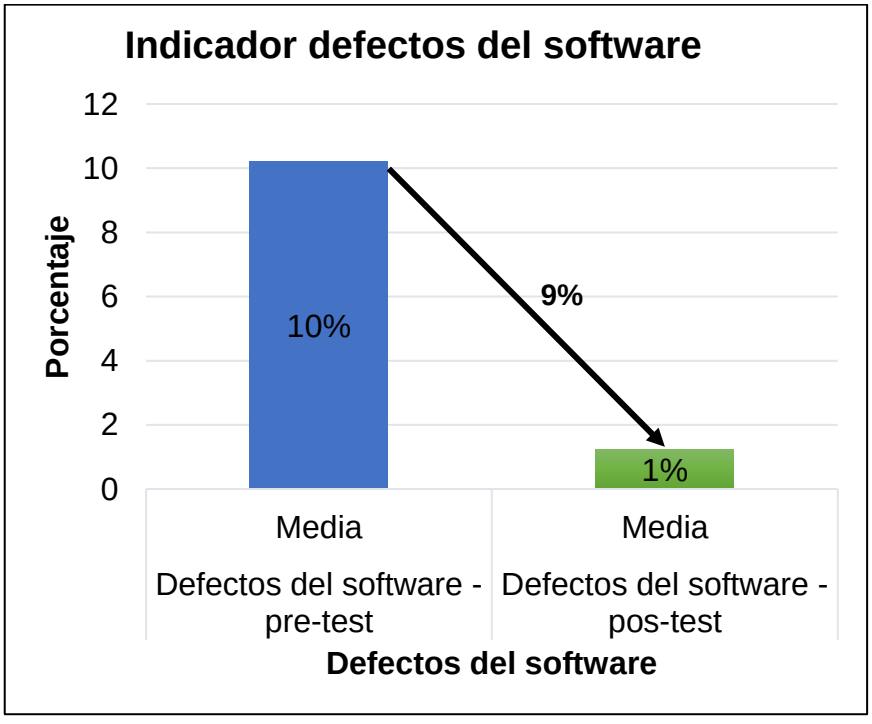
	N	Mínimo	Máximo	Media	Desv. Desviación
Defectos del software - pre-test	50	0	50	10.22	10.465
Defectos del software - pos-test	50	0	17	1.22	4.210
N válido (por lista)	50				

Nota. Base de datos procesado en SPSS Statistics versión 25

En la figura 8, se contempla de forma gráfica el comportamiento del indicador defectos del software en un antes y después de haber aplicado el ISO 25000, se aprecia que existe un bajo porcentaje de defectos después de haber aplicado la ISO 25000.

Figura 8

Medias comparativas del indicador defectos del software



Estadístico descriptivo para el indicador 3 rendimiento del software

En la tabla 3, se contempla el resultado del indicador rendimiento del software, la media obtenida en un pos-test después de la aplicación del ISO 2500 fue de 92% y la media del pre-test antes de la aplicación del ISO 25000 fue del 46%, por lo tanto, se tuvo una mejora en el rendimiento del software con una variación del 46% en su valor medio, en otras palabras, al implementar la ISO 25000 se cumple uno de los objetivos propuestos de manera trimestral el cual es mejorar el tiempo de respuesta de aplicativo y el grado de satisfacción del usuario final.

Tabla 3

Resultado descriptivo estadístico del indicador del rendimiento del software

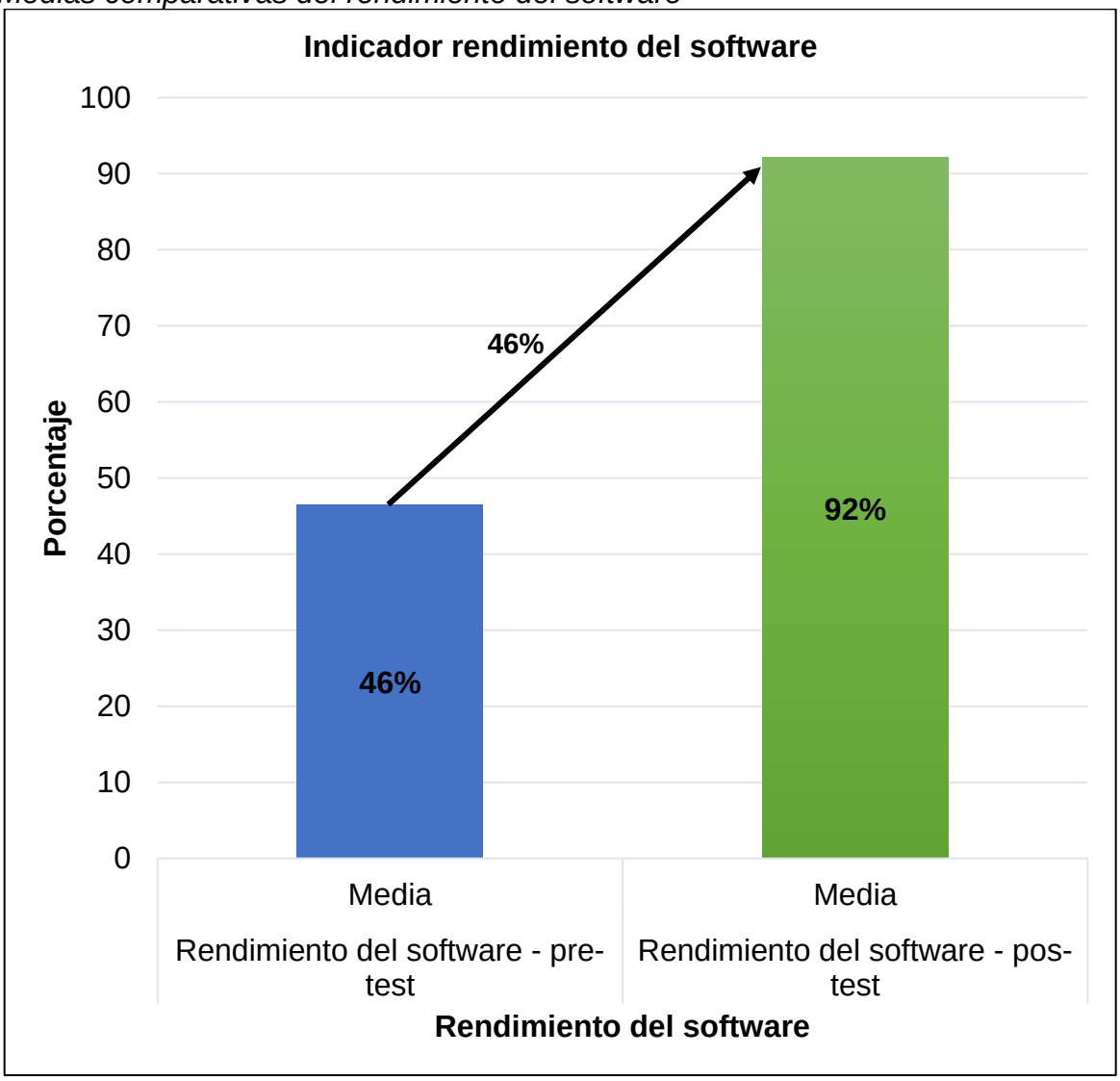
	N	Mínimo	Máximo	Media	Desv. Desviación
Rendimiento del software - pre-test	50	0	83	46.46	13.882
Rendimiento del software - pos-test	50	83	100	92.08	4.720
N válido (por lista)	50				

Nota. Base de datos procesado en SPSS Statistics versión 25

En la figura 9, se contempla de forma gráfica el comportamiento del indicador rendimiento del software en un antes y después de haber aplicado el ISO 25000, se considera que existe un aumento de fiabilidad en el proceso de desarrollo de software. Cabe resaltar que al tener un buen rendimiento del producto software se logra tener un mejor grado de fiabilidad del aplicativo.

Figura 9

Medias comparativas del rendimiento del software



Pruebas de normalidad

Posteriormente, se efectuaron las pruebas de normalidad para los indicadores de la ISO 25000 y el proceso de desarrollo de software a través del método Kolmogorov y Smirnov, a causa de que el tamaño de nuestra muestra estratificada está constituido por 50 proyectos de desarrollo. La prueba se efectuó al ingresar los datos de cada indicador en el software estadístico SPSS 25.0, para un nivel de confiabilidad del 95%, bajo las siguientes condiciones.

Si:

Sig. < 0.05 adopta una distribución no normal.

Sig. ≥ 0.05 adopta una distribución normal.

Indicador 1: Productividad del desarrollador

En la tabla 4, se detallaron los resultados de la prueba mostrando que el Sig. en el indicador productividad del desarrollador para el pre-test estuvo en 0.021, con un valor inferior que 0.05; por el cual, se demuestra una distribución no normal y los datos son no paramétricos. Los resultados del pos-test muestran que el sig. del indicador productividad del desarrollador fue de 0.000, donde el valor es menor que 0.05, por lo tanto, se demuestra una distribución no normal con datos no paramétricos.

Tabla 4
Prueba de normalidad del indicador productividad del desarrollador

Pruebas de normalidad

	Kolmogorov-Smirnov[a]		
	Estadístico	gl	Sig.
Productividad del desarrollador - pre-test	0.137	50	0.021
Productividad del desarrollador - pos-test	0.477	50	0.000

Nota. Base de datos procesado en SPSS Statistics versión 25

Indicador 2: Defectos del software

En la tabla 5, se detallaron los resultados de la prueba mostrando que el Sig. en el indicador defectos del software para el pre-test estuvo en 0.000, con un valor inferior que 0.05; por el cual, se demuestra una distribución no normal y los datos son no paramétricos. Los resultados del pos-test muestran que el sig. del indicador defectos del software fue de 0.000, donde el valor es menor que 0.05, por lo tanto, se demuestra una distribución no normal con datos no paramétricos.

Tabla 5
Prueba de normalidad del indicador defectos del software

Pruebas de normalidad

	Kolmogorov-Smirnov[a]		
	Estadístico	gl	Sig.
Defectos del software - pre-test	0.216	50	0.000
Defectos del software - pos-test	0.534	50	0.000

Nota. Base de datos procesado en SPSS Statistics versión 25

Indicador 3: Rendimiento del software

En la tabla 6, se detallaron los resultados de la prueba mostrando que el Sig. en el indicador rendimiento del software para el pre-test estuvo en 0.000, con un valor inferior que 0.05; por el cual, se demuestra una distribución no normal y los datos son no paramétricos. Los resultados del pos-test indica que el Sig. del indicador rendimiento del software fue de 0.000, con un valor inferior que 0.05; por el cual, se demuestra una distribución no normal y los datos son no paramétricos.

Tabla 6
Prueba de normalidad del indicador rendimiento del software

Pruebas de normalidad

	Kolmogorov-Smirnov[a]		
	Estadístico	gl	Sig.
Rendimiento del software - pre-test	0.214	50	0.000
Rendimiento del software - pos-test	0.287	50	0.000

Nota. Base de datos procesado en SPSS Statistics versión 25

Contrastación de hipótesis

Para entender la influencia de la ISO 25000 dentro del proceso de desarrollo de software se realizó la contrastación de hipótesis el cual se muestra a continuación.

Prueba de hipótesis de indicador 1: Productividad del desarrollador

Hipótesis nula (Ho): El ISO 25000 NO influye en la productividad del desarrollador en el área TI de una financiera.

Hipótesis alterna (H1): El ISO 25000 influye en la productividad del desarrollador en el área TI de una financiera.

Nivel de significancia:

Nivel de significancia o confianza = 0,05

Si p (valor)>= 0,05 no se rechaza la hipótesis nula

Si p (valor)< 0,05 no se rechaza la hipótesis alterna

Para la contrastación de hipótesis se hizo el análisis con estadística no paramétrica de muestras relacionadas, y para eso se utilizó la prueba de rangos de Wilcoxon. Seguidamente, se evidencian los resultados en las tablas 7 y 8.

Tabla 7

Valores de rango para el indicador

Rangos

		N	Rango promedio	Suma de rangos
Productividad del desarrollador - pos-test - Productividad del desarrollador - pre-test	Rangos negativos	1a	7,00	7,00
	Rangos positivos	43b	22,86	983,00
	Empates	6c		
	Total	50		

a. Productividad del desarrollador - pos-test < Productividad del desarrollador - pre-test

b. Productividad del desarrollador - pos-test > Productividad del desarrollador - pre-test

c. Productividad del desarrollador - pos-test = Productividad del desarrollador - pre-test

Nota. Base de datos procesado en SPSS Statistics versión 25

En la tabla 7, se verifican que los valores del rango y suma son significativos, al presentar un rango promedio de 22,86 y suma de rangos con un valor de 983,00; adicional se cuenta con un rango positivo de 43b el cual demuestra la opción “b.

productividad del desarrollador - pos-test > Productividad del desarrollador - pre-test”

Tabla 8

Estadístico de prueba de Wilcoxon del indicador productividad del desarrollador

Estadísticos de prueba[a]

	Productividad del desarrollador - Pos-test - Productividad del desarrollador - pre-test
Z	-5,713[b]
Sig. asintótica(bilateral)	0.000

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos negativos.

Nota. Base de datos procesado en SPSS Statistics versión 25

Según los datos mostrados en la tabla 8, se rechaza la hipótesis nula, aceptando la hipótesis alterna con un 95% de confianza, debido a que el valor Z obtenido se valoró en -5,713 donde el sig. fue de 0.000, entonces no supero el 5% del margen de error por el cual se rechaza la hipótesis nula. Por lo tanto, El ISO 25000 influye en la productividad del desarrollador en el área TI de una financiera.

Prueba de hipótesis de indicador 2: Defectos del software

Hipótesis nula (Ho): El ISO 25000 NO influye en los defectos del software en el área TI de una financiera.

Hipótesis alterna (H1): El ISO 25000 influye en los defectos del software en el área TI de una financiera.

Nivel de significancia:

Nivel de significancia o confianza = 0,05

Si p (valor)>= 0,05 no se rechaza la hipótesis nula

Si p (valor)< 0,05 no se rechaza la hipótesis alterna

Para la contrastación de hipótesis se hizo el análisis con estadística no paramétrica de muestras relacionadas, y para eso se utilizó la prueba de rangos de Wilcoxon. Seguidamente, se evidencian los resultados en las tablas 9 y 10.

Tabla 9

Valores de Rango para el indicador defectos del software

Rangos

		N	Rango promedio	Suma de rangos
Defectos del software - pos-test - Defectos del software - pre-test	Rangos negativos	30a	17,93	538,00
	Rangos positivos	4b	14,25	57,00
	Empates	16c		
	Total	50		

a. Defectos del software - pos-test < Defectos del software - pre-test

b. Defectos del software - pos-test > Defectos del software - pre-test

c. Defectos del software - pos-test = Defectos del software - pre-test

Nota. Base de datos procesado en SPSS Statistics versión 25

En la tabla 9, se verifican que los valores del rango y suma son significativos, al presentar un rango promedio de 14,25 y suma de rangos con el valor de 57,00; adicional se cuenta con un rango positivo de 4b el cual demuestra la opción “b. defectos del software - pos-test > defectos del software - pre-test”

Tabla 10

Estadístico de prueba de Wilcoxon del indicador defectos del software

Estadísticos de prueba[a]

	Defectos del software - pos-test - Defectos del software - pre-test
Z	-4,123[b]
Sig. asintótica(bilateral)	0.000

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos positivos.

Nota. Base de datos procesado en SPSS Statistics versión 25

Según los datos mostrados en la tabla 10, se rechaza la hipótesis nula, aceptando la hipótesis alterna con un 95% de confianza, debido a que el valor Z obtenido se valoró en -4,123 donde el sig. fue de 0.000, entonces no supero el 5% del margen de error por el cual se rechaza la hipótesis nula. Por lo tanto, El ISO 25000 influye en la cantidad de defectos del software en el área TI de una financiera.

Prueba de hipótesis de indicador 3: Rendimiento del software

Hipótesis Nula (Ho): El ISO 25000 NO influye en el rendimiento del software en el área TI de una financiera.

Hipótesis Alterna (H1): El ISO 25000 influye en el rendimiento del software en el área TI de una financiera.

Nivel de significancia:

Nivel de significancia o confianza = 0,05

Si p (valor)>= 0,05 no se rechaza la hipótesis nula

Si p (valor)< 0,05 no se rechaza la hipótesis alterna

Para la contrastación de hipótesis se hizo el análisis con estadística no paramétrica de muestras relacionadas y para eso se utilizó la prueba de rangos de Wilcoxon. Seguidamente, se evidencian los resultados en las tablas 11 y 12.

Tabla 11
Valores de rango para el indicador rendimiento del software

Rangos

		N	Rango promedio	Suma de rangos
Rendimiento del software - pos-test - Rendimiento del software - pre-test	Rangos negativos	0a	,00	,00
	Rangos positivos	50b	25,50	1275,00
	Empates	0c		
	Total	50		

a. Rendimiento del software - pos-test < rendimiento del software - pre-test

b. Rendimiento del software - pos-test > rendimiento del software - pre-test

c. Rendimiento del software - pos-test = rendimiento del software - pre-test

Nota. Base de datos procesado en SPSS Statistics versión 25

En la tabla 9, se verifican que los valores del rango y suma son significativos, al presentar un rango de 25,50 suma de 1275,00, adicional se cuenta con un rango positivo de 50b el cual demuestra la opción “b. rendimiento del software - pos-test > rendimiento del software - pre-test”.

Tabla 12
Estadístico de Prueba de Wilcoxon del indicador rendimiento del software

Estadísticos de pruebaa

	Rendimiento del software - pos-test - Rendimiento del software - pre-test
Z	-6,163b
Sig. asintótica(bilateral)	0.000

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos negativos.

Nota. Base de datos procesado en SPSS Statistics versión 25

Según los datos mostrados en la tabla 12, se rechaza la hipótesis nula, aceptando la hipótesis alterna con un 95% de confianza, debido a que el valor Z obtenido se valoró en -6,163 donde el sig. fue de 0.000, entonces no supero el 5% del margen de error por el cual se rechaza la hipótesis nula. Por lo tanto, El ISO 25000 influye en el rendimiento del software en el área TI de una financiera.

V. DISCUSIÓN

Este apartado muestra las discusiones de los hallazgos de este estudio, en contraste con los resultados y conclusiones de los diversos investigadores mencionados en el marco teórico del estudio, para ello, partiremos comentando los objetivos formulados en la investigación, como objetivo general de la investigación fue determinar de qué manera la aplicación del ISO 25000 ayudara a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023; con los resultados obtenidos de la investigación se buscó comprender el cambio en el proceso de desarrollo de software al momento de aplicar la variable independiente, estos resultados se obtuvieron con un antes y después de aplicar la ISO 25000. Así mismo, se plantearon objetivos específicos por cada dimensión formulada que en total fueron la productividad del desarrollador, defectos en el software y rendimiento del software; se analizaron los 3 resultados obtenidos por cada indicador concluyendo que el valor de significancia para cada uno de ellos fue menor al 0.05 por lo que se rechazó la hipótesis nula.

Para el estudio se determinó cómo influye la variable independiente ISO 25000 en la variable dependiente del proceso de desarrollo de software. En fundamento a los resultados obtenidos después de aplicar T de Wilcoxon con una significancia de 0.000 en los indicadores, se evidencia que se tiene una mejora después de la aplicación de la ISO 25000.

Con respecto al indicador 1 productividad del desarrollador tuvo una estadística no paramétrica, por lo tanto, se utilizó la prueba de rangos de Wilcoxon obteniendo un sig. Bilateral de 0.000 siendo menor < a 0.05 dado este resultado se rechazó la hipótesis nula y se demuestra con la evidencia suficiente de que se acepta la hipótesis alterna, del mismo modo, en la contratación de hipótesis en el pre-test de la muestra el porcentaje de la media obtenida fue de un 44%, con la aplicación del ISO 2500 en el post test este porcentaje incremento al 94%, en otros términos, el porcentaje de productividad del desarrollador aumento en un 50%. Para contrastar la hipótesis del indicador porcentaje de productividad del desarrollador el

nivel de significancia en el post test fue de 0.000 el cual es menor que el valor de 0.05.

Esta información es respaldada por el autor Carrasco (2022), donde se analizó el indicador nivel de comunicación con el área usuaria, el cual se relaciona con el indicador de porcentaje de productividad del desarrollador de la presente investigación, ya que al tener una buena comunicación esto mejora el grado de productividad, el trabajo de investigación fue de tipo pre-experimental, se tuvo 25 trabajadores que representaron a la población ya que eran los encargados de los proyectos de software, para el pre-test se obtuvo un resultado de 40.43%, mientras que, en el pos-test el resultado fue de 95.24%, asimismo, para el pre-test del indicador promedio de nivel de comunicación tuvo un nivel de significancia de 0.061 siendo mayor a 0.05 por el cual se tuvo un nivel de distribución normal, el pos-test del indicador promedio de nivel de comunicación también se analizó por Shapiro-Wilk tuvo un nivel de significancia de 0.000 el cual fue menor a 0.05 por el cual se tuvo un nivel de distribución no normal, aplicaron una prueba no paramétrica entre el pre-test y el pos-test, por lo tanto, se llegó a la conclusión de que al aplicar un prototipo de gestión de proyectos aumenta la comunicación con el equipo de desarrollo aumentando el promedio a un 95.24%, y durante la contrastación de hipótesis el resultado fue que se rechazó la hipótesis nula y se aceptó la hipótesis alterna. Como recomendación indica que es necesario implementar estrategias para alcanzar un buen nivel de comunicación con las personas involucradas en el desarrollo de software.

Para contrastar esta información también tenemos al autor Rojas (2021), quien tuvo el indicador tiempo de entrega del requerimiento en el cual se puede medir la productividad del desarrollador el cual es nuestro indicador en el presente estudio, el trabajo de investigación fue de tipo aplicada, con un diseño cuasiexperimental y con una población de 60 observaciones que fueron elegidas al azar. Para el indicador tiempo de entrega de requerimiento en el pre-test se tuvo un valor de 35.98 horas y en el pos-test fue de 31.42 horas donde se puede validar que se redujo el tiempo de entrega del producto en desarrollo, se tiene una diferencia de 4.56 horas luego de la aplicación de una metodología de gestión de

conocimiento, en las pruebas de normalidad se usó la prueba de Kolmogórov-Smirnov (KS), ya que las fichas observadas fueron mayor a 50, donde el valor de p fue de 0.001 demostrando que no presenta una distribución normal y que se debe trabajar con la prueba no paramétrica, en cuando a la prueba de hipótesis se usó la prueba Wilcoxon donde estadísticamente los resultados fueron w=-6.77; p<0.001, por lo tanto existía bastante evidencia para rechazar la hipótesis nula y aceptar la hipótesis alterna; por consiguiente recomienda que es necesario capacitar a los equipos de desarrollo de software con respecto a la toma de requisitos, para que de esta manera se puedan definir las actividades que lograran que se disminuyan el doble de horas de trabajo destinadas para una sola actividad, así mismo las historias de usuario deben ser intuitivas, los defectos reportados deben ser solucionadas con rapidez para mejorar los resultados con respecto al indicador de disminución de tiempos de entrega de requerimientos dentro del equipo de trabajo.

Con respecto al indicador 2 defectos del software donde se aplicó una estadística no paramétrica, por lo tanto, se aplicó el test de Wilcoxon obteniendo un sig. Bilateral de 0.000 siendo menor < a 0.05 dado este resultado se rechazó la hipótesis nula y se demuestra con la evidencia suficiente de que se acepta la hipótesis alterna, también en la contratación de hipótesis en el pre-test de la muestra el porcentaje de la media obtenida fue de un 10%, con la aplicación del ISO 25000 en el pos-test este porcentaje disminuyo al 1%, por lo tanto, se tuvo una disminución considerable en el porcentaje de detección de defectos con una variación del 9% en su valor medio. Para contrastar la hipótesis del indicador porcentaje de detección de defectos, el nivel de significancia en el post test fue de 0.000 el cual es menor que el valor de 0.05.

Para respaldar el trabajo de investigación se tiene como referencia a Espejo et al. (2016), donde se evaluó el indicador cantidad de defectos hallados durante el proceso de desarrollo de software con un tipo de investigación cuantitativo, donde se compararon 22 proyectos durante el pre y post test, cuyo resultado fue que en el grupo del pre-test se tenía una cantidad de 118 defectos y en el grupo del pos-test se obtuvo un total de 49 defectos, en tal caso se logró reducir la cantidad de 68

defectos en los proyectos de desarrollo de software. En el estudio se aplicó la prueba de t-student donde existe un nivel de significancia de 0.01 donde se concluyó que se logró reducir la cantidad de defectos en el proceso de desarrollo de software.

En cuanto a la similitud de nuestro indicador porcentaje de detección de defectos tenemos al autor Céspedes (2019), quien tuvo como indicador densidad del error, el tipo de estudio fue experimental-aplicada con un diseño pre-experimental y una muestra de 176 casos de prueba para el indicador densidad de error. Asimismo, en sus resultados tuvo en el pre-test un porcentaje de 83.55% y en el pos-test un total de 24.40% donde se demuestra que la densidad de errores disminuyó en un 59.15, para las pruebas de normalidad se aplicó el método Shapiro-Wilk ya que tenía un total de 20 registros, su resultado fue que el nivel de Sig. en el pre-test fue de 0.21 y en el pos-test fue de 0.145, por lo tanto se adoptó una distribución normal, para la contratación de hipótesis se realizó la prueba de T-Student donde el resultado fue de -28.912 ubicándose en la zona de rechazo con un nivel de confianza del 95% por lo tanto se rechazó la hipótesis nula se aceptó la hipótesis alterna. Como conclusión sostuvo que, el software disminuyo el error en un 59.15% en el proceso de asegurar la calidad de software.

En contraste con estos resultados tenemos al autor Caiza (2019), quien implemento métricas para poder analizar la calidad en sus proyectos de desarrollo de software tuvo un caso de estudio con el indicador de cobertura de pruebas el cual se relaciona con nuestro indicador porcentaje de detección de defectos. A todo esto, en el estudio del indicador se logró alcanzar un total de 11.9% del 12.5% de pruebas exitosas realizadas, el cual indica que el restante no se ha podido validar porque aún existen defectos que no han sido corregidos, además de que el tiempo para corregís los errores reportados en las pruebas lleva más tiempo de lo acordado debido a la complejidad de código en los programas que se implementan y por ello tuvieron como resultado un 6.26 de 10 con respecto a la calidad del producto software por los defectos que no fueron corregidos a un tiempo adecuado por la complejidad de código en los proyectos. Como recomendación sostuvo que se debe

elaborar un buen plan para el proceso de pruebas de software, tomando en cuenta que para esto implica un tiempo de análisis para la creación de casos de pruebas.

Con respecto al indicador 3, el rendimiento del software se obtuvo como resultado aplicar la estadística no paramétrica, por lo tanto, se aplicó el test de Wilcoxon obteniendo un sig. Bilateral de 0.000 siendo menor < a 0.05 dado este resultado se rechazó la hipótesis nula y se demuestra con la evidencia suficiente de que se acepta la hipótesis alterna, también en la contratación de hipótesis en el pre-test de la muestra, el porcentaje de la media obtenida fue de un 92%, con la aplicación del ISO 25000 en el pos-test este porcentaje disminuyo al 46%, por lo tanto, se tuvo una mejora en el porcentaje de Fiabilidad con una variación del 46% en su valor medio. Para contrastar la hipótesis del indicador Porcentaje de rendimiento del software el nivel de significancia en el post test fue de 0.000 el cual es inferior al valor de 0.05.

En contraste con estos resultados el autor Llactahuaman (2018), hizo la evaluación de un indicador que permite determinar la eficiencia del proyecto software para aceptar las funcionalidades, con un tipo de estudio de diseño pre-experimental, con una población de 12 proyectos de desarrollo de software, durante las pruebas de normalidad del indicador se obtuvo un nivel de significancia de 0.79 para el pre-test y para el pos-test un 0.077 por lo tanto, se adoptaría una distribución normal y una prueba paramétrica. Durante la contrastación de hipótesis el valor de significancia (Bilateral) fue menor a 0.05, por lo tanto, se rechazó la hipótesis nula y se aceptó la hipótesis alterna con una confianza al nivel de 95%. Se obtuvo como resultados en su pre-test un 53.75% de eficiencia y en el Post-test alcanzo un 94.63%, en el cual claramente se muestra que la eficiencia en el desarrollo de software aumento un 40.88% para los proyectos software de la empresa y estos resultados fueron gracias a la aplicación de una metodología ágil que permitió una mejora significativa para medir la eficiencia de los proyectos software de una empresa privada.

Para respaldar esta información el autor Luna (2017), en su investigación acerca del mejoramiento de procesos en el desarrollo de software logro optimizar el tiempo, recursos y tener una garantía con respecto a la calidad del producto, realizó un análisis con la finalidad de mejorar el rendimiento del software después del desarrollo del software en el área de TI, la metodología que se aplico fue mixta combinando estudios cualitativos y cuantitativos para el estudio tomaron como referencia 44 proyectos que habían finalizado y 47 proyectos que estaban siendo solicitados para poder aplicar un método de calidad en la etapa de desarrollo, al momento de aplicar las buenas prácticas el resultado fue satisfactorio ya que se logró enriquecer el proceso de desarrollo de software disminuyendo un total del 20% de horas que se retrasan durante el proceso de desarrollo, eliminando un 99% de errores en el código en conjunto con la estructura de sus datos, el autor destaca que al aplicar un modelo de calidad no tendrá fin ya que es necesario continuar estableciendo procesos que permitan que la organización alcance la calidad de sus productos software brindados.

VI. CONCLUSIONES

Primero: Se demostró que existe influencia de la ISO 25000 en el proceso de desarrollo de software en el área TI de una financiera, en los resultados descriptivos se evidencio que el porcentaje de productividad del desarrollador incremento en un 50%, el porcentaje de defectos del software redujo en un 9% y el porcentaje de rendimiento del software aumento en un 46%. En el transcurso del proceso de desarrollo de software es importante aplicar criterios de calidad que son aplicados en el ciclo de vida del producto software desde su planificación hasta su puesta en producción, al momento de aplicar la ISO 25000 se logró brindar un producto de calidad que cumpla con los criterios de aceptación para cada nuevo producto software.

Segundo: Se determinó que existe influencia de la ISO 25000 sobre la productividad del desarrollador en el área TI de una financiera, debido a que la prueba de hipótesis aplicando la prueba de Wilcoxon el cual se emplea en las muestras relacionadas no paramétricas, obteniendo un nivel de significancia p = 0.000 por el cual se rechazó la hipótesis nula, del mismo modo se demostró que se tuvo una mejora en el porcentaje de productividad del desarrollador con una variación del 50% en su valor medio, gracias a los resultados obtenidos al analizar el pos-test después de la aplicación del ISO 2500 fue de 94% y la media del pre-test antes de la aplicación del ISO 25000 fue del 44%. El desarrollador incremento el nivel de productividad con respecto a sus tareas por cada sprint, por lo tanto, se logró cerrar las historias de usuario en el tiempo comprometido.

Tercero: Se determinó que existe influencia de la ISO 25000 sobre los defectos del software en el área TI de una financiera, debido a que la prueba de hipótesis aplicando la prueba de Wilcoxon el cual se emplea en las muestras relacionadas no paramétricas, obteniendo un nivel de significancia p = 0.000 por el cual se rechazó la hipótesis nula, del mismo modo se demostró que se tuvo una disminución considerable en el porcentaje de detección de defectos con una variación del 9% en su valor medio, gracias a los resultados obtenidos al analizar

el pos-test después de la aplicación del ISO 2500 fue de 1% y la media del pre-test antes de la aplicación del ISO 25000 fue del 10%. Se analizaron los casos de prueba desde un inicio lo cual permitió que se identificaran los posibles riesgos durante el desarrollo y gracias a ello se logró reducir la cantidad de incidencias reportadas.

Cuarto: Se determinó que existe influencia de la ISO 25000 sobre el rendimiento del software en el área TI de una financiera, debido a que la prueba de hipótesis aplicando la prueba de Wilcoxon el cual se emplea en las muestras relacionadas no paramétricas, obteniendo un nivel de significancia p = 0.000 por el cual se rechazó la hipótesis nula, del mismo modo se demostró que se tuvo una mejora en el porcentaje de rendimiento del software con una variación del 46% en su valor medio, gracias a los resultados obtenidos al analizar el pos-test después de la aplicación del ISO 2500 fue de 92% y la media del pre-test antes de la aplicación del ISO 25000 fue del 46%. Se tuvieron resultados positivos por parte de los usuarios interesados ya que la aplicación funcionó de manera correcta y sin interrupciones.

VII. RECOMENDACIONES

Primero: Se recomienda al jefe de TI organizar todas las reuniones de planeamiento en compañía con el analista de calidad, para que este pueda analizar los posibles riesgos que se tendrá durante el desarrollo, del mismo modo podrá realizar el análisis de casos de prueba que se ejecutaran en el nuevo producto software.

Segundo: Se recomienda al jefe de TI medir el tiempo de desarrollo del programador para medir el porcentaje de productividad el cual incrementará el valor del trabajo y permitirá hacer seguimiento a las tareas que va desarrollando durante la duración del Sprint.

Tercero: Se recomienda al jefe de TI hacer pruebas tempranas durante el desarrollo del software para que se minimice la cantidad de defectos encontrados, así mismo es necesario que el equipo de calidad participe en la definición de requerimientos para que analice los posibles defectos que se pueden ir encontrando, así mismo es necesario tener un tiempo adecuado para definir los casos de prueba para poder abarcar todas las casuísticas necesarias, y así disminuir la cantidad de defectos dentro del software.

Cuarto: Se recomienda al jefe de TI que para mejorar el rendimiento del software es necesario que se defina bien el requerimiento y se sigan los estándares de desarrollo, del mismo modo para garantizar un buen uso de la aplicación se necesita que este pase por pruebas de performance para medir su capacidad de uso ante una gran carga de usuarios.

REFERENCIAS

Ali, A., Keung, J., E-Amin, F., & Abdullah. (2017). *SPIIMM: Toward a Model for Software Process Improvement Implementation and Management in Global Software Development. IEEE Access*. Obtenido de https://ieeexplore.ieee.org/abstract/document/7983363/authors#authors

Arcos, G., & Mauricio, D. (2020). T*he Influence of the Application of Agile Practices in Software Quality Based on ISO/IEC 25010 Standard. International Journal of Information Technologies and Systems Approach (IJITSA)*. Obtenido de https://www.igi-global.com/article/the-influence-of-the-application-of-agile-practices-in-software-quality-based-on-isoiec-25010-standard/252827

Arias, F. (2016). *El proyecto de investigacion* (6 ed.). Venezuela: Editorial episteme. Obtenido de https://abacoenred.com/wp-content/uploads/2019/02/El-proyecto-de-investigaci%C3%B3n-F.G.-Arias-2012-pdf-1.pdf

Azeem, M., Sang, J., Khan, A., E-Amin, A., & Nasrullah. (2017). I*mproving the Quality of Software Development Process by Introducing a New Methodology–AZ-Model. IEEE Access*. Obtenido de https://ieeexplore.ieee.org/abstract/document/8241771/authors#authors

Bautista, R., Guun, S., Velasquez, N., & Ninahualpa, G. (2018). *Software Quality Assessment Applied for theGovernmental Organizations using ISO/IEC 25000. International Conference on eDemocracy & eGovernment (ICEDEG),*. Obtenido de https://www.researchgate.net/publication/325637575_Software_Quality_Assessment_Applied_for_the_Governmental_Organizations_using_ISOIEC_25000

Behutiye, W., Karhapää, P., López, X., Martínez, S., Vollmer, A., Rodríguez, P., . . . Oivo, M. (2020). *Management of quality requirements in agile and rapid software development: A systematic mapping study*. *Information and Software Technology*. Obtenido de https://www.sciencedirect.com/science/article/abs/pii/S095058491930240X

Black, R. (2020). *23 métricas de desarrollo de software que monitorear hoy. computerweekly.es.* Obtenido de https://www.computerweekly.com/es/consejo/23-metricas-de-desarrollo-de-software-que-monitorear-hoy

Caiza, G. (2019). *Implementación de métricas para la evaluación del proceso de control de calidad en proyectos de desarrollo de software para la empresa Logiciel.* Quito. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/20009/1/CD-9449.pdf

Callapiña, Flores, & Huaycho. (2019). *Gestión de calidad en los proyectos de software y su relación con la gestión del riesgo operacional en los sistemas, utilizando la guía del pmbok sexta edición, en una entidad financiera privada en Lima, 2019. Lima, Peru*. Obtenido de file:///C:/Users/GINA/Downloads/Noel%20Callapi%C3%B1a_Selomit%20Flores_Regina%20Haycho_Trabajo%20de%20Investigacion_Maestria_2019%20(1).pdf

Carrasco, J. (2022). *Modelo de Gestión de Proyectos para mejorar el Proceso de Desarrollo de Software de la Corte Superior de Justicia de Lambayeque, 2021.* Trujillo. Obtenido de https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/85518/Carrasco_ZJK-SD.pdf?sequence=1&isAllowed=y

Carrizo, D., & Alfaro , A. (Marzo de 2018). *Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico. Scielo*. Obtenido de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052018000100114

Crespo, A. (18 de Abril de 2018). *ISO 25000: La calidad del producto software*. Obtenido de excentia: https://www.excentia.es/iso-25000#:~:text=Los%20requisitos%20de%20la%20ISO,ISO%202500n%3A%20gesti%C3%B3n%20de%20calidad

Dima, A., & Maassen, M. (2018). *Waterfall to Agile software: Development models in the IT sector. Journal of International Studies*. Obtenido de https://www.jois.eu/files/21_557_Dima.pdf

Espejo, A., Bayona, S., & Pastor, C. (2016). *Aseguramiento de la Calidad en el Proceso de Desarrollo de Software utilizando CMMI, TSP y PSP. risti, 77*. Obtenido de https://pdfs.semanticscholar.org/230c/177c4a620147c2f9dfbf9c1e9c9c7785e207.pdf

Falen, R. (2020). *Scrum en la gestión de proyectos de desarrollo de software en la empresa Innovatec, Magdalena del Mar.* Lima. Obtenido de https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/49080/Falen_RRS-SD.pdf?sequence=1&isAllowed=y

Fidias, G. (2014). *Introducción a la Metodología Científica.* Venezuela: Editorial Episteme. Obtenido de https://books.google.com.pe/books?id=W5n0BgAAQBAJ&dq=libros+de+tecnicas+e+instrumentos+de+investigacion+cientifica&source=gbs_navlinks_s

Flores, E. (2018). *Propuesta de un modelo de mejora de gestión para la calidad del software basado en el modelo de madurez y capacidad integrado (cmmi) en la división de sistemas - coordinacion transferencia tecnológica externa - desarrollo de software de la Uladech Católic.* Chimbote. Obtenido de https://repositorio.uladech.edu.pe/bitstream/handle/20.500.13032/4170/MONITORIZACION%20Y%20CONTROL%20DEL%20PROYECTO_GESTION%20DE%20REQUISITOS_FLORES_FLORES_EDER_RICHAR.pdf?sequence=1&isAllowed=y

Galvan, S., Mora, M., Laporte, C., & Duran, H. (2021). *Reconciliation of scrum and the project management process of the ISO/IEC 29110 standard-Entry profile—an experimental evaluation through usability measures. Software Quality Journal*. Obtenido de https://link.springer.com/article/10.1007/s11219-021-09552-3

Garcia, C., & Mazo, E. (2017). *Guía técnica para evaluación de software.* Colombia. Obtenido de https://dokumen.tips/documents/guia-tecnica-para-evaluacion-de-software-558b0df1ae752.html?page=1

Garcia, S. (16 de 10 de 2022). *IRIM*. Obtenido de Todo lo que necesita saber un Jefe de Mantenimiento del siglo XXI: http://renovetec.com/irim/14-revista-

irim-6/304-indicadores-de-disponibilidad#:~:text=Para%20calcularlo%2C%20es%20necesario%20obtener,tiempo%20total%20del%20periodo%20considerado.

Guffante, F., Chavez, P., & Guffante, T. (2016). *Investigación Científica, El Proyecto de Investigación.* Ecuador: Catedrática Universidad Nacional de Educacion. Obtenido de http://dspace.unach.edu.ec/bitstream/51000/342/3/Investigaci%C3%B3n%20cient%C3%ADfica_el%20proyecto%20de%20investigaci%C3%B3n.pdf

Hernandez, R., Fernandez , C., & Baptista, M. (2014). *Metodología de la Investigación. 6ta Edición.* (sexta ed.). Mexico: Editorial McGraw-Hill Interamericana S.A. Obtenido de https://periodicooficial.jalisco.gob.mx/sites/periodicooficial.jalisco.gob.mx/files/metodologia_de_la_investigacion_-_roberto_hernandez_sampieri.pdf

Hovorushchenko, T. (2018). *Methodology of Evaluating the Sufficiency of Information for Software Quality Assessment According to ISO 25010. Original Scientific Paper*. Obtenido de https://hrcak.srce.hr/file/298237

Hynninen, Kasurenin, Jussi, & Taipale. (2018). *Framework for Observing the Maintenance Needs, Runtime Metrics and the Overall Quality-in-Use*. *Kotka: Journal of Software Engineering and Applications*. Obtenido de https://www.scirp.org/journal/paperinformation.aspx?paperid=83411

Ibarra, S. (2018). *Desarrollo de una herramienta de soporte para el aseguramiento de la calidad en productos de software.* Zacatecas. Obtenido de https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/1041/1/ZAC%20TE%2066.pdf

ISO/IEC 25010. (2011). *Online Browsing Platform (OBP)*. Obtenido de https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

Jimenez, O. (2017). *Pruebas de calidad aplicadas al sitio web Allison.* Colombia. Obtenido de https://dspace.itcolima.edu.mx/bitstream/handle/123456789/725/OSCAR%20PAUL%20Pruebas%20de%20Calidad%20Aplicadas%20al%20Sitio%20

Web%20Allison.pdf;jsessionid=C4155CF3139BE6CCF0F5B19FD2097D2A?sequence=1

Karima Moumane, A., & Abran , A. (2016). *Usability evaluation of mobile applications using ISO 9241 and ISO 25062 standards. SpringerPlus*. Obtenido de https://link.springer.com/article/10.1186/s40064-016-2171-z#citeas

Kyoo. (2016). *Innovative Software Development and Project ManagementFramework for Technology Startups. Arvix*. Obtenido de https://arxiv.org/ftp/arxiv/papers/1708/1708.06900.pdf

Lindsjørn, Y., Sjøberg, , D., Dingsøyr, T., Bergersen, G., & Tore, D. (2016). *Teamwork quality and project success in software development: A survey of agile development teams. Journal of Systems and Software*. Obtenido de https://www.sciencedirect.com/science/article/pii/S016412121630187X#ack0001

Llactahuaman, L. (2018). *Aplicación de una metodología ágil para el desarrollo de proyectos en una empresa privada de software.* Lima. Obtenido de https://hdl.handle.net/20.500.12692/31461

Luna, W. (2017). *Mejoramiento de procesos, basado en el análisis de buenas prácticas. Caso: Área de Desarrollo de la Dirección de Informática de la PUCE.* Quito. Obtenido de https://repositorio.uasb.edu.ec/bitstream/10644/5541/1/T2229-MBA-Luna-Mejoramiento.pdf

Martinez, C., & Gonzales, A. (2014). *Tecnicas e instrumentos de recogida y analisis de datos.* https://books.google.com.pe/books?id=iiTHAwAAQBAJ&dq=investigacion+tipo+explicativa&source=gbs_navlinks_s.

Méndez, M. (2019). *DEC Asociación para el Desarrollo de la Experiencia de Cliente*. Obtenido de Experiencia de Cliente digital. Qué ofrece la norma ISO 25010: https://asociaciondec.org/blog-dec/experiencia-de-cliente-digital-que-ofrece-la-norma-iso-25010/40619/

Mendoza, X. (2018). *Aseguramiento de calidad basado en cmmi en los procesos de mantenimiento de software para la unidad de análisis financiero y económico.* Quito. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/20010/1/CD-9450.pdf

Moyo, S., & Mnkandla, E. (2020). *A Novel Lightweight Solo Software Development Methodology With Optimum Security Practices. IEEE Access*. Obtenido de https://ieeexplore.ieee.org/abstract/document/8978533

Muñoz , C. (2015). *Metodología de la investigación.* Mexico: Oxford University Press. Obtenido de https://books.google.com.pe/books?id=DflcDwAAQBAJ&dq=metodo+de+investigacion+cientifica&source=gbs_navlinks_s

Navarro, A., Fernández, J., & Morales, J. (2013). *Revisión de metodologías ágiles para el desarrollo de software*. Obtenido de file:///C:/Users/GINA/Downloads/6_Revisi%C3%B3n_de_metodolog%C3%ADas_%C3%A1giles_para_el_desarrollo_de_software.pdf

Ormeño, N. (15 de Mayo de 2019). *Medium*. Obtenido de ISO 25010 y el desarrollo de software: https://normeno.medium.com/iso-25010-y-el-desarrollo-de-software-112393a4b341

Owens, D., & Khazanchi, D. (2009). *Calificación de programas informáticos.* Pensilvania: IGI Global.

Perez, M. (2016). *Tools to Support the Assesment of the Quality Characteristics Based on ISO/IEC 25000.* Cuba. Obtenido de https://www.researchgate.net/publication/308994879_Tools_to_Support_the_Assessment_of_the_Quality_Characteristics_Based_on_ISOIEC_25000

Pócsová, J., Bednárová, D., Bogdanovská, G., & Mojžišová, A. (2020). *Implementation of Agile Methodologies in an Engineering Course*. *Education Sciences*. Obtenido de https://www.mdpi.com/2227-7102/10/11/333

Pressman, R. (2010). *Ingeniería del software. Unn enfoque practico.* Mexico: McGraw-Hill Interamericana Editores, S.A. Obtenido de

http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Rahman, A. (2020). *Quality Consideration for e-Learning System Based. Asian Centre of e Learning*. Obtenido de https://ceur-ws.org/Vol-2545/paper-03.pdf

Ramos Palacios, D. (2016). *Diseño de un modelo de evaluación de la calidad de productos de software, basado en métricas externas y usabilidad aplicado a un caso de estudio.* Quito. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/16668/1/CD-7271.pdf

Rashid, J., & Wasif, M. (2016). *How to Improve a Soware Quality. Journal of Computer Science IJCSIS.* Obtenido de https://abacoenred.com/wp-content/uploads/2019/02/El-proyecto-de-investigaci%C3%B3n-F.G.-Arias-2012-pdf-1.pdf

Reyes, L. (2021). *Evaluación de herramientas de software libre para la gestión de incidentes basado en ITIL, utilizando las normas de calidad ISO/IEC 25000.* Esmeraldas. Obtenido de https://repositorio.pucese.edu.ec/bitstream/123456789/2750/1/Reyes%20V%c3%a9lez%20Leonardo%20Gabriel.R.pdf

Ribeiro, E., & Soares, J. (2017). A *Method for Quality Evaluation of Supervision Software Using Fuzzy Concepts and the International Standard ISO/IEC 25000. J Control Autom Electr Syst*. Obtenido de https://doi.org/10.1007/s40313-017-0303-5

Saeedi, K., & Visvizi, A. (2021). *Software Development Methodologies, HEIs, and the Digital Economy.* Education Sciences. Obtenido de https://files.eric.ed.gov/fulltext/EJ1288479.pdf

Suárez, F., & Garzás, J. (2014). *I Jornada sobre Calidad del Producto Software e ISO 25000, Santiago de Compostela, 10 de junio de 2014.* España, Santiago de Compostela: 233 Grados de TI S.L. Obtenido de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://cpeig.gal/sites/default/files/libros/Libro+Jornadas+Galicia+Calidad+Software.pdf

Tavares, B., da Silva, C., & de Souza, A. (2017). *Risk management analysis in Scrum software projects. International Transactions in Operational Research*, https://doi.org/10.1111/itor.12401.

Vaca, T. (2017). *Modelo de calidad de software aplicado al módulo de talento humano del sistema informático integrado universitario – UTN.* Ibarra. Obtenido de http://repositorio.utn.edu.ec/bitstream/123456789/7457/1/PG%20533%20TESIS.pdf

Vargas, L. (2018). S*oftware Quality Methodology to Train Engineers as Evaluators of Information Systems Development Tool. Tamaulipas*. Obtenido de https://www.researchgate.net/publication/329372013_Software_Quality_Methodology_to_Train_Engineers_as_Evaluators_of_Information_Systems_Development_Tools

Xing, X., & Meng. (2019). *Software quality assessment model: a systematic mapping study. Beijing: Science China Information Sciences*. Obtenido de https://xin-xia.github.io/publication/scis181.pdf

ANEXOS

Anexo N°1: Matriz de operacionalización de variables.

VARIABLES DE ESTUDIO	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIÓN	INDICADORES	Técnica	Instrumento	Fórmula	ESCALA DE MEDICIÓN
V1: ISO 25000	La norma ISO 25000 se especializa en la evaluación de productos Software, el objetivo principal de la ISO 25000 es brindar un panorama en general de los contenidos del SQuaRE, además de centrarse en 2 principales procesos las cuales son la especificación de requisitos de calidad de software y la evaluación de la calidad del software, las cuales son soportadas en el proceso de medición de calidad de software	La norma ISO 25000 se especializa en la evaluación de productos Software, el objetivo principal de la ISO 25000 es brindar un panorama en general de los contenidos del SQuaRE, además de centrarse en 2 principales procesos las cuales son la especificación de requisitos de calidad de software y la evaluación de la calidad del software, las cuales son soportadas en el proceso de medición de calidad de software	Seguridad	N° de productos software que aplican la confidencialidad	Fichaje	Ficha de registro	X=A/B Donde X= N° de productos software que aplican la confidencialidad A: N° de datos encriptados/desencriptados correctamente B: N° de elementos de datos que requieren de encriptación/desencriptación	Razón
			Usabilidad	N° de productos software que son accesibles	Fichaje	Ficha de registro	X=A/B Donde X= N° de productos software que son accesibles A: N° de funciones a las que pueden acceder las personas B: N° total de elementos accesibles de la interfaz	
			Mantenibilidad	N° de productos software que pueden ser modificados.	Fichaje	Ficha de registro	X=A/T Donde X= N° de productos software que pueden ser modificados. A: N° de modificaciones T: Tiempo que le toma al desarrollador modificar	
V2: Proceso de desarrollo de software	El proceso de desarrollo de software incluye un conjunto de 5 actividades de un proceso en general: comunicación para entender los objetivos y definir los requerimientos y funciones del software; durante la planeación se definen las especificaciones técnicas que se realizara, probables riesgos, recursos a utilizar, y a definir el programa de actividades; modelado con la finalidad de entender los requerimientos en necesario realizar un bosquejo para entender el panorama total del problema un la solución; la construcción en esta actividad se realizará la generación del código y las pruebas para descartar los errores, y si existen solucionarlo antes de su puesta en producción; despliegue aquí ya se hace la entrega del software funcionando para que el usuario pueda usarlo y evaluarlo de acuerdo a sus necesidades (Pressman, 2010).	Para el proceso de desarrollo de software es vital apoyarse de las métricas ya que ayudan a demostrar de qué forma se desempeña el equipo y el producto software, así que, para tener una mejor medición del desempeño, las organizaciones trabajan bajo KPIs que permiten conocer el rendimiento de los miembros del equipo de desarrollo, el rendimiento del software, la calidad del software y la satisfacción del usuario final (Black, 2020).	Productividad del desarrollador	Porcentaje de velocidad de desarrollo de software	Fichaje	Ficha de registro	VD= (PC/PTH)*100 VD= Velocidad de desarrollo PC= Puntos completados PTH= Puntos totales de historia comprometida	Razón
			Defectos en el software	Porcentaje de detección de defectos	Fichaje	Ficha de registro	D= (DU/DT) * 100 D= Porcentaje de detección de defectos DT=Nro de defectos encontrados antes del lanzamiento DU= Nro de defectos encontrados por el usuario después del lanzamiento	
			Rendimiento del software	Porcentaje de fiabilidad	Fichaje	Ficha de registro	F=(HT-HTM)/HT* 100 F= Porcentaje de Fiabilidad HT= Horas totales HTM= Horas totales por mantenimiento no programado	

Anexo N° 2: Instrumentos de recolección de datos

Guía de observación N° 1. Indicador Porcentaje de velocidad de desarrollo de software

Ficha de registro de medición del indicador Porcentaje de velocidad de desarrollo de software / Pre-test				
Investigador:			Gina Rios Jorge	
Proceso observado:			Productividad del desarrollador	
Pre-test				
Ítem	**Fecha**	**Puntos completados**	**Puntos totales de historia comprometida**	**VD= (PC/PTH)*100 VD= Velocidad de desarrollo PC= Puntos completados PTH= Puntos total de historia comprometida**
1	1/08/2022	1	3	33
2	1/08/2022	1	3	33
3	1/08/2022	2	3	67
4	1/08/2022	0	3	0
5	1/08/2022	2	3	67
6	1/08/2022	1	3	33
.				
N	1/08/2022	1	3	33

Ficha de registro de medición del indicador Porcentaje de velocidad de desarrollo de software / Post-test				
Investigador:			Gina Rios Jorge	
Proceso observado:			Productividad del desarrollador	
Post-test				
Ítem	**Fecha**	**Puntos completados**	**Puntos totales de historia comprometida**	**VD= (PC/PTH)*100 VD= Velocidad de desarrollo PC= Puntos completados PTH= Puntos total de historia comprometida**
1	3/10/2022	3	3	100
2	3/10/2022	3	3	100
3	3/10/2022	3	3	100
4	3/10/2022	2	3	67
5	3/10/2022	3	3	100
6	3/10/2022	3	3	100
.				
N	3/10/2022	3	3	100

Guía de observación N° 2. Porcentaje de detección de defectos

Ficha de registro de medición del indicador Porcentaje de detección de defectos / Pre-test				
Investigador:			Gina Rios Jorge	
Proceso observado:			Defectos en el software	
Pre-test				
Ítem	**Fecha**	**Nro de defectos encontrados antes del lanzamiento**	**Nro de defectos encontrados por el usuario después del lanzamiento**	**D= (DU/DT) * 100 D= Porcentaje de detección de defectos DT=Nro de defectos encontrados antes del lanzamiento DU= Nro de defectos encontrados por el usuario después del lanzamiento**
1	1/08/2022	10	2	20
2	1/08/2022	8	1	13
3	1/08/2022	7	0	0
4	1/08/2022	12	1	8
5	1/08/2022	5	0	0
6	1/08/2022	7	1	14
.				
N	19/09/2022	6	1	17

Ficha de registro de medición del indicador Porcentaje de detección de defectos / Post-test				
Investigador:			Gina Rios Jorge	
Proceso observado:			Defectos en el software	
Post-test				
Ítem	**Fecha**	**Nro de defectos encontrados antes del lanzamiento**	**Nro de defectos encontrados por el usuario después del lanzamiento**	**D= (DU/DT) * 100 D= Porcentaje de detección de defectos DT=Nro de defectos encontrados antes del lanzamiento DU= Nro de defectos encontrados por el usuario después del lanzamiento**
1	3/10/2022	5	0	0
2	3/10/2022	4	0	0
3	3/10/2022	8	1	13
4	3/10/2022	7	0	0
5	3/10/2022	2	0	0
6	3/10/2022	6	0	0
.				
N	14/11/2022	1	0	0

Guía de observación N° 3. Porcentaje de fiabilidad

Ficha de registro de medición del indicador Porcentaje de fiabilidad / Pre-test				
Investigador:			Gina Rios Jorge	
Proceso observado:			Rendimiento del software	
Pre-test				
Ítem	**Fecha**	**Horas totales**	**Horas totales por mantenimiento no programado**	**F=(HT-HTM)/HT* 100 F= Porcentaje de Fiabilidad HT= Horas totales HTM= Horas totales por mantenimiento no programado**
1	1/08/2022	24	12	50
2	2/08/2022	24	14	42
3	3/08/2022	24	14	42
4	4/08/2022	24	10	58
5	5/08/2022	24	16	33
6	6/08/2022	24	8	67
.				
N	19/09/2022	24	15	38

Ficha de registro de medición del indicador Porcentaje de fiabilidad / Post-test				
Investigador:			Gina Rios Jorge	
Proceso observado:			Rendimiento del software	
Post-test				
Ítem	**Fecha**	**Horas totales**	**Horas totales por mantenimiento no programado**	**F=(HT-HTM)/HT* 100 F= Porcentaje de Fiabilidad HT= Horas totales HTM= Horas totales por mantenimiento no programado**
1	1/10/2022	24	2	92
2	2/10/2022	24	0	100
3	3/10/2022	24	4	83
4	4/10/2022	24	3	88
5	5/10/2022	24	0	100
6	6/10/2022	24	2	92
.				
N	19/11/2022	24	2	92

Anexo N° 3: Certificado de validación del instrumento de recolección de datos

CERTIFICADO DE VALIDEZ DE CONTENDO DEL INSTRUMENTO QUE MIDE EL PROCESO DE DESARROLLO DE SOFTWARE

Nº	DIMENSIONES / ítems	Pertinencia[1]		Relevancia[2]		Claridad[3]		Sugerencias
	DIMENSIÓN 1: Productividad del Desarrollador **INDICADOR: Porcentaje de velocidad de desarrollo de software**	Si	No	Si	No	Si	No	
1	VD= (PC/PTH)*100 VD= Velocidad de desarrollo PC= Puntos completados PTH= Puntos totales de historia comprometida							
	DIMENSIÓN 2: Defectos en el software **INDICADOR: Porcentaje de detección de defectos**	Si	No	Si	No	Si	No	
2	D= (DT/DU) * 100 D= Porcentaje de detección de defectos DT=Nro de defectos encontrados antes del lanzamiento DU= Nro de defectos encontrados por el usuario después del lanzamiento							
	DIMENSIÓN 3: Rendimiento del software **INDICADOR: Porcentaje de fiabilidad**	Si	No	Si	No	Si	No	
3	F=(HT-HTM)/HT* 100 F= Porcentaje de Fiabilidad HT= Horas totales HTM= Horas totales por mantenimiento no programado							

Observaciones (precisar si hay suficiencia):_____

Opinión de aplicabilidad: **Aplicable [X]** **Aplicable después de corregir [ ]** **No aplicable [ ]**

Apellidos y nombres del juez validador. Dr/ Mg: **ACUÑA BENITES MARLON FRANK** **DNI: 42097456**

Especialidad del validador:.....

[1]**Pertinencia:**El ítem corresponde al concepto teórico formulado.
[2]**Relevancia:** El ítem es apropiado para representar al componente o dimensión específica del constructo
[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión

20 de octubre del 2022

Firma del Experto Informante.

Anexo N° 5: Carta de aceptación

UNIVERSIDAD CÉSAR VALLEJO

POS GRA DO

"Año del Fortalecimiento de la Soberanía Nacional"

Lima, 11 de noviembre de 2022
Carta P. 1173-2022-UCV-VA-EPG-F01/J

Ing
Paul Cabrera Sanchez
JEFE
Banco de Credito BCP

De mi mayor consideración:

Es grato dirigirme a usted, para presentar a Rios Jorge, Gina Rosario; identificada con DNI N° 70087351 y con código de matrícula N° 6500078266; estudiante del programa de MAESTRÍA EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN quien, en el marco de su tesis conducente a la obtención de su grado de MAESTRA, se encuentra desarrollando el trabajo de investigación titulado:

Aplicación ISO 25000 para el Proceso de Desarrollo de Software en el Área de TI en una financiera, Lima 2023

Con fines de investigación académica, solicito a su digna persona otorgar el permiso a nuestra estudiante, a fin de que pueda obtener información, en la institución que usted representa, que le permita desarrollar su trabajo de investigación. Nuestra estudiante investigador Rios Jorge, Gina Rosario asume el compromiso de alcanzar a su despacho los resultados de este estudio, luego de haber finalizado el mismo con la asesoría de nuestros docentes.

Agradeciendo la gentileza de su atención al presente, hago propicia la oportunidad para expresarle los sentimientos de mi mayor consideración.

Atentamente,

ESCUELA DE POSGRADO UCV JEFATURA CAMPUS LIMA NORTE

Dra. Estrella A. Esquiagola Aranda
Jefa
Escuela de Posgrado UCV
Filial Lima Campus Los Olivos

Autoriza

Paul Cabrera

Somos la universidad de los que quieren salir adelante.

ucv.edu.pe

Anexo N° 6: Matriz de consistencia

PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES E INDICADORES				
			Variable 1: ISO 25000				
			Dimensiones	Indicadores	Instrumento	Niveles y Rango	Fórmula
Problema General:	**Objetivo General:**	**Hipótesis General:**					
¿De qué manera la aplicación del ISO 25000 ayudara a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023?	Determinar de qué manera la aplicación del ISO 25000 ayudara a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023.	La aplicación del ISO 25000 ayudo a mejorar en el proceso de desarrollo de software en el área de TI en una financiera, Lima 2023.	Seguridad	N° de productos software que aplican la confidencialidad	Ficha de registro	Razón	X=A/B Dónde X= N° de productos software que aplican la confidencialidad A: N° de datos encriptados/desencriptados correctamente B: N° de elementos de datos que requieren de encriptación/desencriptación
Problemas específicos:	**Objetivos Específicos:**	**Hipótesis Específicas:**					
Problema Específico 1:	**Objetivo Específico 1:**	**Hipótesis Específicas 1:**					
¿Cómo determinar la influencia del ISO 25000 sobre la Productividad del Desarrollador en el área TI de una financiera?	Determinar la influencia del ISO 25000 sobre la Productividad del Desarrollador en el área TI de una financiera.	El ISO 25000 influye en la Productividad del Desarrollador en el área TI de una financiera.	Usabilidad	N° de productos software que son accesibles	Ficha de registro	Razón	X=A/B Dónde X= N° de productos software que son accesibles A: N° de funciones a las que pueden acceder las personas B: N° total de elementos accesibles de la interfaz
Problema Específico 2:	**Objetivo Específico 2:**	**Hipótesis Específicas 2:**					
¿Cómo determinar la influencia del ISO 25000 sobre los defectos del software en el área TI de una financiera?	Determinar la influencia del ISO 25000 sobre los defectos del software en el área TI de una financiera.	El ISO 25000 influye en los defectos del software en el área TI de una financiera.	Mantenibilidad	N° de productos software que pueden ser modificados.	Ficha de registro	Razón	X=A/T Dónde X= N° de productos software que pueden ser modificados. A: N° de modificaciones T: Tiempo que le toma al desarrollador modificar

Problema Específico 3:	**Objetivo Específico 3:**	**Hipótesis Específicas 3:**	**V2: Proceso de desarrollo de software**				
			Dimensiones	**Indicadores**	**Instrumento**	**Niveles y Rango**	**Fórmula**
¿Cómo determinar la influencia del ISO 25000 sobre el Rendimiento del software en el área TI de una financiera?	Determinar la influencia del ISO 25000 sobre el Rendimiento del software en el área TI de una financiera.	El ISO 25000 influye en el Rendimiento del software en el área TI de una financiera.	Productividad del Desarrollador	Porcentaje de velocidad de desarrollo de software	Ficha de registro	Razón	VD= (PC/PTH)*100 VD= velocidad de desarrollo PC= Puntos completados PTH= Puntos total de historia comprometida
			Defectos en el software	Porcentaje de detección de defectos	Ficha de registro	Razón	D= (DU/DT) * 100 D= Porcentaje de detección de defectos DT=Nro de defectos encontrados antes del lanzamiento DU= Nro de defectos encontrados por el usuario después del lanzamiento
			Rendimiento del software	Porcentaje de fiabilidad	Ficha de registro	Razón	F=(HT-HTM)/HT* 100 F= Porcentaje de Fiabilidad HT= Horas totales HTM= Horas totales por mantenimiento no programado

Anexo Nº 7: Metodología de la aplicación ISO 25000 para el proceso de desarrollo de software

1. Etapa 1: Establecer los requisitos de la evaluación

1.1. Establecer el propósito de la evaluación

El propósito de la evaluación fue evaluar la calidad de los productos desarrollados por la financiera a lo largo de 4 Sprints, basándonos en que los productos software cumplan con los estándares de calidad para evaluar la usabilidad las cuales se establecen en la norma ISO/IEC 25010.

1.2. Obtener los requisitos de calidad del producto

Los requisitos de calidad para la característica de usabilidad fueron determinados por el jefe de TI de la financiera, tomando en cuenta lo planteado en la norma ISO/IEC 25010.

Se consideraron las siguientes subcaracterísticas de usabilidad, se muestran resultados en la tabla 1.

Tabla 1

Subcaracterísticas de usabilidad a evaluar

Subcaracterística de usabilidad	Subcaracterística de usabilidad
Capacidad para reconocer su adecuación	CapAd
Capacidad de aprendizaje	CapAp
Capacidad para ser usado	CapUs
Protección contra errores de usuario	ProErr
Estética de la interfaz de usuario	EstInt
Accesibilidad	Acc

1.3. Obtener los requisitos de calidad del producto

Las partes del producto software que se evaluaron fueron el tiempo de desarrollo, la documentación de las pruebas y tiempo de pruebas.

1.4. Definir el rigor de la evaluación

Para que los proyectos software tengan la condición de aprobado, como criterio de decisión de evaluación debería de ser buena o de alta calidad.

2. Etapa 2: Especificar la evaluación

2.1. Definir el rigor de la evaluación

Para cada subcaracterística de usabilidad el jefe de TI atribuyo un nivel de importancia. El grado de importancia se muestran en la tabla 2.

Tabla 2

Nivel de Importancia para la valoración de cada subcaracterística de la usabilidad

Nivel de importancia	Simbología	Significado
Alto	A	El grado de importancia de la característica y subcaracterística es alto por ende se realizará las mediciones
Medio	M	La característica y subcaracterística no es tan relevante, pero puede o no ser medida dependiendo del criterio del evaluador
Bajo	B	La característica y subcaracterística no tiene relevancia y no será medida
No Aplica	NA	Este valor se dará a la característica y subcaracterística que no se pueden medir dependiendo de diferentes factores

El jefe de TI, realizo la valoración de las subcaracterísticas de usabilidad de los productos software, teniendo en cuenta la tabla 3. Esta valoración sirvió de base para determinar la ponderación y de acuerdo a ello obtener la medida de usabilidad del producto software.

Tabla 3

Valoración de las subcaracterísticas de usabilidad

Subcaracterística de usabilidad	Nivel de importancia
Capacidad para reconocer su adecuación	M
Capacidad de aprendizaje	M
Capacidad para ser usado	A
Protección contra errores de usuario	A
Estética de la interfaz de usuario	A
Accesibilidad	M

A continuación, se elaboró el User Test el cual es presentado en la tabla 4. El presente test establece un instrumento de medición de usabilidad considerando los indicadores para cada subcaracterística, en base a la norma ISO/IEC 25010.

Tabla 4

Test del Usuario para la medición de la usabilidad de los productos software de la financiera.

Subcaracterística de usabilidad	Indicador
Capacidad para reconocer su adecuación	Claridad en las instrucciones Grado de eficiencia Grado de importancia
Capacidad de aprendizaje	Manual de usuario Pop-up de ayuda
Capacidad para ser usado	Facilidad de uso

	Facilidad de logueo Facilidad de cerrar sesión
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado. Mensajes orientados al uso
Estética de la interfaz de usuario	Combinación de colores Textos claros Gráficos interactivos
Accesibilidad	Contraseña encriptada Protección de datos sensibles Pruebas de seguridad

2.2. Definir los criterios de decisión para las métricas

Para realizar la medición de la usabilidad del producto software se aplicó al User Test una escala de evaluación del 1-4.

Según la valoración obtenida de cada indicador, utilizando la escala de Likert, se determinaron los criterios de decisión para las métricas que se presenta en la tabla 5.

Tabla 5

Criterios de decisión para las métricas

Escala	Puntuación	Criterio de decisión para las métricas
1	0 puntos	Bajo
2	2 puntos	Regular
3	3 puntos	Bueno
4	5 puntos	Excelente

2.3. Definir los criterios de decisión de la evaluación

La puntuación de cada subcaracterística de usabilidad está dada por la sumatoria de las puntuaciones obtenidas de los indicadores que corresponden a cada subcaracterística según la tabla **M**.

Para determinar la medida de la característica de usabilidad se tiene en cuenta los niveles de importancia asignadas a cada subcaracterística según la tabla **Z**. La puntuación de las subcaracterísticas valoradas como nivel de importancia Medio (M) se multiplica por 1 y la puntuación de las subcaracterísticas valoradas como nivel importancia Alto (A) se multiplica por 2; siendo la ecuación de la medida de usabilidad la siguiente:

Medida usabilidad= 1*CapAd + 1*CapAp + 2*CapUs + 2*ProtErr + 2*EstInt + 1*Acc

Los criterios de decisión de la evaluación para cada subcaracterística se determinaron según el rango de puntuación obtenido de la sumatoria de las puntuaciones de los indicadores que conforman cada subcaracterística. Estos criterios de decisión de la evaluación para cada subcaracterística se presentan en la tabla 6.

Tabla 6

Criterio de decisión de la evaluación para las subcaracterísticas de usabilidad

Rango de puntuación	Criterio de decisión de la evaluación para las subcaracterísticas
[10 – 9 puntos]	Alta Calidad
[9 – 6 puntos]	Buena Calidad
[6 – 2 puntos]	Regular Calidad
[2 – 0 puntos]	Mala Calidad

Los criterios de decisión de la evaluación para la característica de usabilidad del producto software se determinaron según el rango de puntuación obtenido aplicando la ecuación 1. Estos criterios de decisión de la evaluación para la característica de usabilidad se detallan en la tabla 7.

Tabla 7

Criterio de decisión de la evaluación para la característica de usabilidad

Rango de puntuación	Criterio de decisión de la evaluación para la característica de usabilidad
[80 – 70 puntos]	Alta Calidad
[69 - 40 puntos]	Buena Calidad
[39 – 20 puntos]	Regular Calidad
[19 – 0 puntos]	Mala Calidad

3. Etapa 3: Diseñar la evaluación

.

3.1. Definir los criterios de decisión de la evaluación

Las actividades de la evaluación del producto software de un juego fueron las siguientes:

a. Presentación del software desarrollado.
b. Definir escenarios para medir la usabilidad del software
c. El jefe de TI tomará el rol del usuario y responderá el User Test en la tabla 4, para poder medir la usabilidad del producto software

4. Etapa 4: Ejecutar la evaluación

En esta etapa se ejecutan las actividades de evaluación obteniendo las métricas decalidad y aplicando los criterios de evaluación

4.1. Realizar las mediciones

Se realizó las mediciones de producto software académicos correspondientes a cinco productos de desarrollo. Las mediciones se presentan en la tabla 8.

Tabla 8

Mediciones de producto software académicos

Subcaracterística de usabilidad	Indicador	Juego				
		anx7	rest	artu	tdsa	raas
Capacidad para reconocer su adecuación	Claridad en las instrucciones	2	3	5	3	5
	Grado de eficiencia	3	3	5	5	5
Capacidad de aprendizaje	Manual de usuario	3	3	3	2	3
	Pop-up de ayuda	2	3	2	3	2
Capacidad para ser usado	Facilidad de uso	3	5	2	5	3
	Facilidad de logueo	3	2	3	2	5
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	2	3	3	2	3
	Mensajes orientados al uso	3	2	3	2	5
Estética de la interfaz de usuario	Combinación de colores/	5	3	3	2	5

	Gráficos interactivos					
	Textos claros	5	3	2	5	3
Accesibilidad	Contraseña encriptada	5	5	5	5	5
	Protección de datos sensibles	5	3	3	2	5

4.2. Aplicar los criterios de decisión para las métricas

Considerando la tabla 5, se aplican los criterios de decisión para las métricas según los puntajes obtenidos de los indicadores del Test de Usuario, lo cual se detalla en la tabla 9**.**

Tabla 9

Aplicación de los criterios de decisión para las métricas a las mediciones de producto software académicos

Subcaracterísti ca de usabilidad	**Indicador**	**Juego**				
		anx7	**rest**	**artu**	**tdsa**	**raas**
Capacidad para reconocer su adecuación	Claridad en las instruccion es	Regular	Bueno	Excelent e	Bueno	Excelent e
	Grado de eficiencia	Bueno	Bueno	Excelent e	Excelent e	Excelent e
Capacidad de aprendizaje	Manual de usuario	Bueno	Bueno	Bueno	Regular	Bueno
	Pop-up de ayuda	Regular	Bueno	Regular	Bueno	Regular
Capacidad para ser usado	Facilidad de uso	Bueno	Excelent e	Regular	Excelent e	Bueno
	Facilidad de logueo	Bueno	Regular	Bueno	Regular	Excelent e

Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	Regular	Bueno	Bueno	Regular	Bueno
	Mensajes orientados al uso	Bueno	Regular	Bueno	Regular	Excelente
Estética de la interfaz de usuario	Combinación de colores/ Gráficos interactivos	Excelente	Bueno	Bueno	Regular	Excelente
	Textos claros	Excelente	Bueno	Regular	Excelente	Bueno
Accesibilidad	Contraseña encriptada	Excelente	Excelente	Excelente	Excelente	Excelente
	Protección de datos sensibles	Excelente	Bueno	Bueno	Regular	Excelente

4.3. Aplicar los criterios de decisión para las métricas

A nivel de subcaracterísticas se realiza la suma de las puntuaciones adquiridas por los indicadores que conforman cada una de las subcaracterísticas, obteniendo un valor por cada subcaracterística. A este valor se aplica los criterios de decisión de la evaluación según la tabla 6.

Teniendo en cuenta los valores consolidados por cada subcaracterística de la usabilidad, se aplica la ecuación 1 obteniendo un valor a nivel de la característica de usabilidad. A este valor se aplica los criterios de decisión de la evaluación según la tabla 7. Las tablas del 10 al 14 presentan los criterios de decisión de la evaluación a nivel de la característica de usabilidad y a nivel de sus subcaracterísticas para cada uno de los productos software académicos evaluados en esta investigación.

Tabla 10

Criterios de decisión de la evaluación a nivel de la característica de usabilidad y a nivel de sus subcaracterísticas de la aplicación Anx7

Subcaracterística de usabilidad	**Indicador**	**Aplicación: anx7**					
		A nivel de subcaracterísticas			**A nivel de característica de usabilidad**		
Capacidad para reconocer su adecuación	Claridad en las instrucciones	2	5	Regular Calidad	1*5	62	Buena Calidad
	Grado de eficiencia	3					
Capacidad de aprendizaje	Manual de usuario	3	5	Regular Calidad	1*5		
	Pop-up de ayuda	2					
Capacidad para ser usado	Facilidad de uso	3	6	Buena Calidad	2*6		
	Facilidad de logueo	3					
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	2	5	Regular Calidad	2*5		
	Mensajes orientados al uso	3					
Estética de la interfaz de usuario	Combinación de colores/ Gráficos interactivos	5	10	Alta Calidad	2*10		
	Textos claros	5					
Accesibilidad	Contraseña encriptada	5	10	Alta Calidad	1*10		
	Protección de datos sensibles	5					

Tabla 11

Criterios de decisión de la evaluación a nivel de la característica de usabilidad y a nivel de sus subcaracterísticas de la aplicación Rest

Subcaracterística de usabilidad	**Indicador**	**Aplicación: rest**				
		A nivel de subcaracterísticas			**A nivel de característica de usabilidad**	
Capacidad para reconocer su adecuación	Claridad en las instrucciones	3	6	Buena Calidad	1*6	56 Buena Calidad
	Grado de eficiencia	3				
Capacidad de aprendizaje	Manual de usuario	3	6	Buena Calidad	1*6	
	Pop-up de ayuda	3				
Capacidad para ser usado	Facilidad de uso	5	7	Buena Calidad	2*7	
	Facilidad de logueo	2				
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	3	5	Regular Calidad	2*5	
	Mensajes orientados al uso	2				
Estética de la interfaz de usuario	Combinación de colores/ Gráficos interactivos	3	6	Buena Calidad	2*6	
	Textos claros	3				
Accesibilidad	Contraseña encriptada	5	8	Buena Calidad	1*8	
	Protección de datos sensibles	3				

Tabla 12

Criterios de decisión de la evaluación a nivel de la característica de usabilidad y a nivel de sus subcaracterísticas de la aplicación Artu

Subcaracterística de usabilidad	**Indicador**	**Aplicación: artu**					
		A nivel de subcaracterísticas			**A nivel de característica de usabilidad**		
Capacidad para reconocer su adecuación	Claridad en las instrucciones	5	10	Alta Calidad	1*10	55	Buena Calidad
	Grado de eficiencia	5					
Capacidad de aprendizaje	Manual de usuario	3	5	Regular Calidad	1*5		
	Pop-up de ayuda	2					
Capacidad para ser usado	Facilidad de uso	2	5	Regular Calidad	2*5		
	Facilidad de logueo	3					
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	3	6	Buena Calidad	2*6		
	Mensajes orientados al uso	3					
Estética de la interfaz de usuario	Combinación de colores/ Gráficos interactivos	3	5	Regular Calidad	2*5		
	Textos claros	2					
Accesibilidad	Contraseña encriptada	5	8	Buena Calidad	1*8		
	Protección de datos sensibles	3					

Tabla 13

Criterios de decisión de la evaluación a nivel de la característica de usabilidad y a nivel de sus subcaracterísticas de la aplicación Tdsa

Subcaracterística de usabilidad	**Indicador**	**Aplicación: tdsa**				
		A nivel de subcaracterísticas			**A nivel de característica de usabilidad**	
Capacidad para reconocer su adecuación	Claridad en las instrucciones	3	8	Buena Calidad	1*8	56 Buena Calidad
	Grado de eficiencia	5				
Capacidad de aprendizaje	Manual de usuario	2	5	Regular Calidad	1*5	
	Pop-up de ayuda	3				
Capacidad para ser usado	Facilidad de uso	5	7	Buena Calidad	2*7	
	Facilidad de logueo	2				
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	2	4	Regular Calidad	2*4	
	Mensajes orientados al uso	2				
Estética de la interfaz de usuario	Combinación de colores/ Gráficos interactivos	2	7	Buena Calidad	2*7	
	Textos claros	5				
Accesibilidad	Contraseña encriptada	5	7	Buena Calidad	1*7	
	Protección de datos sensibles	2				

Tabla 14

Criterios de decisión de la evaluación a nivel de la característica de usabilidad y a nivel de sus subcaracterísticas de la aplicación Raas

Subcaracterística de usabilidad	**Indicador**	**Aplicación: raas**				
		A nivel de subcaracterísticas			**A nivel de característica de usabilidad**	
Capacidad para reconocer su adecuación	Claridad en las instrucciones	5	10	Alta Calidad	1*10	73 Alta Calidad
	Grado de eficiencia	5				
Capacidad de aprendizaje	Manual de usuario	3	5	Regular Calidad	2*5	
	Pop-up de ayuda	2				
Capacidad para ser usado	Facilidad de uso	3	8	Buena Calidad	2*8	
	Facilidad de logueo	5				
Protección contra errores de usuario	Mensajes de alerta en caso se dé un uso inadecuado.	3	8	Buena Calidad	2*8	
	Mensajes orientados al uso	5				
Estética de la interfaz de usuario	Combinación de colores/ Gráficos interactivos	5	8	Buena Calidad	2*8	
	Textos claros	3				
Accesibilidad	Contraseña encriptada	5	10	Alta Calidad	1*10	
	Protección de datos sensibles	5				

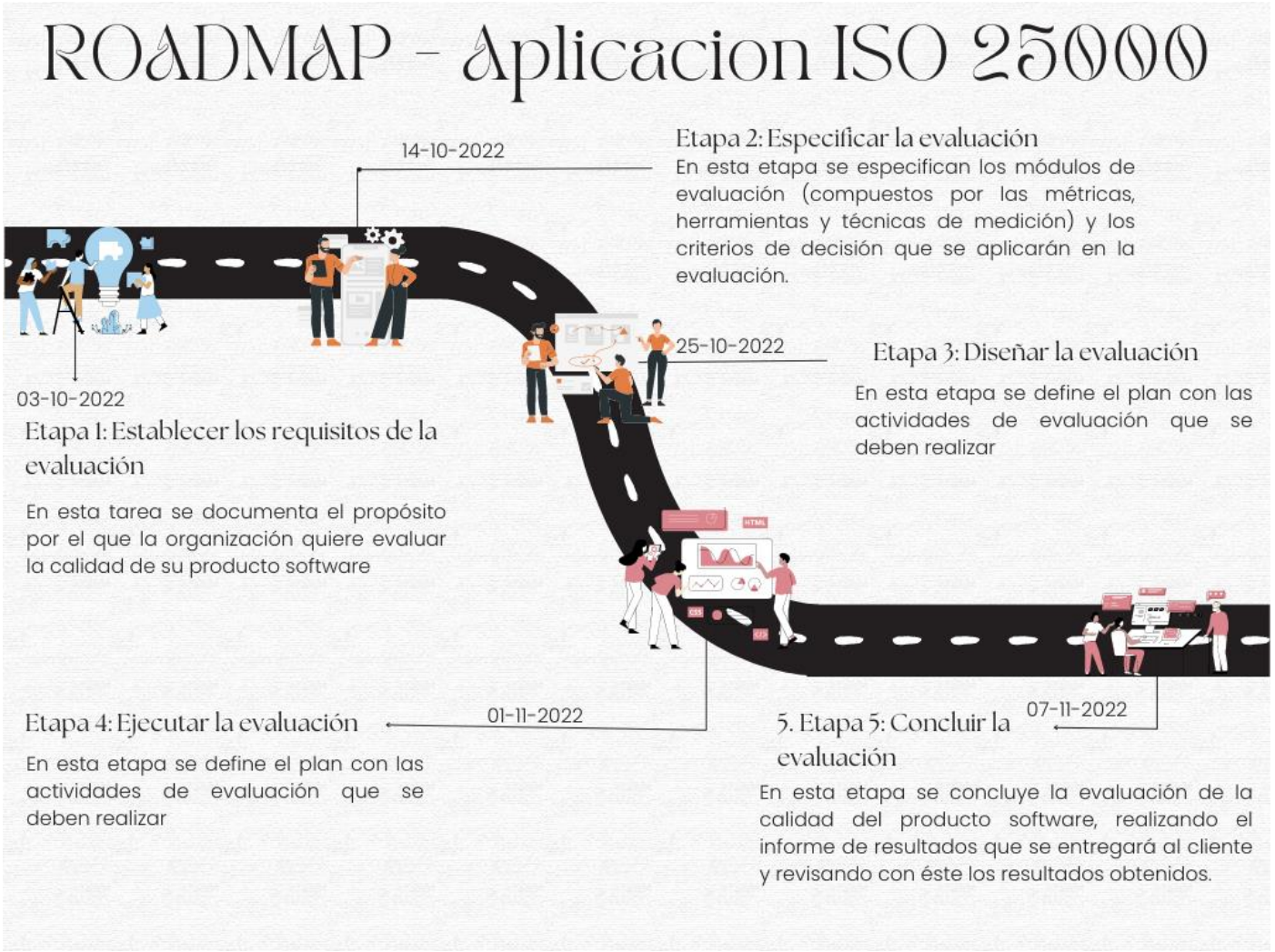
5. Etapa 5: Concluir la evaluación

Luego de haber realizado el proceso de medición y evaluación se determinaron las siguientes conclusiones.

a. De los 5 productos software evaluados a nivel de la característica usabilidad 1 producto software alcanzó el criterio de decisión de evaluación de alta calidad y 4 productos software alcanzaron buena calidad.
b. Según el rigor de la evaluación definido en la etapa 1.4, se determinó que de los cinco productos software evaluados 5 alcanzaron la condición de aprobados que fueron anx7, rest, artu, tdsa y raas.
c. Se recomienda retroalimentar a el equipo de desarrollo acerca de las subcaracterísticas que se deben tener en cuenta al momento de implementar un nuevo producto software o al realizar la modificación, así mismo es necesario que el equipo de calidad tome en cuenta los casos de uso para hacer las pruebas con las subcaracterísticas de la usabilidad.

Figura 1:

ROADMAP - Aplicacion ISO 25000



UNIVERSIDAD CÉSAR VALLEJO

ESCUELA DE POSGRADO

MAESTRÍA EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Declaratoria de Autenticidad del Asesor

Yo, ACUÑA BENITES MARLON FRANK, docente de la ESCUELA DE POSGRADO MAESTRÍA EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA NORTE, asesor de Tesis titulada: "Aplicación ISO 25000 para el Proceso de Desarrollo de Software en el Área de TI en una Financiera, Lima 2023", cuyo autor es RIOS JORGE GINA ROSARIO, constato que la investigación tiene un índice de similitud de 15.00%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

LIMA, 06 de Enero del 2023

Apellidos y Nombres del Asesor:	**Firma**
ACUÑA BENITES MARLON FRANK **DNI:** 42097456 **ORCID:** 0000-0001-5207-9353	Firmado electrónicamente por: MACUNABE el 06-01-2023 15:07:26

Código documento Trilce: TRI - 0511440

INVESTIGA UCV